



17

LANBIDE HEZIKETARAKO Materialak

DATU-BASEAK

Euskara Zerbitzua
Ikasmaterialak

Toribio Etxebarria
Lanbide Heziketarako Materialak

17

Datu-baseak

Joan Aldamizetxebarria

Xiomara Gezuraga

Idoia Mugartegi

Gorka Guenaga

Rubén San Martín

EUSKO JAURLARITZA



GOBIERNO VASCO

HEZKUNTZA, UNIBERTSITATE
ETA IKERKETA SAILA

DEPARTAMENTO DE EDUCACIÓN,
UNIVERSIDADES E INVESTIGACIÓN

Eusko Jaurlaritzaren Argitalpen Zerbitzu Nagusia

Servicio Central de Publicaciones del Gobierno Vasco

Vitoria-Gasteiz, 2009

Lan honen bibliografia-erregistroa Eusko Jaurlaritzako Liburutegi Nagusiaren katalogoan aurki daiteke:
<http://www.euskadi.net/ejgvbiblioteca>

ARGITARATUTAKO IZENBURUAK

1. Prototipo elektronikoen garapena eta eraikuntza
2. Finantza kudeaketa
3. Giza baliabideak
4. Kultur animazioa
5. Analisi kimiko eta tresna bidezkoa
6. Laborategiko antolaketa eta kudeaketa
7. Farmazia- eta parafarmazia-produktuen prestaketa
8. Esku-hartze komunitarioen metodologia
9. Taldeko gorputz- eta kirol-ekintzak
10. Biltegiatze-lanak
11. Komunitate-garapena
12. Jolasaren oinarri teorikoak eta umeen jolasak: (animaziorako jokoak eta jolas-ekintza fisikoak)
13. Lan prestakuntza eta orientabidea: lan zuzenbidea
14. Fabrikazio mekanika material osagarria
15. Enpresa txikiaren kudeaketa: lana, zerga-sistema eta administrazioa
16. Aurrekontuak eta hornikuntza ostalaritzako enpresetan
17. Datu-baseak

Hezkuntza, Unibertsitate eta Ikerketa Sailak onetsia 2008/04/14

Argitaraldia: 1. a, 2009ko iraila

Ale-kopurua: 500

© Euskal Autonomia Erkidegoko Administrazioa
Hezkuntza, Unibertsitate eta Ikerketa Saila
internet: www.euskadi.net <<http://www.euskadi.net>>

Argitaratzailea: Eusko Jaurlaritzaren Argitalpen Zerbitzu Nagusia
Servicio Central de Publicaciones del Gobierno Vasco
Donostia-San Sebastián, 1 - 01010 Vitoria-Gasteiz

Egilea: Joan Aldamizetxebarria, Xiomara Gezuraga, Idoia Mugartegi,
Gorka Guenaga eta Rubén San Martín

Fotokonposizioa: Eusko Printing Service, S.L.

Inprimaketa: Gráficas Santamaría, S.A.

ISBN: 978-84-457-2979-3

L.G. VI-147/2009

AURKIBIDEA

1. DATU-BASEEN IBILBIDEA HISTORIAN ZEHAR	7
1.1. SARRERA	9
1.1.1. Datu-baseak vs fitxategiak.....	11
1.1.2. Datu-baseak vs DBKS	12
1.2. EREDU HIERARKIKOA.....	14
1.3. SARE EREDUA	15
1.4. EREDU ERLAZIONALA	16
1.5. DATU-BASE BANATUAK.....	16
1.6. OBJEKTUEI ZUZENDUTAKO DATU-BASEAK.....	17
1.7. DATU-BASEAK, ZERGATIK?	17
1.8. DATU-BASEEN SISTEMEN EZAUGARRIAK.....	18
1.9. ONDORIOAK.....	19
2. DATUEN EREDU ERLAZIONALA	21
2.1. SARRERA	23
2.2. DATU-BASE ERLAZIONALEN EGITURA.....	25
2.3. EREDU ERLAZIONALA ETA ALJEBRA ERLAZIONALA.....	28
3. ARKITEKTURA ETA OSAGAIK	33
3.1. SARRERA.....	35
3.2. DATU-BASEEN SISTEMEN ARKITEKTURA.....	35
3.2.1. Egitura fisikoa: barneko eskema.....	36
3.2.2. Egitura logikoa: eskema kontzeptuala.....	36
3.2.3. Kanpoko eskema.....	36
3.3. DATU-BASEEN SISTEMA ERLAZIONALEN OSAGAIK.....	37
3.3.1. Datuak.....	37
3.3.2. Softwarea: DBKS	38
3.3.3. Erabiltzaileak.....	39
3.3.4. Datu-hiztegia.....	41
3.3.5. Indizeak.....	41
3.4. DBKS-AK LAN EGITEKO DUEN ERA, ZENBAIT MODULUREN IKUSPUNTUTIK.....	43

4. ANALISIA ETA DISEINUA	45
4.1. SARRERA.....	47
4.2. DATUEN ANALISIRAKO ETA ESPEZIFIKAZIORAKO TEKNIKA. E/R EREDUA.....	48
4.2.1. Oinarrizko eredua (E/R).....	48
4.2.2. Eredu hedatua (EE/R).....	51
4.2.3. Martin eredua.....	58
4.3. DATUEN DISEINUA.....	60
4.3.1. Eredu erlazionala lortzeko arauak.....	61
4.4. NORMALIZAZIO PROZESUA.....	66
4.4.1. Adibideak.....	73
5. DATUEN ERABILERA. LENGOAIK	79
5.1. SARRERA.....	81
5.2. DML.....	81
5.2.1. Datuak atzitzeko sententziak. SELECT.....	82
5.2.2. Datu-basearen aldaketak.....	92
5.2.3. DCL.....	94
5.3. DDL.....	94
5.3.1. Sententziak.....	94
5.4. DATU HIZTEGIA.....	97
6. DATU-BASEEN ADMINISTRAZIOA	99
6.1. SARRERA.....	101
6.2. DATU-BASEEN EGITURAREN ADMINISTRAZIOA.....	101
6.3. KONFIDENTZIALTASUNAREN KUDEAKETA.....	102
6.3.1. Elementuen gaineko baimenak.....	103
6.3.2. Sistemaren baimenak.....	104
6.3.3. Baimenak kentzea.....	105
6.3.4. Erabiltzaileak sortu eta ezabatzea.....	106
6.3.5. Rolak.....	107
6.3.6. Profilak.....	108
6.4. ATZIPEN KONTROLAREN AUDITORIA.....	109
7. DATU-BASEEN OPTIMIZAZIOA	111
7.1. KONTZEPTUA.....	113
7.2. OPTIMIZAZIOAREN ARDURADUNAK.....	114
7.2.1. Datu-basearen diseinatzailearen optimizazioak.....	114
7.2.2. Datu-basearen administratzailearen optimizazioak.....	117
7.2.3. Datu-basearen erabiltzailearen optimizazioak.....	118
7.2.4. Datu-basearen sistemaren ardurapeko optimizazioak.....	120
8. DATUEN SEGURTASUNERAKO TEKNIKAK ETA PROZEDURAK	125
8.1. SEGURTASUNA: ZERTAZ ARI GARA?.....	127

8.2. DBKS-AK ETA SEGURTASUNA.....	129
8.2.1. Konfidentziasuna	130
8.2.2. Erabilgarritasuna.....	130
8.2.3. Osotasuna.....	137
9. DATU-BASE BANATUAK.....	143
9.1. SARRERA	145
9.2. DATU-BASE BANATUEN SISTEMAK.....	145
9.3. DATU-BASE BANATUEN DISEINUA.....	147
9.3.1. Zatiketa eta kokapena	150
9.3.2. Errepikapena eta kokapena.....	154
9.4. KONTSULTEN PROZESAKETA.....	155
9.4.1. Gardentasuna eta kontsulten prozesaketa.....	156
9.4.2. Datuen transferentziak.....	157
9.5. KONKURRENTZIAREN KONTROLA.....	159
9.5.1. Blokeoaren koordinatzaile bakarra (lehentasunezko lekua).....	159
9.5.2. Blokeoaren koordinatzaile banatua (lehentasunezko kopia)	160
9.5.3. Gehiengo bidezko blokeo-koordinatzailea	160
9.6. BERRESKURAPENA.....	160
9.6.1. Bi faseko konpromisoa (<i>two-phase commit, 2PC</i>).....	161
9.6.2. Hiru faseko konpromisoa (<i>three-phase commit, 3PC</i>).....	163
ERANSKINA.....	180
1. SARE EREDUKO DATU-BASEAK.....	180
2. EREDU HIERARKIKOA	192
GLOSARIOA.....	201
BIBLIOGRAFIA.....	207

*Datu-baseen ibilbidea
historian zehar*

1

1.1. SARRERA

Ukaezina da historian zehar informazioa jaso ahal izateak izan duen garrantzia. Belaunaldiz belau-naldi hainbat euskarritan pasatu eta gorde izan da, garaian garaikoak erabiliz. Era berean, ezinbestekoa da informazioa ondo antolatzea informazio hori kudeatuko dutenen lana erraztu nahi bada. Demagun, adibidez, banketxeak sortu ziren garaia. Sasoi hartan liburuetan idazten zituzten diruen mugimen-duak, informazioa gordetzeko era bakarra zelako. Informazio hori ezin zen jatorrizko kokagunetik kanpo irakurri, ezin ziren bezero jakin baten mugimenduak zerrendatu,... Gaur egun inolako arazorik gabe egiten dira horrelako eragiketak. Erroldena da beste adibide bat: erromatarren garaitik egiten dira, baina herri bakoitzeko gizabanakoak zenbatzeko eta informazio hori gordetzeko erak zeharo aldatu dira. XX. mendera arte, zenbaketak eskuz egin izan dira. Geroztik, makinek egin dute zenbaketa, askoz ziurrago eta arinago, gainera.

Bestalde, informazioaren bolumena izugarri hazi da. Bolumen hori kudeatu beharrak datuak biltzeko eta kudeatzeko sistemak erabat hobetsi ditu. Duela 3.000 urte inguru, ahozko transmisioa nahikoa izan zitekeen. Hala ere, mendeetan aurrera egin ahala, idatzizkora pasatu beharra egon da, informazio asko galdu egiten baita. Idatzizko informazioa ere era antolatuan gorde izan da harrezke-ro, liburuen eta liburutegien bitartez. Mende luzetan, liburutegiak izan dira informazioaren gordailu antolatu bakarrak, XX. mendean ordenagailua agertu arte. Garai hartan imajinatu ere ezin zitezkeen informazio edo datu-kopuruak (ikus 1.1 testu-taula) biltegitratzen dira gaur egun ordenagailuetan, era arautuan. Aurrerapauso hori, informazioa atzitzeko sistema berriekin batera etorri da, sistema sekuen-zialetatik abiatu eta ausazko sistemetara heldu arte.

Atzipen sekuentzialeko lehenengo sistemetan datuak fitxategietan biltzen ziren. Orokorrean, aplikazio batek hainbat egituratako datuak erabiltzen zituenez, fitxategi asko erabiltzen ziren: batzuk datuak hartzeko, beste batzuk, ateratzeko, esate baterako. Fitxategiak zabaldu, errenkadak irakurri, errenkadak gehitu eta fitxategia ixteko instrukzioak baino ez zituzten eskaintzen hasierako lengoia horiek.

Baina datuak pilatzeko sistema sekuentzial hori ez zen behar bezain eraginkorra, errenkada as-koko fitxategietan datuak bilatzea oso geldoa baitzen. Disko magnetikoen agerpenak, zintak ordeztuz, datuen pilaketaren kontzepzio berria ekarri zuen: atzipen sekuentzialaren ordeztuz ausazko atzipena nagusitu zen. Hori fitxategi-aplikazioetan¹ aurrerapauso handia izan zen.

Hala ere, fitxategien erabilerak arazoak zekartzan; datuak eguneratzeko orduan, esate baterako. Fi-txategi batean datu jakin bat aldatu arren besteetan aldatzeke geratuko balitz, informazio okerrarekin arituko ginateke lanean. Hori dela eta, datuak biltzeko sistemak erabat aldatu ziren 60ko hamarkadan, datu-baseen sorrerarekin. Sistema horiek datuak era osoan eta egonkorrean kudeatzen dituzte: bista-ratu egiten dituzte, aldatu, txertatu... Horretarako, datu-basea kudeatzeko sistema (DBKS) deritzon programa-multzoa du.

¹ "Fitxategi-aplikazio" hitza erabili da datuak gordetzeko fitxategiak erabiltzen dituzten aplikazioez hitz egiteko.

Informazioa vs datuak:

“Informazio” eta “datu” kontzeptuak sinonimoak balira bezala erabili ditugu orain arte. Baina ez dira, adibide honetan ikusten denez: demagun pertsona bati buruzko hiru datu daudela, hala nola *15402323-T, Peio, Uriarte*. Hiru datu horiek pertsona horren Nortasun-agiria (NA), izena eta abizena dira. Datuok hauteskunde-mahai bateko zerrendan baleude, “bozkatzeko aukera du” informazioa emango lukete. Aldiz, datu horiek kiroldegian baleude, “kiroldegiko bazkide da” esan nahiko lukete. Ikusten denez, datu berberak informazio ezberdina transmititzen dute.

Datuek esanahi ezberdina hartzen dute kontestuaeren arabera. Datuak zerbait objektiboa dira; informazioa, aldiz, zerbait subjektiboa, ikusi den bezala datu berek bi irakurketa ezberdin sortzen baitituzte. Hauteskundeetako emaitzen kasua ere adibide argia da. Datu berberak irakurketa ezberdinak izaten dituzte, alderdi politikoen arabera.

Egungo edozein ekintza planifikatzeko, ezinbestekoa da informazioa prozesatzea eta manipulatzea. Edozein arlotan, batez ere enpresa munduan, datu pila manipulatu behar da eta, era berean, datu horiek prozesatzeko abiadura ere azkartu beharra dago. Zenbakizko magnitudeak, izenak edo ikur-multzoak, esaldiak, irudiak, soinuak eta abar izan daitezke datuak. Kode edo lengoia formal batez adieraziko dira komunikatu edo prozesatu ahal izateko. Baina datuek, beraiek baka-rik, ez dute beharrezko ezagupena emango; horretarako, datu horiek prozesatu egin behar dira. Datuak eraldatzearen emaitza da informazioa. Edo, beste era batera esanda, agerbide arauak erabiliz datuei ematen zaien esanahia da informazioa transmititzea. Informazioa emateko, datuak identifikatu, bilatu, elkartu eta abar egin behar dira. Ondorengo taulan bildu dira zenbait adibide. Kasuan kasu, datu berberak informazio ezberdina ematen dutela ikusten da. *21.345.456* zenbakia NA zenbakia izan daiteke edo, nola ez, kontu korrante baten saldoa ere bai.

Datua	<i>2002-05-27</i>	<i>21.345.456</i>	<i>53</i>
Informazioa	VISAren iraugitze-data	Na zenbakia	Langilearen adina
	Datu-baseen azterketa-eguna	Kontuaren saldoa	Lanbidearen kodea

1.1 testu-taula. Informazioa *versus* datuak.

1.1.1. Datu-baseak vs fitxategiak

Datu-baseen aitzindari izan ziren fitxategiak, baina aldaketa ez zen bat-batekoa izan. Izan ere, ez da etenaldi zehatzik egon non fitxategien erabilera bertan behera geratu eta datu-baseen sistemak erabiltzen hastea nagusitu izan baiten. Are gehiago, gaur egun oraindik erabili egiten dira fitxategi-aplikazioak.

Fitxategiek denbora luzerako eta kopuru handian pilatzen dituzte datuak. Baina ezbeharren bat gertatuz gero, informazioa galtzea ohikoa izaten da, eta berreskuratzeko era bakarra segurtasun-kopiak egitean datza. Gainera, segurtasun-kopiak egiteko erantzukizuna bete-betean dagokio erabiltzaileari, bera izango baita egunero kopia horiek egin beharko dituen. Hala ere, ustekabeen datuak galduz gero, segurtasun-kopiatik berreskuratu beharko dira. Baina azken kopiatik ezbeharra izan arte egingdako lana galdu egiten da. Datu-baseekin, ordea, alferrik galdutako lana erabat minimizatzen da.

Horretaz aparte, fitxategi-aplikazioetan ez da batere erosoak datuak kontsultatzeko eta manipulatzeko lengoia edo interfazea. Lehenengo fitxategi-aplikazioetan datuak ondo bereizitako ataletan biltzen ziren eta oso txikia zen haien arteko erlazioa. Era berean, gerora, fitxategi-aplikazioek ez dituzte aurreikusi normalean hainbat erabiltzailek aldi berean egingdako atzipenak.

Aplikazioen analisia eginez gero, alde batetik fitxategiek erabiltzen dituzten aplikazioak daude eta, bestetik, datu-baseek erabiltzen dituztenak.

Lehenengoei, **gertaerei zuzendutako sistema** deritze, egin beharko dutena analizatu eta gero aplikazioak beharrezkoak dituen fitxategiak sortzen baititu. Ondoren, beste aplikazioen bat behar izanez gero, horren analisia egin eta beharrezkoak dituen fitxategiak sortzen dira; hau da, fitxategiak aplikazio bakoitzaren mendekoak dira eta sortutako fitxategiak, elkarren artean guztiz independenteak. Hori dela eta, aplikazio-kopurua hazten doan heinean, fitxategiak ugaritu egiten dira. Gainera, sarritan datuak errepikatuta agertzen dira hainbat fitxategitan, memoria alferrik betez. Eta, are txarrago dena: datuen sendotasunik ezak zentzugabeko informazio-pilaketa eragin lezake, datu berberak hainbat fitxategitan zenbait balio ezberdinekin errepikatuta agertu baitaitezke. Hona hemen adibidea:

Demagun eskola bateko idazkaritzan ikasleak kudeatzeko aplikazioa ezarri beharra dagoela eta beste aplikazio bat irakasleen gelan, irakasleek ikasleen notak kudeatzeko:

Idazkaritzako aplikazioak ikasleak matrikulatu, ezabatu eta ikasle guztien zerrendak atera beharko ditu. Horretarako, IKASLEAK izeneko fitxategia erabiliko du. Berorren egitura 1.1 taulan islatzen da.

Irakasleen gelan, notak sartzeko aplikazioarekin ikasleen notak sartu eta klase bateko ikasleen noten zerrenda egin nahi da. Horretarako, besteak beste, 1.2 taulako TALDEAK eta 1.3 taulako NOTAK bezalako erregistro-egitura duten fitxategiak erabiliko dituzte.

MATRIKULA-ZENBAKIA	NA	IZENA	ABIZENA	HELBIDEA
--------------------	----	-------	---------	----------

1.1 taula. IKASLEAK fitxategiaren egitura.

NA	IZENA	ABIZENA	HELBIDEA	TALDEA
----	-------	---------	----------	--------

1.2 taula. TALDEAK fitxategiaren egitura.

PROGRAMAZIOA	SAREAK	DATU-BASEAK	HELBIDEA	TALDEA	IZENA	ABIZENA
--------------	--------	-------------	----------	--------	-------	---------

1.3 taula. NOTAK fitxategiaren egitura.

Aipatutako adibidean, idazkaritzako aplikazioaren bidez ikasleren bat ezabatzen bada, eragiketa hori ez da TALDEAK fitxategian islatuko, bigarren aplikazio horrek ez baitu fitxategi hori kudeatzen. Hori dela eta, fitxategian ez dagoen ikasle baten notak kudeatzen aritzea gerta liteke. Ondorioz, prozesuei begira garatutako aplikazioetan datuek programarekiko daukaten mendekotasuna nabaritzen da alde batetik, eta malgutasun falta bestetik. Gainera, edozein helbide-aldaketak arazoak ekar litzake.

Eragozpen horiek direla eta, datuei zuzendutako sistemek garrantzi handia hartu dute. Horien artean koka daitezke datu-baseen sistemak.

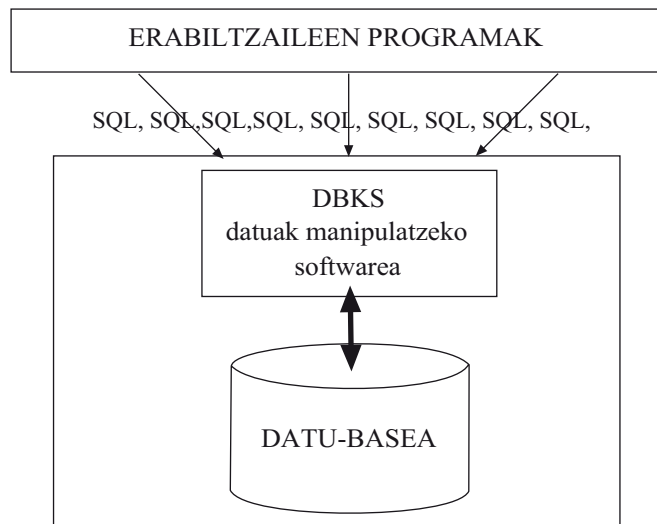
Hirurogeiko hamarkadan, informazioaren egitura datu-base batean deskribatzeko aukera ematen zuten ereduak agertzen hasi ziren. Horrela, aplikazioen eta datuen pilaketaren arteko independentzia handiagoa lortu zen. Izan ere, elkarrekin erlazionatutako datuen bilduma baino ez da datu-basea. Bilduma horrek mundu errearen zati bat adieraziko du. Adierazi nahi den mundu errearen zatia bada ere, pertsona bakoitzak bere erara ikusiko du, era subjektiboan, eta era jakin batean adieraziko du, unean uneko komunikazio-tresna egokiena erabiliz. Hauteskundeetako emaitzena dugu adibide aproposa, gertakari erreala, non alderdi politiko bakoitzak bere irakurketa propioa egiten dien eta bere usteak hitzez, idatziz eta beste hainbat eratara adierazten dituen.

Munduaren zati hori datu-baseetan gordetzean, datu bihurtzen da informazioa. Datu horien hainbat ikuspegi sor daitezke eta erabiltzaile bakoitzarentzat nolabaiteko mundu subjektiboa eraiki.

1.1.2. Datu-baseak vs DBKS

Datu-basea deitzen zaio denbora-tarte batean existitzen den datu-bilduma egituratuari (adibidez, bideoklub bateko bazkideen datuak: izena, helbidea, telefonoa, NA zenbakia...). Egungo datu-baseak ordenagailuei loturik daude eta haien kudeaketa guztiz automatizatuta dago, software berezien bidez. Software hori osatzen duten aplikazioek **Datu-baseak Kudeatzeko Sistema** (DBKS) izenez ezagutzen dena sortzen dute. Datu-baseak kudeatzeko sistema hori datu-base fisikoaren eta datu-base horren erabiltzaileen artean dagoen softwarea edo programen bilduma da. Erabiltzaileek datu-basea atzitzeko, bertako datuak irakurri, aldatu edo ezabatzeko erabiliko dute bilduma hori.

Teorian, datu-basea eta DBKS-a kontzeptu ezberdinak direnez, batetik software-etxe bateko datu-basea eta, bestetik, beste software-etxe bateko DBKS-a izan ditzakegu. Baina hori, normalean, ez da horrela izaten. Etxe bakoitzaren DBKS-a eta datu-basea ezin apur daitekeen bikotea da. Datuak formatu ezberdinetan gordetzen dituzte eta, ondorioz, eurek bakarrik dakite nola erabili. Erabiltzaileek estandarizatutako zenbait lengoaiak erabili ditzakete (SQL da ezagunena) datuekin lan egiteko eta lengoaiak horiek erabiliz egindako galderak DBKS bakoitzak nahi dituen bezala kudeatuko ditu, betiere gutxiengo ezaugarri batzuk betez. Horregatik, nahiz eta DBKS bakoitzak datuak modu ezberdinean erabili, erabiltzailea ez da ohartzen, berak DBKS-arekin soilik dituelako harremanak eta, horretarako, SQL lengoaiak estandarizatuarekin nahikoa duelako. Beraz, erabiltzailearen ikuspuntutik berdina dio zer DBKS/datu-base bikote erabili, komunikatzeko beti SQL erabiliko baitu (ikus 1.1 irudia).



1.1 irudia: Datu-basea kudeatzeko sistemaren kokapena, erabiltzaileen eta datu-basearen artean.

Horrekin guztiarekin lasai esan daiteke DBKSA datu-basearekin lan egitea ahalbidetzen duen programa-bilduma dela.

Esate baterako, demagun A armairua D dokumentuz beteta dagoela. Dokumenturen bat atzitu nahi denean ezin da armairua zuzenean ireki, horretarako kudeatzaile-lanak egiten dabilen K langilea baitago. D1 dokumentua nahi denean Kri eskatuko zaio; horrek protokolo jakin bati jarraituko dio eta, lehenik, eskaria nork egin duen ziurtatuko du. Ondoren, armairua irekitzeko eta K1 dokumentua hartzeko baimenik duen ala ez begiratuko du... Hori guzti hori betetzen bada, armairutik dokumentua hartu eta erabiltzaileari emango dio. Bitartean, erabiltzailea ez da ohartu K-k egin dituen eragiketez, ez daki eskatutako dokumentua osoa den ala beste baten zatia... D eta A datu-basea izango dira. K-k eta berorren zereginek osatzen duten multzoa, berriz, DBKS-a.

DBKS-a software edo programa-bilduma bat da, baina ez da edozein software. DBKS-tzat har dadin, software horrek betebeharrak izango ditu eta, ondorioz, DBKS-aren helburuak betekiko dira. Batetik, datu-base/DBKS bikotea datuei zuzendutako sistema denez, datuen independentzia eta informazioaren abstrakzioa bilatzen du. Bestetik, atzitzeko beharizanaren arabera ikusi/erabiliko dituzte hango datuak aplikazioek, datuen egituraz, gordetzeko eraz eta beste hainbat zereginenez batera arduratu gabe. Aplikazio batek eremuak edo errenkadak gehitzen/kentzen baditu, besteak ez du eragin okerrik nabaritutako, horixe baita DBKS-aren lanetako bat. Hori bai, osotasuna (datuen zuzentasuna) bermatu beharko da eta, gainera, posible izango da bi aplikazioak datu bereberekin batera egikaritzea (konkurrentzia). Arazo horiek era egokian zuzendu beharko dira, datu-basea egoera sendoan egon dadin, datuek informazio koherentea uneoro eman dezaten.

Adibidez, kontzeptuz, datu-baseetan ez dago erredundantziarik (datuek kopia soil bat dute), baina, azken finean, kasu jakin batzuetan erredundantzia minimoa onartzen da, sistema azkarragoa izan dadin. Zer gertatzen da datuaren kopietako bat aldatzen denean? DBKS-ak horretaz ohartu beharko du eta, sendotasuna mantentzeko, gainerako kopiak ere eraldatu egin beharko ditu. Ez da erabiltzailearen esku geratuko, erabiltzaileak ez baitu jakingo kopiarik dagoen ala ez; datu-basearen diseinatzailearen erabakia izango da hori.

DBKS-ak segurtasuna ere bermatu egin behar du; alde batetik, erabiltzaileen kudeaketaz eta atzipenatarako izango dituzten baimenez arduratuko da, eta bestetik, datuen segurtasunaz: datu-basearen segurtasun-kopiak egiteko tresnak izango ditu, istripuak gertatzen direnean datu-basea egoera sendoan uzteko gai izan beharko du... Baina horrek guztiorrek beste ezaugarri bat azaleratzen

du: berreskurapena. Datu-basea azken egoera sendora ekartzeko tresnez hornituta egon beharko du DBKS-ak.

Datu-baseek eta, jakina, DBKS-ek bilakaera sendoa izan dute 60ko hamarkadan aurkeztu zirenetik gaur arte. Hasieran, eredu hierarkikoa zen nagusi; geroago, sare ereduak agertu zen, eta azkenik, gaur egun merkatuan nagusi den eredu erlazionala. Egundak, ikerketak aurrera darraiela, datu-base banatuak, *Data Mining* teknikak, objektuei zuzendutako datu-baseak... egundak badaude, baina oraindik merkatuaren gutxiengoa baino ez dira. Hori guztiori dela eta, liburu honetan batez ere eredu erlazionalaz arituko gara. Beste ereduak eta etorkizuneko teknikak ere jorratuko ditugu, baina maila apalagoan. Ekin diezaiozun datu-baseen ibilbide horri.

1.2. EREDU HIERARKIKOA

Gizakia Ilargira heltzeko Apollo proiektuan datu-baseen sistemen hasiera. Garai haietan ez omen zegoen proiektuak behar zuten informazio-kopuru erraldoia kudeatzeko sistemarik. Proiektuan lanean ari zen lehenengo enpresak, NAA hain zuzen, GUAM izeneko softwarea garatu zuten. Geroago, IBMk eta NAAk, GUAMetik abiatuta, gaur egungo IMS izenez ezagutzen den produktua garatu zuten. Software horren funtsezko ideia datuak zuhaitz-egiturari jarraituz antolatzea zen. **Sistema hierarkikoa** esaten zaio horrelako antolaketa edo egitura duen datu-baseen sistemari.

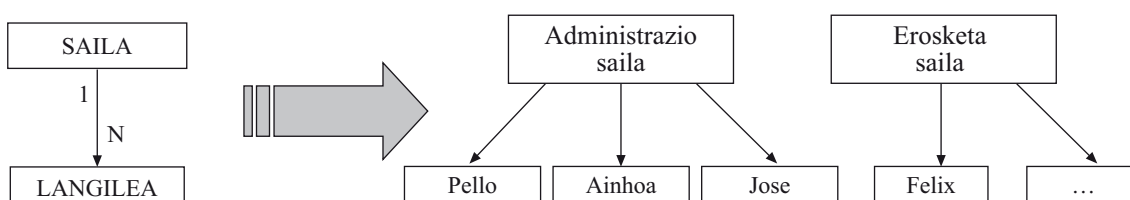
Eredu hierarkikoa ez zen arrazionalizazioaren eta estandarizazioaren ondorio izan, IBMk 1960ko hamarkadaren hasieran erabilitako IMS datu-base hierarkikoaren hedapenaren ondorio baizik. Horrela, IMS gainerako datu-base hierarkikoentzat estandar bilakatu zen, berorren egitura eta ezaugarriak hartu baitziren oinarritzat.

Datu-baseen lehen ereduak izan zen. Taulak eta erlazioak dira datu-base horien osagarriak. Taulek errenkadetan pilatzen dituzte datuak. Taula bateko errenkada guztiek atributu berberak izango dituzte. Adibidez, taula-mota bat IDAZLE izan daiteke, honako atributu hauekin: ‘Izena’, ‘Abizena’ eta ‘Nazionalitatea’. Edo LIBURUA, atributu hauekin: ‘Izenburua’, ‘ISBN’, ‘Generoa’ eta ‘Argitaletxea’. IDAZLE taulako errenkada bi izan litezke; esate baterako, *José—Saramago—Portugaldarra* eta *Bernardo—Atxaga—Euskalduna* (ikus 1.1 taula).

IZENA	ABIZENA	NAZIONALITATEA
<i>José</i>	<i>Saramago</i>	<i>Portugaldarra</i>
<i>Bernardo</i>	<i>Atxaga</i>	<i>Euskalduna</i>
...

1.4 taula. IDAZLEA taula eta bertako agerraldiak.

Taulek elkarren artean dituzten loturak zehazteko erabiltzen dira erlazioak. Eta hortxe agertzen da ereduaren ahulunea. Berez onartzen dituen erlazioak hierarkikoak baino ez dira; hau da, aita-seme erlazioak (ikus 1.2 irudia).



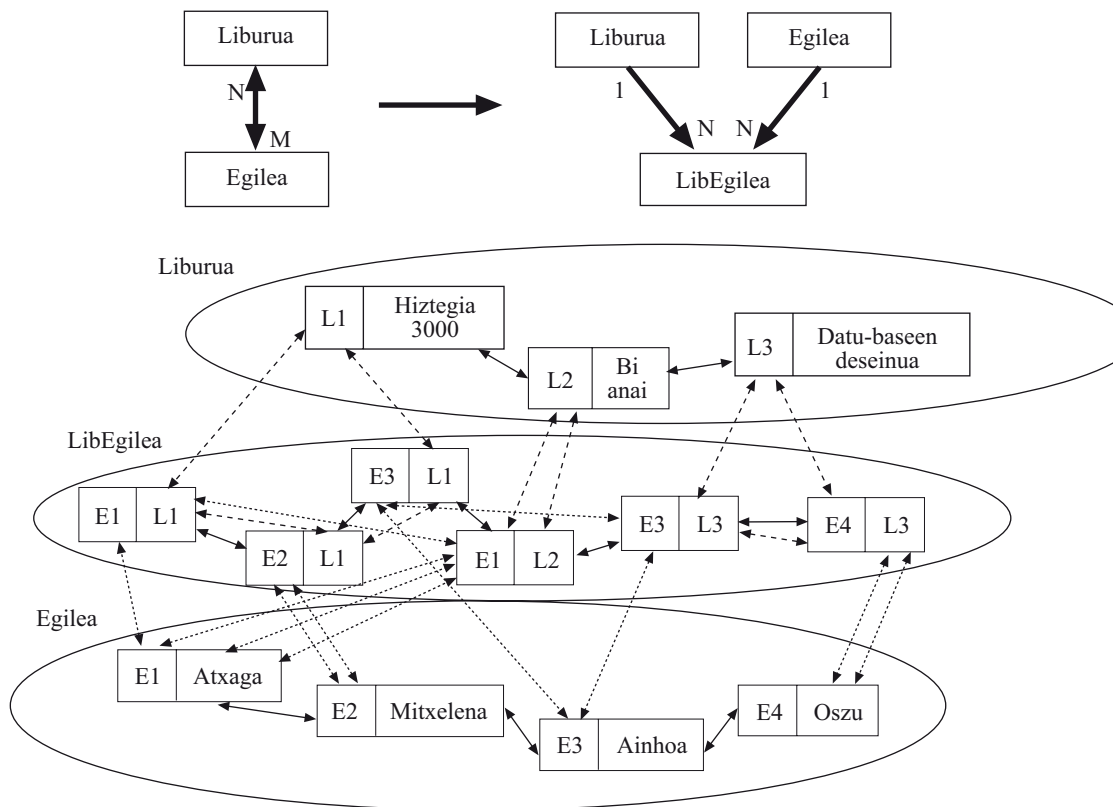
1.2 irudia: Datu-base hierarkiko baten egitura eta bertako agerraldiak, erabiltzaileak ikusiko dituen bezala.

Arazoak, esan bezala, taulen arteko erlazioak konplexuagoak direnean sortzen dira. Adibidez: nola adierazi idazle batek liburuak idazten dituela baina liburuak sarritan idazle batek baino gehiagok idatziak direla? Badaude moduak, baina konplexuagoak dira. Arazo horiek konpontzera etorri ziren sare eredu jarraitzen duten datu-baseak.

1.3. SARE EREDUA

Hirurogeiko hamarkadaren erdi aldera, IDS sortu zen, General *Electric*ek garatuta, Charles Bachmann-en zuzendaritzapean. Datu-baseetarako sistema-mota berria zen, sare-sistema deiturikoa. Ez da nahastu behar datu-basea komunikazio-sare batean banatuta egotearekin. Sistema hierarkikoen bidez islatzeko zailak ziren datu arteko erlazioak biltzeko asmatu zen sistema berri hori. Gainera, datu-baseen estandarra sortu nahi izan zen eta, horretarako, AEBko gobernuak, CODASYL eta enpresa garrantzitsu batzuk bildu zituen DBTG sortuz. DBTGren helburua datu-baseen definizio eta manipulazio estandarra sortzea zen. DBTGk azken txostena 1971n aurkeztu zuen eta, ANSI erakundeak onartu ez arren, estandar modura funtzionatzen hasi zen.

Aurreko puntuko adibideari eutsiz (idazleak eta liburuak), azaldutako teknikak modu logikoan islatzen ditu erlazioak (ikusi 1.3 irudia).



1.3 irudia. Sare ereduko datu-base baten egitura eta agerraldiak. Hiru liburu daude (*Hiztegia*, *Bi anai* eta *Datu-baseen diseinua*). Kolore urdina duten esteei jarraituz gero, *Hiztegia* idatzi dutenak izango ditugu. Kolore berdeko estekak *Bi anai* liburuko egileak izango ditu, eta kolore gorriko estekak, *Datu-baseen diseinua* izeneko liburukoak. Kolore ezberdinez baina puntukako marrekin autore bakoitzaren liburuak jarraitu ahal izango dira. Kolorezko baina etengabeko marrek taula ezberdinetako errenkadak lotzen dituzte: LIBURUA, LIB-EGILEA eta EGILEA.

1.3 irudian ikusten da eredu honi izena nondik datorrikon: pilaturiko datu-errenkadek beraien artean erlazionatzeko erabiltzen dituzten esteekek sare itxura hartzen dutelako.

Sistema hierarkikoak eta sare-sistemak desabantaila handia dute: tauletako errenkadak banaka-banaka erabili behar dira; ezin dira multzo modura erabili. Ondorioz, erabiltzaile-programatzailearen lana izango da errenkada guztiak banan-banan prozesatzea.

1.4. EREDU ERLAZIONALA

1970. urtean, Codd-ek, IBMko ikerkuntza sailean ari zelarik, eredu erlazionala aurkeztu eta, aldi berean, aurreko eredu desabantailak agertu zituen. Hasiera batean, ez zuen arrakastarik izan, baina, berak esan bezala, beste eredu desabantailak agertzen hasi zirenean eredu erlazionala hasi zen nagusitzen. Ordutik, sistema erlazional asko sortu izan dira. Lehenengoetarikoa IBMren *System R* izan zen, eredu erlazionalaren eraginkortasuna frogatzeko garatutakoa. Horren ekarpen nagusiak bi izan ziren: batetik, galdeketak egiteko SQL lengoaia, eredu erlazionalen estandar bilakatu zena, eta bestetik, datuak multzoka erabiltzeko erraztasunak.

Artean, taulak eta erlazioak ziren eredu osagaiak. Baina eredu erlazionalan erlazioak baino ez daude (2. gailan sakonduko dugu).

Egun, ugari dira DBKS erlazionalak. Hona hemen horietako batzuk: DB2, SQL/DS, ORACLE, INGRES, Informix, Sybase, PostgreSQL, Paradox, dBase IV, Access, SQL SERVER, FOXPRO, MySQL...

DBKS-en eredu erlazionalak bigarren belaunaldia ere izan zuen, sistema erlazionalak ere hobetu egin baitziren. Hasieran, datuak modelatzeko malgutasuna oso urria zen. 1976. urtean Chen-ek Entitate/Erlazio eredu proposatu zuen (E/R), diseinurako prozesu gehienetan erabiltzen dena. Eredu horren garrantzia nabarmena denez, kapitulu osoa eskainiko diogu liburu honetan, laugarrena hain zuzen ere. Errealitateko mundua era egokiagoan islatzeko eginiko ahaleginek hobekuntza aipagarriak ekarri zituzten.

Datu-baseak erabiltzen dituzten aplikazioen zailtasuna gero eta handiago izanik, eredu berriak sortu dira eta datu-baseen sistemen belaunaldi berria eratu dute. Euron artean, eredu erlazionalaren zabalkuntza baino ez diren datu-base banatuak eta objektuei zuzendutako datu-baseak daude.

1.5. DATU-BASE BANATUAK

Deszentralizazioa eta komunikazio-sareen arrakastaren ondorio da eredu hau. Eredu honetan, sarearen bidez konektatutako ordenagailuetan gordetzen dira datuak. Pentsa dezagun IKEA moduko multinazional bateko datu-basean: delegazioak mundu zabaleko hainbat lekutan egongo dira. Datuak, normalean, gehien erabiltzen diren delegazioan gordeko dira. Datu horiek delegazioak berak eguneratuko ditu. Izan ere, egungo teknologiaekin, ez du logikarik Madrilgo bezeroen datuak Bartzelonan gordetzeak.

Hainbat ordenagailutan DBKS bakarrak eragiten duenean, datu-baseen sistema banatua daukagu (DBKS berbera hainbat delegaziotan edo nodotan). Baina DBKS-ak ezberdinak badira (delegazio bakoitzean bertako DBKS-a erabiltzen da, nahiz eta batera lan egin) gehiago edo gutxiago teilkatuko diren DBKS askoren kasua da.

Datu-basea sare bitartez erabiltzaile askok atzitzen badute, baina datuak kokagune bakar batean badaude, hori ez da datu-base banatua. Horretarako, datuek kokagune ezberdinetan egon behar dute.

1.6. OBJEKTUEI ZUZENDUTAKO DATU-BASEAK

Laurogeiko hamarkadan, objektuei zuzendutako lengoaien agerraldiarekin batera, filosofia hori datu-baseen sistemetara eramatea pentsatu zen. Lehenengo kasua *Gemstone* prototipoarena izan zen. Oso famatua izan zen *Objectstore* sistema ere. Laurogeita hamarreko hamarkadaren hasieran merkatura plazaratzen hasi ziren objektuei zuzendutako datu-baseen sistemak.

Eredu horretan, informazioa errenkada batean gorde beharrean, objektu iraunkor baten bidez pilatzen da. Horri esker, diskoan gordetzeko orduan etekin hobea lortzen da. Galdeketak ere errazagoak dira. Gainera, objektuei zuzenduriko programazio-lengoaien ezaugarriak etekina ateratzen diote; adierazpenerako ahalmena, esate baterako.

Hori dela eta, logikoena izango litzateke azken eredu horrek eredu erlazionala ordeztuko duela pentsatzea, baina ez da horrela gertatu, eta gaur egun, objektuei zuzendutako DBKSak erlazionalen aldean gutxi dira oraindik. Hainbat arrazoi dago. Nahiz eta erakundeek zenbait aplikaziotarako objektuei zuzenduriko lengoaiak erabili, datu-baseekin lan egiteko ez dute erabili nahi, oraindik guztiz heldu izan ez den teknologia balitz bezala. Orduan, ondo egokitutako eredu, zertarako aldatu?

1.7. DATU-BASEAK, ZERGATIK?

Fitxategi-sistemekin konparatuz, datu-baseen sistemek abantaila asko eskaintzen dituzte:

- **Datuen sendotasuna eta erredundantziaren kontrola:** fitxategi-sistemetan datu bera hainbat fitxategitan errepikaturik egon daiteke, eta horrek lekua alferrik galtzea eta datuen sendotasunik eza eragiteko arriskua dakar. Printzipioz, datu-baseen sistemetan fitxategi guztiak daude integratuta, eta horregatik, ez dago datuak errepikatzerik. Dena den, alde aurretik aipatu den bezala, batzuetan, datuen arteko erlazioak errazago irudikatzeko edo prestakuntzak hobetzeko, erredundantzia minimoa mantentzen da. Hori bai, datua behin baino gordez ez bada, behin baino ez da eguneratu behar izango, eta automatikoki eskuragarri egongo da erabiltzaile guzti-entzat. Datua errepikatuta badago, baina sistemak hori jakin badaki, sistemak berak eguneratuko ditu beste kopia guztiak.
- **Segurtasuna eta datuak konpartitzea:** fitxategi-sistemetan fitxategia erabiltzen duena da fitxategiaren jabea. Datu-baseetan datuak erabiltzaile guztien eskura daude, baina erabiltzaile bakoitzak bere baimenen arabera atzitu ditu. Gainera, sorturiko aplikazio berriek dagoeneko existitzen diren datuak erabil ditzakete, eta segurtasun-kopien eta berreskuratze-zerbitzuen hobekuntza ere badago, fitxategi-sistemetan ez bezala. Azken horietan erabiltzailearen ardura da segurtasun-kopiak egitea. DBKS-ek, berriz, minimora gutxitzen dute alferrik galdutako lana. Eta segurtasun-kopiak egiteko tresnak/erraztasunak eskaintzen dituzte.
- Datuak integratuta dauden heinean errazagoa da estandarrak betetzea datuen formatuan, dokumentazioan, atzipen-arauetan eta eguneratze-prozeduretan. Gainera, datuen independentziari esker, mantentze-lana errazago egiten da. Fitxategi-sistemetan, fitxategien egitura programetan deskribatzen denez, datuen egitura edo datuak pilatzeko eran aldaketak gertatzen direnean, programak aldatu egin behar dira. Baina DBKS-etan ez da halakorik gertatzen, datuen eta aplikazioen deskripzioak banaturik daudelako. Banaketa horri **independentzia** deritza eta bi mailatan gautzen da: batetik, independentzia fisikoak bermatzen du datuak biltegitatuta dauden erak egitura logikoan eraginik ez izatea. Hau da, nahiz eta biltegitatze fisikoan aldaketak egon, datu horietara heldu behar duen erabiltzaileak ez du programa aldatu behar izango. Eta bestetik, independentzia

logikoa dago: datu-baseari elementuak gehitzeak, kentzeak edo aldatzeak datu-basea kudeatzen duten programetan eraginik ez izatea ahalbidetzen du.

- Datuen atzipen hoberako, DBKS-ek **galdeketak egiteko lengoaiak** eskaintzen dituzte. Lengoaiak horiek erabiliz gero, aplikazioak programatzea ez da beharrezkoa. Ondorioz, produktibitatea hobea da. Horrez gain, DBKS askok laugarren belaunaldiko inguruneak dituzte, edo beraiekin bat egiten dute eta datuak atzitzen dituzten aplikazioen garapena asko errazten dute.
- DBKS-ek datu berberei aldi berean atzipen ugari egitea baimentzen dute; hots, konkurrentziarik badago, bera arduratuko da dena kontrolatzeaz, datuen osotasuna eta sendotasuna uneoro zaintzeko.

Desabantailak ere badituzte, ordea:

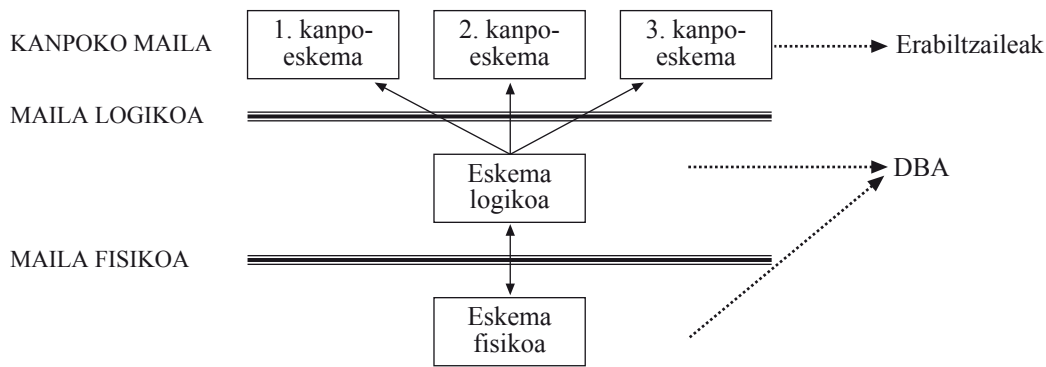
- Zailtasuna eta tamaina: DBKS-ak programa-multzo handiak eta konplexuak dira eta, etekin ona lortzeko, ondo ulertu behar dira. Horrez gain, programa eta datu-multzo handi eta konplexu horrek leku asko behar du, bai diskoan, bai memorian.
- Arlo ekonomikoa: DBKS/datu-base bikoteak ematen dituen abantailak ordaindu egin behar dira. Batzuetan fitxategi-sistema datu-baseen sistema batekin ordeztzea oso garestia da; aplikazioa aldatzeaz gain datu-baseen sistema ezarri behar da, enpresakoei erabilera irakatsi behar zaie... Gainera, ordenagailuek baliabide egokidunak eta pilatzeko espazio handikoak izan behar dute, DBKS-ak ondo funtziona dezan. Sarri DBKS/datu-baseak berak bakarrik ordenagailu osoa behar izaten du.
- Prestazioak: normalean fitxategi-sistemak aplikazio jakin baterako sortzen dira eta, beraz, oso prestazio onak eskaintzen dituzte. DBKS-ak, berriz, aplikazio ugari erabiltzen dituzte eta, horregatik, aplikazio batzuk lehen baino astiroago joan daitezke.

Dena den, abantailak gehiago dira; hala ez balitz, datu-baseak egun dauden tokira ez ziren inoiz heldu izango.

1.8. DATU-BASEEN SISTEMEN EZAUGARRIAK

DBKSen historiaz hitz egiterakoan, datuetara zuzendutako sistemak direla esan dugu. DBKSen lehenengo helburua erabiltzaileei informazioaren ikuspegi abstraktua ematea da. Horrela, erabiltzaileak ez du jakin behar datuak nola biltegituta dauden edo mantentze-lana nola egin. Hori posible izateko, datu-basea hiru abstrakzio-mailatan banatzen da. Abstrakzio-maila bakoitzean datuen deskripzioa egiteko, eskemak erabiltzen dira.

- **Maila fisikoa:** datuen biltegitatze fisikoaren deskripzioarekin du zerikusia. Maila honetan eskema fisikoa definitzen da.
- **Maila kontzeptuala:** datuen antolaketa logikoari lotuta dago. Maila honetan eskema kontzeptuala definitzen da.
- **Kanpoko mailak:** datuen inguruan erabiltzaileek dituzten ikuspegiaren deskripzioekin du zerikusia. Erabiltzaileengandik hurbilen dagoen maila da. Maila honetan kanpoko eskemak definitzen dira. Kanpoko eskema bat baino gehiago egon daiteke, bakoitza datu-base osoaren zati baten adierazpen abstraktua izanik. Horrela, erabiltzaile bakoitzari, beraren beharrezko arabera dagoen informazioa erakutsiko zaio. Horrelako informazio-multzo bakoitzari **ikuspegi** esaten zaio (ikus 1.4 irudia).



1.4 irudia. Datu-baseen abstrakzio-mailak eta eskemak.

DBKSak, datuak definitu eta kudeatu ahal izateko, ondoren aurkezten diren bi lengoaiak erabiltzen ditu:

- Datuak Definitzeko Lengoaia (DDL): datu-basea osatzen duten elementuak definitzeko erabiliko da. DDLa erabiliz egindako definizioak, murrizketak, osotasun-arauak, datuen egiturak... datu-hiztegia gordetzen dira.
- Datuak Manipulatzeko Lengoaia (DML): datu-basea definitu ostean, bertako datuekin lan egin ahal izateko, DBKSak ulertuko duen eragiketa-multzoa behar da. Horixe da DMLa.

Eredu erlazionalean, DML eta DDL sententziak, denak egiten dira SQL lengoaia erabilita (ikus 5. gaia).

Datu-baseen sistema baten osagaiak aipatzen badira, bi bereizten dira bereziki: DBKSa eta datuak. Erabiltzaileak datu-basea atzitzeko, bertako datuak irakurri, aldatu edo ezabatzeko egiten dituzten eskaerak DBKSak onartu eta bideratzen ditu. Horretarako, hainbat eragiketa-mota eskaintzen ditu. DBKSak hardware mailako xehetasunetatik aldentzen ditu datu-basearen erabiltzaileak, programazio-lengoaiekin programatzaileekin egiten dutenaren antzera.

DBKSaren erabiltzailea DBA motakoa edo erabiltzaile arrunta izan daiteke. DBA datu-baseko administraria da eta baimen guztiak dauzka. Beste muturrean erabiltzaileak daude. Erabiltzaileok DBAk ematen dizkien baimenak baino ez dituzte izango.

1.9. ONDORIOAK

Informatikaren historian fitxategiek duten tokia ez da alde batera uztekoa, baina gaur egun datu-baseen sistemak gero eta ugariagoak dira edonon. Aplikazioak garatzen mantentze-lana askoz hobea da, datuen egitura zerbait aldatzen denean aplikazioak ez baitira ukitu behar. Horregatik, erakundearen erabaki estrategiko baten ondorioz datuen egitura aldatu beharra badago, aplikazioek bere horretan martxan jarraituko dute. Aldaketak egiteaz DBA arduratuko da eta erabiltzaileak ez dira konturatuko. Erabiltzaileek, DML eta DDL sententziak erabiliz, berdin-berdin atzitu dute datu-basea DBKSari esker.

Datu-basean datuak eta beraien deskripzioak gordeko dira, eta DBKSaren ardura izango da beraien kudeaketa egokia egitea, osotasuna eta sendotasuna zainduz.

*Datuen eredu
erlazionala*

2

2.1. SARRERA

Datu-baseak kudeatzeko sistema hierarkikoak eta sare ereduak zituzten eragozpenak ikusita, 1970ean Codd-ek, IBMko ikerkuntza sailetik, eredu erlazionala aurkeztu zuen. Datu-baseen egitura sinplifikatzeko aurkeztu zuen eredu berri hori.

Codden dokumentuak erlazioen teorian oinarritutako eredia proposatzen du, non datuak erlazio modura egituratzen diren. Erlazio horien adierazpen grafikoa taulen bidez egiten denez, zenbait argitalpenetan “taula” ere esaten zaie.

1970ean datuen kudeaketarako Coddek proposatutako ereduak honako helburu hauek bete nahi zituen:

- Independentzia fisikoa: datu-basearen egitura datuak biltegitatuta dauden erak ez dezala egitura logikoan eraginik izan; hau da, erabiltzaileak ez dezala biltegitatze fisikoari buruz ezer jakin behar izan.
- Independentzia logikoa: datu-baseari elementuak gehitzeak, kentzeak edo aldatzeak (eremuak gehitu, kendu...) eraginik ez izatea, programa edo eta datu-basearen azpimultzoa (ikuspegia) erabiltzen dabilen erabiltzailearengan; hau da, ez dezala eraginik izan erabiltzaileak datu-basearekiko duen ikuspegian.
- Malgutasuna: erabiltzaile bakoitzak, berak nahi duen eran datuak aurkezteko gaitasuna izatea; hots, hainbat ikuspegi izatea.
- Uniformetasuna: datuak era uniformearen aurkeztea, egitura logikoa erabiliz (erlazioak).
- Sinpletasuna: guztiz ulerkorra eta erraz erabiltzeko modukoa izatea.

Aipatutako helburu horiek lortzeko, Coddek **erlazio** kontzeptua aurkeztu zuen. Datu-baseak kudeatzeko sistemako datu guztiak erlazioetan aurkezten dira eta erlazio baten edukia aldatu egingo da denbora aurrera doan heinean.

“Erlazio” eta “taula” kontzeptuak oso antzekoak izan arren, ez dira gauza bera. Askotan, nahastu egiten dira, erlazioak irudikatzeko taulak erabiltzen direlako. Baina erlazio batean ezin dira bi errenkada berdin egon; tauletan bai, ordea. Gainera, erlazioetan, errenkaden ordenak ez du garrantzirik. Eta amaitzeko, erlazio batean zutabe eta errenkaden arteko gurutzaketetan balio bakarra egon daiteke (balio atomikoa, 4. gaian, normalizazioaren arloan sakonduko da ezaugarri hori).

Itxuraz, erlazioa errenkada-kopurua da. Terminologia erlazionalan, gainera, aljebra erlazionalako eragileak erabiltzen dira erlazioekin.

1985. urtean Coddek datu-base erlazional guztiek bete beharreko 12 arau argitaratu zituen. Haien bidez definitzen da datu-base erlazional ideala. Aldi berean, datuen eredu erlazionalaren diseinurako pausoak finkatzeko lagungarri izan dira arau horiek. Datu-base erlazionalen 12 arauak ondoren azaltzen dira:

1. Informazioaren araua: datu-baseak kudeatzeko sistema erlazioetan, informazio guztiak era logikoan egituratuta egon behar du erlazioetan. Adibide gisa, imajina dezagun bideo-denda bateko bezeroen, filmen eta alokairuen informazioa islatzen duen datu-basea. Eredu erlazionalan honela egituratuta agertuko litzateke informazioa (ikusi 2.1, 2.2 eta 2.3 taulak).

NifBez	Izena	Abizena	Helbidea
1111111L	Amaia	Gómez	Artekale 3, Irun
2222222J	Aitor	Agirre	Guen kalea, Oiartzun
3333333T	Josune	Astorena	Telleria 5, Bilbao

2.1. taula. BEZEROA erlazioa.

ZenbAlok	FilmKodea	Data	NifBez
18	22	02/1/1	1111111L
14	22	03/5/28	2222222J
22	25	04/6/22	2222222J
42	22	05/6/2	3333333T

2.2. taula. ALOKAIRUA erlazioa.

FilmKodea	Izenburua	Kopurua	Prezioa
22	Mar adentro	2	1.5
25	Ninja dortokak	6	2
43	Seven	9	

2.3 taula. FILMA erlazioa.

- Ziurtatutako atzipena: datu-base erlazionalako datu bakoitza, era logikoan, erlazioaren izena, gakoa eta zutabearen izena jakinda lortu ahal izango da. Demagun 'FilmKodea' eremuan (gako nagusia) 22 balioa duen filmaren prezioa jakin nahi dela. Horretarako, FILMA erlazioa joan behar da, bertan 22 kodea duen errenkada aurkitu eta 'Prezioa' zutabetik 1.5 balioa atera. Betiere errenkada bakar bat egongo da kode horrekin.
- Balio hutsen tratamendu sistematikoa: balio hutsak, informaziorik eza adierazteko erabiltzen dira. Adibide gisa jo dezagun film berri bat gehitu zaiola FILMA erlazioari eta 'Kodea' 43 dela, 'Izena' *Seven*, 'Kopurua' 9, baina 'Prezioa' oraindik finkatu gabe geratu dela. Orduan, 'Prezioa' eremuan film horrek balio hutsa duela esaten da.
- Katalogo dinamikoa: datu-basearen berezko egituraren deskripzioa datu-baseko informazioa denez, erlazioen bitartez adierazten da. Horrela, erabiltzaileak gainerako datuei aplikatzen zaien lengoaia erlazional bera erabiliko du datu-baseari buruzko informazioa atzitzeko. Adibide gisa, datu-baseak kudeatzeko sistema erlazional bateko hiztegia har daiteke (ikusi 2.4 taula).

ERLAZIOAREN EDOTA IKUSPEGIAREN IZENA	EDUKIA
CHECK_CONSTRAINTS	CHECK motako murrizketak
COLUMNS	uneko erabiltzaileak atzitu ahal dituen zutabeak
TABLES	uneko erabiltzaileak atzitu ahal dituen taulak
TABLE_CONSTRAINTS	uneko erabiltzaileak kudeatu ahal dituen taulen gainean definitutako murrizketak
TABLE_PRIVILEGES	uneko erabiltzaileak kudeatu ahal dituen taulen gainean definitutako baimenak (txertatu, ezabatu...)

2.4 taula: Datu-base erlazional bateko hiztegi baten edukiaren adibidea.

5. Azpilengoaia oso baten araua: ondoren aipatutako puntu guztiak definitzeko gai den azpilengoiaren bat baldin badago, azpilengoaia osoa dela esaten da:
 - datuen definizioa
 - ikuspegien definizioa
 - datuen manipulazioa
 - murrizketen definizioa
 - baimenak
 - trukaketen mugak
6. Ikuspegiak eguneratzeko araua: datu-basea, erabiltzaile bakoitzari konbinazio logiko batzuk erabiliz aurkeztu ahal zaio, erabiltzaile bakoitzak dituen baimenen eta beharizanen arabera (adibidez, irakasle batek notak sartzerakoan bere ikasleak baino ez bistaratzea). Konbinazio logiko horiei **ikuspegi** esaten zaie. Ikuspegi bakoitzean datu-baseko erlazio batek jaso ditzakeen manipulazio-eragiketa guztiak egin daitezke, teorian. Praktikan, eguneratzeko eta ezabatzeko eragiketak ez dira ikuspegietan ematen (5. gaian landuko da sakonago).
7. Txertatzea, eguneratzea eta ezabatzea: datu-baseetako erlazioak, eragigai modura, berreskuratzeko prozesuetan ez ezik, txertatzeko, eguneratzeko eta ezabatzeko prozesuetan ere erabiltzen dira.
8. Datuen independentzia fisikoa: erabiltzaileak ez dauka zertan jakin informazioa fisikoki nola biltegitatuta dagoen.
9. Datuen independentzia logikoa: datu-baseari elementuak gehitzeak, kentzeak edo aldatzeak (eremuak gehitu, kendu...; hau da, egitura logikoaren aldaketak) ez du eraginik izan behar erabiltzailearen programetan. Beraz, ez luke eraginik izan behar erabiltzaileak datu-basearekiko duen ikuspegian.
10. Osotasunarekiko independentzia: aipatutako azpilengoaian definitzen dira murrizketak. Katalogoan gordetzen dira eta ez aplikazio-programetan. 2.4 taulan ikusten da, besteak beste, CHECK motako murrizketak datu-hiztegian gordetzen direla, CHECK_CONSTRAINTS delako erlazioan. Adibidez, eremu batean pertsonaren adina sartzerakoan, datu-hiztegian kontuan hartu bada datu horrek zenbaki positiboa izan behar duela, programek ez dute horretaz arduratu behar, nahiz eta batzuetan komeni izan.
11. Banaketaren independentzia: datu-baseak kudeatzeko sistema erlazional batek erabiltzailearen eta datuen kokapenaren arteko independentzia bermatu behar du, hau da, erabiltzaileak ez dauka zertan jakin erlazioa hainbat datu-basetan banatuta dagoen ala ez (9. gaian sakonduko dugu).
12. Iraulketarik ezaren araua: sistema erlazionalek behe-mailako lengoia baldin badaukate, lengoia hori ezin da erabili goi-mailako lengoian definitutako murrizketak iraultzeko.

2.2. DATU-BASE ERLAZIONALAREN EGITURA

Erlazioa da eredu erlazionalaren oinarritzko elementua. Taula gisa adierazten da. Erlazioan zutabe-kopuru jakina dago eta horietako bakoitzak **atributu** izena hartzen du. Atributuek erlazioaren ezaugarriak adierazten dituzte. Gainera, errenkada-kopuruari **tuplo** deritzo. Errenkada-kopuruak

erlazioaren **kardinaltasuna** adieraziko du eta zutabe-kopuruak, **gradua**. Horrek guztiak bi kontzeptu garrantzitsuren definiziora ekarri gaitu: eskema eta hedapena.

Erlazioak bi zatitan banatzen dira: batetik **eskema** edo goiburua dago, hots, erlazioaren egitura definitzen den tokia. Alde estatikoa da. Adibidez, IKASLEA (NA, Izena, Abizena). Bestetik, **hedapena** dago, hots, alde dinamikoa. Errenkada-kopuruak zehaztuko duenez, denboran zehar aldatuz joan daiteke. Adibidea 2.1 irudian agertzen da.

NA	Izena	Abizena
1111	Txomin	Aguirre
2222	Alazne	Lopez
44444	Agurtzane	Arriaga
55555	Aner	Fernandez

} ESKEMA (ESTATIKOA)
} HEDAPENA (DINAMIKOA)

2.1 irudia. Eskemaren eta hedapenaren adibidea.

Beste alde batetik, badira eredu erlazionalak onartzen ez dituen zenbait egitura, “erlazio” definitziorik eratorriak. Hona hemen eredu horren murrizketak:

- Ez daude bi errenkada berdin.
- Errenkaden ordenak ez du garrantzirik.
- Zutabeen ordenak ez du garrantzirik.
- Atributu bakoitzak dagokion domeinutik har ditzake balio posibleak. Adibidez, ‘Prezioa’ atributuaren domeinua zenbaki positiboek osatuko dute. Edo ‘Kolora’ atributua bagenu eta domeinua gorria, zuria, beltza eta urdina hitzek osatuko balute, balio bakarra hartuko luke; hau da, balio atomikoak baino ez dira onartuko. 2.2 irudian atributua ‘Prezioa’ izango litzateke, domeinua zenbaki positiboak, eta agerraldiak, 100 eta 250.

PiezaZenb	Kolora	Prezioa
1	Urdina	100
2	Gorria, Beltza	250

→ Ez dago onartuta

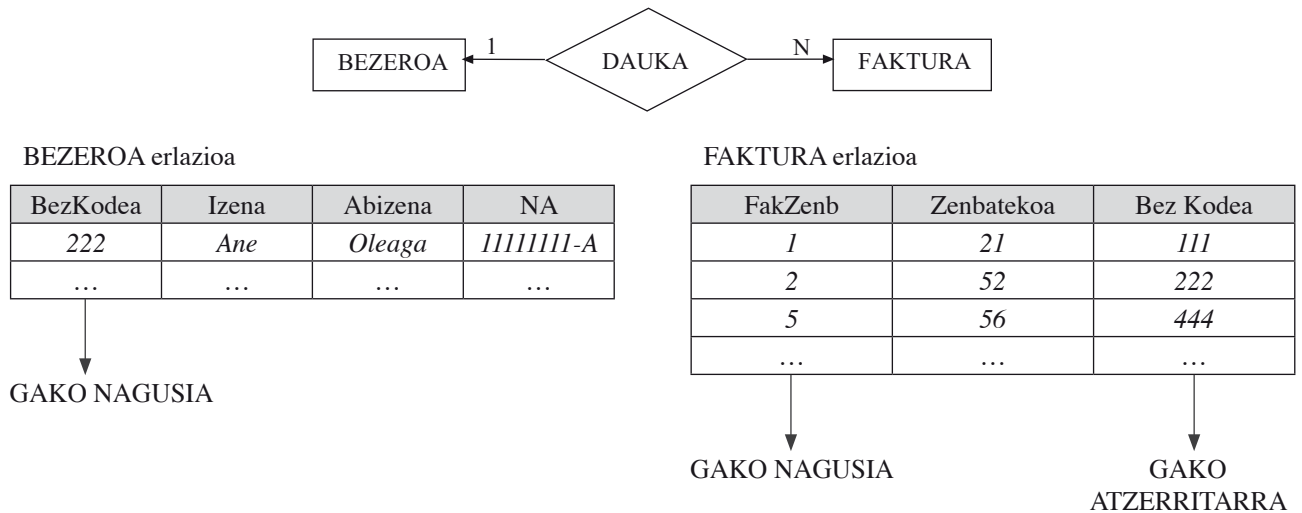
2.2 irudia. Balio atomikoen adibidea.

Atal honi bukaera emateko, **gako** kontzeptua azalduko dugu. Gakoari esker, eremuaren hainbat murrizketa betearazten dira; erlazio batean bi errenkada berdin ez agertzea, besteak beste.

Gakoa: datu-basearen diseinua egiterakoan, tuplo bakoitza era bakarrean definituko duen atributua edo atributu-multzoa da. Erlazio bakoitzak gako bat baino gehiago izan ditzake. Adibidez, ‘NA’ eta ‘BezKodea’ izan daitezke gakoak 2.3 irudiko BEZEROA erlazioan. Beraz, eremu honetako/haue-tako datuak ezin izango dira erlazioan errepikatu eta eremuok ezin izango dute balio hutsik izan.

Gako nagusia: gakoaren artean egokiena aukeratzen da gako nagusitzat. Gako nagusia osatzen duten eremuak azpimarratuta adierazi ohi dira adibideetan.

Gako atzerritarra: bi erlazio erlazionatzen direnean, lehenengo erlazioan gako nagusia osatzen duen atributu-multzoa bigarren erlazioan agertzen denean, gako atzerritar modura ezagutzen da. Gako nagusiarekin gertatzen ez den bezala, gako atzerritarretan balio hutsak onartzen dira (ikus 2.3 eta 2.4 irudiak, non gako atzerritarrak letra lodiz agertzen diren).



2.3 irudia. Gako nagusia eta gako atzerritarra eremuek osatzen duteneko adibidea.

2.3 irudiko BEZEROA erlazioan, gako nagusia 'BezKodea' atributuak osatzen du. Horregatik, ez dira bi bezero egongo 'BezKodea' berarekin, ezta 'BezKodea' gabeko bezerorik ere. Bi ezau-garri horiek bermatzen dute ez direla bi errenkada berdin egongo. Bestalde, FAKTURA erlazioko gako atzerritarra 'BezKodea' atributua izango da, atributu horrek erlazionatzen baititu BEZEROA eta FAKTURA erlazioak; hau da, 'BezKodea' atributuak esango du faktura bakoitza zein bezerori dagokion.

Baliteke, denbora iragan ahala, zenbait bezero datu-basetik ezabatzea. Ondorioz, beraiei dagoz-kien fakturak gorde edo ezabatu egin beharko dira. Gordetzekotan, gako atzerritarrak balio hutsak edukitzea baimendu beharko da, datu-basean dagoeneko ez dauden bezeroen fakturak gordeko baititugu. Ondorioz, 'BezKodea' atributuak balio hutsak hartuko ditu. Bigarren kasuan, oster, ez da hori gertatuko eta bezeroa ezabatzean bezero horri dagokion faktura guztiak ere ezabatu egingo zaizkio.

Zenbait kasutan, gako nagusia atributu batek baino gehiagok osatzen dute. 2.4 irudiko adibidean, PERTSONA erlazioko gako 'Izena', 'Abizena1' eta 'Abizena2' atributuek osatzen dute. Ondorioz, KOTXEA erlazioko gako atzerritarra ere atributu horiek guztiek osatuta dago.

2.4 adibidean garbi ikusten da gako nagusia atributu askok eratzen dutenean agertu ohi den beste arazo bat. Gako hori beste erlazio batean gako atzerritar modura erabiltzen denean arazoak ematen ditu, gako osoa (eremu guztiak) kopia behar da eta. Gainera, kontuan izan behar da, gero erlazioak azkarrago ibil daitezzen indizeak sortzen direnean, ez duela kostu bera eremu batekiko indizeak edo eremu askorekiko indizeak mantentzeak.

Hori konpontzeko, orokorrean lehenengo erlazioari (aitari, 2.4 irudiko adibidean PERTSONAri) beste eremu bat jarri ohi zaio, 'Kodea' motakoa (2.4 adibidean 'PertsonaKodea' izan liteke). Horixe izango da gako berria; tuplo bakoitzari dagokion zenbakia gordeko da gako horretan (lehenengoari 1, bigarrenari 2 eta horrela azken tuplorarte). Ondorioz, bigarren erlazioak (semeak) gako atzerritar modura atributu bakarra izango du ('PertsonaKodea') eta datu-basea arinago ibiliko da.



PERTSONA erlazioa

Izena	Abizena1	Abizena2	Helbidea	Herria
<i>Gorka</i>	<i>Lopez</i>	<i>Ayusti</i>	<i>San Ignazio</i>	<i>Bilbo</i>
<i>Maria</i>	<i>Quintana</i>	<i>Rodriguez</i>	<i>Goenkale</i>	<i>Bilbo</i>
<i>Patxi</i>	<i>Leaniz</i>	<i>Teiletxe</i>	<i>Zeharkale</i>	<i>Basauri</i>



KOTXEA erlazioa

Matricula	Marka	Erosi	Izena	Abizena1	Abizena2
<i>BI 7823 BB</i>	<i>VECTRA</i>	<i>1985</i>	<i>Gorka</i>	<i>Lopez</i>	<i>Ayusti</i>
<i>3434 BSK</i>	<i>ASTRA</i>	<i>2002</i>	<i>Gorka</i>	<i>Lopez</i>	<i>Ayusti</i>
<i>5125 DLT</i>	<i>GOLF</i>	<i>2006</i>	<i>Patxi</i>	<i>Leaniz</i>	<i>Teiletxe</i>



2.4 irudia. Gako nagusia eta gako atzerritarra eremu batek baino gehiagok osatzen duteneko adibidea

2.3. EREDU ERLAZIONALA ETA ALJEBRA ERLAZIONALA

Eredu erlazionalaren egitura eta egitura horrek bete beharreko murrizketak definitu ondoren, datu-basea osatuko duten datuekin egin daitekeen eragiketa-multzoa definitzea da hurrengo pausoa.

1970. urtean, eredu erlazionala definitu ostean, aljebra erlazionala osatzen zuten oinarrizko eragileak aurkeztu zituen Codd-ek. Eragile horiek multzoen teorian oinarrituta daude, hala nola bilketa, ebaketa, kenketa, biderkadura cartesiarra, proiektzioa, eta abarretan. Ondoren, JOIN moduko beste eragile eratorriak aurkeztu zituen.

Datuen eredu erlazionala eta dagokion aljebra eredu matematikoak dira. Benetako datu-base erlazionala sortzerakoan, ereduak finkatzen dituen erlazioak sortzeko eta kudeatzeko, lengoaiari erlazional baten premia dago. Aljebra erlazionalan oinarritutako lengoaiak DDL erlazionala eta DML erlazionala dira (ikus 5. gaia). Adibidez, SQL lengoaiari. Aljebra erlazionalerako eragile garrantzitsuenak aztertuko ditugu ondoren.

- **Bilketa** ($E = S \cup U$): S eta Uko tuploak bildu eta E erlazio berria sortuko da, non tuplo errepikatuz agertuko ez den. Bilketa egin ahal izateko, S eta U erlazioek atributu-kopuru berekoak izan behar dute eta Sko n-garren atributuak eta Uko n-garrenak domeinu bera izan behar dute (adibidez, 'Herria' atributuak ezin du erlazio batean erreala eta bestean zenbaki osoa izan).

Adibidez, demagun Ikasleak datu-basea IKASLE1 eta IKASLE2 erlazioez osatuta dagoela (ikus 2.5 eta 2.6 taulak. Bien bilketa 2.7 taulan agertzen da):

NA	Izena	Abizena	Herria
<i>21542155J</i>	<i>Ane</i>	<i>Gomez</i>	<i>Ea</i>
<i>33333333J</i>	<i>Alazne</i>	<i>Garate</i>	<i>Dima</i>
<i>22222222T</i>	<i>Eduarne</i>	<i>Telletxea</i>	<i>Getxo</i>

2.5 taula: Ikasleak datu-baseko IKASLE1 erlazioa.

NA	Izena	Abizena	Herria
<i>11111111M</i>	<i>Jon</i>	<i>Urrutia</i>	<i>Zalla</i>
<i>21542155J</i>	<i>Ane</i>	<i>Gomez</i>	<i>Ea</i>

2.6 taula: Ikasleak datu-baseko IKASLE2 erlazioa.

NA	Izena	Abizena	Herria
21542155J	Ane	Gomez	Ea
33333333J	Alazne	Garate	Dima
22222222T	Edurne	Telletxea	Getxo
11111111M	Jon	Urrutia	Zalla

2.7 taula: IKASLE3 = IKASLE1 U IKASLE2.

- **Kenketa** ($E = S - U$): S erlazioan badauden baina U erlazioan ez dauden tuploak bueltatuko ditu E erlazio berriak. Adibideari jarraituz, 2.8 taulan IKASLEBERRIA erlazioa sortuko litzateke.

NA	Izena	Abizena	Herria
11111111M	Jon	Urrutia	Zalla

2.8 taula: IKASLEBERRIA = IKASLE2 - IKASLE1.

- **Ebaketa** ($E = S \cap U$): E erlazio berria sortzen du, S eta U erlazio bietan dauden tuploekin. 2.9 taulan IKASLEBERDINA izena eman zaio.

NA	Izena	Abizena	Herria
21542155J	Ane	Gomez	Ea

2.9 taula: IKASLEBERDINA = IKASLE2 \cap IKASLE1.

- **Hautaketa** ($S = \sigma_{\langle \text{BALDINTZA} \rangle}(E)$): baldintza betetzen duten E erlazioko tuploak bueltatuko ditu. Konparazio eragileak \geq (handiagoa edo berdina), \leq (txikiagoa edo berdina), $<$ (txikiagoa), $=$ (berdina), $>$ (handiagoa), $<>$ (ezberdina) eta AND (eta), OR (edo) eta NOT (kontrakoa) eragile logikoak onartzen dira baldintzak idazterakoan. Jo dezagun LANGILEA erlazioa (2.10 taula) daukagula eta bertatik 1.500 euro baino gehiago irabazten duten langileak bi \leq direla. Erabili beharreko aljebra adierazpena $\sigma_{\text{Soldata}>1500}(\text{LANGILEA})$ litzateke eta emaitza 2.11 taulan agertzen da.

NA	Izena	Abizena	Soldata	DeptuZenb
11125252K	Jose Angel	Eguren	1200	3
23323565L	Ane Jone	Urrutia	950	5
78898562S	Txomin	López	1750	5
12345125H	Benito	Txurruka	2500	3

2.10 taula: LANGILEA.

NA	Izena	Abizena	Soldata	DeptuZenb
78898562S	Txomin	López	1750	5
12345125H	Benito	Txurruka	2500	3

2.11 taula: $S = \sigma_{\text{Soldata}>1500}(\text{LANGILEA})$ hautaketa egin ondoren lortutako emaitza.

Bestalde, 1.500 euro baino soldata handiagoa duten langileak eta 5 zenbakia duen departamentuan lan egiten dutenak hautatzeko, honako hiru adierazpen hauek erabil daitezke:

$$\sigma_{Soldata > 1500}(\sigma_{DeptuZenb=5}(LANGILEA))$$

$$\sigma_{DeptuZenb=5}(\sigma_{Soldata > 1500}(LANGILEA))$$

$$\sigma_{DeptuZenb=5 \text{ AND } Soldata > 1500}(LANGILEA)$$

Hiruren emaitza berbera izango da, 2.12 taulan adierazitakoa hain zuzen.

NA	Izena	Abizena	Soldata	DeptuZenb
78898562S	Txomin	López	1750	5

2.12 taula: $S = \sigma_{DeptuZenb=5}(\sigma_{Soldata > 1500}(LANGILEA))$ hautaketaren emaitza.

- **Proiektzioa** ($S = \Pi_{\langle \text{atrib1}, \text{atrib2}, \dots, \text{atribN} \rangle}(E)$): E erlazioaren ikuspegi bat bueltatuko du, atrib1, atrib2... atribN atributuez mugatua. Adibide gisa, imajinatu 1.500 euro baino gehiago irabazten duten langile guztien NA eremua atera nahi dela. Horretarako $\Pi_{NA}(\sigma_{Soldata > 1500}(LANGILEA))$ erabiliko litzateke eta emaitza 2.13 taulan adierazitakoa izango da.

NA
78898562S
12345125H

2.13 taula: $E = \Pi_{NA}(\sigma_{Soldata > 1500}(LANGILEA))$ proiektzioa egin ondoren lortutako emaitza.

- **Biderkadura kartesiarra** ($E = E1 \times E2$): E erlazio berria bueltatuko du. Egitura E1 eta E2 erlazioen egituren batuketa izango da, eta tuplo-kopurua, E1 x E2. Adibide gisa, jo dezagun 2.14 taulako DEPARTAMENTU erlazioa eta LANGILEA erlazioetan oinarrituta LANGILEA x DEPARTAMENTU biderkadura lortu nahi dela. Emaitza 2.15 taulan adierazitakoa da.

Zenb	Izena
5	Informatika
3	Erosketak

2.14 taula: DEPARTAMENTUA erlazioa.

NA	Izena	Abizena	Soldata	DeptuZenb	Zenb	Izena
11125252K	Jose Angel	Eguren	1200	3	3	Erosketak
11125252K	Jose Angel	Eguren	1200	3	5	Informatika
23323565L	Ane Jone	Urrutia	950	5	3	Erosketak
23323565L	Ane Jone	Urrutia	950	5	5	Informatika
78898562S	Txomin	López	1750	5	3	Erosketak
78898562S	Txomin	López	1750	5	5	Informatika
12345125H	Benito	Txurruka	2500	3	3	Erosketak
12345125H	Benito	Txurruka	2500	3	5	Informatika

2.15 taula: $E = LANGILEA \times DEPARTAMENTUA$.

- **Konbinazioa edo JOIN (E = E1 θ E2):** E1 x E2 biderkadura cartesiarra egin ondoren, a eta b atributuek θ baldintza betetzen duten tuploak bueltatuko ditu, non baldintza horretan E1eko atributuren bat (edo batzuk) E2ko atributuren batekin (edo batzuekin) erlazionatzen den (edo diren). θ eragiketa bitarra da (<, <=, >=, =). Kasu berezia da biderkadura arruntarena (E1 |x| E2), non baldintzan agertzen den atributu bakoitzak bi erlazioei dagozkien eskemen intersektzioan agertu behar duen. Baldintza, gainera, eragiketa = eragiketa izango da. Emaitzan proiektzioa ere egiten da eta proiektzio horretarako hautaturiko atributuak bi erlazioen eskemen arteko baturak emandakoak izango dira.

$$E1|x|E2 = \prod_{e1 \gg e2} (\sigma_{e1.a1=e2.a1 \text{ AND } e1.a2=e2.a2 \text{ AND } e1.a3=e2.a3 \dots} E1xE2))$$

e1 eta e2 erlazioen eskemak lirateke eta a1, a2, a3..., atributuak.

Adibidez, langile guztien zerrenda nahi izanez gero, departamentuen datuekin, DEPTZENB izango litzateke bi erlazioen eskemak lituzkeena. Hau izango litzateke eragiketa:

$$LANGILEA|x|DEPARTAMENDUA = \prod_{L \gg D} (\sigma_{LANGILEA.DeptuZerb=DEPARTAMENTUA.Zerb}(LANGILEAxDEPARTAMENTUA)) \quad L \text{ litzateke LANGILEA erlazioaren eskema eta D, DEPARTAMENTUArena.}$$

NA	Izena	Abizena	Soldata	DeptuZerb	Izena
11125252K	Jose Angel	Eguren	1200	3	Erosketak
23323565L	Ane Jone	Urrutia	950	5	Informatika
78898562S	Txomin	López	1750	5	Informatika
12345125H	Benito	Txurruka	2500	3	Erosketak

2.16 taula: E = LANGILEA |x| DEPARTAMENTUA

- **Berrizendatu: S = ρS(E):** erlazioaren izena aldatuko du. Adibidez, ρ_{IKAS(IKASLE)} eragiketak IKASLE erlazioari IKAS izena jartzen dio

Gaiari bukaera emateko, aljebra erlazionaleko eragiketa bat baino gehiago batzen dituen adibidea aurkeztuko dugu ondoren. Demagun informatika departamentuan lan egiten duten langileen izena bistaratu nahi dela. Erabili beharreko adierazpen posibleetako batzuk honako hauek lirateke:

$$\prod_{Izena} (\sigma_{DEPARTAMENTUA.Izena="Informatika"} (\sigma_{LANGILEA.DeptuZerb=DEPARTAMENTUA.Zerb} (DEPARTAMENTUAxLANGILEA)))$$

edo

$$\prod_{Izena} (\sigma_{DEPARTAMENTUA.Zerb=LANGILEA.DeptuZerb} (\prod_{Zerb} (\sigma_{Izena=Informatika} (DEPARTAMENTUA)))xLANGILEA))$$

Lehenengo eta bat, LANGILEA x DEPARTAMENTUA biderkadura cartesiarra egingo litzateke eta emaitzak 8 tuplo izango lituzke. Biderkadura horri, s (LANGILEA.Deptzenb = DEPARTAMENTUA.Zerb) hautaketa aplikatuko zaio eta orduan emaitza 4 tuplokoa izango da (LANGILEA X DEPARTAMENTUA biderkadurako 1., 4., 6. eta 7. tuploak). Tuplo horiei s(DEPARTAMENTUA.

Izena="Informatika") hautespena aplikatuko zaie, hau da, departamentuaren izena "Informatika" dutenak baino ez dira hautatuko. Azkenik, bueltatutako tuploetatik, 'Izena' atributua baino ez da proiektatuko, emaitza 2.17 taulakoa izanik.

Izena
<i>Ane Jone</i>
<i>Txomin</i>

2.17 taula: $S = \prod_{Izena} (\sigma_{DEPARTAMENTUA.Izena="Informatika"} (\sigma_{LANGILEA.DeptuZenb=DEPARTAMENTUA.Zenb} (DEPARTAMENTUA \times LANGILEA)))$

*Arkitektura eta
osagaiak*

3

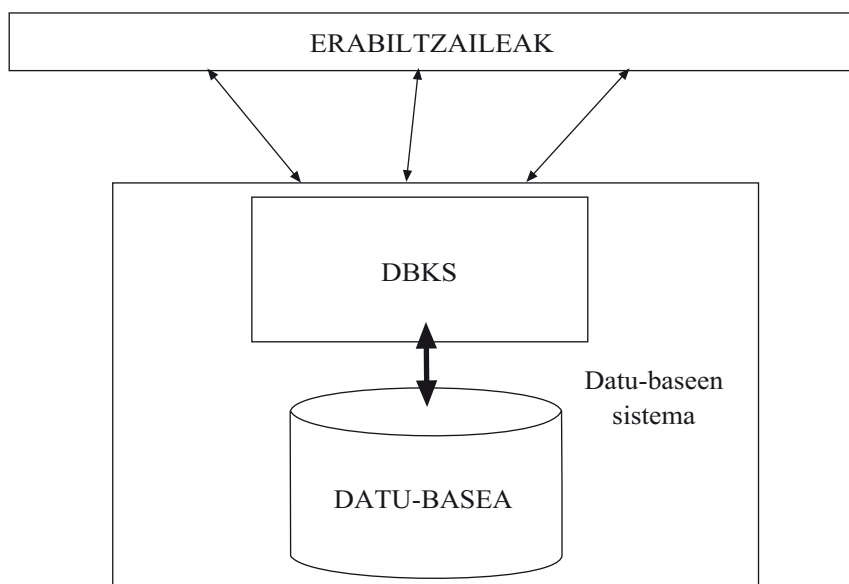
3.1. SARRERA

Denbora-tarte batean existitzen den datu-bilduma egituratuari **datu-base** esaten zaio. Datu horien arteko erredundantzia minimoa izango da eta, ahalik eta eduki semantiko gehien biltzeko asmoz, datu-eredu batean oinarrituta elkarren artean lotuta egongo dira. Gainera, datu horiek konpartitu egingo dituzte, bai erabiltzaileek, bai hainbat aplikaziok.

Hona hemen datu-base baten adibidea: bideoklub bateko bazkideen datuak (izena, helbidea, telefonoa, NA zenbakia eta abar), bideoklubean dauden filmen datuak (izenburua, zuzendaria, aktoreak eta abar), bazkideek egindako alokairuen datuak (bazkide bakoitzak zein film alokatu dituen) eta abar biltzen dituzte.

Egungo datu-baseak ordenagailuei lotuta daude eta haien kudeaketa guztiz automatizatuta dago software berezien bidez. Software hori osatzen duten aplikazioek **Datu-basea Kudeatzeko Sistema (DBKS)** eratzen dute. Datu-baseak kudeatzeko sistema datu-base fisikoaren eta erabiltzaileen artean dagoen software edo programa-bilduma da. Erabiltzaileek datu-basea ezarri, atzitu, bertako datuak irakurri, aldatu edo ezabatzeko erabiliko dute. Ikusi 3.1 irudia.

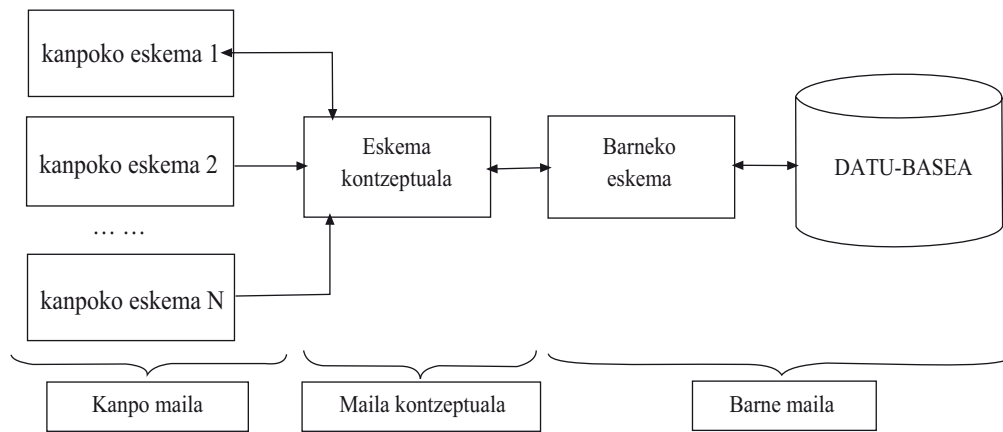
Datu-baseak eta DBKS-ak osatutako taldeari **DATU-BASEEN SISTEMA** deitzen zaio.



3.1 irudia: Datu-basea kudeatzeko sistemaren kokapena, erabiltzaileen eta datu-basearen artean.

3.2. DATU-BASEEN SISTEMEN ARKITEKTURA

Datu-baseen sistema baten helburu garrantzitsua da erabiltzaileei informazioaren ikuspegi abstraktua ematea. Horrela, erabiltzaileak ez du jakin behar datuak nola dauden biltegitatuta edo nola egin datuon mantentze-lana. Hori dena posible izateko, datu-baseen sistema zenbait abstrakzio-mailatan banatzen da. Ikusi 3.2 irudia:



3.2 irudia: 3 mailatan banatutako DBKS baten arkitektura.

Maila bakoitzean definitutako deskripzioak **eskema** bidez adierazten dira:

3.2.1. Egitura fisikoa: barneko eskema

Hemen, datuak fisikoki biltegitzeko modua (datu-basea osatzen duten artxiboen izena, antolaketa-mota, zein unitatetan biltegitzen diren eta atzipenerako metodoa, taulen deskripzioa, datu-hiztegiaren eta datu-direktorioaren deskripzioa) eta kokatzeko estrategiak (atzitzeko metodoak, gako indizeen eta erakusleen zehaztapena, datuak konprimitzeko teknikak, kriptografiatzekoak eta abar) deskribatzen dira.

Egia esan, gaur eguneko DBKSeK barneko eskeman aldaketak egiteko oso aukera gutxi uzten dute, automatikoki kudeatzen dituzte-eta beharrezko aldaketak.

3.2.2. Egitura logikoa: eskema kontzeptuala

Datuek datu-basean duten antolaketa eta datuen arteko loturak deskribatuko ditu. Eskema kontzeptualak datua berez den denean erakusten du eta ez erabiltzailea ikustera behartuta dagoen bertsio mugatua. Horretaz gain, beste zenbait ezaugarri ere zehatz daitezke; esaterako, segurtasunerako kontrol-aginduak eta eremuetarako baimentzen diren murrizketak edo balioak. Maila hori eta fisikoa datu-basearen kudeatzaileak baino ez ditu erabiltzen.

3.2.3. Kanpoko eskema

Kanpoko maila da erabiltzaileengandik hurbilena. Kanpoko hainbat ikuspegi egongo dira, eta horietako bakoitza datu-base osoaren zati baten adierazpen abstraktua da. Horrela, erabiltzaile bakoitzari beharizanen arabera dagokion informazioa erakutsiko zaio. Horrelako informazio-multzo bakoitzak **ikuspegi** izena jasotzen du. Kanpoko ikuspegi horiek kanpoko eskeman definituko dira.

Barneko eskema eta eskema kontzeptuala bakarrak dira; kanpoko eskema ugari daude, aldiz. Horrela, datu-baseak dituen adina erabiltzaile izango dira egon dauden kanpoko eskemak.

Hiru mailako sistema batean, DBKSak maila batetik besteetarako datuen transferentzia bermatu behar du. Prozesu horri **datuen transformazio** edo **maparaketa** deitzen zaio. Eskema jakin bat beste maila bateko eskemara transformatu aurretik, jatorrizko eskema egiaztatu eta onartu egin behar da.

Hiru transformazio-maila daude:

- Datuak kanpoko eskematik kontzeptualera transformatzea, eta alderantzizkoa. Erabiltzaileak kontsulta bat egikaritu nahi badu, datu-baseak kudeatzeko sistemak eskaria interpretatu, erabiltzaile horrentzat definitua dagoen kanpoko eskema egiaztatu eta eskaria kanpoko mailatik maila kontzeptualeraino transformatu du.
- Bigarren transformazioa eskema kontzeptualetik barne-mailarakoa da. Eskema kontzeptuala egiaztaturik dagoenean, kontsultarako eskaria barneko eskemara pasatzen denean gertatuko da transformazioa.
- Hirugarren transformazioa barneko eskematik datu fisikoetarakoa da. Transformazio horretan, erabiltzaileak egindako kontsulta-eskariari erantzuna emateko, biltegiatutako datuen artean beharrezkoa den erlazioa edo erlazioak hautatu eta horrelaxe gauzatu da kontsulta.

Erabiltzaileari kontsultaren emaitzak bueltatzeko, hiru transformazioak alderantziz prozesatu beharko dira.

Datu-baseetan dauden mota nagusiak honako hauek dira: hierarkikoa, sarekoa eta erlazionala. Arkitektura azaltzeko orduan, guztientzat balio duen azalpena eman dugu. Datu-base baten osagaiak zehazterakoan, ostera, osagaiak zerikusia dute datu-basearen motarekin. Liburu hau datu-base erlazionaletan zentratu dugu. Horregatik, hemendik aurrerako azalpen eta zehaztapen gehienak era horretako datu-baseei buruzkoak izango dira.

3.3. DATU-BASEEN SISTEMA ERLAZIONALEN OSAGAIK

3.3.1. Datuak

Datuen balioa berauei ematen zaien erabileraren araberrakoa da. Erabakiak hartzen lagunduko duen prozesu baten beharra dute datuek, baina sarritan makinak irakurtzeko prestatuta dagoen formatuan gordeta daude eta erabaki-prozesuek ezin dituzte erabili. Datuak formatu egokian ez egotearen programa berriak eratzen eta zaharrak aldatzen ematen den denbora eta dirua izugarria da askotan.

Datuak ahal den erabilgarrienak egiteko eta sistemaren garapenerako kostua kontrolatzeko, ezinbestekoa da datu-sistemaren diseinu egokia.

Lortu nahi den datu-basean, zerbaiti edo norbaiti buruzko datuak gorde nahi direnean, datuak erlazioetan gordetzen dira.

Demagun, adibidez, bezero bakoitzari buruz datu hauek gorde nahi direla: bezeroaren kodea ('BezKodea'), NA, izena, bi abizenak eta posta-kodea. 3.1 taulan ikus daitekeenez, sei atributu behar dira bezeroei buruzko informazioa gordetzeko. Erlazio horretan bezeroen bi agerraldi besterik ez daude.

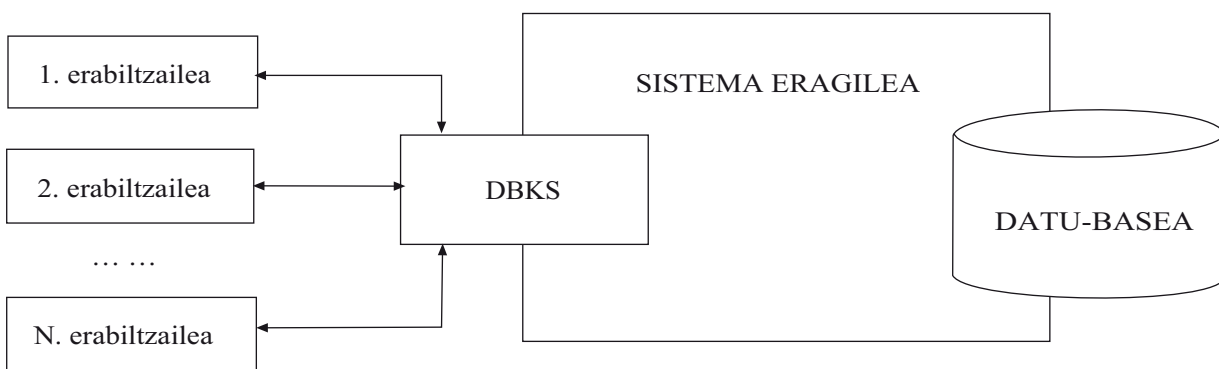
BezKodea	NA	Izena	Abizena1	Abizena2	PostaKodea
10	12345678-J	Iker	Arando	Etxebarria	48880
24	23416783-K	Nerea	Urkidi	Goitiz	48280

3.1 taula. BEZERO erlazioa.

3.3.2. Softwarea: DBKS

Datu-baseko datuak aplikazio batentzat baino gehiagorentzat balio dezaten, datuok aplikazioetatik era independentean gordeta egon beharko dute. Independentzia hori lortzeko beharrezkoa izango da aplikazioen eta datuen bitartean interfaze-lanak egingo dituen programa-multzo batez baliatzea. Programa-multzo horri **Datu-basea Kudeatzeko Sistema** deitzen zaio.

Datu-basea Kudeatzeko Sistema elkarlanean dago sistema eragilearekin. Bi ezaugarri izan behar diru DBKSak: alde batetik, datu-basearen erabiltzaileei (erabiltzaile ez-informatikoak, analistak, programatzaileak eta datu-basearen administratzaileak) biltegitatuta dauden datuak deskribatu, datuen atzipena lortu eta gordeta dauden datuak manipulatu ahal izateko aukera eman (betiere datuen segurtasuna, konfidentzialtasuna eta integritatea mantenduz), eta bestetik, elkarren artean koordinaturik dauden programa, prozedura eta lengoaietz osaturiko multzoa eratu (ikusi 3.3 irudia).



3.3 irudia: Datu-basea Kudeatzeko Sistemak sistema eragilearekin elkarlanean dihardu.

DBKS-ak datu-base bat baino gehiago kudea ditzake.

Hori guztiori maila teorikoan gertatzen da, baina praktikan beste arazo batzuk sortzen dira: nola bereizi datu-basea eta DBKS-a? Egia da datu-baseen sistema batean gehienetan DBKS-ak erraz kudea ditzakeela zenbait datu-base, baina betiere beraren mota berekoak. Hau da, software garatzaile bakoitzak “bere” datu-base motak baino ez ditu kudeatzen. Estandarizazioa puntu horretan ia nulua da. Adibidez: Microsoften *SQLServer* DBKS-ak ez du arazo handirik *SQLServer* datu-base sorta maneiatzeko, baina ezin du *Oracle* edo *PostgreSQL* bitartez sortutako datu-baserik kudeatu. Lehenik, DBKS sortzaileak beste DBKS-ak ere ulertuko duten formatu batera esportatu behar du datu-basea (XML, testu soila,...) eta, ondoren, bigarren DBKS-ak formatu horretatik berera inportatu beharko du. Baina hori behin-behineko irtenbidea besterik ez da.

Ondorioz, datu-basearen eta DBKS-aren arteko independentzia teorian baino ez dago; praktikan lotuta daude eta ezin dira banatu. Horregatik esaten da datu-baseen sistemak itxiak direla.

DBKS-a osatzen duten programa, prozedura eta lengoaiak horietatik honako hauek nabarmenduko genituzke (5. gaien sakonduko da):

- DDLa (Datuak definitzeko lengoia): datuak zenbait abstrakzio-mailatan (fisikoa, logikoa eta kanpokoa) definitzeko DDL lengoia erabiltzen da. Lengoaia horren bidez, datu-basean parte hartuko duten datuen deskripzio logikoa jorratzen da; hau da, batetik datuen definizioa, eta bestetik datu horien ezaugarriak eta beraien arteko loturak. Gehien nabarmentzen diren aginduak honako hauek dira: datu-basean erlazioak sortzen edo ezabatzen dituzten CREATE eta DROP,

eta murrizketak eta segurtasun semantikorako arauak definitzen dituzten CHECK eta CONSTRAINT aginduak.

DDL lengoaiaren barruan beste bi instrukzio-multzo ere badaude: DCLa eta DSDLa.

- DCLa (Datu-basea kontrolatzeko lengoiaia): alde batetik, datuen kontrolaz eta segurtasunaz arduratuko den azpilengoaia dago. Datu-baseen sistemara jotzen duten eta sistemarekiko elkarreraginean diharduten mota askotako erabiltzaileak daude, eta horietako bat DBKS-ko administratzailea (DBA) da. Datuak kudeatzeko garaian, ardura guztiak bere gain ditu DBAk eta beraren esku dago segurtasunerako politika guztia. Erabiltzaile arruntek izaten ez dituzten goi-mailako agindu batzuk erabili ahal ditu eta, beraien bitartez, erabiltzaile kontuak sortzeko, erabiltzailei pribilegio batzuk emateko edo kentzeko eta segurtasun-mailak kudeatzeko gaitasuna du. Horretarako GRANT eta REVOKE aginduez baliatuko da, beste batzuen artean.
- DSDLa (Datuen biltegitratzea definitzeko lengoiaia): beste alde batetik, ikuspuntu fisikoa jorratzen duen azpilengoaia dago, hau da, datuen biltegitratzea definitzen duen lengoiaia. Beraren bitartez deskribatuko dira datuak biltegitratuko diren unitate fisikoak, erabiltzen diren bolumenak eta artxiiboak, eta informaziorako atzipen-metodoak: clusterrak, blokeak, indizeak, erlazioak eta abar.
- DMLa (Datuak kontsultatu eta manipulatzeko lengoiaia) datuen manipulazioa gauzatuko duen lengoiaia. Datuak manipulatzeko (betiere datu-base administratzaileak emandako pribilegioak eta segurtasun-neurriak errespetatuz) gehien nabarmentzen diren aginduak honako hauek dira: hurrenez hurren, datuak kontsultatzen, txertatzen, aldatzen eta ezabatzen dituzten SELECT, INSERT, UPDATE eta DELETE aginduak.

Datu-baseko datuak kontsultatu eta manipulatzeko bi era daude:

- DML sententziak zuzenean erabiltzea.
- C, C++, VBasic, Java, PHP, COBOL edo bestelako lengoiaia ostalariak erabilia sortzen diren aplikazioak. Aplikazio horiek ingurune grafikoetan oinarritutako menuen bitartez diseinatu-ta daude. Lengoiaia hori erabiliko da erabiltzaileari datu-basearen datuak manipulatzeko bidea emango dioten aplikazio-programak egiteko. Programa horien iturri-kodean DML sententziak daude txertaturik.

DML sententziak prozedurazkoak edo ez-prozedurazkoak izan daitezke. Prozedurazkoen kasuan, erabiltzaileak atzipena pausoz pauso zehaztu beharko luke, zein datu eta nola lortu espezifikatuz. Ez-prozedurazkoen kasuan, erabiltzaileak zein datu lortu nahi dituen zehaztu beharko du eta ez nola. Erabiltzaileei bideratutako lengoiaiek ahal denik eta ez-prozedurazkoenak izan behar dute. Horrela, datu-base erlazionaletan DMLa ez-prozedurazkoa da (SQL); aldiz, datu-base hierarkiko eta sareko datu-baseetan DMLa prozedurazkoa da.

Amaitzeko, komeni da aipatzea, datu-baseen sistemez hitz egitean, DBKSaz gain beste software gehigarri batzuk ere badaudela, hala nola garapeneko softwareak, lengoiaia ostalariaren konpiladoreak eta abar.

3.3.3. Erabiltzaileak

Erabiltzaile askok nahi dute horrenbesteko konplexutasuna duen sistemarekin lan egin. Lehenengo eta behin, erabiltzaile terminalen betebeharrak zehazten dituzten espezifikazioak garatuko

dituzten analistak daude. Ondoren, diseinatzaileak arduratzen dira betebeharrak diseinatzen, hau da, datu-basean parte hartuko duten datuak, datuon egiturak eta datuen arteko loturak eratzen, eta abar. Komentatu beharra dago analisi eta diseinuko lan horietan etorkizunean datu-base administratzailea izango denak ere parte hartu beharko lukeela. Erabiltzaile-motak:

- Datu-basearen administratzaileak (DBA). Datu-basea administratuko dute eta pribilegio-mailarik altuenaren jabe izango dira.
- Programatzaileak betebeharrak inplementatzeaz arduratzen dira. Horretarako, 3. eta 4. belaunaldietan oinarritutako programazio-inguruneak eta menuak erabiliz diseinatuta dauden aplikazioak garatuko dituzte. Programa horien iturri-kodean DML sententziak daude txertatuta. Horrela, erabiltzaile terminalak datu-baseko datuetarako atzipena lortu ahal izango du.
- Erabiltzaile terminalek, aplikazio-programen bidez, datu-basearekiko elkarreraginean dihardute. Ez dute zertan adituak izan. Erabiltzaile-mota horien artean, adibidez, ikastetxe bateko idazkaria kokatu ahal da. Idazkari horrek aurretik diseinatuta dagoen aplikazio-programaren bat erabiliko du ikasleen txostena ateratzeko, esate baterako.
- Ustekabeko erabiltzaileak. Erabiltzaile sofistikatu izenez ere ezagutzen dira, eta datu-baseko datuekin era interaktiboan lan egiten dute. Horretarako, kontsultak onartzen dituen lengoia batez idazten dituzte aginduak.

Erabiltzaile moten artean daukan garrantziagatik, administrariaren (DBA) lana zehaztuko dugu. DBAk, datu-baseak definitu eta kontrolatzeko ardura ez ezik, erabiltzaile guztiei DBKSari buruz behar duten laguntasuna ematearena ere beregan baitauka. Datu-basearen administratzaile pertsona bakarra edo gehiago izan daitezke.

Normalean, zenbait DBKS ezagutuko ditu eta datu-baseen diseinuan, sistema eragileen arloan, sareen arloan eta hardware eta softwareari dagozkien gaietan aditua izango da.

Hauek dira DBAri dagozkion funtzio garrantzitsuenak:

- Datu-basearen eskema fisikoa definitu eta mantendu, hau da, datu-basean parte hartuko duten artxibo, erlazio eta erregistroen definizioa. Horretaz gain, datuetarako sarbide-metodoak eta datuak konprimitzeko eta zifratzeko teknikak deskribatuko ditu besteak beste.
- Datu-basearen eskema logikoa inplementatu eta mantendu, hau da, DBAk datuen antolaketari eta loturei buruzko deskripzioak, segurtasunaren kontrolerako aginduak eta eremuetarako baimentzen diren murrizketak edo balioak inplementatuko ditu.
- Hainbat ikuspegi sortu eta mantendu, erabiltzaile bakoitzari dagokion informazioa baino ez erakusteko.
- Datuen pribatutasuna eta segurtasuna kontrolatzea. Horretarako, hainbat eratako pribilegioak emango zaizkie erabiltzaileei (batzuek informazioa irakurri baino ezin izango dute egin, beste batzuek irakurri eta aldaketak egin...)
- Gai informatikoez aparte, DBAk enpresako legeak eta politikak ezagutu beharko ditu. Horregatik, DBAk administratu behar duen datu-baseak kudeatzeko sistemak enpresako datuen administrazio-politikaren arauak beteko ditu, eta DBAren helburu nagusietako bat horixe izango da.

DBAk funtzio guztiak gauzatzeko DDL eta DML sententziak erabiliko ditu, eta informazio hori guztia datu-hiztegian gordeta geratuko da. Izan ere, hurrengo puntuan ikusiko den bezala, datu-hiztegi horixe da DBAk aurrean komentatutako funtzioak gauzatzeko duen tresnarik garrantzitsuen.

3.3.4. Datu-hiztegia

Datu-hiztegia biltegiara daitezkeen datuei buruzko informazio-bilduma da, hau da, metadatu biltzuma. Datu-baseko erabiltzaileek irakur dezaketen informazioa da eta hau guztiau biltegitan bertan:

- Datu-basearen eskema fisikoaren, logikoaren eta kanpoko eskemen deskripzioa. Baita eskema batetik beste batera pasatzeko beharrezkoak diren erregelak ere.
- Datu-basean sortutako ikuspegiaren definizioak, ikuspegi bakoitzaren erabiltzailea nor den eta zer sarbide-eskubide dituen.
- Segurtasunarekin lotuta dauden erregelak, arauak eta murrizketak.
- Datuak gordeta daudeneko eremua, erregistroa eta erlazioa, eta beraien arteko loturen deskripzioak.

Definitutako datu-basearen barruan egoten da hiztegia eta, edozein datu-baserekin egiten den eran, berari galdeketak egiteko aukera ematen du. Hots, baimena duten erabiltzaileek hiztegiaren kontra kontsultak, txertaketak eta abar egin ditzakete.

3.3.5. Indizeak

Datu-base batean gordeta dauden datuen aurka DML sententziak erabiliz egiten dira kontsultak. Gainera, erabiltzaileek aldi berean datu baten eguneraketa bat baino gehiago egikaritu ahal dituzte; eta hala denean, konkurrentzia arazoak sortzen dira. Hori dela eta, sistemaren errendimenduaren ikuspuntutik, oso interesgarria izan daiteke kontsulta bakoitzaren emaitzak ateratzeko behar duen denbora aztertzea.

Kontsultak azkartzeko sortzen dira indizeak. Indizeok erregistro bateko atributu batengan edo gehiagorengan jartzen dira. Kontsultan bilaketak egiteko indizatutako atributuren bat erabiltzen bada, datuak askoz azkarrago bilatzen dira. Indizerik erabiltzen ez bada, DBKSak erlazioko erregistro guztiak irakurri beharko ditu era sekuentzian lehenengotik hasita; hau da, erlazioa oso-osorik arakatu beharko du. Arazoa areagotu egingo da erlazioko erregistro-kopurua handia bada. Horrela, estatistikek diotenez, 1.000 erregistroko erlazio txikietan ere indizeak erabiliz egindako kontsultak indize gabeko kontsultak baino 100 aldiz azkarragoak dira. Erlazio handietan lortzen den abantaila oraindik ere handiagoa izango da.

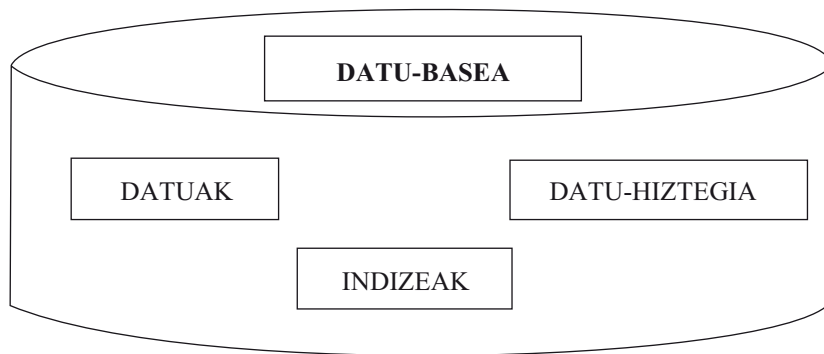
Erlazioa goitik behera arakatzea ekidin egin behar da ahal den neurrian, ondoren aipatzen diren arrazoiengandik:

- Prozesatzeko unitate zentralaren gainkarga.
- Konkurrentzia: DBKSa erlazioaren miaketa egiten dabilen bitartean, erlazioa blokeatuta egongo da. Beste erabiltzaileek erlazio horretan datuak irakurri baino ezin izango dute egin; hauda, ezin izango dute erregistrorik eguneratu ez ezabatu.

- Datuak gordeta dituen diskoaren gainkarga. Erlazio handien eskaneok sarrera/irteerako eragiketa asko egitera behartzen du diskoa.

Erlazioko eremu bat edo gehiago indizatzen direnean, DBKSak diskoan beste datu-egitura bat sortzen du indizatutako eremuen balioekin. Datu-egitura berri hori zuhaitz-motakoa izango da eta DBKSak oso azkar egingo ditu bilaketak datu-egitura berri horretan.

Orain erraza da konturatzea, datu-baseari buruz hitz egiterakoan, datuez gain datu-basearen barruan kokatutako indizeez eta datu-hiztegiak ere hitz egiten dela (ikusi 3.4 irudia).



3.4 irudia: Datuak, indizeak eta datu-hiztegia dira datu-basearen osagaiak.

Indize mota bat baino gehiago daude. Erabilienak **gako nagusia** eta **gako atzerritarra** dira.

Gako nagusia PRIMARY KEY klausularekin sortzen da eta ezin du balio nulurik (NULL) izan. Gainera, erregistro bateko eremu batean edo gehiagotan aplikatu ahal da. Gako nagusiari esker, erregistro hori erlazioko erregistro guztietatik era bakarrean identifikatuta geratuko da.

Gako nagusia aplikatzeko adibide argia da honako hau: pentsatu datu-base batean ikastetxe bateko ikasleen datuak gorde behar direla erlazio honetan:

NA	Izena	Abizena1	Abizena2	Helbidea	Herria	Telefonoa
----	-------	----------	----------	----------	--------	-----------

3.2 taula. IKASLEAK erlazioaren egitura.

Ikasle guztiek dutela NA pentsatuko dugu. Beraz, NA eremua gako nagusi bihurtuko da eta indizatu egingo da.

IKASLEAK erlazioa sortzeko DDL sententzia hau erabiliko da:

```
CREATE TABLE IKASLEAK (NA varchar(9), Izena varchar(15), Abizena1 varchar(20), Abizena2 varchar(20), Helbidea varchar(30), Herria varchar(15), Telefonoa varchar(9), PRIMARY KEY (NA));
```

Esan bezala, indizeei esker kontsultak azkarrago egikaritzen direnez, erregistroko eremu guztiak indizatzeko tentazioa izan dezakegu. Baina indizeak erabiltzeak “kostua” duela jakin behar da. Datuen erlazio bateko erregistro batean INSERT, UPDATE, REPLACE edo DELETE aginduekin datua txertatu, eguneratu, aldatu edo ezabatzen denean, sortutako indizeen zuhaitz-egitura ere eguneratu egingo da, eta ondorioz, sistemaren errendimendua jaitsiko da. Beraz, indizeak sortzeko prozesu horrek gehieneko muga dauka. Bi aldagai horien (galdeketak bizkortu vs eguneraketak geldotu) arteko erlazioak ezartzen du topea. Oro har, erlazio batek ez ditu bi edo hiru indize baino gehiago izango.

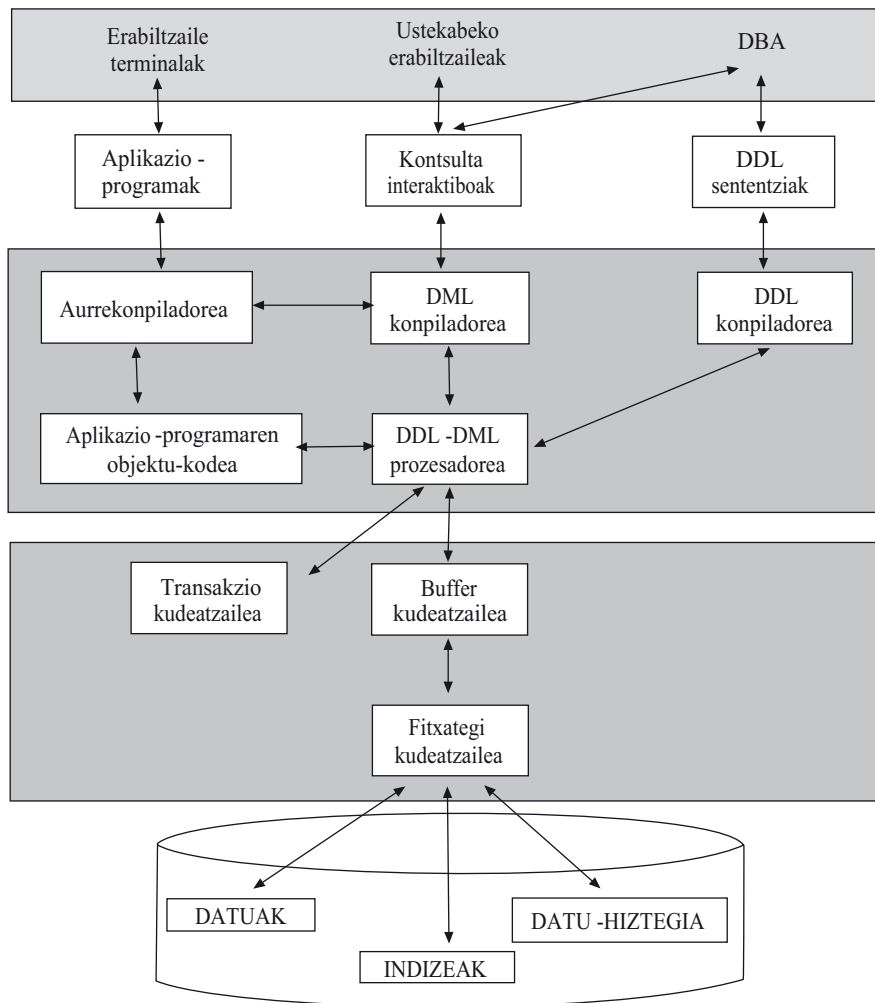
Indizeen potentzia hori kontsultan indizatutako eremua WHERE klausulan erabiltzen denean aprobetxatzen da gehien bat. Horretarako, indizeak sortu aurretik, kontsulta gehienak zein eremu kontuan hartuz egingo diren jakitea komeni da. Hori dela eta, askotan, datu-basearen diseinua egi-terakoan, oraindik ezin izango dira indize guztiak aurreikusi. DBAk sortuko ditu aurrerago beharko dituen indizeak.

Esan bezala, datuak deskribatu, datuen atzipena lortu eta gordeta dauden datuak manipulatzeko koordinaturik dauden programek, prozedurek eta lengoaiak osatutako multzoa da DBKSa. Ondoren- go puntuan programa horien koordinazioa eta elkarlana hainbat moduluren bitartez nola gauzatzen den ikusiko da.

3.4. DBKS-AK LAN EGITEKO DUEN ERA, ZENBAIT MODULUREN IKUSPUNTUTIK

DBKSa honako modulu hauek osatzen dute:

- **Biltegiratze-kudeatzailea:** datuetan, indizeetan eta datu-hiztegian egiten diren sarbideak kontrolatzen ditu. Horretarako, memoria nagusian dauden *bufferen* eta diskoan dauden fitxategien kudeaketaz arduratzen da.
- **DDL-DML prozesadorea:** sortzeko, kontsultatzeko eta eguneratzeko aginduak jasotzen ditu eta haiek egikaritzeko modurik egokiena bilatzen du. Ondoren, biltegiratze-kudeatzaileari aginduak bidaltzen dizkio.
- **Transakzioen kudeatzailea:** datu-basearen sendotasuna bermatzen du. Edozein akats gertatzen denean, oso garrantzitsua da datu-basea egoera sendoan geratuko dela ziurtatzea. Horretarako, transakzio izeneko instrukzio-multzoa definitzen da. Transakzioko instrukzio guztiak (transakzioa bera) batera doaz, hau da, guztiak egikaritutako dira edo bat bera ere ez; hasieran, transakzioa egikaritu aurretik, datu-basea egoera sendoan egon ohi da; transakzioa bukatzen denean, datu-baseak egoera sendoan jarraitu beharko du. Bien bitartean (transakzio erdian) datu-basea memento batzuetan egoera ez-sendoan egon daiteke.
- **DDL konpiladorea:** DBAk agindutako DDL sententziak prozesatu eta datu-hiztegian gordetzen ditu. Sententzia horien bidez datuen deskripzio logikoa, datuen pribatutasuna, segurtasunaren kontrola eta biltegiratzea kontrolatzen dira.
- **DML konpiladorea:** DML sententziak konpilatzen ditu. Sententzia horiek bi bidetatik jasoko ditu: alde batetik, ustekabeko erabiltzaileek edo erabiltzaile sofistikatu deiturikoez era interaktiboan datu-basearen kontra egingo dituzten kontsulta eta eguneraketa bidez idatzitako DML sententziak. Beste alde batetik, aurrekonpiladoreak ematen dizkion DML sententziak.
- **Aurrekonpiladorea:** datu-base baten kontra aplikazio-programa bat erabiltzen denean, DML sententziak lengoia ostalariak erabilia sortzen diren aplikazioetatik jasotzen dira. Aurrekonpiladoreak DML sententziak eta lengoia ostalariko kodea bereizten ditu. Modulu horien funtzionamendua 3.5 irudian agertzen da:



3.5 irudia: DBKsa osatzen duten moduluen funtzionamendua.

Analisis eta diseinua

4

4.1. SARRERA

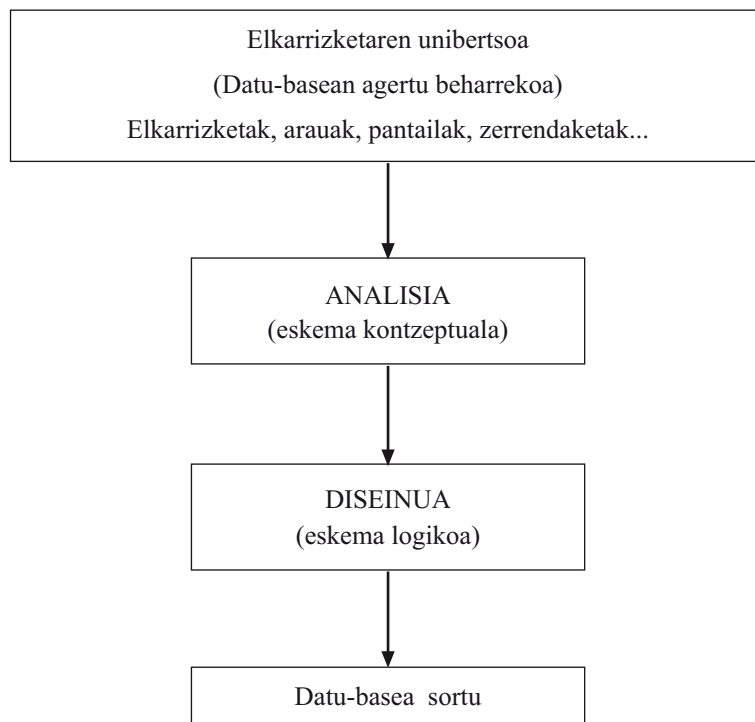
Hirurogeiko hamarkadatik aurrera datu-baseek garapen izugarria izan dute. Hori dela eta, ezinbestekoa ikusten da datu-baseen analisia eta diseinua egiteko zenbait pauso jarraitzea. Datu-baseen diseinua ondo egiteko 4.1 irudian azaltzen den eskema proposatzen da.

Lehenengo pausoa elkarrizketaren unibertsoa definitzea da, hots, datu-base bidez ezarri nahi dena, orokorrean diseinatzaileak eta enpresako informazio-sistemako arduradunak erabakitzen dutena. Diseinatzaileak, informazioa bildu ondoren, elkarrizketaren unibertsoa definitzen du, enpresako informazio-sistemari buruz duen ikuspegia agertuz. Hasierako datu-hiztegia sortzen da, sisteman garrantzitsuak diren osagai guztien zerrendarekin. Osagai bakoitza ondo definituta egotea komeni da, horri esker erabiltzaileak eta diseinatzaileak datuen ikuspegi berbera izango dute eta.

Sistema informatizatzeko, lehenik eta behin, beharrianak azertu behar dira. Informatika ikuspuntutik, beharrian horiek, aurrera ateratzeko, diseinuko faseari ekiten zaio, identifikatutako arazo bakoitzarentzat diagrama bidezko soluzioa planteatuz. Horrela, behin-behineko datuen eskema kontzeptuala lortuko da. Datuen eskema horrek elkarrizketaren unibertsoa datuak deskribatuko ditu.

Konponketak eginez, behin betiko eskema kontzeptuala lortuko da; hau da, Entitate/Erlazio (E/R) diagrama. Bertan, geroago 4.2 puntuan definitutako diren entitateak, beraien arteko erlazioak, atributuak, murrizketak eta abar agertzen dira.

E/R ereduak, edozein datu-base motatako datuen analisia eta berezitasunak zehazteko erabiltzen da. Datu-baseen eredu jakin bat sortzeko, aurretik lortutako E/R diagramaren eraldaketa behar-beharrezkoa da, datuen eskema logikoa lortzeko. Adibidez, sortu nahi den datu-basea eredu erlazionalekoa balitz, orduan E/R eskematik datuen eredu erlazionaleko eraldaketa egin beharko litzateke (ikus 4.3 puntua). Ondoren, diseinua bukatzeko, normalizazio-prozesua aplikatuko zaio, diseinu normalizatua lortzeko.



4.1 irudia. Datu-basea sortzeko pausoak.

4.2. DATUEN ANALISIRAKO ETA ESPEZIFIKAZIORAKO TEKNIKA. E/R EREDUA

Elkarrizketaren unibertsoa definitu ondoren, datuen eskema kontzeptuala eratu behar da. E/R ereduak da eskema hori adierazteko gehien erabiltzen dena. E/R ereduarekin aplikazio informatikoaren atal estatikoa adierazi nahi da; hau da, aplikazioarentzako interesgarriak izango diren datuak. Horretarako ikur eta erregela bereziak erabiltzen dira.

E/R ereduak Peter Chenek aurkeztu zuen. Erabiltzailearen eta diseinatzailearen arteko elkar ulertzea errazten zuen etedu berri horrek. Hasiera batean aurkeztutako ereduari “oinarrizko eredu” deitu zitzaion (ikus 4.2.1 puntua). Geroago, diseinatzaileek aurkeztutako ereduak gabezia batzuk zituela konturatu ziren. Gabezia horiei erantzun asmoz, eredu hedatua aurkeztu zuten (ikus 4.2.2 puntua).

Baina datuen analisisia beste eredu batzuk jarraituz ere egin daiteke, Martin ereduak erabiliz adibidez. Horrekin, eskema kontzeptuala era malguagoan agertzen da (ikus 4.2.3 puntua).

4.2.1. Oinarrizko ereduak (E/R)

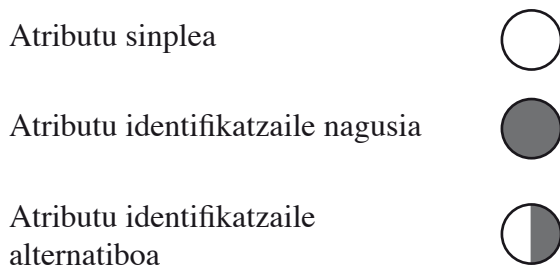
Datuen eskema kontzeptual honetan hiru osagai nagusi bereizten dira:

- **Entitateak:** lortu nahi den datu-basean, elkarrizketaren unibertsoan definitu den zerbaiti edo norbaiti buruzko informazioa gorde nahi denean, zerbait edo norbait hori entitate bilakatzen da. Entitateak laukitxo baten barruan adierazten dira, 4.2 irudian ikusten den bezala.

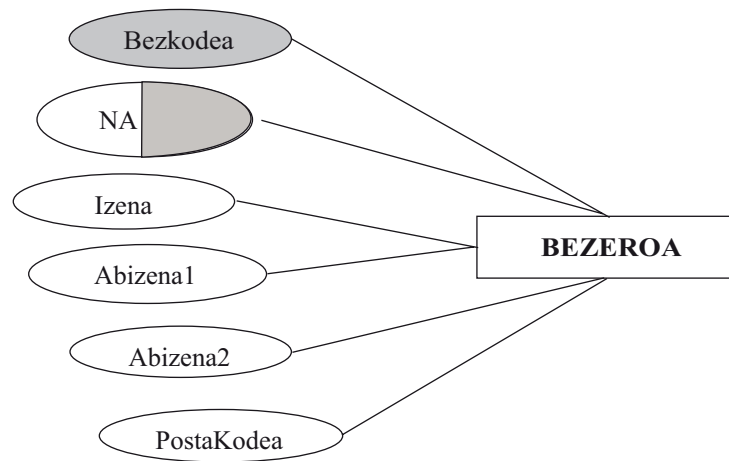


4.2 irudia. Entitate baten adibidea, laukizuzenaren barruan adierazita.

- **Atributua:** entitate edo erlazio batek dituen ezaugarriak deritze. Era honetan adierazten dira:

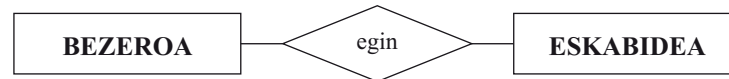


Demagun, adibidez, BEZEROA entitatearen atributuak honako hauek izan daitezkeela: bezeroaren kodea ('BezKodea'), nortasun-agiria (NA), izena, 1. abizena, 2. abizena eta posta-kodea. Atributu horiek guztiak ez dira mota bereberkoak. Adibidez, bezeroaren kodea atributu identifikatzaile nagusia izango da, guztien artean bezero bakoitza bakarka identifikatzeko balio duelako. NA atributuak ere bezeroa bakarka identifikatuko du; beraz, NA atributu identifikatzaile alternatibo modura jar daiteke. Beste atributu guztiak sinpleak dira (ikus 4.3 irudia).



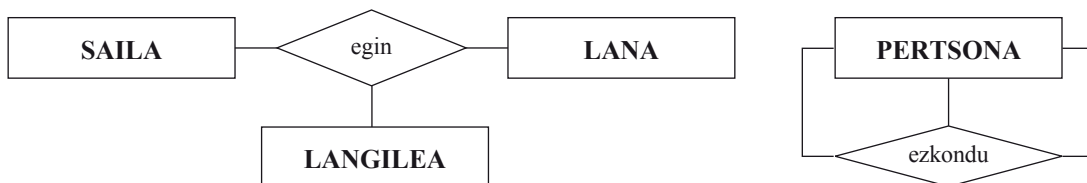
4.3 irudia. Entitatea zenbait atributuekin.

- **Erlazioak:** entitateen arteko ekintzak dira. Erronboz adierazten dira eta barruan erlazioarekin zerikusia duen aditza idazten da. 4.4 irudian ikusten denez, bezero batek eskabideak *egin* ahal ditu, eta eskabidea bezeroak *egina* izango da.



4.4 irudia. 'Egin' erlazioak BEZEROA eta ESKABIDEA lotzen ditu.

- **Gradua:** erlazioetan parte hartzen duen entitate-kopuruari erlazioaren **gradu** deitzen zaio. 4.4 irudian agertzen den *egin* erlazioaren gradua bi (2) da. 4.5 irudiko *egin* erlazioaren gradua hiru (3) da, eta *ezkondu* erlazioaren gradua bat (1) da.



4.5 irudia. 'Egin' eta 'ezkondu' erlazioak.

- **Kardinaltasuna:** kontzeptu hau ulertzeko, lehenengo, agerraldia zer den jakin behar da. BEZEROA entitateko agerraldi bati dagokion datu-multzoa izango da; hau da, bezero konkretu baten datuek osaturiko multzoa (ikusi 4.1 erlazio-taula).

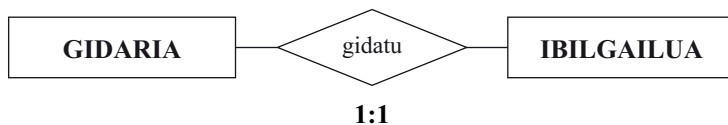
BezKodea	NA	Izena	Abizena1	Abizena2	PostaKodea
10	12345678-J	Iker	Arando	Etxebarria	48880

4.1 taula. BEZEROAK entitateko agerraldia.

Bigarren graduko erlazioetan, entitate bateko agerraldi jakin bat beste entitate bateko zenbat agerraldirekin lotuta dagoen adierazten du kardinaltasunak. Kontzeptu hori datuen eskema kontzeptualetik (E/R) datuen eskema erlazionalera (eskema logikora) pasatzerakoan oso kontuan hartu beharrekoa da.

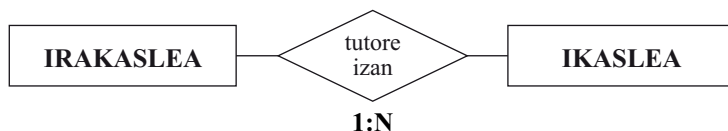
Hiru kardinaltasun-mota daude: batetik baterakoa (1:1), batetik askotarakoa (1:N) eta askotatik askotarakoa (N:M). Entitate batekin erlazionatutako agerraldi-kopurua bat (1) bada, orduan marra soil batez adierazten da. Aldiz, entitate baten erlazionatutako agerraldi-kopurua batekoa (1) baino handiagoa bada, orduan marra eta geziz adierazten da.

Adibidez, garraio-enpresa bateko gidari bakoitzak ibilgailu bat baino ez du gidatzen (marrarekin adieraziko da), eta ibilgailu bakoitza beti gidari berak gidatzen du (marrarekin). Egoera horren E/R eskema 4.6 irudikoa litzateke.



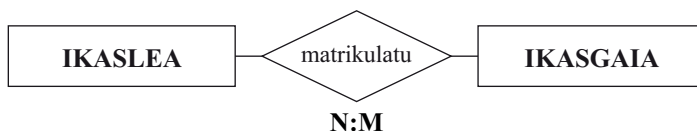
4.6 irudia. ‘Gidatu’ erlazioak GIDARIA eta IBILGAILUA entitateak lotzen ditu.

Beste adibide honetan, ikastetxe batean irakasle bat ikasle askoren tutore izan ahal dela joko da (marra eta geziarekin adieraziko da) eta aldiz, ikasle batek tutore bakarra baino ez duela izango (marrarekin bakarrik). Egoera horren E/R eskema 4.7 irudikoa izango litzateke:



4.7 irudia. “Tutore izan” erlazioak IRAKASLEA eta IKASLEA entitateak lotzen ditu.

Bestalde, N:M motako kardinaltasuneko E/R diagramaren adibidea 4.8 irudikoa da. Ikasleak ikasgai askotan matrikulatuta egon daitezke, eta ikasgai batean ikasle bat baino gehiago matrikula daitezke.



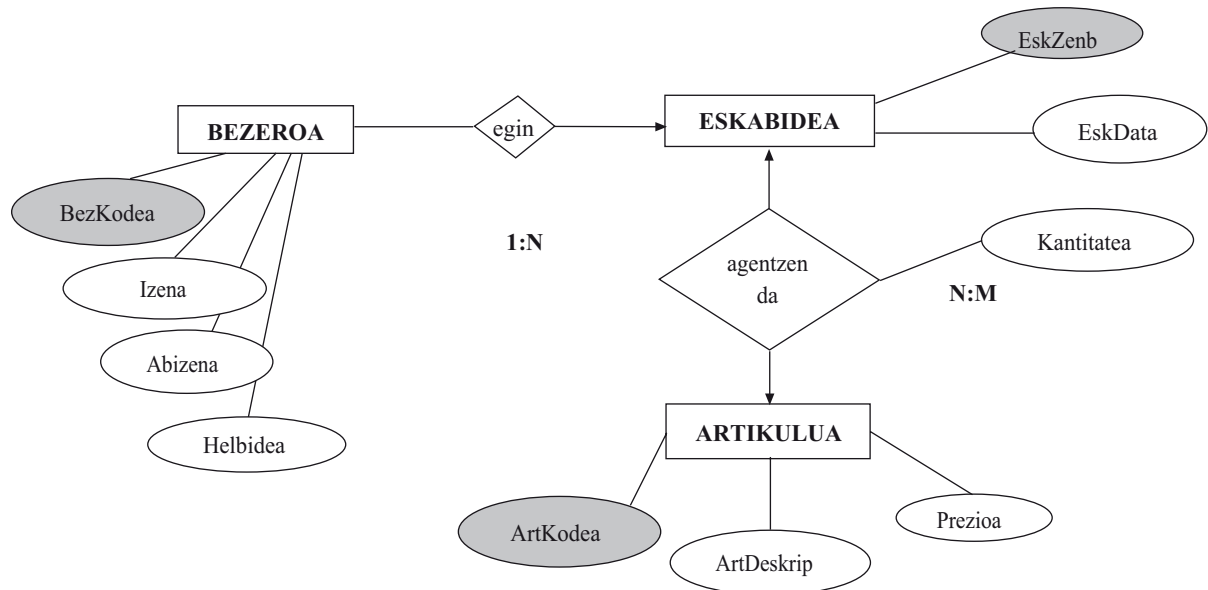
4.8 irudia. “Matrikulatu” erlazioak IKASLEA eta IKASGAIA entitateak lotzen ditu.

Datu-baseetan, entitatea, erlazioa, gradua, kardinaltasuna eta atributuen kontzeptuak batera joaten dira. Adibide errealagoa aurkeztuko da ondoren:

Bezero batek eskabide asko egin ditzake, baina eskabide bakoitza bezero batek egina izango da. Bestalde, bezero bakoitzeko honako ezaugarri hauek gorde beharko dira: bezeroaren kodea, izena, abizena eta helbidea. Eskabide bakoitzeko gorde beharreko ezaugarriak eskabidearen zenbakia eta data izango dira.

Eskabide bakoitzean artikulua asko agertu ahal izango dira eta, logikoa denez, artikulua bakoitza eskabide askotan ere agertuko da. Artikulu bakoitzeko, kodea, deskripzioa eta prezioa gorde beharko dira. Eskabide bakoitzean artikulua bakoitzeko eskatzen den kantitatea ere jakin beharko da.

Egoera horren E/R eskema 4.9 irudian ikus daiteke:



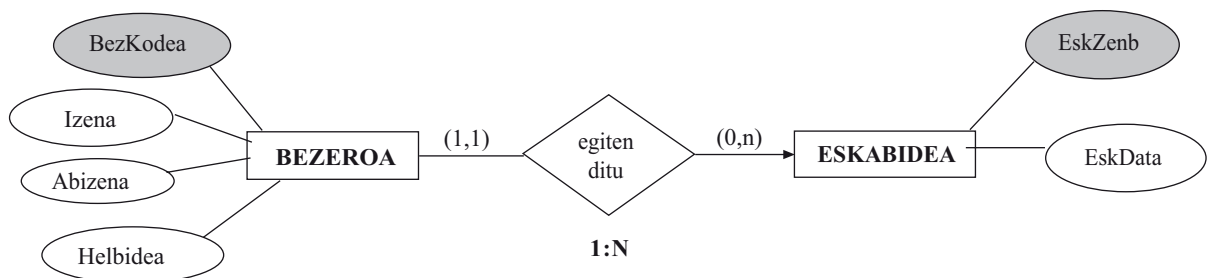
4.9 irudia. BEZEROA, ESKABIDEA eta ARTIKULUA entitateei dagokien E/R eskema.

Eskema kontzeptuala lortu eta gero, eskema kontzeptualeko osagai estatikoak (ikus 2. gaiko 2.1 irudia) adierazteko DDL lengoia erabiltzen da.

4.2.2. Eredu hedatua (EE/R)

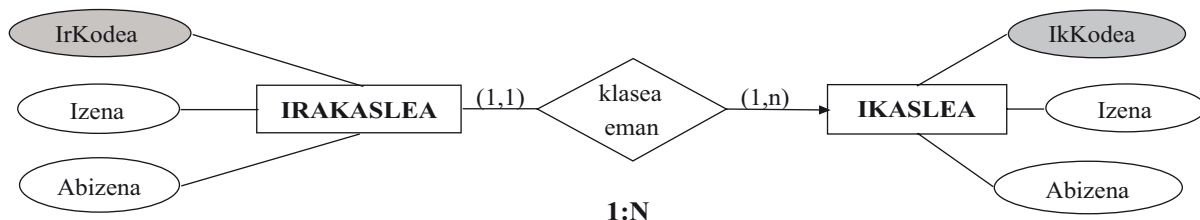
Beste autore batzuek P. Chenek aurkeztutako eredu hobetu zuten eta horixe da, hain zuzen ere, gaur egun erabiltzen dena. Hala ere, oraindik E/R modura izendatzen da, EE/R bada ere laburdura zuzena. Hobekuntzen artean daudenetako bat, entitate bakoitzak erlazioan dituen agerraldi-kopurua adieraztea da. Entitate batek izan dezakeen gertaera-kopuru handiena eta txikiena adierazten ditu (x,y) , non x gutxieneko kopurua den eta y , berriz, gehienekoa; kopuru minimo eta maximo ere esaten zaie.

Adibide honetan, bezeroak n eskabide egin ditzake gehienez, eta zero gutxienez. Alderantziz, eskabide bakoitza bezero batek baino ezin du egin. E/R eskema 4.10 irudian agertzen da:



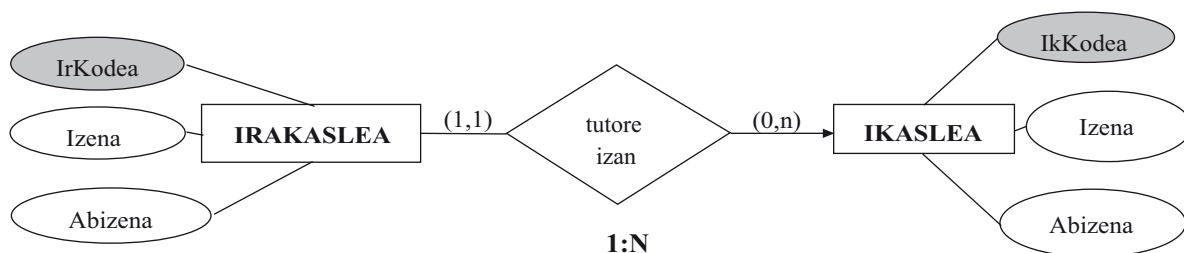
4.10 irudia. “Egiten ditu” erlazioak BEZEROA eta ESKABIDEA entitateak lotzen ditu.

Beste adibide honetan, demagun irakasle batek klasea ikasle askori eman diezaiokeela (gutxienez ikasle bati); aldiz, ikasle batek irakasle bakar batengandik (gutxienez eta gehienez) jasoko ditu klaseak. Egoera horren E/R eskema 4.11 irudian ikus daiteke:



4.11 irudia. "Klasea eman" erlazioak IRAKASLEA eta IKASLEA entitateak lotzen ditu.

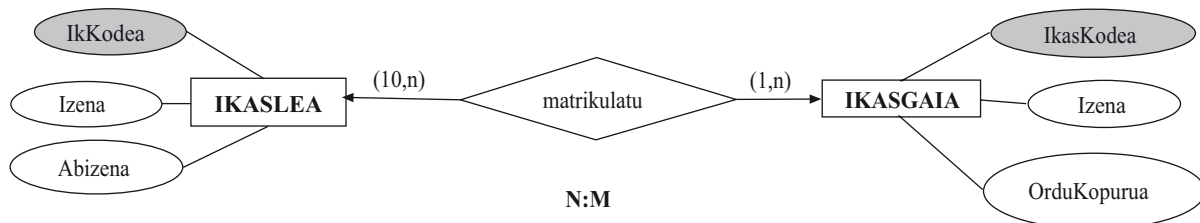
Hirugarren adibide honetan, ikastetxeko irakasle guztiek ez dute tutore papera beteko; posible da irakaslea tutore ez izatea, edo ikasle askoren tutore izatea. Aldiz, ikasle batek irakasle bat bakarrik izango du tutore gisa. Egoera horri dagokion E/R eskema 4.12 irudian agertzen da:



4.12 irudia. "Tutore izan" erlazioak IRAKASLEA eta IKASLEA entitateak lotzen ditu

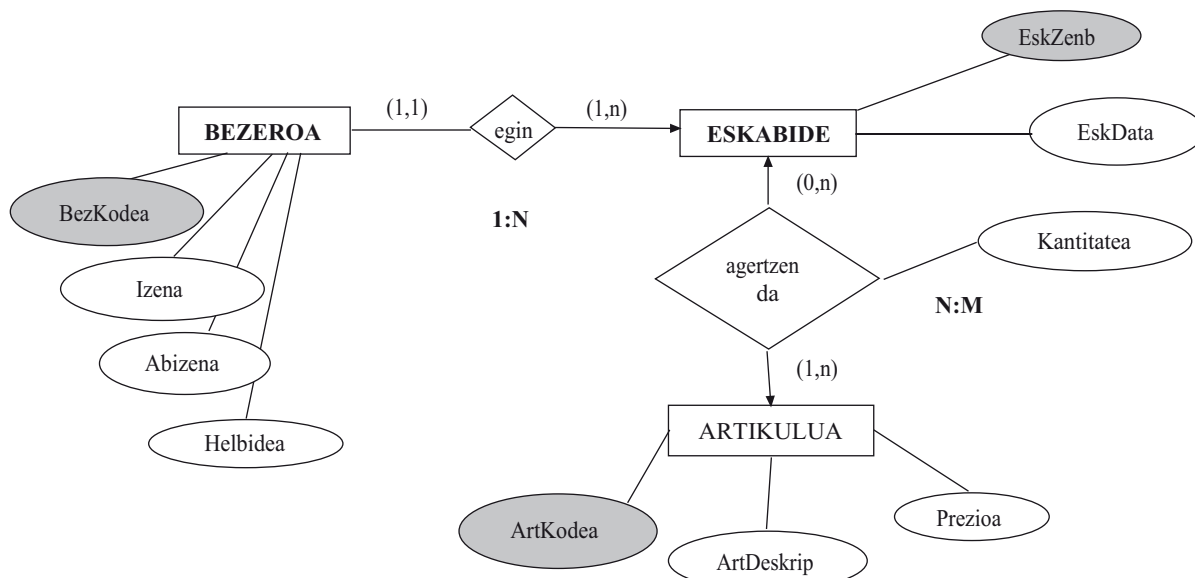
4.11 eta 4.12 irudietako E/R eskemek kardinaltasun berbera dute (1:n), baina badago bien artean ezberdintasun bat, agerraldietan hain zuzen ere. 4.11 irudian, irakasle batek klasea gutxienez ikasle bati emango dio, baina, 4.12 irudian ikusten den bezala, posible da irakasle bat tutore ez izatea, mintegiburua, ikasketaburua edo horrelako karguren bat duelako, esate baterako.

4.13 irudian agertzen den adibideko E/R eskeman, ikasle bat ikasgai askotan (eta batean gutxienez) matrikulatu ahal izango da. Gainera, ikasgai batean ikasle bat baino gehiago matrikulatu ahal dira (10 gutxienez).



4.13 irudia. 'Matrikulatu' erlazioak IKASGAIA eta IKASLEA entitateak lotzen ditu.

Ondorengo adibidean ikusten den bezala, bezero batek eskabide asko egin ditzake (1 gutxienez), baina eskabide bakoitza bezero bakar batek eginga izango da. Eskabide bakoitzeko, artikulua asko ager daitezke (1 gutxienez); aldiz, artikulua bakoitza eskabide askotan (edo batean ere ez) egon daiteke. Egoera horren E/R eskema 4.14 irudian ikus daiteke:



4.14 irudia. BEZEROA, ESKABIDEA eta ARTIKULUA entitateei buruzko E/R eskema.

- **Mendekotasuneko erlazioak (entitate eta erlazio ahulak):** entitateen artean sendoak eta ahulak daude. Orain arte ikusitako adibideetako entitateak sendoak izan dira. Entitate ahulen agerraldiak egoteko, erlazonatutako beste entitate batean agerraldiek egon behar dute, hasierako entitate hori entitate ahula izango delarik. Entitate ahulak lerro bikoitzeko laukizuzen batez adierazten dira, 4.15 irudian agertzen den moduan.



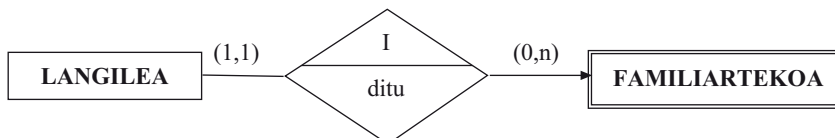
4.15 irudia. Entitate ahulak lerro bikoitzeko laukizuzenarekin adierazten dira.

Entitate ahulak bi eratakoak izan daitezke:

- Existentzian (E)
- Identifikazioan (I)

Hobeto ikusteko, adibide bana azalduko dugu. Demagun datu-base batean enpresa bateko langileei buruzko datuak gorde nahi direla. Gainera, langile horien familiartekoen buruzko informazioa ere gorde nahi da.

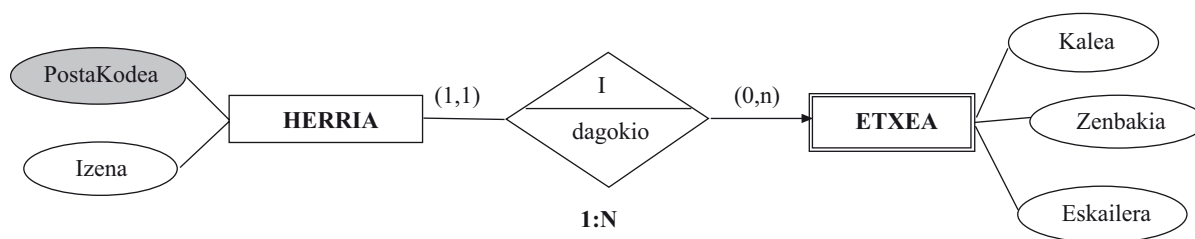
Langilea enpresatik kanporatzen badute, haren familiartekoen informazioa ez da beharrezkoa izango; hau da, enpresarentzat familiartekoen informazioa langilea enpresan dagoen bitartean baino ez da interesgarria. Aipaturiko baldintza adierazteko, -E- hizkia (existentzian) erabiltzen da, 4.16 irudian agertzen den moduan.



4.16 irudia. FAMILIARTEKOA entitate ahula da.

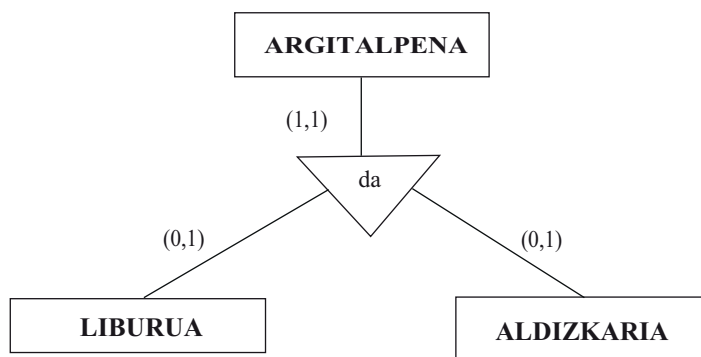
Beste adibide honetan, probintzia bateko herriei eta herri bakoitzeko etxeei buruzko datu-basea diseinatu nahi da. Herri bakoitzean etxe asko egongo dira eta etxe bat herri bakar batekoa izango da. Herria entitatearen gakoa posta-kodea izan daiteke; horrela, agerraldi bakoitza bakarka identifikatzeko aukera izango da. Etxea entitateko gakoa helbidea (kalea, zenbakia, eskailera) izan daitekeela pentsa daiteke, baina, Bizkaian adibidez, posible da etxe batek baino gehiagok helbide bera izatea (herri ezberdinetakoak izanda, noski). Hori dela eta, etxea entitateko agerraldien helbideek ez dute bakarka identifikatuko agerraldi bakoitza. Jakin beharko litzateke etxe bakoitza zein herritakoa den, hau da, ETXEA entitateko agerraldi bat bakarka identifikatzeko, HERRIA entitateko gako identifikatzailea ere beharko litzateke. Horrela, ETXEA entitatea ahula dela esan daiteke.

Aipaturiko baldintza adierazteko, -I- hizkia (identifikazioan) erabiltzen da, ETXEA entitatea identifikazioan ahula baita. 4.17 irudian ikusten da kasua.



4.17 irudia. ETXEA entitate ahula da.

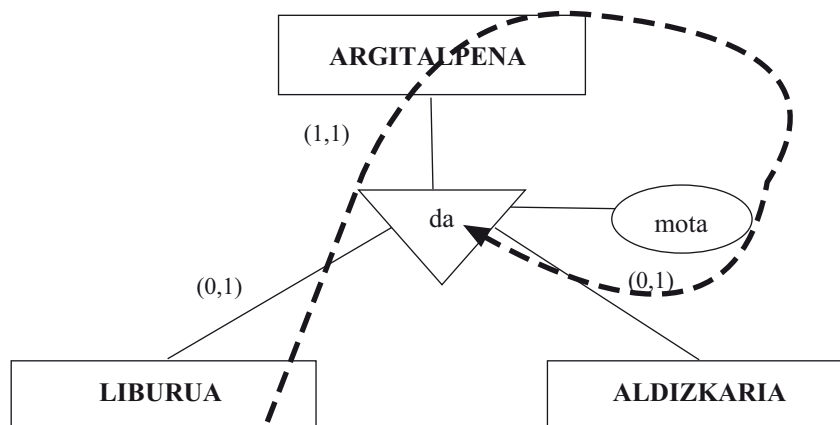
- **Entitate mota eta entitate-azpimotak (erlazio hierarkikoak):** Datuen eredu kontzeptuala lortzeko, kasu batzuetan entitate bat beste entitate-azpimota batzuetan deskonposatu beharra dago. Horretarako, 'da' etiketa erabiltzen da. Erlazioa hiruki alderantzuz adierazten da, 4.18 irudian ikusten den eran.



4.18 irudia. ARGITALPENA entitate-mota izango da eta LIBURUA eta ALDIZKARIA, entitate-azpimotak.

Liburuak eta aldizkariak argitalpen motak direnez, azpimota modura jarri dira. Gainera, kardin- altasunari dagokionez, argitalpen entitateko agerraldi bat liburu edo aldizkari bat (gehienez) izango da. LIBURUA entitateko agerraldi bakoitza beti izango da ARGITALPEN entitateko agerraldia ere. Alderantzizkoa ez da gertatzen, hau da, liburu bat argitalpena izango *da*, baina argitalpena ez *da* beti liburu bat izango.

Erlazio hierarkikoak atributuen arabera diseinatzen dira. Atributua (mota) erlazioaren ondoan jartzen da. Erlazio hierarkikoak behetik gora irakurtzen direla esan daiteke, 4.19 irudiko geziaren norabideak adierazten duen bezala (liburu bat argitalpen-mota bat *da*).

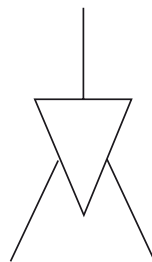


4.19 irudia. LIBURUA eta ALDIZKARIA, ARGITALPENAREN motak dira.

LIBURUA eta ALDIZKARIA entitateek ARGITALPENA entitateko atributuak heredatzen dituzte. Gako nagusia ARGITALPENA entitatean jarriko da. LIBURUA eta ALDIZKARIA entitateetan bakoitzari dagozkion atributuak baino ez dira zehaztuko. Atributu horiek normalean ezberdinak izango dira edo, atributu berdinak izateko kasuan, balio ezberdinak hartuko dituzte.

Bi irizpideren arabera, erlazio hierarkikoak hainbat motatan bana daitezke. Batetik eksklusibo edo inklusibo modura bereiz daitezke.

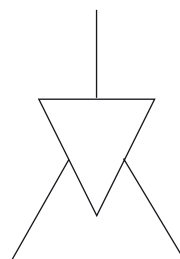
- **Esklusiboak:** entitate motako agerraldia entitate-azpimota bakar batekoa izan daitekeenean gertatzen da. Erlazioaren azpian arku batez adierazten da, 4.20 irudian erakusten den moduan.



4.20 irudia. Erlazio hierarkiko eksklusiboa

Beraz, 4.18 eta 4.19 irudietan, arkua marraztea faltako litzateke.

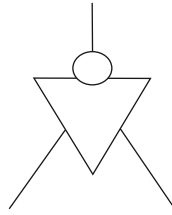
- **Inklusiboak:** entitate motako agerraldia, entitate-azpimota batekoa baino gehiagotakoa izan ahal da. Erlazioaren azpian arku barik adierazten da, 4.21 irudian ikus daitekeen moduan.



4.21 irudia. Erlazio hierarkiko inklusiboa.

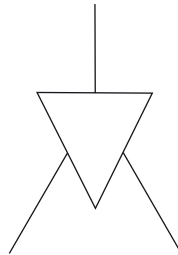
Bestetik, osoak edo partzialak izan daitezke erlazio hierarkikoak:

- **Osoak:** entitate motako agerraldia, derrigorrez, entitate-azpimota batekoa denean. Erlazioaren goialdean biribil batekin adierazten da (ikusi 4.22 irudia).



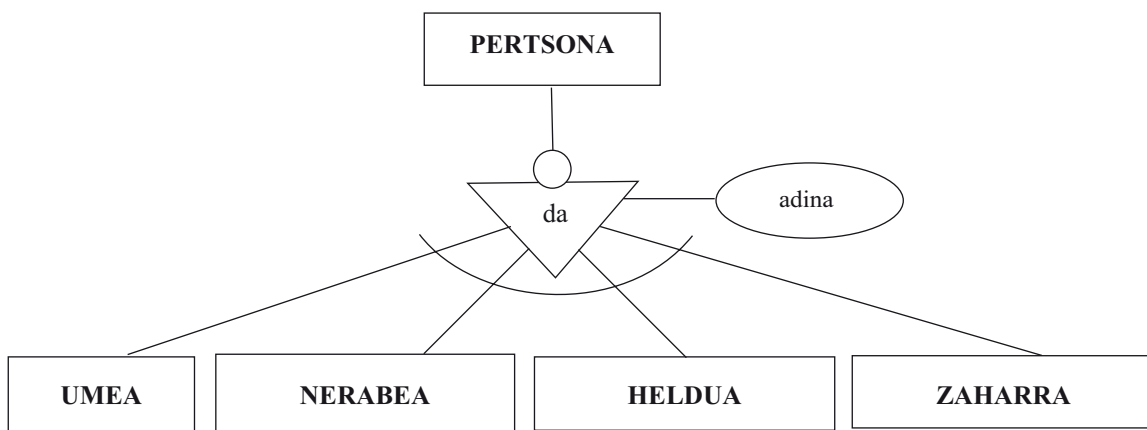
4.22 irudia. Erlazio hierarkiko osoa.

- **Partzialak:** entitate motako agerraldi batek ez du zertan entitate-azpimotako agerraldia izan. Erlazioaren goialdean biribil barik adierazten da, 4.23 irudian agertzen den moduan.



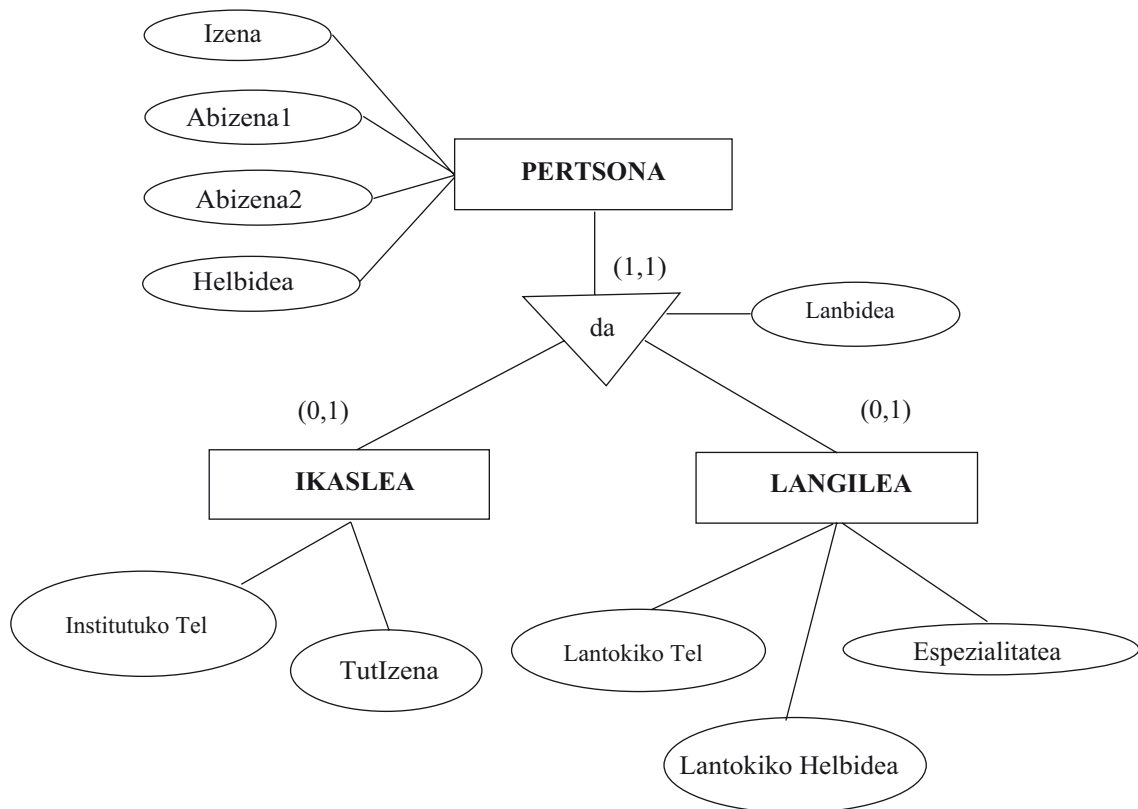
4.23 irudia. Erlazio hierarkiko partziala.

Horren guztiorren ondorioz, erlazio hierarkikoa lau motatakoa izan daiteke: eskusibo osoa, eskusibo partziala, inklusibo osoa edo inklusibo partziala. Adibidez, adinaren arabera, pertsona bat umea, nerabea, heldua edo zaharra izan daiteke. Erlazio hierarkiko hori eskusibo osoa da. Ikusi 4.24 irudia:



4.24 irudia. Erlazio hierarkiko eskusibo osoa.

Beste adibide honetan, pertsona bat, duen lanbidearen arabera, ikasle edo langile modura har daiteke (ikusi 4.25 irudia).



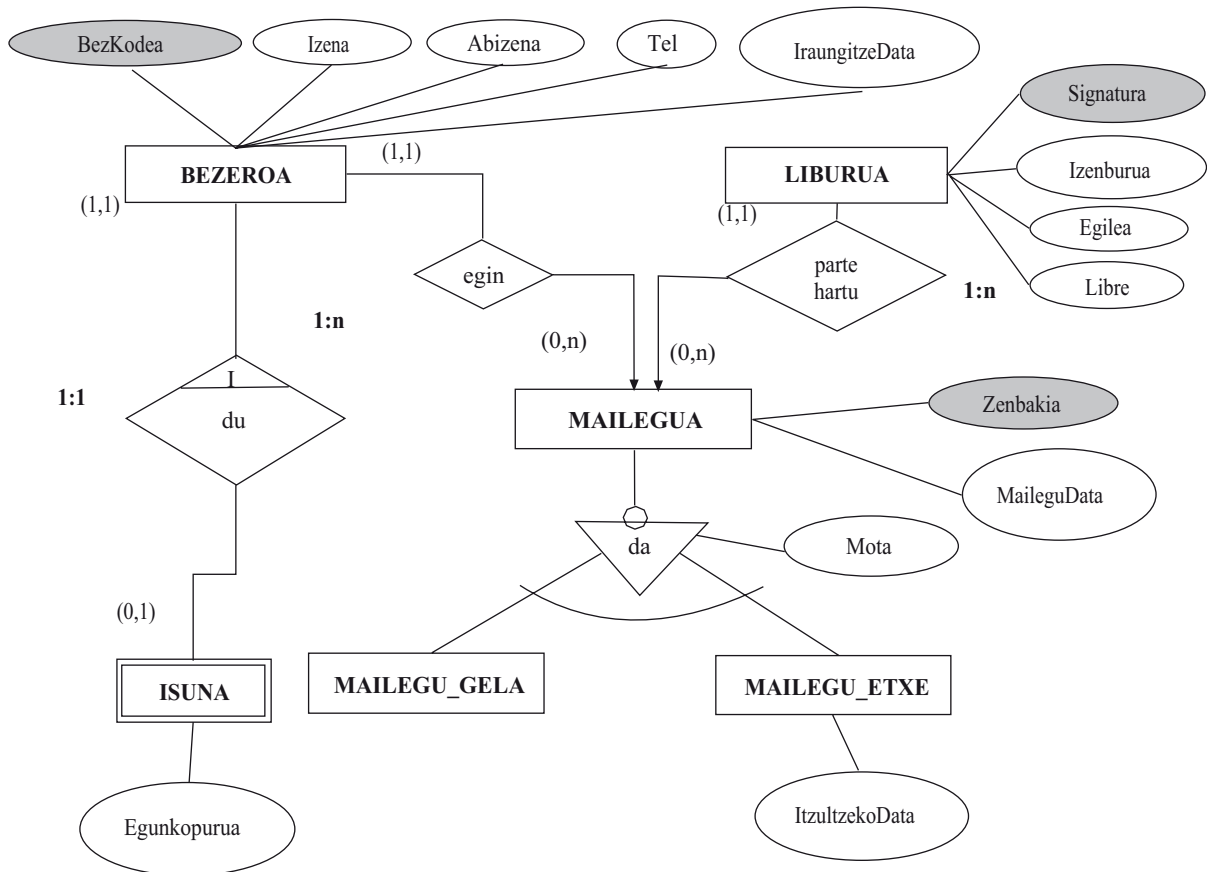
4.25 irudia. Erlazio hierarkiko inklusibo partziala.

4.25 irudian ikusten denez, lanbidearen arabera, pertsona bakoitza ikaslea edo langilea izan daiteke. Posible da PERTSONA entitateko agerraldi bat IKASLEA edo LANGILEA entitateko agerraldi bakar batean ere ez agertzea (adibidez, langabezian dagoen norbait); horregatik, erlazioa partziala izango da. Gainera, PERTSONA entitateko agerraldiren bat ikaslea zein langilea izan daiteke aldi berean. Kasu horretan, erlazioa, partziala izateaz gain, inklusiboa ere izango da.

Atal honi bukaera emateko, adibide orokorrago bat aztertuko dugu, berrikusketa gisa. Demagun liburutegi batean gordetzen diren liburuei eta bezeroek egiten dituzten maileguei buruzko datu-basea diseinatu nahi dela, ondoren adierazten diren murrizketa semantikoak kontuan hartuz:

- Bezero bakoitzak bere kodea izango du eta mailegu asko egin ditzake. Gerta daiteke bezeroren batek mailegurik ez egitea.
- Mailegu bakoitza bezero batek egina izango da.
- Mailegu bakoitzak bere zenbakia izango du eta liburu bati buruzkoa izango da.
- Bi mailegu mota daude:
 - a) Liburua liburutegiko gelan bertan irakurtzeko denean.
 - b) Liburua etxera eramatekoa denean. Kasu horretan, mailegua egiterakoan, liburua bueltatzeko epea jarriko zaio bezeroari. Epe barruan liburutegiko ekartzen ez badu, orduan isuna jarriko zaio, berandu ekarri duen egun-kopuruaren arabera; hots, 5 egun berandutuz gero, 5 eguneko isuna jarriko zaio. Bezero batek gehienez isun bat izan dezake, isuna betetzean desagertu egiten baita.


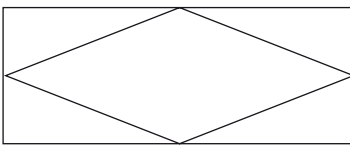
Baldintza horiei dagokien E/R eskema 4.26 irudian agertzen da.



4.26 irudia. Liburutegiko ariketari dagokion E/R eskema.

4.2.3. Martin eredua

Orain arte ikusi diren E/R eskemak Peter Chenen ereduari jarraituz diseinatu dira. Baina datuen analisia beste eredu batzuk jarraituz ere egin ahal da; Martin eredua erabiliz adibidez. Horren bidez, eskema kontzeptuala era malguagoan adierazten da. Bi ereduen arteko ezberdintasun batzuk 4.1 diagraman ikus daitezke.

E/R P. Chen	E/R Martin
Erlazioak $\left\{ \begin{array}{l} 1:1 \\ 1:N \end{array} \right.$	 Erronboak desagertzen dira, kardinaltasuna mantenduz.
Erlazioak $\left\{ \begin{array}{l} N:M \end{array} \right.$	 Entitate berria sortzen da
Atributuak agertzen dira	Ez da atributurik agertzen

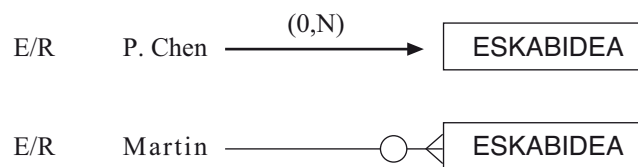
4.1 diagrama. Bi ereduen arteko ezberdintasun batzuk.

Agerraldi kopurua mantendu egiten da, baina beste era batean adierazita. Ikusi 4.2 diagrama.

Agerraldi-kopurua	
0	
1	
n	

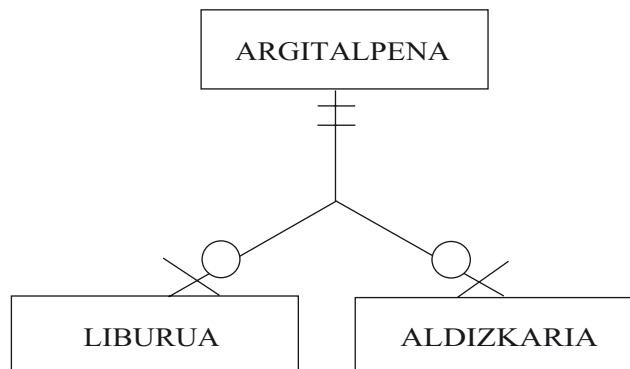
4.2 diagrama: Bi ereduaren arteko ezberdintasunak agerraldien kopuruen ikuspuntutik.

Agerraldi kopuru txikiena beti entitatetik urrunen agertuko da eta handiena, berriz, entitatetik hurbilen. Ikusi 4.3 diagrama:



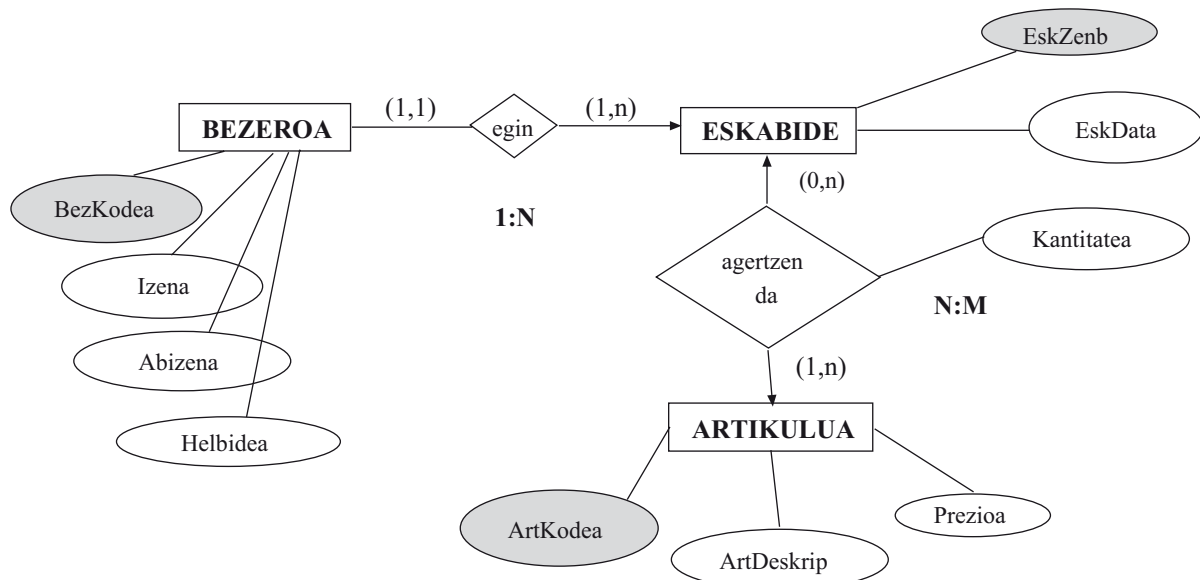
4.3 diagrama. Agerraldi-kopuru txikiena beti entitatetik urrunen agertuko da eta handiena, berriz, entitatetik hurbilen.

Erlazio hierarkikoak 4.27 irudian bezala adierazten dira:



4.27 irudia. Erlazio hierarkikoak

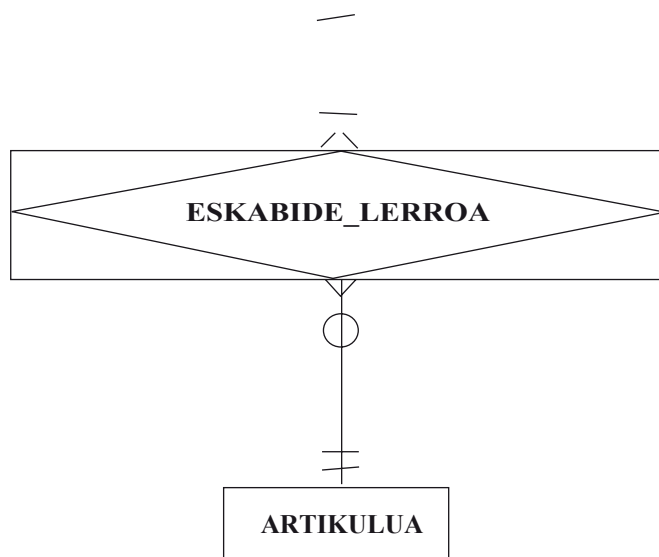
4.28 irudiko adibidean, 4.14 irudian P. Chenen eredu jarraituz lortutako E/R eskema ageri da. Martinen eredu jarraituz lortzen den datuen eskema kontzeptuala 4.29 irudian ikus daiteke.



4.28 irudia. 4.14 irudian lortutako E/R eskema



ESKABIDEA eta ARTIKULUA lotzeko ESKABIDE_LERROA erabiliko dugu. ESKABIDEAN, artikulu bakoitzeko lerro bat agertzen da, hau da, ESKABIDE batean 5 artikulu ezberdi eskatzen badira, orduan ESKABIDE horrek 5 lerro izango ditu. Posible da artikulu bat eskabide-lerro batean ere ez edo eskabide-lerro askotan agertzea



4.29 irudia. 4.14 irudian lortutako E/R eskema Martinen eredu.

4.3. DATUEN DISEINUA

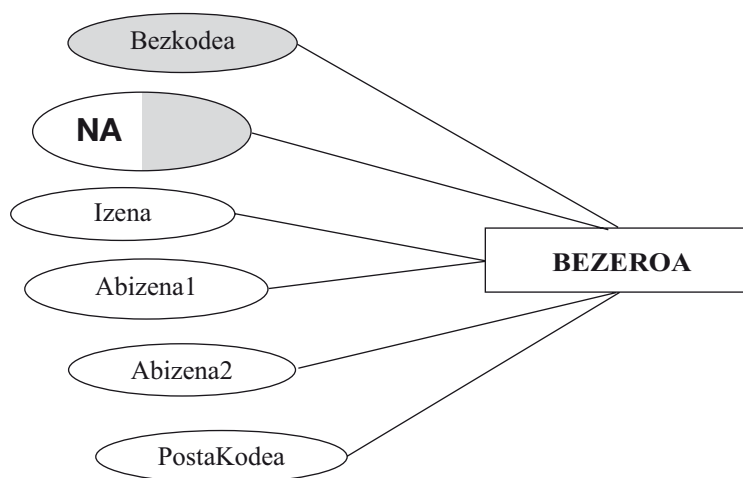
Datuen analisisan datu-basearen E/R eskema lortzen da. E/R ereduarekin aplikazio informatikoaren atal estatikoa adierazten da, aplikazioarentzat interesgarriak diren datuak identifikatzen ditu eta.

Datuen diseinuak, berriz, aurreko fasean identifikatutako datuen antolaketa ekarriko du; hau da, datu horiek ordenagailuak manipulatu ahal izateko zer-nolako egiturak erabili behar diren esango du, DBKSak ezartzen dituen murrizketak kontuan hartuz. Fase horretan, datuen eskema logikoa lortuko da.

Sortu nahi den datu-basea mota erlazionalekoa bada, orduan E/R eskematik datuen eredu erlazio-nerako eraldaketa egin beharko da. E/R ereduaren osagaiak entitateak, erlazioak eta atributuak diren bezala, eredu erlazionaleko osagai nagusi erlazioak dira. E/R ereduko erlazioak eta erlazionaleko erlazioek hainbat kontzepturi egiten diete erreferentzia. Eredu erlazionalean, agerraldiak grafikoki taulen bidez adierazten direnez, atal honetan, errakuntzak ekiditearren, eredu erlazionaleko erlazioei **erlazio-taula** deituko zaie.

4.3.1. Eredu erlazonala lortzeko arauak

- **Entitateak:** entitate bakoitza erlazio-taula bilakatuko da. 4.30 irudian ikusten den bezala, BEZEROA entitatea BEZEROA erlazio-taula bihurtu da, 'BezKodea' identifikatzaile nagusi modura mantenduz (azpimarratuta) eta 'NA' identifikatzaile alternatibo (puntuka azpimarratuta) modura. E/R eskeman atributu zirenei, eskema erlazionalean, **eremu** deitzen zaie.



BEZEROA (Bezkodea, NA, Izena, Abizena1, Abizena2, PostaKodea)

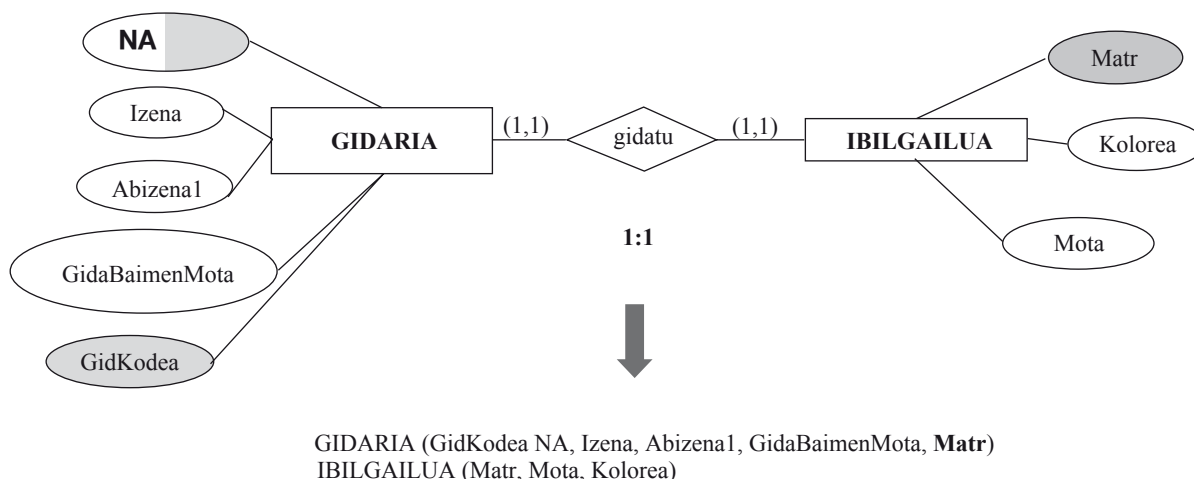
Bezkodea	NA	Izena	Abizena 1	Abizena 2	PostaKodea
<i>10</i>	<i>12345678-J</i>	<i>Iker</i>	<i>Aranda</i>	<i>Etxebarria</i>	<i>48880</i>
<i>12</i>	<i>23145689-T</i>	<i>Ane</i>	<i>Gomez</i>	<i>Garai</i>	<i>48300</i>
<i>14</i>	<i>14258369-K</i>	<i>Zihara</i>	<i>Bengoa</i>	<i>Orbe</i>	<i>48210</i>

4.30 irudia. BEZEROA entitatea BEZEROA erlazio-taula bihurtu da, eremu eta guzti.

- **Erlazioak:**

- 1:1 motako erlazioak: orokorrean ez dute erlazio-taularik sortzen. 4.31 irudian ageri den bezala, GIDARIA eta IBILGAILUA entitateak erlazio-taula bana bihurtu dira, baina erlazioa ez da erlazio-taula bilakatu. Zer gertatzen da orduan 'gidatu' erlazioarekin? IBILGAILUA erlazio-taulako identifikatzaile nagusia GIDARIA erlazio-taulara pasatzen bada, bi erlazio-taulak erlazionatuta geratuko dira. GIDARIA erlazio-taulako identifikatzaile nagusia 'GidKodea' eremua izango da egoera berrian ere eta **Matr** (hizki lodiz) eremua identifikatzaile atzerritar bihurtuko da. Horrela, GIDARIA eta IBILGAILUA erlazio-taulak erlazionatuta geratuko dira. Gidari batek zein ibilgailu gidatzen duen jakitea erraza izango da. Kontrakoa ere, ibilgailu jakin bat nork gidatzen duen jakitea, oso erraza izango da. Identifikatzaile atzerritarrek badute berezitasunik. GIDARIA erlazio-taulako datuak sartzeko, lehenengo eta behin, IBILGAILUA erlazio-taulakoak sartzten hasi beharko da, **Matr** gako atzerritarra GIDARIA erlazio-taulan

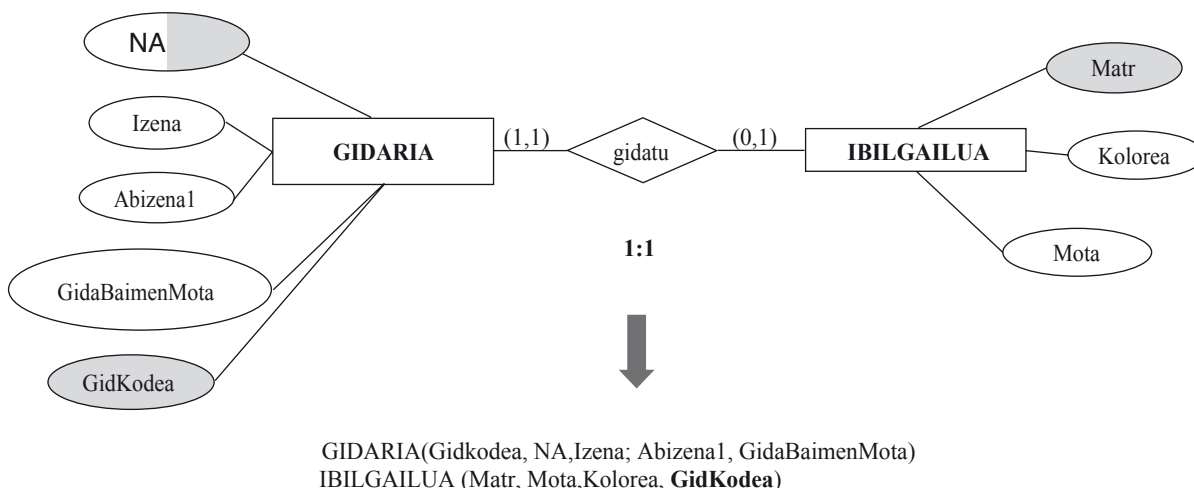
baitago. Hau da, **Matr** gako atzerritarrean datua sartu aurretik, datu horrek **Matr** eremua gako nagusi den erlazio-taulan sartuta egon beharko du (ikusi 4.31 irudia).



4.31 irudia. 1:1 motako erlazioek ez dute erlazio-taula berririk sortzen

1:1 motako erlazioetan alderantzizkoa ere gerta daiteke; hau da, GIDARIA erlazio-taulako identifikatzaile nagusia IBILGAILUA erlazio-taulara pasatzea. Horrela, IBILGAILUA erlazio-taulako identifikatzaile nagusia 'Matr' eremua izango da eta 'GidKodea' eremua, gako atzerritarra. Irizpide bata edo bestea gauzatu da, datuetara egin beharreko kontsulten antolaketaren arabera.

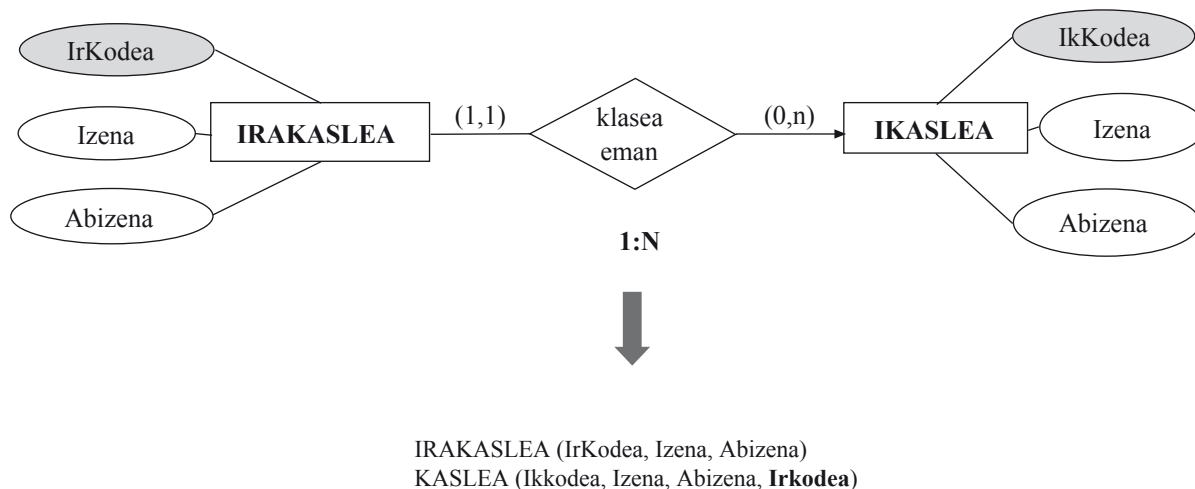
Entitate batean gutxieneko agerraldia 0 denean, kasu berezi baten aurrean gaude. Eman dezagun posible dela gidari batek ibilgailurik bat ere ez gidatzea (adibidez, bulegoko lanetan jarduten duelako). Balio huts horiek saihesteko, gutxieneko agerraldia 0 dagokion erlazio-taulak beste erlazio-taulako identifikatzaile nagusia atzerritar gisa hartu beharko du. 4.32 irudian ikusten den bezala, GIDARIA erlazio-taulako identifikatzaile nagusia IBILGAILUA erlazio-taulara pasatzen da; bestela, GIDARIA erlazio-taulako **Matr** eremua gidari batzuentzat daturik gabe (balio hutsak) geratuko da (ikusi 4.32 irudia).



4.32 irudia. 1:1 motako erlazioa, entitate batean gutxieneko agerraldia 0 denean.

b) 1:N motako erlazioak. Orokorrean ez dute erlazio-taularik sortzen. Gehieneko agerraldia 1 den erlazio-taularen identifikatzaile nagusia beste erlazio-taulara pasatuko da identifikatzaile

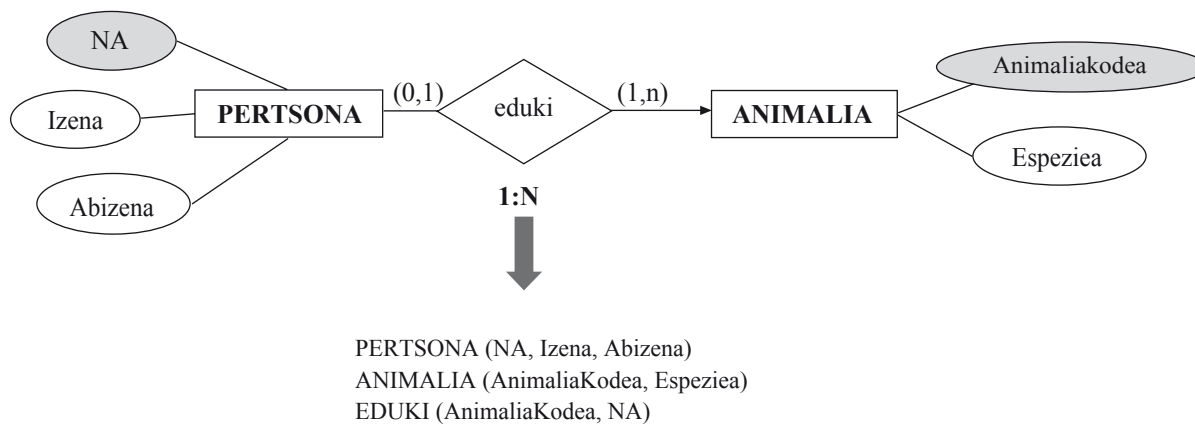
atzerritar gisa (lodiz). Identifikatzaile atzerritarrek erlazio-geziaren norabidea darama, hau da, IRAKASLEA erlazio-taulatik IKASLEA erlazio-tulara. 4.33 irudian ikusten den bezalaxe, IRAKASLEA erlazio-taulako identifikatzaile nagusia IKASLEA erlazio-tulara pasatu da, gako atzerritar modura. Horrela, erlazio-taula biak erlazionatuta geratuko dira.



4.33 irudia. 1:N motako erlazioa.

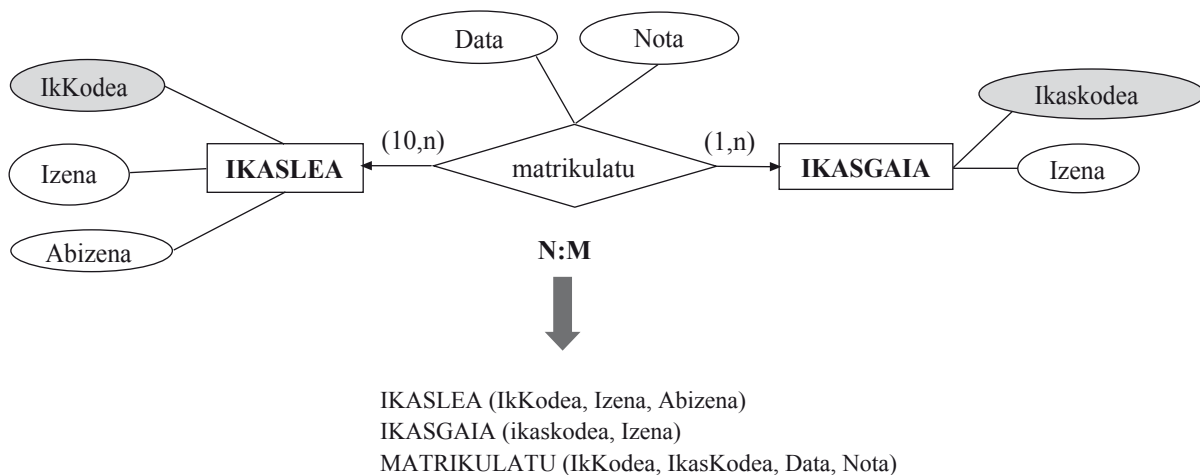
Gehieneko agerraldia n den entitatean, gutxieneko agerraldia 0 bada, orduan gako atzerritarra tokiz aldatuko da, 4.33 irudian agertzen den moduan. Beraz, ez da ezer aldatzen.

Gehieneko agerraldia 1 den entitatean, gutxieneko agerraldia 0 bada, orduan erlazio-taula berria sortu behar da. Erlazio-taula berri horretako eremuak, erlazionatutako entitate bietako identifikatzaile nagusiekin sortzen dira, baina erlazio-taula berriko identifikatzaile nagusia gehieneko agerraldia n deneko entitatearen identifikatzaile nagusia izango da (ikusi 4.34 irudia).



4.34 irudia. 1:N motako erlazioa, gehieneko agerraldia 1 deneko entitatean, gutxieneko agerraldia 0 denean.

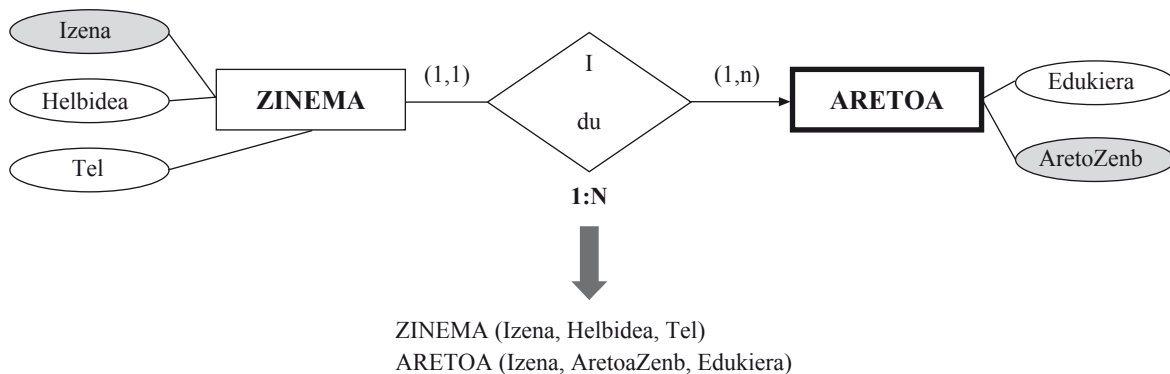
c) N:M motako erlazioak: erlazio-taula berria sortzen da. Erlazionatutako entitateen gakoekin osatzen da erlazio-taula berri horren gakoak. Erlazioak atributuak izan baditu, orduan erlazio-taula berri horretara pasatuko dira (ikusi 4.35 irudia).



4.35 irudia. N:M motako erlazioetan erlazio-taula berri bat sortzen da.

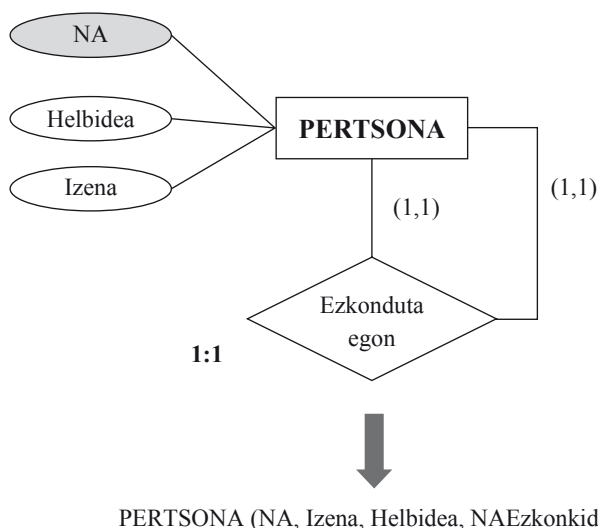
Entitateren batean gutxienero 0 bada, berdin jokatuko da.

d) Mendekotasuneko erlazioak: normalean, 1:1 eta 1:N motakoak direnez, ez dute erlazio-taula berririk sortzen, 4.36 irudian ikusten den moduan.



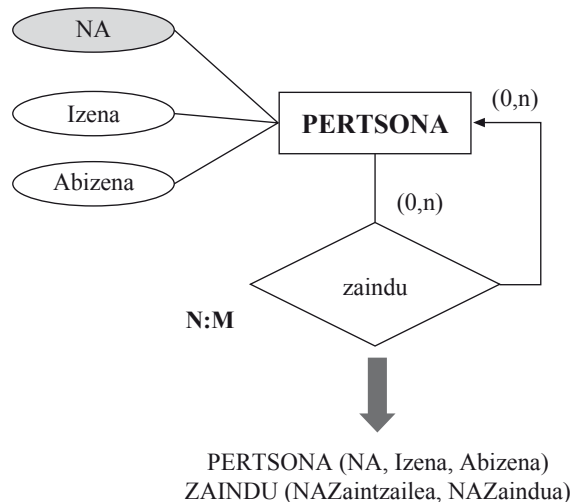
4.36 irudia. Mendekotasuneko erlazioak ez du erlazio-taularik sortzen.

e) Erlazio errekursiboak: Kardinaltasunaren arabera, erlazio-taula sortuko da edo ez. Adibidez, 1:1 eta 1:N kardinaltasuneko erlazioa bada, ez da erlazio-taularik sortuko (ikusi 4.37 irudia).



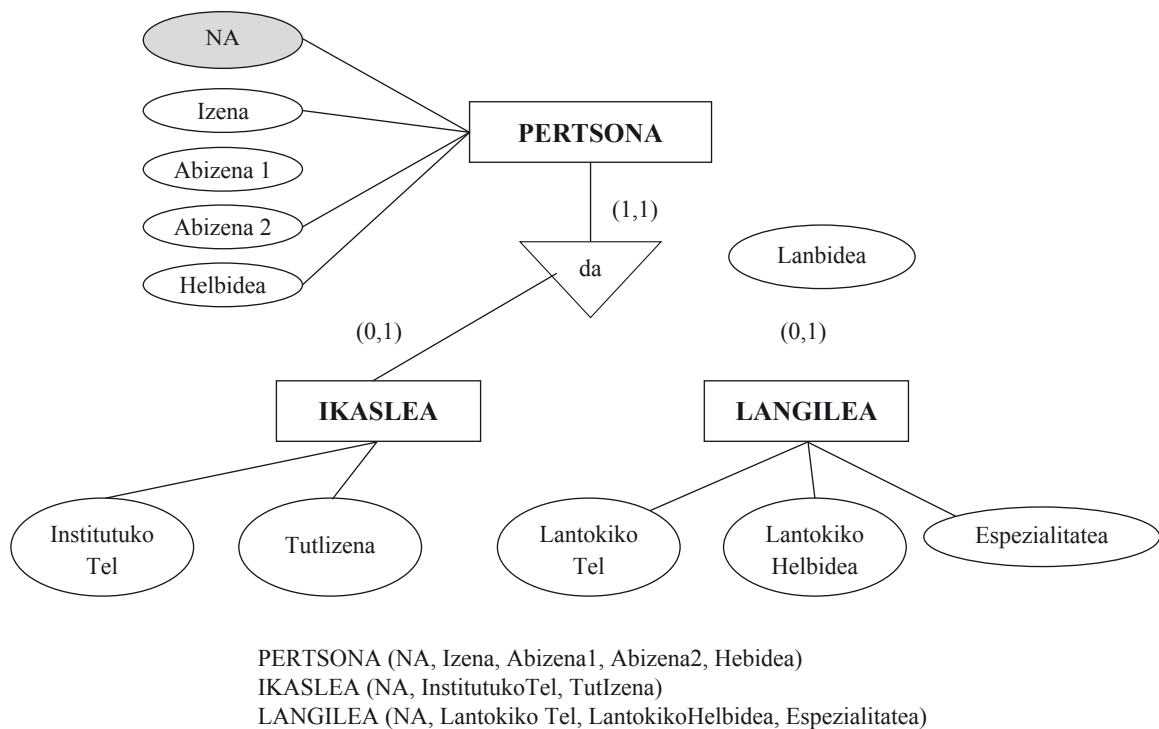
4.37 irudia. Erlazio errekursiboak, 1:1 kardinaltasunarekin.

N:M motakoa bada, orduan erlazio-taula berria sortuko da (ikusi 4.38 irudia).



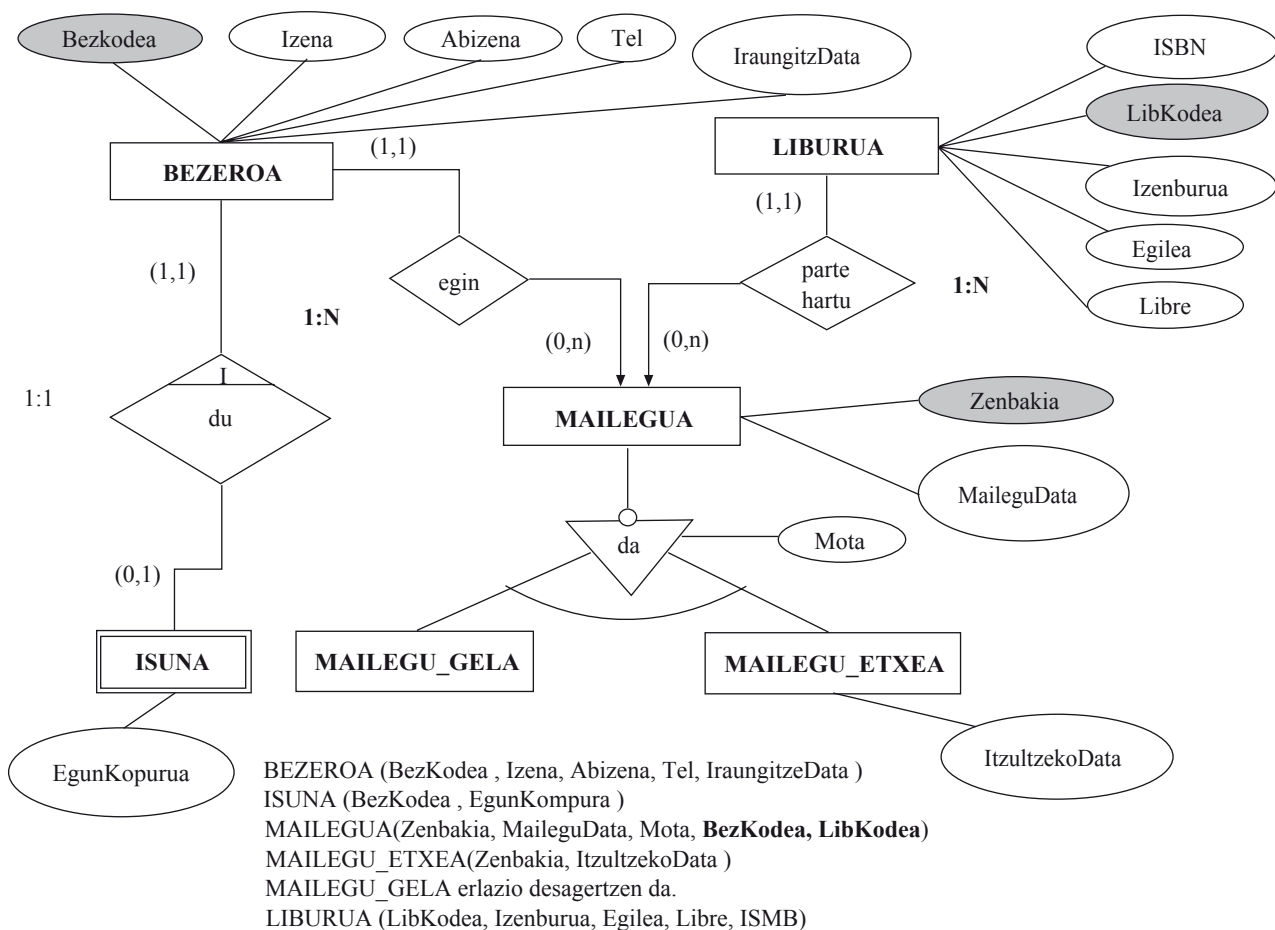
4.38 irudia. Erlazio errekurtsiboak, $n:m$ kardinaltasunarekin.

- f) Erlazio hierarkikoak: Entitate motarentzako erlazio-taula sortuko da (4.40 irudian PERTSONA erlazio-taula). Entitate-azpimotek, atributurik ez badute eta beste entitate batekin erlazionatu barik badaude, orduan desagertu egingo dira (ikusi 4.41 irudiko MAILEGUGELA). Bestalde, entitate-azpimotek atributuak badituzte, orduan erlazio-taula bat sortuko da eta entitate-motaren gako heredatuko dute (4.40 irudian IKASLEA eta LANGILEA erlazio-taula). *da* erlazioa eskusiboa bada, orduan erlazioaren atributua entitate-motari dagokion erlazio-taulara pasatuko da (ikusi 4.40 irudia).



4.39 irudia. Erlazio hierarkikoa.

Adibide orokor modura, eskema erlazionalera pasatuko dugu 4.26 irudian agertzen den E/R eskema (ikusi 4.41 irudia).



4.40 irudia. Liburutegi bateko liburuei eta bezeroek egiten dituzten maileguei buruzko eskema erlazionala.

4.4. NORMALIZAZIO PROZESUA

Normalizazioa datu-eredu erlazionalen diseinu egokia lortzeko erabiltzen da. Eredu erlazionalaren diseinu egoki batez ari garenez gero, diseinu egokitzat onartuko da prozesuak finkatzen dituen forma normal guztiak edo gutxienez, lehenengo hirurak, betetzen baditu. Forma normalen kontzeptua, horiek kontuan hartzen dituzten murrizketak eta forma normal batetik bestera pasatzeko pausoak aztertuko dira atal honetan.

Normalizazioa diseinuari dagokion prozesua denez, eredu erlazionalan datu-basearen diseinua bi eratara lor daitekeela aipatu behar da:

- Errealitatearen analisia egin ondoren, eskema erlazionala zuzen lortzea. Horretara eginez gero, ez da bitarteko E/R ereduak lortuko.
- Diseinua bi pausotan lortzea: errealitatearen analisiaren ostean. E/R ereduak lortzea eta, hortik, eredu erlazionala. Orain arte gai honetan era horixe landu dugu.

Era batera edo bestera eredu erlazionala lortu ondoren, diseinua egokia edo desagokoa gerta daiteke, honako arazo hauek agertzen badira:

- Erreduantzia: Erlazio batean edo datu-baseko erlazio desberdinetan datuak errepikatuta agertzen direnean erreduantzia dagoela esaten da. Eguneratzeko arazoak ekar litzake, jakina, eta espazioa alferrik galtzea.

- Txertatzean, ezabatzean eta aldatetak egitean anomaliak egitea: Erlazio bateko atributuen elkarrekiko erlazioa dela eta, baliteke tuplo bat ezabatzerakoan galdu nahi ez den informazio-multzoa galtzea edo atributuren bat aldatzean erlazioak gordetzen duen informazioa sendotasun barik geratzea.

Arazo horiek agertzen dituen eredu erlazional baten adibidea 4.2 taulan ikusten da.

ESKATZEN_DU (Bezeroa, Artikulua, Kopurua, Prezioa, Herria, Telefonoa, PK)

Bezeroa	Artikulua	Kopurua	Prezioa	Herria	Telefonoa	PK
<i>Bez1</i>	<i>Art1</i>	22	200	MARKINA	946166230, 660562389	48270
<i>Bez2</i>	<i>Art1</i>	23	200	GERNIKA	946851232, 659869523	48563
<i>Bez3</i>	<i>Art1</i>	55	200	EIBAR	943625152	45695
<i>Bez1</i>	<i>Art2</i>	10	300	MARKINA	946166230, 660562389	48270
<i>Bez2</i>	<i>Art2</i>	12	300	GERNIKA	946851232, 659869523	48563
<i>Bez4</i>	<i>Art3</i>	55	400	BERMEO	946836548, 660395286	48956
<i>Bez3</i>	<i>Art3</i>	50	400	EIBAR	943625152	45695

4.2 taula. Bezeroen eta bakoitzak erositako artikuluen informazioa batzen duen eredu erlazionala ESKATZEN_DU erlazioa osatu ondoren.

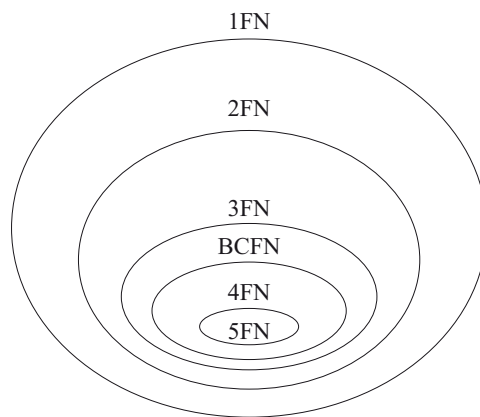
4.2. taulan honako arazo hauek agertzen dira:

- Erredundantzia: herria, telefonoa eta posta-kodea bezero berari dagokion tuplo bakoitzean errepikatzen dira. Prezioa ere, artikulua berari dagokion tuplo guztietan errepikatzen da.
- Anomaliak aldatzean: artikulua baten prezioa aldatu nahi izanez gero, artikulua horri dagozkion tuplo guztietan aldatu behar da prezioa. Tuploren batean aldatuta egiten ez bada, erlazioa sendotasun gabe geratuko da.
- Anomaliak txertatzean: artikulua berri baten datuak gorde nahi izanez gero erlazioan, artikulua hori ez bada inork eskatzen, ezinezkoa da berari buruzko informazioa gordetzea, gako nagusia osatzen duten atributuek balio hutsik ezin hartu baitute.
- Anomaliak ezabatzean: bezero bat ezabatu nahi izanez gero, berari dagozkion tuplo guztiak ezabatu behar dira; ondorioz, artikuluei buruzko informazioa galduko da.

Arazo horiek guztiak daudenez, 4.2 taulako eredu erlazionala gaizki diseinatuta dagoela egiaztatzen da. Normalizazioa, arazo horiek guztiak konponduko dituen prozesua da. Azkenean, diseinu egokia aurkeztuko duen eredu erlazional normalizatu berria lortuko da.

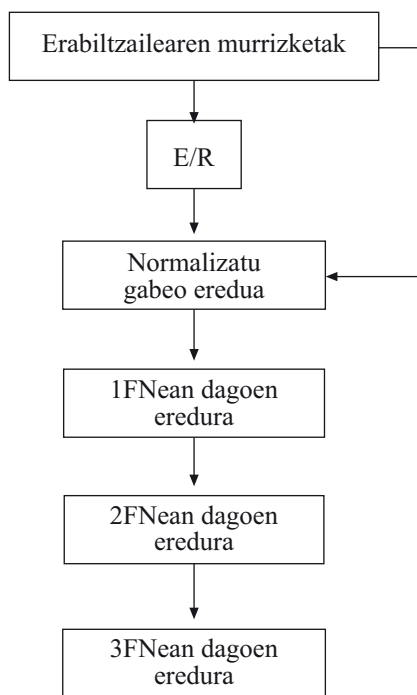
Eredu erlazional normalizatuak hainbat forma normal bete behar ditu, nahiz eta forma normal bakoitza murrizketa batzuek definitzen den.

Eredu erlazionala 5. forma normalera (FN) arte normaliza daiteke. 5. forma normalean dagoen eredu erlazionalak beste forma normal guztiak ere betetzen ditu, 4.42 irudian adierazten den bezala.



4.41 irudia. Forma normalen arteko erlazioa

Normalizazioan 5 forma normal ezagutzen diren arren, 3. forma normala betetzen duen eredu erlazionala eredu normalizatua dela esaten da. Hori horrela izanik, gai honetan 4FNeraino dauden forma normalak jorratuko dira, 5FNa argitalpen honen helburutik kanpo baitago. Prozesu horretan, hasierako pausoa, eredu 1FNean jartzea da, ondoren 2FNera pasatzea, gero 3FNera eta, horrela, nahi izanez gero, 4FNeraino. Adibide gisa, 4.42 irudia proposatzen da.



4.42 irudia. Normalizazio-prozesua.

Prozesua era sekuentzialean, pauso batetik bestera pasatuz, egin behar dela argitu ondoren, forma normal bakoitza definitzen duten ezaugarriak eta murrizketak zehaztuko ditugu:

- a) 1. forma normala (1FN): erlazio batean atributu guztiak atomikoak direnean, 1FN betetzen da, hau da, erlazioa 1. forma normalean jartzeko, multzoak ezabatu egin behar dira. Adibidez, 4.2 taulako erlazioan, “Telefonoa” atributuan multzoak antzematen dira bezero baten telefono finkoa eta eskukoa gorde nahi direnean. Multzo horiek ezabatu ondoren, eredu 1FNean egongo da, 4.3 taulan adierazten den moduan.

Bezeroa	Artikulua	Kopurua	Prezioa	Herria	TelefonoFinkoa	EskukoTelefonoa	PostaKodea
Bez1	Art1	22	200	MARKINA	946166230	660562389	48270
Bez2	Art1	23	200	GERNIKA	946851232	659869523	48563
Bez3	Art1	55	200	EIBAR	943625152		45695
Bez1	Art2	10	300	MARKINA	946166230	660562389	48270
Bez2	Art2	12	300	GERNIKA	946851232	659869523	48563
Bez4	Art3	55	400	BERMEO	946836548	660395286	48956
Bez3	Art3	50	400	EIBAR	943625152		45695

4.3 taula. ESKATZEN_DU eredu erlazionala 1FNean (multzorik gabeko erlazioa).

- b) 2. forma normala (2FN): Eredu erlazionala 2FNean egongo da (eta, beraz, 1FNean ere bai) gakoko partaide ez den atributu bakoitzak gakoarekiko erabateko mendekotasun funtzionala badauka, hau da, gakoarekiko mendekotasun funtzional partzialik ez badago. Aipatutakoa ulertzeko, erabateko mendekotasun funtzionalaren kontzeptua azalduko dugu:

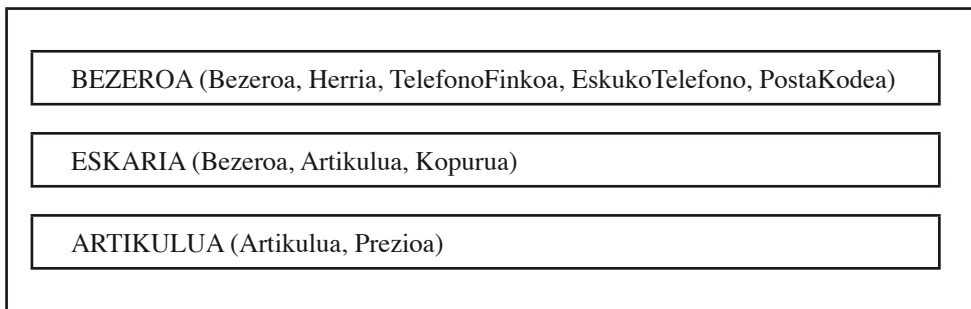
Erabateko mendekotasun funtzionala ($x \Rightarrow y$): y atributuak x -rekiko erabateko mendekotasun funtzionala du, x -rekiko mendekotasun funtzionala daukanean x -ren azpimultzoz batekiko mendekotasunik eduki gabe. Beste modu batera esanda, x -ren atributu guztiak beharrezkoak dira y atributua finkatzeko, eta ez bakarrik x -ren atributu batzuk.

Erlazioa 2FNean jartzeko, gakoarekiko mendekotasun partzialak ($x \textcircled{R} y$) ezabatu egin behar dira. Adibide gisa 4.3 taulako ereduaren agertzen diren mendekotasunak 4.1 testu-taulan batu dira:

Bezeroa \rightarrow Herria
Mendekotasun partziala: 'Herria' atributua 'Bezeroa' atributuaren mendekoa da eta ez gako nagusia osatzen duten 'Bezeroa' eta 'Artikulua' atributu bien mendekoa.
Bezeroa \rightarrow TelefonoFinkoa
Mendekotasun partziala: 'TelefonoFinkoa' atributua 'Bezeroa' atributuaren mendekoa da eta ez gako nagusia osatzen duten 'Bezeroa' eta 'Artikulua' atributu bien mendekoa.
Bezeroa \rightarrow EskukoTelefonoa
Mendekotasun partziala: 'EskukoTelefonoa' atributua 'Bezeroa' atributuaren mendekoa da eta ez gako nagusia osatzen duten 'Bezeroa' eta 'Artikulua' atributu bien mendekoa.
Bezeroa \rightarrow PostaKodea
'PostaKodea' atributua 'Bezeroa' atributuaren mendekoa da eta ez gako nagusia osatzen duten 'Bezeroa' eta 'Artikulua' atributu bien mendekoa.
Artikulua \rightarrow Prezioa
'Prezioa' atributua 'Artikulua' atributuaren mendekoa da eta ez gako nagusia osatzen duten 'Bezeroa' eta 'Artikulua' atributu bien mendekoa.
Bezeroa, Artikulua \rightarrow Kopurua
Erabateko mendekotasuna: 'Kopurua' atributua 'Bezeroa' eta 'Artikulua' atributuaren mendekoa da; gako nagusia osatzen duten atributu bien mendekoa, beraz.

4.1 testu-taula. Mendekotasun funtzionalak.

‘4.1 testu-taula’ irudian aztertutako mendekotasun partzialak ezabatzeko, erlazio berriak gehituko dira eredu erlazionalera: erabateko mendekotasuna bermatzen duten erlazioak hain zuzen. Eredu erlazional berria 4.2 testu-taulan agertzen den moduan geratuko da 2FNean:

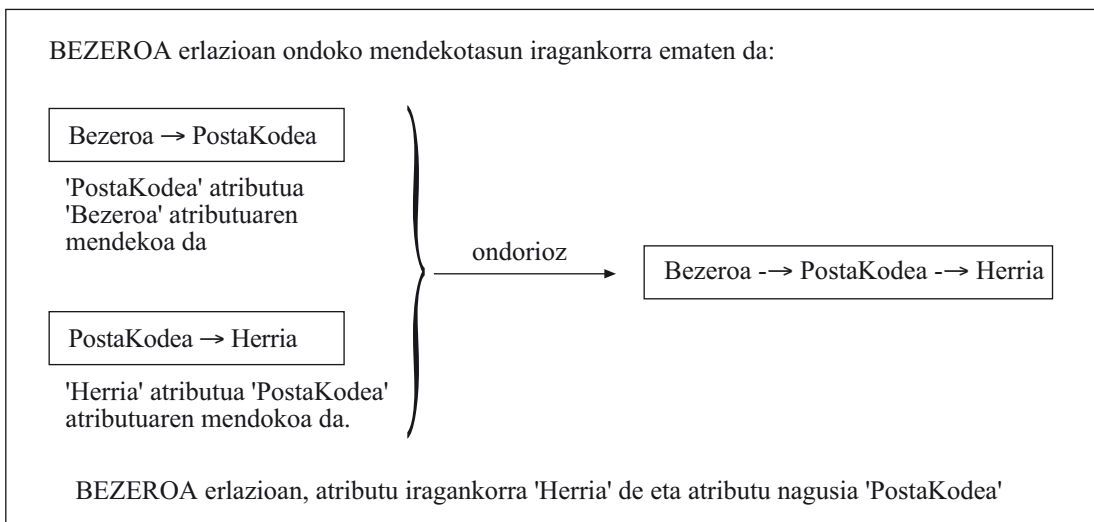


4.2 testu-taula. ESKATZEN_DU eredu erlazional berria

c) 3. forma normala (3FN): erlazioa 3FNean egongo da (eta, beraz, 2FNean ere bai) gakoko partaide ez den edozein atributu zuzen-zuzen gakoak zehazten badu, eta ez besteren batek; hau da, mendekotasun iragankorrik ez badago. Mendekotasun iragankorrek ezabatzeko, honako pauso hauek jarraitu behar dira:

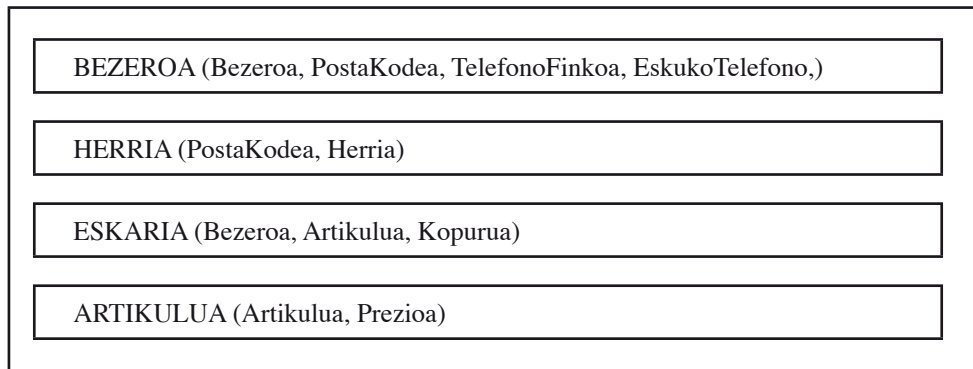
- Gakoarekin eta iragankorrek ez diren atributuekin erlazio bat osatuko da.
- Erlazio berria sortuko da atributu iragankorrek eta atributu nagusiarekin (azken hori erlazio berriko gakoa izango da). Aipatutako atributu nagusi horren bitartez mantenduko da iragankortasuna.

Eredu erlazionala 3FNean jartzeko, mendekotasun iragankorrek ezabatu egin behar dira. Adibide gisa, BEZEROA erlazioa 3FNean jartzeko, mendekotasun iragankorrek 4.3 testu-taulan islatzen dira:



4.3 testu-taulan. ESKATZEN_DU eredu erlazionala mendekotasun iragankorrek

Mendekotasun iragankorrek aurkitu ondoren, aipatutako pausoak jarraituz, 3FNean egongo den eredu erlazional berria lortuko da, 4.4 testu-taulan azaltzen den bezala.



4.4 testu-taulan. ESKATZEN_DU eredu erlazional berria 3FNean

d) BOYCE-CODDEN forma normala (BCFN): E erlazioan, gakoa A eta B atributuek osatzen badute eta 4.4 diagramako egoera ematen bada, erlazioa ez dagoela BCFNean esaten da.

$E(A, B, C)$

$A, B \rightarrow C$

$C \rightarrow B$

4.4 diagrama. Erlazioa BCFNean egoteko gertatu behar ez diren murrizketak.

Aipatutako mendekotasun funtzionalak badaude, erlazioa, ez dagoela BCFNean esaten da. BCFNean jartzeko, aldaketa batzuk egin behar dira:

- Gakoko partaide den atributu independentearekin (A) eta gakoko partaide ez diren atributu guztiekin (C) E1 erlazioa sortuko da.
- Gakoko beste atributuekin (B) eta gakoko atributu hori determinatzen duen bigarren mailako atributuarekin (C) beste erlazio bat osatuko da, eta azken hori 2. taulako gakoa izango da.

4.4 diagramako adibidea deskonposatu ondoren, erlazio bik osatuta geratuko da BCFNa betetzen duen erlazioa: E1(A,C) eta E2(C,B)

Adibide gisa, 4.4 taula azaltzen da:

SOKATIRA (Tiralaria, Taldea, Entrenatzailea)

Tiralaria	Taldea	Entrenatzailea
<i>Barandika</i>	<i>Abadiño A</i>	<i>Edu Mendizabal</i>
<i>Barandika</i>	<i>Abadiño B</i>	<i>Xabier Ardanza</i>
<i>Lizundia</i>	<i>Abadiño A</i>	<i>Asier Ugarte</i>
<i>Lizundia</i>	<i>Abadiño B</i>	<i>Xabier Ardanza</i>
<i>Mendi</i>	<i>Arrizku</i>	<i>Gorka Zubiaurre</i>
<i>Barandika</i>	<i>Sokarri</i>	<i>Gorka Landaluze</i>

4.4 taula. SOKATIRA erlazioa.

Aipatutako erlazioan, erabiltzaileak jarritako honako murrizketa semantiko hauek bete behar dira:

1. Izen berarekin 2 tiralari egon daitezke 2 talde ezberdinetan; beraz, ez da beteko
Tiralaria \rightarrow Taldea.
2. Talde batek, entrenatzaile bat baino gehiago izan ditzake, baina tiralari bati, talde batean, entrenatzaile bakarra dagokio; beraz, $A, B \rightarrow C$ betetzen da.
3. Entrenatzaile bakoitza talde bati baino ez dagokio; beraz, Entrenatzailea \rightarrow Taldea betetzen da.
4. Ez dira bi talde izen berarekin egongo.

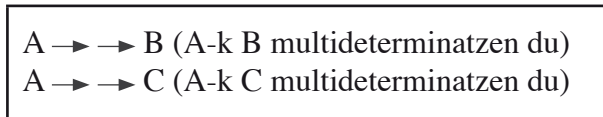
2. eta 3. murrizketa semantikoak direla eta, erlazioa ez dago BCFNean, $C \rightarrow B$ eta $A, B \rightarrow C$ mendekotasunak ematen baitira, C atributua entrenatzailea, B atributua Taldea eta A atributua Tiralaria izanik (ikusi 4.5 diagrama).

Tiralaria, Taldea \rightarrow Entrenatzailea
Entrenatzailea \rightarrow Taldea

Erlazio hori BCFNean jartzeko, arestian aipatutako pausoak jarraitu ondoren, honelaxe geratuko da:

TALDEA 1 (Tiralaria, Entrenatzailea)
TALDEA 2 (Entrenatzailea, Taldea)

- e) 4. forma normala (4FN): Eredu erlazionalak ez du behar 3FNetik gorako forma normalik. Dena den, 4FNa tresna baliagarria da datuen analisirako, bai eguneratze-anomalien aurrean, zein murrizketa batzuk jartzeko orduan. 4FNa aztertze, mendekotasun multibaloratuaren kontzeptua aurreratuko dugu (ikusi 4.5 diagrama):



4.5 diagrama. Mendekotasun multibaloratuaren kontzeptua.

Mendekotasun multibaloratu dagoela esaten da gakoak bi multzo errepikari eta elkarrekiko independenteak determinatzen baditu. Adibidea, 4.5 taulan agertzen da.

IRAKASLE (IrKodea, Taldea, Kirola)

IrKodea	Taldea	Kirola
<i>100</i>	<i>2B</i>	<i>Tenisa</i>
<i>100</i>	<i>3D</i>	<i>Futbola</i>
<i>200</i>	<i>4C</i>	<i>Surfa</i>
<i>200</i>	<i>4C</i>	<i>Tenisa</i>

4.5 taula. IRAKASLE erlazioa.

Erlazio horretan agerikoak dira erredundantziak, baina 2FNaren edota 3FNaren ezaugarri diren mendekotasun funtzionalak agertzen ez direnez, ezin da erlazioa deskonposatu. Erredundantzia horiek anomaliak sorraraziko dituzte eguneratze-prozesuetan, ikasle batek kirolik egiten ez badu, adibidez, ezin baita datu-basean agertu (gako nagusia osatzen duten atributuek ezin dute balio hutsik

izan). Hori dela eta, mendekotasun funtzionalak ez dira aski diseinu egokia lortzeko; kasu honetan, adibidez, ez da ‘Taldea’ eta ‘Kirola’ atributuen arteko independentziarik jaso. Erlazio horretan, bi mendekotasun multibaloratu daude (ikusi 4.6 diagrama):



4.6 diagrama. Mendekotasun multibaloratuaren adibidea.

Erlazioa 4FNean dagoela esaten da mendekotasun multibaloraturik ez bada agertzen; beraz, erlazio hori 4FNean jartzeko, honela geratuko da:

IRAKASLE 1 (IrKodea, Taldea)
 IRAKASLE 2 (IrKodea, Kirola)

4.4.1. Adibideak

Forma normal guztien berezitasunak eta bete beharreko murrizketak aztertu ondoren, datu-base erlazionalen diseinuari dagokionez, 3FNa betetzen duen datu-eredu erlazionala egoki diseinatutako eredutzat onartzen da. Normalizazio-prozesuak hasierako eredu behar du, E/R batetik edo mundu errealeko analisi zuzenetik lortua. Ondoren, eredu hori aldatuz joango da forma normal bakoitzak zehaztutako murrizketak betetzeko, azken eredu normalizatua lortu arte. Kasu bakoitzeko, adibide bana aurkeztuko dugu jarraian:

- a) Errealitatetik E/R eredura eta hortik eredu erlazionalera. Adibide gisa, ospitale baterako datu-basearen diseinua lortu nahi da. Bertan, gaixoen sarrerak erregistratu nahi dira, sarrereguna, gela, ospitaleratuta egongo den egun-kopurua eta berataz arduratuko diren medikuen datuekin.

Horrez gain, ospitaleak dituen erizainei buruzko informazioa ere gorde nahi da, bakoitzak daukan zeregina erregistratuz. Ospitaleak solairu ugari dituzenez, bakoitza arlo jakin baterako dela (psikologia, haur-medikuntza..) esan digute, solairu bakoitzean hainbat erizain dabiltzala eta erizain bat hainbat solairutan ari daitekeela, zeregin berdinean edo ezberdinarekin. Aritzen den zereginaren kodea eta kode horri dagokion deskripzioa oso interesgarriak dira.

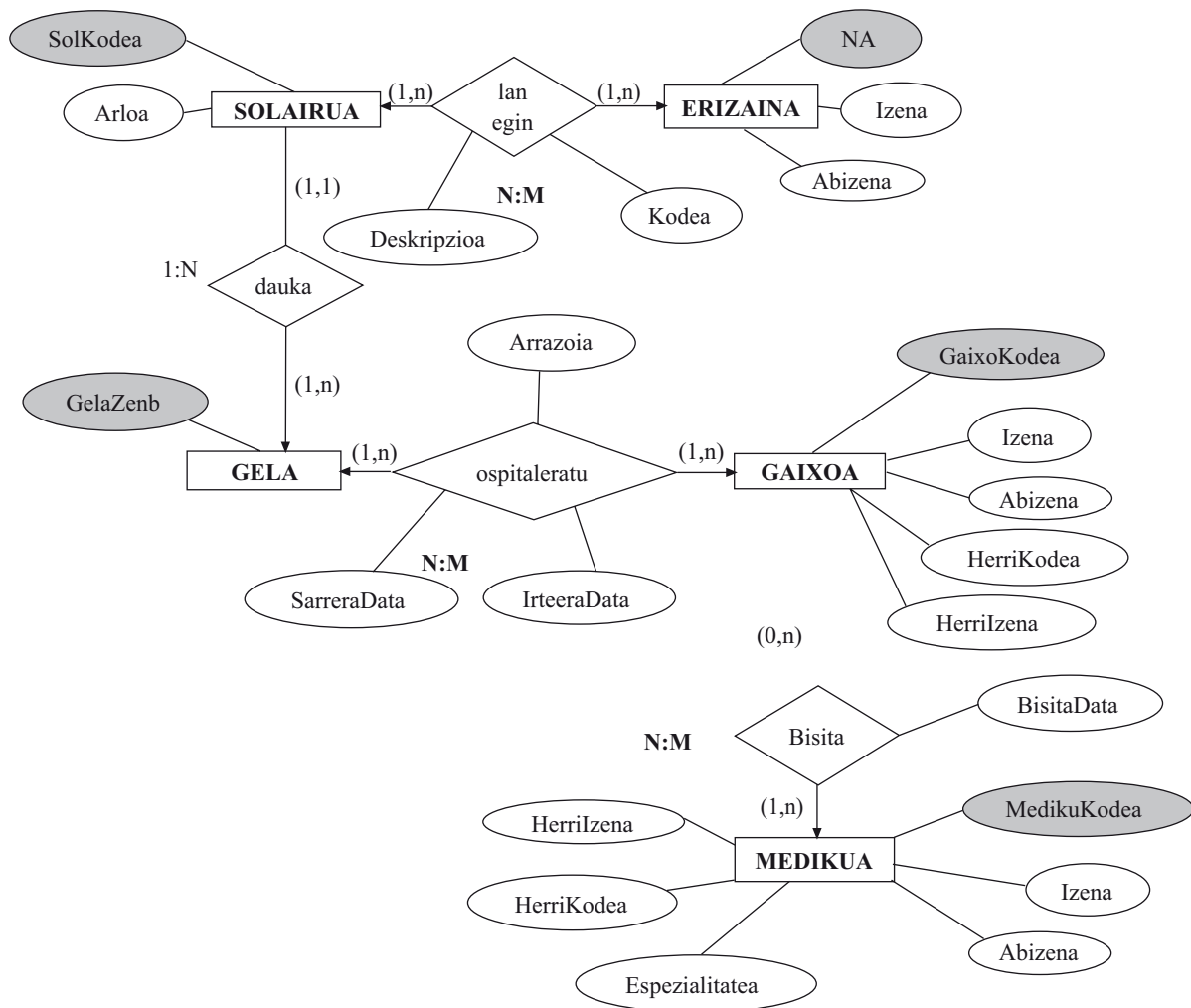
Datu-baseari egingo zaizkion ohiko kontsulten zenbait adibide agertzen dira ondoren:

- Gaixoaren kodearekin, ospitaleratuta egon den gelaren zenbakia, egun-kopurua eta arrazoia bistaratzea. Kontuan izan behar da gaixo batek ospitalizazio bat baino gehiago izan ditzakeela.
- Mediku jakin batek astean bisitatutako gaixoen zerrenda.

Murrizketa semantikoak:

- Gela bakoitzak zenbaki ezberdina dauka.
- Gaixoa mediku ugari bisita dezakete ospitalizazio batean. Bisita bakoitzaren data da interesgarria.
- Gaixo eta mediku bakoitzeko ohiko datuez gain, herriaren kodea eta izena ere jakin nahi dira.

Datu horiei dagokien E/R eredu:



4.43 irudia. Ospitalea adibidearen E/R eredu.

E/R honi dagokion eredu erlazionala:

SOLAIRUA (SolKodea, Arloa)

ERIZAINA (NA, Izena, Abizena)

LANEGIN (SolKodea, NA, Kodea, Deskripzioa)

GELA (GelaZenb, SolKodea)

OSPITALERATU (GelaZenb, GaixoKodea, SarreraData, Arrazoia, IrteeraData)

GAIXOA (GaixoKodea, Izena, Abizena, HerriKodea, HerriIzena)

BISITA (GaixoKodea, MedikuKodea, BisitaData)

MEDIKUA (MedikuKodea, Izena, Abizena, HerriKodea, HerriIzena, Espezialitatea)

Eredu erlazional hori ondo diseinatuta egoteko, 3FNeraino normalizatu beharra dago.

- 1FNa: multzoak ezabatu behar dira:

Adibidean, ez dago multzorik; beraz, eredu dagoeneko 1FNean dago

- 2FNa: mendekotasun funtzional partzialak ezabatu behar dira. Adibideko mendekotasun funtzional partzialak honako hauek dira:

LANEGIN erlazioan: Kodea ® Deskripzioa

‘Deskripzioa’ atributua ‘Kodea’ atributuarekiko da mendeko, eta ez gakoa osatzen duten atributu guztien mendeko.

Eredua 2FNean honela geratuko da:

SOLAIRUA (SolKodea, Arloa)

ERIZAINA (NA, Izena, Abizena)

LANEGIN (SolKodea, NA, Kodea)

LANA (Kodea, Deskripzioa)

GELA (GelaZenb, SolKodea)

OSPITALERATU (GelaZenb, GaixoKodea, SarreraData, Arrazoia, IrteeraData)

GAIXOA (GaixoKodea, Izena, Abizena, HerriKodea, HerriIzena)

BISITA (GaixoKodea, Medikukodea, BisitaData)

MEDIKUA (MedikuKodea, Izena, Abizena, HerriKodea, HerriIzena, Espezialitatea)

- 3FNa: mendekotasun iragankorrak ezabatu behar dira. Adibideko mendekotasun iragankorrak:

GAIXOA eta MEDIKUA erlazioetan:

MedikuKodea → HerriKodea → HerriIzena.

GaixoKodea → HerriKodea → HerriIzena.

‘HerriIzena’ atributua iragankorra da; ‘HerriKodea’, atributu nagusia

Eredua 3FNean honela geratuko da:

SOLAIRUA (SolKodea, Arloa)

ERIZAINA (NA, Izena, Abizena)

LANEGIN (SolKodea, NA, Kodea)

LANA (Kodea, Deskripzioa)

GELA (GelaZenb, SolKodea)

OSPITALERATU (GelaZenb, GaixoKodea, SarreraData, Arrazoia, IrteeraData)

GAIXOA (GaixoKodea, Izena, Abizena, HerriKod)

BISITA (GaixoKodea, Medikukodea, BisitaData)

MEDIKUA (MedikuKodea, Izena, Abizena, HerriKodea, Espezialitatea)

HERRIA (HerriKodea, HerriIzena)

- b) Errealitatetik eredu erlazionala zuzen lortu ondoren, futboleko liga honako informazio honekin kudea daiteke:

Ligan parte hartzen duen futbolari bakoitzaren izena, NA, jaiotza-urtea, soldata, hitz egiten dituen hizkuntzak (gaztelania, ingelesa, euskara...), jokatzen duen taldearen izena, taldeari dagokion herria, taldeburua, jokatutako partiden kodea, data eta emaitza. Aipatutako informazioarekin, honako eredu erlazional hau lortuko da:

LIGA(NA, Izena, Jaio, Soldata, Hizkuntza, Taldea, Herria, Burua, PartidaKodea, Data, Emaitza)

Erlazio horrek gorde dezakeen informazioaren adibidea 4.6 taulan ikusten da:

NA	Izena	Jaio	Soldata	Hizkuntza	Taldea	Herria	Burua	PostaKodea	Data	Emaitza	
78825623	Jorge	1975	200	Euskara	Artibai	Markina	Juan Ansotegi	452	2004/10/23	1	
				Gaztelania				455	2004/12/2	X	
								356	2004/03/11	2	
14562238	Gorka	1974	300	Euskara	Artibai	Markina	Juan Ansotegi	455	2004/12/2	X	
				Gaztelania				356	2004/03/11	2	
				Ingelesa							
78658956	Aimar	1975	250	Gaztelania	Tabira	Durango	Txema Zamalloa	452	2004/10/23	1	
									244	2004/09/12	X
									356	2004/03/11	2

4.6 taula. LIGA erlazioaren agerraldiak.

Eredu erlazional hori ondo diseinatuta egoteko, 3FNeraino normalizatu beharra dago.

- 1FNa: multzoak ezabatu behar dira:

Adibidean, jokalaria bakoitzak hizkuntza bat baino gehiago hitz egiten baditu, multzoak agertzen dira 'Hizkuntza' atributuan. Jokalari bakoitzeko ere, partidei buruzko informazioan, multzoak ageri dira. Arazo horiek konpontzeko, aldaketak egin behar dira:

LIGA (NA, Izena, Jaio, Soldata, Taldea, Herria, Burua)

HIZKUNTZA (NA, Hizkuntza)

PARTIDA (NA, PartidaKodea, Data, Emaitza)

- 2FNa: mendekotasun funtzional partzialak ezabatu behar dira. Adibidean mendekotasun funtzional partzial bakarra dago:

PartidaKodea → Data, Emaitza

'Data' atributua kodearen mendeko baino ez da, eta ez 'NA' eta 'Kodea' ren mendeko; beraz, mendekotasun funtzional partziala dago. Eredua 2FNean geratzeko:

LIGA (NA, Izena, Jaio, Soldata, Taldea, Herria, Burua)

HIZKUNTZA (NA, Hizkuntza)

PARTIDA (PartidaKodea, Data, Emaitza)

JOKATU (NA, PartidaKodea)

- 3FNa: mendekotasun iragankorrek ezabatu behar dira. Adibideko mendekotasun iragankorrek:

NA → Taldea → Herria

NA → Taldea → Burua

‘Herria’ eta ‘Burua’ atributu iragankorrek dira, ‘Taldea’ atributu nagusia izanik. Eredua 3FNean uzteko:

LIGA (NA, Izena, Jaio, Soldata, Taldea)

HIZKUNTZA (NA, Hizkuntza)

PARTIDA (PartidaKodea, Data, Emaidza)

JOKATU (NA, PartidaKodea)

TALDEA (Taldea, Herria, Burua)

Datuen erabilera.
Lengoaiak

5

5.1. SARRERA

Datu-base erlazionala diseinatu ondoren (ikus 4. gaia), datu-basea ezarri egin behar da, diseinuan erabakitakoa idatziz. Horretarako, kapitulu honetan datu-base erlazionalak kudeatzeko lengoia estandarra erabiliko da: SQL².

1986an ANSI erakundeak SQL estandarra argitaratu zuen eta gaur egun produktu berri guztiek onartzen dute SQL, bai ordenagailu pertsonalek, bai enpresa handietako ordenagailuek. Datu-baseetan kontsultak egiteko edota definizio, kontrol eta kudeaketa funtzioak egiteko erabiltzen denez, bi sententzia-motatan banatzen da.

- DML: Datuak Manipulatzeko Lengoia
- DDL: Datuak Definitzeko Lengoia

5. gai honetan, bi sententzia-multzo horien ezaugarriak aztertuko ditugu.

5.2. DML

DBKSetan datuen manipulazioa edo erabilera eragiketa hauetan laburbiltzen da:

- Datu-basean biltegitutako informazioa kontsultatzea
- Datu-basean informazio berria txertatzea.
- Datu-basean biltegitutako informazioa aldatzea
- Datu-basean biltegitutako informazioa ezabatzea

Kontsultak eta aldaketak (txertatu, aldatu, ezabatu) egiteko DMLa erabiltzen da. Bi motatako DMLak daude:

- Prozedurazkoak: DML prozedurazkoetan erregistroak banan-banan manipulaten dira. Behe-mailako DMLa ere esaten zaio. Erabiltzaileak zein datu behar duen, nola lor daitekeen eta datua lortzeko atzipen-eragiketa egokiak dituen prozedura idatzi beharko ditu. Gero, prozedura horri jarraituz, DBKSak erregistro bat berreskuratu eta prozesatuko du. Ondoren, lortutako emaitzetan oinarrituta, beste erregistro bat berreskuratzera joko du, prozesatzeko, eta abar. Eskatutako datu guztiak lortu arte jarraituko du prozesuak.
- Ez-prozedurazkoak: DML ez-prozedurazkoetan, erregistro-multzoekin lan egiten da. Era horretako lengoaietan, eskatutako datua adierazpen sinple batean zehatz daiteke. Erabiltzaileak ez daki ezer datu-egiturari buruz; ezta biltegitutako datuei eragiten dieten algoritmoez ere. Nahi duen datua zehaztu beharko du, baina nola lortu behar den esan gabe.

Ikasteko eta erabiltzeko, prozedurazkoak baino errazagoak dira DML ez-prozedurazkoak; kasu horretan, DBKSak egiten du lanik handiena. Eredu erlazionalako sistemek DML ez-prozedurazkoak erabiltzen dituzte oro har. Sare eredu eta eredu hierarkikoek, berriz, prozedurazkoak erabiltzen dituzte.

DMLaren eragiketak erabiltzaileak era interaktiboan defini ditzake edo, bestela, programa baten barnean. Programaren aukera hartuz gero, **lengoia ostalaria** deituko den programazio-lengoia (PHP, Visual Basic, Java, C++,...) erabili beharko du. Bestalde, eragiketa interaktiboak idatzi ahal izateko sintaxia ematen duen DMLaren zatiari **galdeketa-lengoia** deitzen zaio.

² Codd-ek eredu erlazionala atera zuenean, IBMk datu-baseak kontsultatzeke SQL lengoia sortu zuen.

5.2.1. Datuak atzitzeko sententziak. SELECT

SQLk datuak atzitzeko **SELECT** sententzia erabiltzen du, zeinen sintaxi orokorra ondoren adierazten den:

SELECT atributuak FROM erlazioak [WHERE baldintzak [...]]

SELECT sententzia hiru atal nagusitan banatzen da:

- **SELECT**: aljebra erlazionaleko proiektzioari dagokio eta kontsulta bateko emaitzan bistaratu nahi diren atributuak aukeratzeko erabiltzen da.
- **FROM**: aljebra erlazionaleko biderkadura kartesiarrari dagokio eta espresioa ebaluatzerakoan aztertu beharreko erlazioak zerrendatzeko erabiltzen da.
- **WHERE**: aljebra erlazionaleko hautaketa-agindu bati dagokio eta baldintza hori FROM klausulan agertzen diren erlazioetako atributuei aplikatzen zaie.

Demagun bideoklub batean bezeroek alokatzen dituzten filmen kudeaketa errazteko datu-base simple bat sortzen dela. 4. gaian azaltzen den bezala, datu-basea diseinatzeko pausoak jarraituz, normalizaturik dagoen eskema erlazionala lortuko da. Demagun ariketa honetako eskema erlazionala hauxe dela:

BEZEROA (NifBez, Izena, Abizena, Helbidea)

ALOKAIRUA (ZenbAlok, FilmKodea, Data, NifBez)

FILMA (FilmKodea, Izenburua, Kopurua, Prezioa)

Hiru erlazio horiek 5.1, 5.2 eta 5.3 tauletan agertzen dira.

NifBez	Izena	Abizena	Helbidea
1111111L	AMAIA	GOMEZ	ARTEKALE 3, IRUN
2222222J	AITOR	AGIRRE	GUEN KALEA 6, OIARTZUN
3333333T	JOSUNE	ASTORENA	TELLERIA 5, BILBAO

5.1 taula. BEZEROA erlazioa.

ZenbAlok	FilmKodea	Data	NifBez
18	22	02/1/1	1111111L
14	22	03/5/28	2222222J
22	25	04/6/22	2222222J
42	22	05/6/2	3333333T

5.2 taula. ALOKAIRUA erlazioa.

FilmKodea	Izenburua	Kopurua	Prezioa
22	UN, DOS, TRES	2	1.5
25	NINJA DORTOKAK	6	2

5.3 taula. FILMA erlazioa.

Datuak atzitzeko erak aztertu asmoz, zenbait adibide aurkeztuko ditugu ondoren. Hasteko, demagun bezeroen izenak zerrendatu nahi ditugula. Erabili beharreko sententzia honako hau da, eta emaitza, 5.4 taulan agertzen dena:

SELECT Izena FROM BEZEROA;

Izena
AMAIA
AITOR
JOSUNE

5.4 taula. SELECT Izena FROM BEZEROA.

1.5€ baino prezio handiagoa duten filmak zerrendatzeko, DML sententzia hau erabili ahal da:

SELECT * FROM FILMA WHERE Prezioa>1.5;

* ikurrak eremu guztiak adierazten dituzenez, honako hau izango da emaitza:

FilmKodea	Izenburua	Kopurua	Prezioa
25	NINJA DORDOKAK	6	2

5.5 taula. SELECT * FROM FILMA WHERE Prezioa>1.5.

Noizbait alokatu diren filmen kodea zerrendatzeko, honako sententzia hau erabiliko da; Emaitza 5.6 taulan agertzen da:

SELECT FilmKodea FROM ALOKAIRUA;

FilmKodea
22
22
25
22

5.6 taula. SELECT FilmKodea FROM ALOKAIRUA.

• ***DISTINCT eta ALL***

Errepikatutako lerroak ezabatu nahiko balira, **DISTINCT** erabili beharko litzateke:

SELECT DISTINCT FilmKodea FROM ALOKAIRUA;

Emaitzan filmen kodeak errepikatu gabe agertuko lirateke:

FilmKodea
22
25

5.7 taula. SELECT DISTINCT FilmKodea FROM ALOKAIRUA.

Alderantziz, **ALL** hitza errepikatutako lerroak ez ezabatzeko erabiltzen da. Dena den, errepikatutako emaitza berez agertzen denez, normalean ez da **ALL** erabiltzen. Emaitza 5.6 taulakoa da.

```
SELECT ALL FilmKodea FROM ALOKAIRUA;
```

• **UNION, INTERSECT eta MINUS**

Eragiketa hauek oso erabilgarriak dira askotan. SQL lengoaiak **UNION** (bilketa), **INTERSECT** (ebaketa) eta **MINUS** (kenketa) eragiketak barneratzen ditu (2. gaian adierazitako aljebra erlazionalako eragiketei dagozkie).

Demagun lehengo bideoklubak beste bideoklub bat eskuratu duela eta bigarren bideoklub horretako bezeroak 5.8 taulan adierazitakoak direla:

NifBez	Izena	Abizena	Helbidea
2222222J	AITOR	AGIRRE	GUEN KALEA 6, OIARTZUN
6666666J	YOLANDA	LOMA	AJURIA 3, ZALDIBAR
5555555M	ASIER	USKOLA	BARRENKALE 5, ZORNOTZA

5.8 taula. BEZEROA2 erlazioa.

Orain bezeroak bi erlaziotan banatuta daude, eta guztiak zerrendatu nahiko balira, honako DML sententzia hau erabili beharko litzateke:

```
SELECT * FROM BEZEROA
```

```
UNION
```

```
SELECT * FROM BEZEROA2
```

Emaitza 5.9 taulan ikus daiteke:

NifBez	Izena	Abizena	Helbidea
1111111L	AMAIA	GOMEZ	ARTEKALE 3, IRUN
2222222J	AITOR	AGIRRE	GUEN KALEA 6, OIARTZUN
3333333T	JOSUNE	ASTORENA	TELLERIA 5, BILBAO
6666666J	YOLANDA	LOMA	AJURIA 3, ZALDIBAR
5555555M	ASIER	USKOLA	BARRENKALE 5, ZORNOTZA

5.9 taula. BEZEROA eta BEZEROA2 erlazioen bilketa.

UNION eragiketak, berez, emaitza errepikatuak ez ditu zerrendatzen. Beraz, tuplo errepikatuak zerrendatzea nahiko balitz, **UNION ALL** hitza erabili beharko litzateke.

Eskuratutako bideoklubean bezero berriak zein diren jakin nahiko balitz; hau da, lehendik bezero ez direnak:

```
SELECT * FROM BEZEROA2
```

```
MINUS
```

```
SELECT * FROM BEZEROA
```

Emaitza 5.10 taulan ikus daiteke:

NifBez	Izena	Abizena	Helbidea
66666666J	YOLANDA	LOMA	AJURIA 3, ZALDIBAR
55555555M	ASIER	USKOLA	BARRENKALE 5, ZORNOTZA

5.10 taula. BEZEROA eta BEZEROA2 erlazioen kenketa.

Eskuratutako bideoklubean, aurretiaz bezeroak direnak zerrendatu nahiko balira:

```
SELECT * FROM BEZEROA
```

```
INTERSECT
```

```
SELECT * FROM BEZEROA2
```

Emaitza 5.11 taulan ikus daiteke:

NifBez	Izena	Abizena	Helbidea
22222222J	AITOR	AGIRRE	GUEN KALEA 6, OIARTZUN

5.11 taula. BEZEROA2 eta BEZEROA erlazioen ebaketa.

• ORDER BY

Zerrendatutako emaitzak ordenatzeko aukera ematen du SQL lengoaiak **ORDER BY** erabiliz. Ordenatzeko erabiliko den atributua adierazi beharko da.

Izenaren arabera alfabetikoki ordenatu nahi izanez gero, DML sententzia hau erabiliko da:

```
SELECT NifBez, Izena, Helbidea FROM BEZEROA
```

```
ORDER BY Izena;
```

Emaitza:

NifBez	Izena	Helbidea	Helbidea
22222222J	AITOR	GUEN KALEA 6, OIARTZUN	GUEN KALEA 6, OIARTZUN
11111111L	AMAIA	ARTEKALE 3, IRUN	
33333333T	JOSUNE	TELLERIA 5, BILBAO	

5.12 taula. Izenaren arabera alfabetikoki ordenatutako adibidea.

Berez, ordena beti goranzko eran egiten du. Ordena hori aldatu nahi izanez gero, ASC eta DESC hitzak erabiliko dira. Adibidez:

```
SELECT NifBez, Izena, Helbidea
```

```
FROM BEZEROA
```

```
ORDER BY Izena DESC;
```

Emaitza:

NifBez	Izena	Helbidea
33333333T	JOSUNE	TELLERIA 5, BILBAO
11111111L	AMAIA	ARTEKALE 3, IRUN
22222222J	AITOR	GUEN KALEA 6, OIARTZUN

5.13 taula. Izenaren arabera alfabetikoki era beherakorrean ordenatutako adibidea

• **Biderkadura cartesiarra edo JOIN**

Join eragiketa, taula bi edo gehiagoren biderkadura cartesiarra egin ondoren, informazioa ateratzeko erabiltzen den eragiketa da. Horretarako, SQL lengoaian, FROM klausulan, biderkaduran erabiliko diren erlazioak adierazten dira. Horrez gain, erlazioak lotzen dituzten atributuak WHERE klausulan jarri beharko dira. SELECTak, ondoren, proiektzioa egingo du. Emaitza 5.14 taulan ikus daiteke:

Jo dezagun alokairuren bat egin duten bezeroen datuak bistaratu nahi direla:

```
SELECT * FROM BEZEROA, ALOKAIRUA
WHERE BEZEROA.Nifbez=ALOKAIRUA.Nifbez;
```

ZenbAlok	FilmKodea	Data	NifBez	Izena	Abizena	Helbidea
18	22	02/1/1	11111111L	AMAIA	GOMEZ	ARTEKALE 3, IRUN
14	22	03/5/28	22222222J	AITOR	AGIRRE	GUEN KALEA 6, OIARTZUN
22	25	04/6/22	22222222J	AITOR	AGIRRE	GUEN KALEA 6, OIARTZUN
42	22	05/6/2	33333333T	JOSUNE	ASTORENA	TELLERIA 5, BILBAO

5.14 taula. Biderkadura cartesiarraren adibidea.

Emaitza berbera lortuko litzateke ondorengo sententzia hau erabiliz gero:

```
SELECT * FROM BEZEROA NATURAL JOIN ALOKAIRUA;
```

Aipatzekoa da, beste join mota bat ere badagoela: OUTER JOIN, (+) ikurrarekin adierazten dena. Join mota hori, erlazio batean erlazonatutako zutaberik ez izan arren, beste erlazioko informazioa atera nahi denean erabiltzen da. Adibide gisa:

```
SELECT * FROM BEZEROA, ALOKAIRUA
WHERE BEZEROA.NifBez=ALOKAIRUA.NifBez(+);
```

Edo,

```
SELECT * FROM BEZEROA, LEFT JOIN ALOKAIRUA;
```

Goiko kasuetan, bezero guztiak agertuko dira nahiz eta alokairurik ez izan.

RIGHT JOIN motaren kasuan, lehenengo erlazioan erlazonatutako zutaberik ez izan arren, bigarren erlazioko informazioa atera nahi denean erabiltzen da. Adibide gisa:

```
SELECT * FROM BEZEROA, ALOKAIRUA
WHERE BEZEROA.NifBez(+)=ALOKAIRUA.NifBez;
```

Edo,

```
SELECT * FROM BEZEROA, RIGHT JOIN ALOKAIRUA;
```

• *Azpikontsultak*

Azpikontsultek asko errazten dituzte sententziak. SELECT sententziaren emaitzari beste SELECT sententzia bat ezartzean ematen dira. Kontuan hartu behar da lehenbizi azpikontsulta egikaritzen dela eta, ondoren, kontsulta. Demagun, adibidez, 1'5 €ko prezioa baino handiagoa duten filmei buruz egin diren alokairuen datuak bistaratu nahi direla. Biderkadura cartesiarra eginez:

```
SELECT FilmKodea, Data, NifBez
FROM ALOKAIRUA, FILMA
WHERE ALOKAIRUA.FilmKodea=FILMA.FilmKodea
AND FILMA.Prezioa>1.5;
```

Azpikontsulta erabiliz, berriz:

```
SELECT FilmKodea, Data, NifBez
FROM ALOKAIRUA WHERE FilmKodea IN(SELECT FilmKodea FROM FILMA
WHERE Prezioa>1.5);
```

Goiko bi sententzien emaitza bera da, 5.15 taulan agertzen dena hain zuzen:

FilmKodea	Data	NifBez
25	04/6/22	22222222J

5.15 taula. Azpikontsultaren adibidea.

• *IN*

IN eragileak, elementu jakin bat multzo bateko partaide den egiaztatzen du; **NOT IN** eragileak, ostera, multzo bateko partaide ez dela. Adibidez, zerrendatu hasierako bideoklubeko bezero ez ezik bigarrenekoak ere badirenak:

```
SELECT * FROM BEZEROA
WHERE NifBez IN (SELECT NifBez FROM BEZEROA2);
```

Emaitza 5.16 taulan jaso da.

NifBez	Izena	Abizena	Helbidea
22222222J	AITOR	AGIRRE	GUEN KALEA 6, OIARTZUN

5.16 taula. IN sententziaren adibidea

Gainera, IN eragilea atributu bati baino gehiagori aplikatu dakioke:

```
SELECT * FROM BEZEROA
```

```
WHERE (NifBez, Helbidea) IN (SELECT NifBez, Helbidea FROM BEZEROA2);
```

Emaitza 5.16 taulako berbera izango litzateke.

• **Tuploko aldagaiak, AS eta konektoreak**

Sarritan SELECT sententziak erabat nahasten dira. Ulergarriagoak egiteko erabiltzen dira tuploko aldagaiak. Tuploko aldagai bakoitza erlazio bati erlazonatuta egoten da. Aldagaiak FROM klausulan definitzen dira. Horien erabilera ulertzeko, hurrengo adibidean, alokairuren bat duten bezeroen datuak aterako dira.

```
SELECT B.NifBez, Izena
```

```
FROM BEZEROA B, ALOKAIRUA A
```

```
WHERE B.NifBez=A.NifBez;
```

Adibide horretan, B eta A dira tuploko aldagaiak.

Erlazioetako eremuekin eragiketak egiteko edo konparazio logikoak egin ahal izateko, SQL lengoaiak AND, OR eta NOT eragile logikoak eta +, -, * eta / eragile aritmetikoak erabiltzen ditu. Kalkulatutako zutabeak lortzeko erabil daitezke eragile aritmetikoak. Adibidez: Kalkulatu film bakoitzeko prezioaren %10 (ikus 5.17 taula).

```
SELECT FilmKodea, Izenburua, Kopurua, Prezioa, Prezioa*10/100
```

```
FROM FILMA;
```

FilmKodea	Izenburua	Kopurua	Prezioa	Prezioa*10/100
22	UN, DOS, TRES	2	1.5	0.15
25	NINJA DORTOKAK	6	2	0.2

5.17 taula. Adibidea eragile aritmetikoekin.

Ikusten denez, sortutako eremu berriak ez du izenbururik. Izenburua jartzeko AS hitza erabiltzen da sententzian:

```
SELECT FilmKodea, Izenburua, Kopurua, Prezioa, Prezioa*10/100 AS '%10'
```

```
FROM FILMA;
```

Emaitza, 5.18 taulan:

FilmKodea	Izenburua	Kopurua	Prezioa	%10
22	UN, DOS, TRES	2	1.5	0.15
25	NINJA DORTOKAK	6	2	0.2

5.18 taula. Adibidea eragile aritmetikoekin, AS hitza izenburuak adierazteko erabiliz.

Ondorengo adibide honetan AS eta tuploko aldagaiak erabiltzen dira. A, B eta P dira aldagaiak. || ikurrak karaktere-kateak kateatzen ditu. Beraz, B.Izena || B.Abizena jarriz gero, emaitza “izena-

abizena” jarraian izango da. Izena eta abizena bananduta agertzeko ‘ ‘ espazioa ere lotu beharko da, ondoren adierazten den bezala.

```
SELECT B.Izena || ' ' || B.Abizena AS 'Bezeroaren izen-abizenak',
A.Data, P.Izenburua
FROM BEZEROA B, ALOKAIRUA A, FILMA P
WHERE ((A.NifBez=B.NifBez) AND (A.FilmKodea=P.FilmKodea));
```

Eraitza 5.19 taulakoa da.

Bezeroaren izen-abizenak	Data	Izenburua
<i>Amaia Gomez</i>	<i>02/1/1</i>	<i>UN, DOS, TRES</i>
<i>Aitor Agirre</i>	<i>03/5/28</i>	<i>UN, DOS, TRES</i>
<i>Aitor Agirre</i>	<i>04/6/22</i>	<i>NINJA DORDOKAK</i>
<i>Josune Astorena</i>	<i>05/6/2</i>	<i>UN, DOS, TRES</i>

5.19 taula. Adibidea, tuploko aldagaiak, konektoreak eta AS erabiliz

Tuploko aldagaiak 1. graduko erlazioetan oso lagungarria dira. Jo dezagun enpresa batean langile batzuk daudela. Langile horien artean, langile batzuk beste batzuen buru izan daitezke.

LANGILEA(NALangilea, Izena, Abizena, Helbidea, NABurua)

Buruak diren langileen izenak eta abizenak zerrendatu nahiko balira, DML sententzia hau erabili beharko da:

```
SELECT J.Izena, J.Abizena
FROM LANGILEA E, LANGILEA J
WHERE E.NABurua=J.NALangilea
```

• **BETWEEN eta LIKE**

Bestalde, SQL lengoaiak BETWEEN eragilea ere badauka bi balioen arteko konparazioa egiteko. Prezioa 1.8 eta 2 euro tartean duten filmen izenburuak zerrendatzeko sententzia adierazten da ondoren.

Bi sententziek emaitza bera lortzen dute

```
{
SELECT Izenburua FROM FILMA
WHERE Prezioa BETWEEN 1.8 AND 2;
SELECT Izenburua FROM FILMA
WHERE Prezioa >= 1.8 AND Prezioa <=2;
```

Gainera, SQL lengoaiak karaktere-kateak konparatzeko badu LIKE eragilea. Erabileran ikur ezberdinak darabiltza:

% ⇒ Edozein azpikateari dagokio.

_ ⇒ Edozein karaktereri dagokio.

Adibidez, zerrendatu helbidean ‘AR’ azpikatea duen edozein bezeroren izena eta helbidea (ikusi 5.20 taula):

```
SELECT Izena, Helbidea FROM BEZEROA  
WHERE Helbidea LIKE ‘%AR%’;
```

Izena	Helbidea
AMAIA	ARTEKALE 3, IRUN
AITOR	GUEN KALEA 6, OIARTZUN

5.20 taula. ‘AR’ azpikatea helbidean duten izenen zerrenda.

• GROUP BY

Datu-basean taldeak sortu nahi direnerako, SQL lengoaiak **GROUP BY** agindua darabil. Sortzen diren taldeetan hainbat funtzio aplikatu daitezke:

- Minimoa: MIN
- Maximoa: MAX
- Batura: SUM
- Zenbatu: COUNT
- Batez bestekoa: AVG

Adibidez, film bakoitzeko egindako alokairu-kopurua atera nahiko balitz:

```
SELECT FilmKodea, COUNT(FilmKodea) AS ‘ZenbAlok’  
FROM ALOKAIRUA  
GROUP BY FilmKodea;
```

Emaitza 5.21 taulan agertzen da.

FilmKodea	ZenbAlok
22	3
25	1

5.21 taula. GROUP BY adibidea.

Edo bideoklub horretan dagoen bezero-kopurua zenbatu nahiko balitz:

```
SELECT COUNT (NifBez) AS ‘Bezero-kopurua’  
FROM BEZEROA;
```

Emaitza honako hau izango da:

Bezero-kopurua
3

5.22 taula. COUNTen adibidea.

Bideoklubeko zuzendaritzak alokairu guztiekin irabazitako dirua kalkulatu nahi badu:

```
SELECT SUM(Prezioa) AS 'Diru-kopurua'
FROM ALOKAIRUA A, FILMA P
WHERE A.FilmKodea=P.FilmKodea
```

Emaitza 5.23 taulakoa izango da:

Diru-kopurua
6.5

5.23 taula. SUM adibidea.

Bideoklubeko zuzendaritzak alokairua duen film bakoitzeko irabazitako dirua kalkulatu nahi badu:

```
SELECT A.FilmKodea, P.Izenburua, SUM(Prezioa) AS 'Diru-kopurua'
FROM ALOKAIRUA A, FILMA P
WHERE A.FilmKodea=P.FilmKodea
GROUP BY A.FilmKodea, P.Izenburua;
```

Emaitza, 5.24 taulan:

FilmKodea	Izenburua	Diru-kopurua
22	UN, DOS, TRES	4.5
25	NINJA DORDOKAK	2

5.24 taula. SUMen adibidea.

• **HAVING**

Batzuetan, taldeei dagozkien baldintzak ere defini daitezke: adibidez, izan daiteke aurreko kasuan bi alokairu baino gehiago dituzten filmak baino ez zerrendatu nahi izatea. Horretarako, **HAVING** erabiltzen da.

```
SELECT FilmKodea, COUNT(FilmKodea) AS 'Bi_baino_gehiago'
FROM ALOKAIRUA
GROUP BY FilmKodea
HAVING COUNT(FilmKodea)>2;
```

5.25 taulan dugu sententzia horren emaitza:

FilmKodea	Bi_baino_gehiago
22	3

5.25 taula. GROUP BY eta HAVING adibidea.

5.2.2. Datu-basearen aldaketak

Orain arte ikusitako sententzia guztiak datu-basetik informazioa atzitzeko sententziak izan dira. Orain berriz, datu-baseko informazioa aldatzeko sententziak aztertuko dira. Sententziok erlazioetan dituzten emaitzak ikusi nahiko balira, sententzien ondoren SELECT egin beharko litzateke.

• **DELETE**

Tuploak datu-basetik ezabatzeko, **DELETE** erabiltzen da. Demagun, prezioa 1.3 eta 1.5 balioen artean duten filmak ezabatu egin nahi direla:

```
DELETE FROM FILMA
```

```
WHERE Prezioa BETWEEN 1.3 AND 1.5;
```

Esan bezala, datu-basean aldaketak eragiten dituzten sententziak erabiltzerakoan, erabiltzaileari ez zaizkio aldaketak erakusten, nahiz eta datu-basearen erlazioetan eman. DELETE sententziaren adibidean, FILMA erlazioa honela geratuko da (ikus 5.26 taula):

FilmKodea	Izenburua	Kopurua	Prezioa
25	NINJA DORTOKAK	6	2

5.26 taula. FILMA erlazioa geratuko litzatekeen modua.

• **INSERT**

Tuploak erlazio batean txertatzeko INSERT erabiltzen da. Esate baterako:

```
INSERT INTO FILMA
```

```
VALUES('26', 'GHOST', 5, 2.3);
```

FilmKodea	Izenburua	Kopurua	Prezioa
25	NINJA DORTOKAK	6	2
26	GHOST	5	2.3

5.27 taula. INSERT adibidea.

Erlazioan definitutako ordena berean txertatzen dira atributuak. Ordena gogoratzen ez badugu, erabiltzaileak defini dezake:

```
INSERT INTO FILMA
```

```
(Izenburua, FilmKodea, Prezioa, Kopurua)
```

```
VALUES('GHOST', '26', 2.3, 5);
```

Emaitza 5.27 taulan agertzen dena izango litzateke, nahiz eta, kasu horretan ere, erabiltzaileak ez lukeen ikusiko (horretarako SELECT egin beharko luke). Bestalde, aipaturiko bi INSERT sententziak hurrenez hurren aplikatuko balira, bigarrenengan errorea agertuko litzateke, bietan sartu nahi den informazioaren gako nagusia berbera delako.

Baliteke erlazio batean beste erlazio batetik kontsultatutako balioak txertatu nahi izatea. FILMA erlazioaren egitura berarekin FILMA2 deritzon erlazioa sortuko da hurrengo adibidean, bertan 1.9€ baino prezio handiagoa duten filmak gordetzeko:

```
INSERT INTO FILMA2
SELECT * FROM FILMA
WHERE Prezioa>1.9;
```

Erlazioa 5.28 taulan erakusten den moduan geratuko litzateke:

FilmKodea	Izenburua	Kopurua	Prezioa
25	<i>NINJA DORDOKAK</i>	6	2
26	GHOST	5	2.3

5.28 taula. INSERT adibidea.

• UPDATE

SQL lengoaiak, tuplo bateko balioak aldatzeko (eguneratzeko), UPDATE erabiltzen du. Arestiko adibideari jarraituz, demagun filmei prezioa %15 igo nahi zaiela:

```
UPDATE FILMA
SET Prezioa=Prezioa+Prezioa*0.15;
```

Erlazioa 5.29 taulan erakusten den moduan geratuko litzateke:

FilmKodea	Izenburua	Kopurua	Prezioa
25	<i>NINJA DORDOKAK</i>	6	2.3
26	<i>GHOST</i>	5	2.64

5.29 taula. UPDATE adibidea.

Eta prezioa 2.5 € baino handiagoa dutenei prezioa %15 handitzeko:

```
UPDATE FILMA
SET Prezioa=Prezioa+Prezioa*0.15
WHERE Prezioa >2.5;
```

Erlazioa 5.30 taulan erakusten den bezala geratuko litzateke:

FilmKodea	Izenburua	Kopurua	Prezioa
25	<i>NINJA DORDOKAK</i>	6	2.3
26	<i>GHOST</i>	5	3.04

5.30 taula. UPDATE adibidea.

• NULL

Datu-baseetan, bereziki gako nagusi diren eremuetan, kontuz ibili behar da balio hutsekin. Atributu batek balio hutsa duen frogatzeko, **NULL** hitza erabiltzen du SQL lengoaiak. Adibidez, preziorik ez duten filmen datuak ateratzeko:

```
- SELECT * FROM FILMA WHERE Prezioa IS NULL;
```

Prezioa dutenenak zerrendatzeko, berriz:

```
SELECT * FROM FILMA WHERE Prezioa IS NOT NULL;
```

5.2.3. DCL

DCLk erabiltzaile ugariako sistemetan datuen babesa kudeatzea ahalbidetzen duten sententziak ditu (segurtasuna mantentzeko, atzipen-murrizketak ezartzeko, konkurrentzia kudeatzeko...). GRANT eta REVOKE dira sententzia nagusiak (ikusi 5.31 taula). Datu-baseen sistema erlazioaletan sententzia horiek datu-hiztegiarekin asko erlazionatzen dira. DBA da horretaz arduratzen dena, 6. gaian aztertuko denez.

DCL SENTENTZIA	FUNTZIOA
GRANT SELECT ON ErlazioaIzena TO Erabiltzailea	Erabiltzaileari erlazioan SELECT egiteko baimena ematen dio
GRANT SELECT ON ErlazioaIzena TO Erabiltzailea WITH GRANT OPTION	Erabiltzaileari ERLAZIOIZENA erlazioan SELECT egiteko baimena ematen dio eta, gainera, ERABILTZAILA erabiltzaileak baimen hori beste erabiltzaile bati ematea ahalbidetzen du
GRANT CONNECT TO Erabiltzailea IDENTIFIED BY Password	Erabiltzailearen konexioa sortzeko
GRANT ALTER ZutabelIzena ON ErlazioaIzena TO Erabiltzailea	Adierazitako erlazioko eremuaren egituran aldaketak egiteko baimena emateko erabiltzaileari
GRANT DELETE ON ErlazioaIzena TO Erabiltzailea	Adierazitako erlazioan gauzak ezabatzeko baimena emateko erabiltzaileari
REVOKE SELECT ON ErlazioaIzena FROM Erabiltzailea	Erabiltzaileari erlazioan SELECT egiteko baimena kentzeko
REVOKE ALL ON ErlazioaIzena FROM Erabiltzailea	Erabiltzaileak erlazioarekiko dituen baimen guztiak kentzeko

5.31 taula: Datu-base erlazionalak kudeatzeko zenbait DCL sententziaren adibideak.

5.3. DDL

Datuen definizioa datu-basearen administratzaileak egiten du. Definizioa DDL (datuen definizio edo deskripziorako lengoia) erabiliz egiten da. Abstrakzio-maila bakoitzeko, datu-basea osatzen duten elementuen definizioa egiten da.

Datu-base erlazionala osatzen duten elementuak zein diren jakinik, hau da, erlazioak, beraiek osatzen dituzten eremuak, erlazioek bete behar dituzten murrizketak (gako nagusia, gako atzerritarra...), erabiltzaileei erakutsi behar zaizkien ikuspegiak eta abar hainbat eskematan definitzen direla jakinda (eskema kontzeptuala, barneko eskema eta kanpoko eskema) datu-basea definitu ahal izateko erabiltzeko den DDLa aztertuko dugu ondoren.

5.3.1. Sententziak

SQL lengoia datu-baseetan kontsultak egiteko erabiltzen da gehienbat, baina baita definizio, kontrol eta kudeaketa funtzioak egiteko ere. Oraingoan, datu-basearen definizioa egiteko duen gaitasuna aztertuko dugu.

• **Domeinuak, datu motak, konstanteak eta balio hutsak**

Datuentzako domeinuak sortzeko CREATE DOMAIN sententzia erabiltzen da. Adibidez:

```
CREATE DOMAIN KoloreMota AS CHAR(1)
CHECK (VALUE IN('B','H','N'));
```

Dena den, produktu komertzial guztiek ez dute uzten domeinuak sortzen; beraz, kasu horretan, datu mota hauek daude erabilgarri:

- DECIMAL (p,s): zenbaki dezimalak, *s* dezimal-kopurua izanez eta *p*, zehaztasuna.
- INTEGER: zenbaki osoak definitzeko.
- SMALLINT: zenbaki osoak eta txikiak definitzeko.
- FLOAT, REAL, DOUBLE: zenbaki errealak definitzeko.
- DATE: datak adierazteko.
- TIME: orduak adierazteko.
- CHAR(*n*): luzera finkoa duten karaktere-kateak definitzeko, non *n* karaktere-kopurua izango den.
- VARCHAR(*n*): luzera aldakorra duten karaktere-kateak definitzeko, non *n* gehienezko karaktere-kopurua den.
- LONG VARCHAR: luzera mugagabea duten karaktere-kateak definitzeko.
- BLOB: objektuak definitzeko: irudiak, programak, soinu-fitxategiak...

Gainera, SQL lengoaiak konstanteen erabilera ahalbidetzen du.

Baliteke tuploren bat txertatzerakoan atributu batzuei baliorik ez ematea; beraz, balio hutsak egon daitezke datu-basean. Egoera hori atributu batzuetan onargarria gerta daiteke, baina beste atributu batzuentzat (gako nagusiak, adibidez) sekula ez. Dena den, ezaugarri hori DBKS batzuek berez dakarte ezarrita. Beste atributuren batean balio hutsak ez onartzeko NOT NULL erabiltzen da atributuaren domeinuaren deklarazioan. Adibidez,

```
Izena VARCHAR(20) NOT NULL;
```

• **Erlazioak sortzen eta eraldatzen. Indizeak**

Erlazioak sortzerakoan, gehien erabiltzen diren aginduak honako hauek dira:

- Erlazioak sortzeko: CREATE TABLE
- Gako nagusia zehazteko: PRIMARY KEY(<zutabe1>,<zutabe2>,...)
- Gako atzerritarra zehazteko: FOREIGN KEY<Gakoaren izena> (<zutabe1>,<zutabe2> REFERENCES <erlazioaren izena> [(<zutabe1>,<zutabe2>)[(ON DELETE|ON UPDATE) (RESTRICT|CASCADE|SET NULL|SET DEFAULT)])
- Bete beharreko baldintzaren bat zehazteko: CHECK (<Baldintza>)

Hona hemen adibidea:

```
CREATE TABLE ikasleak (
```

```
Izena VARCHAR(20),
```


NA VARCHAR(10),
Helbidea VARCHAR(25),
JaiotzaData DATE,
Taldea VARCHAR(3),
PRIMARY_KEY NA,
CHECK (JaiotzaData BETWEEN '1/1/1960' AND '31/12/1992')
FOREIGN_KEY Taldea REFERENCES Taldeak);

- Erlazioa ezabatzeko: DROP TABLE
- Erlazioaren egitura aldatzeko: ALTER TABLE

Adibidez:

```
ALTER TABLE ikasleak  
ADD Abizena VARCHAR(25);
```

Azkenik, indizeak sortzeko, agindu-mota hau erabiliko da:

```
CREATE[UNIQUE] INDEX ON <erlazioaren izena> (<zutabea> [(ASC|DESC)]...)
```

Horrela, gakoan erabilitako zutabe bakoitzeko indize bat sor daiteke. Optimizazioa jorratzerakoan (7. gaia) gehiago landuko dira.

• *Disparadoreak edo triggerak*

Disparadorea definitzeko, CREATE TRIGGER agindua erabiltzen da, disparadorea gertaeraren aurretik, ondoren edo gertaeraren ordeztar den zehaztuz. Gertaera horiek, txertaketak, ezabatzeak edo atributuen aldatetak izan daitezke. Adibidez, demagun honako erlazio hauek ditugula eta ikasleren bat taldez aldatzen dela. Beraz, zegokion taldeari ikasle-kopurua unitate batean murriztu beharko zaio eta talde berriari, ostera, bat gehitu.

IKAS (NA, Helbidea, Telefonoa, Taldea)

TALDEAK (TaldeZenb, Kokapena, Tutorea, IkasKop)

Hauxe izango litzateke *trigger*-aren garapena:

```
CREATE TRIGGER IkasAldatu  
AFTER UPDATE OF Taldea ON Ikas  
REFERENCING OLD AS IkasZ NEW AS IkasB  
FOR EACH ROW  
BEGIN  
UPDATE Taldeak SET IkasKop=IkasKop+1
```

```

WHERE TaldeZenb=IkasB.Talde;

UPDATE Taldeak SET IkasKop=IkasKop-1

WHERE TaldeZenb=IkasZ.Talde;

END

```

Disparadore hori, ‘Taldea’ eremua IKAS erlazioan aldatu ostean aktibatuko da, AFTER UPDATE OF Taldea ON Ikas lerroak adierazten duen modura. Ondoren, TALDEAK erlazioan ‘IkasKop’ eremua eguneratuko da, talde berriari (NEW AS IkasB) ikasle bat gehituz eta zaharrari (OLD AS IkasZ), bat kenduz.

• *Ikuspegiak*

Ikuspegia definitzeko CREATE VIEW agindua erabiltzen da. Adibidez, ikasleak bi erlazioetan banatuta agertzen badira (IKASLEAK eta IKASLEAK2 erlazioak) eta guztiak ikuspegi batean batu nahi izanez gero:

```

CREATE VIEW IkasleGuztiak AS

(SELECT * FROM Ikasleak)

UNION

(SELECT * FROM Ikasleak2);

```

Adibide horretan, IKASLEAK eta IKASLEAK2 erlazioetako ikasle guztiak aukeratuko dira. Orain, ikuspegi hau erabiliz, “Astorena” abizena duten ikasle guztiak zerrenda daitezke:

```

SELECT * FROM IkasleGuztiak

WHERE Izena LIKE ‘ASTORENA’;

```

Ikuspegien arazoa, tuploak eguneratzerakoan agertzen da. Demagun zenbait aldaketa egin nahi direla:

```

INSERT INTO IkasleGuztiak

VALUES(‘KOSME’, ‘125689F’, ‘,’, ‘HANDIA GETXO’, 1976/02/11, ‘2A’);

```

Agindu horrekin ezin da jakin IKASLEAK ala IKASLEAK2 erlazioa, bietariko zein eguneratuko den. Hori dela eta, ikuspegi batean INSERT (txertatzea), DELETE (ezabatzea) eta UPDATE (aldatzea edo eguneratzea) eragiketak posible izateko, ikuspegi horrek erlazio bakar batean definituta egon behar du. Beraz, IKASLEGUZTIAK deritzon ikuspegian, eragiketa horiek ezin izango dira egin, IKASLEAK eta IKASLEAK2 erlazioetan oinarrituta dago eta.

5.4. DATU HIZTEGIA

Sistemak erabiltzen duen datu-base modura har daiteke **datu-hiztegia**. Hiztegi horrek “datuei buruzko datuak” (metadatuak) ditu; hau da, sistemaren beste objektuei buruzko definizioak gordetzen ditu. Datu-basea sortzen doan heinean, hiztegia handituz joango da, datu-baseko elementuei buruzko informazioa gordeko baitu.

Datu-base erlazioetan, hiztegia datuei buruzko datuak gordetzen dituen erlazio-multzo modura ulertzen da eta horregatik dago erlazio batzuez osatuta. Beraz, SQL erabiliko da bertoko informazioa atzitzeko (ikusi 5.32 taula).

ERLAZIOAREN edo IKUSPEGIAREN Izena	EDUKIA
CHECK_CONSTRAINTS	CHECK motako murrizketak
COLUMNS	Uneko erabiltzaileak atzitu ditzakeen zutabeak
TABLES	Uneko erabiltzaileak atzitu ahal dituen erlazioak
TABLE_CONSTRAINTS	Uneko erabiltzaileak kudeatu ahal dituen erlazioen gainean definitutako murrizketak
TABLE_PRIVILEGES	Uneko erabiltzaileak kudeatu ahal dituen erlazioen gainean definitutako pribilegioak (txertatu, ezabatu...)
VIEW	Uneko erabiltzaileak atzitu ditzkeen ikuspegiak

5.32 taula: Datu-base erlazional bateko hiztegiaren edukiaren adibidea.

Definitutako datu-basearen barruan egoten da hiztegia eta, edozein datu-baserekin egiten den eran, berari buruzko galderak egiteko aukera ematen du; hau da, hiztegiari kontsultak, txertaketak... egin dakizkioke. Horrela, taula batean erabiltzaileak dituen baimenak, atzitu ahal dituen taulak eta abar, hiztegia begiratzuz jakin daitezke. Adibidez, 5.33 taulak dakarrena.

DML SENTENTZIA	FUNTZIOA
SELECT TABLE_NAME FROM TABLES;	Erabiltzaileak atzitu ahal dituen erlazioak bueltatzen ditu
SELECT CONSTRAINT_NAME FROM TABLE_CONSTRAINTS WHERE TABLE_NAME='IKASLEAK';	<i>Ikasleak</i> erlazioan definitutako murrizketak bueltatzen ditu

5.33 Taula: Datu-base erlazionalak kudeatzeko zenbait DML sententziaren adibideak.

*Datu-baseen
administrazioa*

6

6.1. SARRERA

Egungo DBKSetan, bakoitzak dituen baimenen arabera, hainbat erabiltzaile-maila egon daitezke, datu-basean kontsultak eta aldaketak egiteko dituzten aukeren arabera. Orokorrean, DBKSetan bi erabiltzaile-mota sailka daitezke:

- DBA motako erabiltzaileak: datu-basea administratuko dute eta baimen-mailarik gorenaren jabe izango dira.
- Bestelako erabiltzaileak baimendutako ekintzak egin ahal izango dituzte soilik, hala nola datuak atzitu edota aldatu, datuak irakurri eta abar.

DBAk datu-baseak definitzeko eta kontrolatzeko ardura dauka. Horretaz aparte, aplikazioen garapenaren arduradunei eta erabiltzaileei behar duten laguntasuna emateko ardura du. Datu-basearen administratzailea pertsona bakarra edo gehiago izan daitezke. Lagungarri izango ditu DBKS ugari ezagutzea, datu-baseen diseinuan jakintsua izatea eta sistema eragileen arloan, sareen arloan eta hardware eta softwareari dagozkien gaietan ere aditua. Gai informatikoez aparte, DBAk erakundeko legeak eta politikak ezagutu beharko ditu datuen administrazioari dagokion arloan, bera baita informazioa erabiltzaileei helarazi edo ezkatatzeaz arduratuko dena. Hori dela eta, DBAk administratu behar duen DBKSak enpresako datuen administrazio-politikaren arauak beteko ditu eta horixe izango da DBAren helburu nagusietako bat. DBAren ardurapean dauden funtzioak hurrengo ataletan azaltzen dira.

DBKS barruko jarduera izugarria da, izan ere, erabiltzaileen arteko konkurrentziak ezinbestean jarraitu beharreko estandar batzuk definitzera derrigortzen du. Arlo hori segurtasunari dagokion hurrengo gaien zehazten da gehiago, datu-basearen segurtasuna bermatzea DBAri dagokion beste eginkizun garrantzitsuenetariko bat baita.

6.2. DATU-BASEEN EGITURAREN ADMINISTRAZIOA

Eginkizun honetarako oso komenigarria da datu-basearen hasierako diseinuan parte hartzea, ondoren ager daitezkeen aukerak aztertzea, datu-basearen betebeharrak guztiak kontrolatzea, erabiliko den DBKСа menperatzea eta, azkenik, lortuko den diseinua ezagutzea. Izan ere, DBA diseinu-taldearen kudeatzaile modura ere ezagutzen da.

Datu-basearen diseinua lortuta dagoenean, datu-base hori abian jartzeko garaia dator, eta horretarako, DBKSaren instalazioari ekitea da lehenengo eginkizuna. DBKS baten instalazioa produktuaren beraren eta oinarri izango duen sistema eragilearen arabera izango da. Egun, lan-estazioekin sare bidez konektatutako zerbitzari ertainak eta handiak dituzten inguruneetan, instalazio bikoitza egin behar izaten da. Lehenengo, zerbitzarian, eta, ondoren, lanerako estazioetan (azken horiek beste sistema eragile bat izan ohi dute).

Dena den, gehienetan, ondoren zehazten diren pauso hauek ematen dira:

- DBKSaren motorra edo nukleoa zerbitzari(et)an instalatu.
- Administraziorako tresnen instalazioa: zerbitzarian instalatuko diren lehenengo aplikazioak izango dira.
- Bezeroen instalazioa: datu-basea atzitu duten lan-estazioetan beharrezkoa den softwarea instalatuko da.

- Komunikaziorako aplikazioen instalazioa: bezero-zerbitzari inguruneetan bezeroen eta zerbitzariaren arteko komunikazioa sarearen bitartez eratzen da.
- Atzipenerako tresnen instalazioa: edozein DBKS erlazionalek SQL kontsola du datu-baseko datuak atzitzeko. Sortuta dagoen datu-base bateko erlazioen atzipen eta manipulazio zuzena egiteko aukera ematen du.
- Garapenerako tresnen instalazioa: DBKSetako zati oso garrantzitsua dira, datu-basearen erabilera erraztu egiten baitute eta, gainera, formulario, txosten eta abarren diseinua ahalbidetzen dutelako. Hainbatean, tresnak erabili ahal izateak edo ez izateak, eta horien ezaugarriek, produktuaren salmenta baldintzatzen dute. Egun, elementuei zuzendutako ezaugarriak dituzte eta bezero-zerbitzari aplikazioak garatzeko gai dira. Batzuk aipatzearen, Oracle Developer eta Power Builder nabarmenduko genituzke.

DBKSaren instalazioa egin ostean, behar adina datu-base sortzen dira, kasuan kasuko beharriaren arabera. Horretarako, edozein produktu komertzialetan, bi aukera daude:

- DBKSkon erremintetako bat erabiltzea.
- SQL kontsolatik, CREATE DATABASE, CREATE DBSPACE motako sententziekin datu-basea eskuz sortzea.

Kasu bietan, datu-basearen ezaugarriak finkatuko dituen parametro-multzoa zehaztu beharko du administratzaileak; besteak beste, datu-basearentzako izena, datu-baseko konfigurazio-parametroak gordeko dituen fitxategiaren izena (*Oracle-n*, INIT.ORA izena hartzen du), beste fitxategien izenak (kontrolakoa, bitakora koaderno...),...

Datu-basea sortzeko prozesua honako pauso hauetan labur daiteke:

- Datu-basea bera sortzea, bai fitxategi nagusiak, bai beste guztiak.
- Erlazioak sortzea.
- Hainbat prozesutarako beharrezkoak izan daitezkeen beste erlazio eta ikuspegi batzuk sortzea.
- Aurreikusten diren prozedurak eta *trigger*-ak sortzea.

Datu-basea sortu ahala, datu-baseko hiztegia haziz joango da, hiztegiak, egiturari buruzko informazioa, datu-baseak dituen datu-motak, datu moten eragin-eremuak, segurtasun mugatzaileak eta antzeko informazioak gorde behar baititu. Hori guztia kudeatzearen arduraduna DBA izango da eta DBKS bakoitzak eskaintzen dituen erremintak erabiliko ditu.

6.3. KONFIDENTZIALTASUNAREN KUDEAKETA

Etixerako erabileratik kanpoko edozein sistematan, datu-basea hainbat erabiltzailek erabili ahal izango du: gutxi batzuek edo milioika erabiltzailek, hain zuzen. Horrek informaziorako sarbidea zehatz kudeatzea eskatuko du, datu-base batean oinarritutakoak diren datuen osotasuna eta konfidentzialtasuna ziurta daitezkeen.

- **Osotasuna:** datu-baseak, errealitate jakina islatzen duen heinean, errealitate horren arau eta murrizketa batzuk bete beharko ditu. Horretarako, datu-basea diseinatzerakoan, osotasun hori betetzen laguntzen duten gako nagusiak, gako atzerritarrak eta abar erabili daitezke. Beraz, osotasuna, datu-basearen diseinu faseari dagokion ezaugarria da; datu-basearen administratzaileari dagokio osotasun hori bermatzea, ahal den neurrian. Izan ere, erabiltzaileek gaizki

egindako aplikazioek osotasun hori apurtu egin dezakete. Behar denean, babes-kopiak egin edo datu-basea irauli egiten du.

- **Konfidentzialtasuna:** baimenik gabeko erabiltzaileek informaziorako sarbiderik ez izatean datza ezaugarri hau. Konfidentzialtasuna lortzeko, atzipenak baimentzeko politika egokia hartu behar da eta, horretarako, administratzaileak hainbat baimen-mota banatuko ditu erabiltzaileen artean. DBAren lana izango da erabiltzaileak sortzea, baimenak eman eta kentzea, rolak (ikus 6.2.5 atala), profilak zehaztea... Orokorrean, baimenak sistemen gainekoak edo elementuen gainekoak izan daitezke.

6.3.1. Elementuen gaineko baimenak

Elementuak atzitu eta elementuotan aldaketak egitea ahalbidetzen duten baimenak dira. Hona hemen elementuen gaineko baimenak:

- ALTER: Erlazioen egitura aldatzeko.
- DELETE: Erlazioetako datuak ezabatzeko.
- EXECUTE: Prozedurak egikaritzeko.
- INDEX: Indizeak sortzeko.
- INSERT: Erlazioetan datuak txertatzeko.
- REFERENCES: Gako atzerritarrak sortzeko CREATE TABLE edo ALTER TABLE sententziak erabiltzerakoan.
- SELECT: Erlazioetako datuak jasotzeko.
- UPDATE: Erlazioetako datuak aldatzeko.
- ALL: Zerrendaturiko pribilegio guztiak emateko.

Elementuen gainean baimenak emateko agindua GRANT da. Adibidez, *Oracle*ko datu-baseetako SQL erabiliz:

```
GRANT SELECT, INSERT
ON erlazio1
TO Leire
WITH GRANT OPTION;
```

Goiko adibidean, Leire izeneko erabiltzaileari, ERLAZIO1 izeneko erlazioan, SELECT eta INSERT eragiketak egitea baimendu zaio. Baimenik ematen ez zaion eragiketarik ezin izango du egin. WITH GRANT OPTION delakoarekin, baimenaren jasotzaileari baimenak banatzeko aukera eman zaio. Gure kasuan, Leirek ERLAZIO1 erlazioaren gainean SELECT eta INSERT eragiketak egiteko baimena eman diezaieke beste erabiltzaile batzuei.

Beste adibide batzuk:

```
GRANT UPDATE BezKodea
ON Bezeroa
TO E1, E2, E3;
```

E1, E2 eta E3 erabiltzaileei BEZEROA erlazio2 'BezKodea' atributurako UPDATE baimena ematen zaie.

```
GRANT UPDATE Izena ON erlazio2 TO Ane;
```

Aneri 'Izena' aldatzeko pribilegioa eman zaio ERLAZIO2 erlazioan.

```
GRANT ALL ON erlazio3 TO PUBLIC;
```

Pribilegio guztiak eman zaizkio mundu guztiari ERLAZIO3 erlazioan eragiketak egiteko (PUBLIC = erabiltzaile guztiak). Esan beharra dago oso ariketa arriskutsua izan daitekeela aurreko adibidean aipaturikoa.

Jo dezagun ERLAZIO1 erlazioaren jabea Ainhoa erabiltzailea dela, berak sortu duelako edo haren jabetza eman diotelako. Lehenengo adibidean, Leireri SELECT eta INSERT eragiketak egiteko baimena eman zaio, eta WITH GRANT OPTION sententziarekin, baimen horiek banatzeko aukera ere bai. Demagun orain, Leirek ERLAZIO1 erlazioaren gainean SELECT egiteko baimena eman nahi diola Txomin izeneko erabiltzaile bati:

```
GRANT SELECT  
ON Ainhoa.erlazio1  
TO Txomin;
```

Goiko adibidean, Leire erabiltzaileak, Ainhoaren ERLAZIO1 erlazioaren gainean, SELECT egitea baimendu dio Txomin erabiltzaileari. Erlazioaren izenaren aurrean, jabearen izena jarri behar izaten da, baimena ematen duena jabea bera ez bada.

6.3.2. Sistemaren baimenak

SQL aginduak egikaritzeko baimena ematen dute. Baimen-mota horietan WITH ADMIN OPTION erabiltzen da beste erabiltzaile batzuei baimenak banatu ahal izateko, aurrekoetan WITH GRANT OPTION aukerarekin bezala. Hona hemen sistemaren gaineko baimen batzuk:

- CONNECT: datu-basera konektatu.
- CREATE/ALTER/DROP ANY CLUSTER: edozein *cluster* sortu/aldatu/ezabatu.
- ALTER DATABASE: datu-basearen egitura aldatu.
- CREATE/ALTER/DROP ANY INDEX: edozein indize sortu/aldatu/ezabatu.
- GRANT ANY PRIVILAGE: edozein baimen banatu.
- CREATE/ALTER/DROP/EXECUTE ANY PROCEDURE: edozein prozedura sortu/aldatu/ezabatu/egikaritu.
- CREATE/ALTER/DROP ANY PROFILE: edozein profil sortu/aldatu/ezabatu.
- CREATE/ALTER/DROP ANY ROLE: edozein rol sortu/aldatu/ezabatu.
- CREATE/ALTER SESSION: datu-basera erabiltzailea konektatu/erabiltzailea aldatu.
- CREATE/ALTER/DROP ANY TABLE: edozein erlazio sortu/aldatu/ezabatu.
- SELECT/INSERT/UPDATE/DELETE ANY TABLE: edozein erlaziotan SELECT, INSERT, UPDATE eta DELETE eragiketak egin.

- CREATE/ALTER/DROP ANY VIEW: edozein ikuspegi sortu/aldatu/ezabatu.
- CREATE/ALTER/DROP ANY USER: edozein erabiltzaile sortu/aldatu/ezabatu.
- CREATE/ALTER/DROP ANY TRIGGER: edozein *trigger* sortu/aldatu/ezabatu.
- DBA: DBA izateko baimenak eman.

Adibidez:

GRANT CREATE SESSION TO Peru; Peruri lan-saioa sortzeko baimena eman zaio.

GRANT CONNECT TO Peru, Txomin; Peru eta Txomini datu-basera konektatzeko baimena eman zaie.

GRANT SELECT ANY TABLE TO PUBLIC; edozein erlaziotan SELECT egiteko baimena eman zaio mundu guztiari.

GRANT DBA TO Ane; Aneri administratzailearen baimenak eman zaizkio.

GRANT DROP USER TO Miren WITH ADMIN OPTION;

Mireni erabiltzaileak ezabatzeko baimena eman zaio. Gainera, Mirenek baimen hori beste erabiltzaile batzuei bana diezaieke. Sistemaren gaineko baimenak administratzaileak berak banatu ohi ditu, elementuen baimenak elementuen jabeek banatzen dituzten modura.

6.3.3. Baimenak kentzea

Baimenak ematen diren antzera, kendu ere egin daitezke. Horretarako, REVOKE agindua erabiltzen da. Adibidez:

REVOKE SELECT, INSERT

ON erlazio1

FROM Leire;

Leireri ERLAZIO1 erlazioaren gainean SELECT eta INSERT egiteko baimena kendu zaio.

REVOKE DROP USER

FROM Miren;

Mireni erabiltzaileak ezabatzeko baimena kendu zaio.

Banatutako baimen guztiak DBKSkon hiztegian islatuko dira. DBAk une jakin batean banatu dituen elementuen gaineko baimenak edo sistemaren gaineko baimenak ezagutu nahi baldin baditu, hiztegia kontsultatzea besterik ez du. Datu-baseko informazioa kontsultatzeko, DML arrunta erabiltzen da. Aipatutako informazioa gordetzen duten ikuspegiak *Oraclen*, besteak beste, 6.1 taulakoak dira:

IKUSPEGIAREN IZENA	IKUSPEGIAK EMATEN DUEN INFORMAZIOA	IKUSPEGIAK DITUEN ZUTABEAK
SESSION_PRIVS	Uneko erabiltzaileak dituen baimenak	PRIVILEGE: Baimenaren izena
DBA_SYS_PRIVS	Rol eta erabiltzaileei banatutako sistemaren gaineko baimenak	GRANTEE: Baimena jaso duenaren izena (Rola edo erabiltzailea) PRIVILEGE: Jasotako baimena ADMIN_OPTION: Pribilegioa banatzeko baimena
USER_TAB_PRIVS	Uneko erabiltzaileak jaso edo banatutako elementuen gaineko baimenak	GRANTEE: Baimena jaso duen erabiltzailearen izena OWNER: Elementuaren jabea TABLE_NAME: Erlazioaren izena (Elementua) GRANTOR: Baimena banatu duen erabiltzailearen izena PRIVILEGE: Jasotako baimena GRANTABLE: Baimena banatzeko baimena
USER_SYS_PRIVS	Uneko erabiltzaileari banatutako sistemaren gaineko baimenak	USERNAME: Baimena jaso duenaren izena PRIVILEGE: Jasotako baimena ADMIN_OPTION: Pribilegioa banatzeko baimena

6.1 Taula: Hiztegia gordetzen diren baimenen ikuspegi batzuk.

6.3.4. Erabiltzaileak sortu eta ezabatzea

Erabiltzaileak sortzeko baimena DBAk baino ez izatea da ohikoena. Hala ere, badago aukera erabiltzaileak sortzeko baimen hori beste erabiltzaile bati emateko. Baimena jaso duenak beste erabiltzaile batzuk sortu ahal izango ditu; horretarako, CREATE USER sententzia erabili beharko du. Adibidez:

```
CONNECT DBA/ADMINISTRATZAILEA;
```

Administratzailea konektatu da, DBA erabiltzaile izena eta ADMINISTRATZAILEA pasahitzarekin.

```
GRANT CREATE USER TO Txomin;
```

Txomin erabiltzaileari, erabiltzaile berriak sortzeko baimena eman dio DBAk.

```
GRANT CREATE SESSION TO Txomin WITH ADMIN OPTION;
```

Txomin erabiltzaileari lan-saioa sortzeko baimena eta baimen hori beste batzuei banatzeko aukera ematen dio. Ondoren, konektatu ahal izango da.

```
GRANT CONNECT TO Txomin WITH ADMIN OPTION
```

```
CONNECT Txomin/tx;
```

Txomin erabiltzailea datu-basera konektatuko da. Adibidean, Txomin izango da uneko erabiltzailea. Baina besteak ere badaude, ez baitira deskonektatu.

```
CREATE USER Lorea
```

```
IDENTIFIED BY lore;
```

Txominek Lorea izeneko erabiltzaile berria sortu du, "lore" pasahitzarekin.

GRANT CREATE SESSION TO Lorea;

Loreari lan-saio bat sortzeko baimena eman dio Txominek.

GRANT CONNECT TO Lorea;

Loreari datu-basera konektatzeko baimena eman dio Txominek.

CONNECT Lorea/lore;

Lorea datu-basera konektatu da; beraz, uneko erabiltzailea Lorea izango da. Erabiltzailea ezabatu nahi izanez gero, hori egiteko baimena eduki behar du uneko erabiltzaileak. Aurreko kasuan bezala, printzipioz baimen hori DBAk baino ez du, baina badago aukera erabiltzaileak ezabatzeko baimena beste erabiltzaile batzuei ere emateko. Beste erabiltzailearen batek baimen hori jasotzen badu, erabiltzaileak ezabatu ahal izango ditu. Horretarako, DROP USER sententzia erabili beharko du. Adibidez:

DROP USER Lorea;

Lorea erabiltzailea ezabatu da.

DROP USER Lorea CASCADE;

Lorea erabiltzailea ezabatu da atxikita dituen elementu guztiekin, hau da, dituen erlazio guztiekin. Erabiltzaileen aukerak, ALTER USER sententzia erabiliz alda daitezke. Adibidez:

ALTER USER Lorea IDENTIFIED BY loyo;

Lorea erabiltzaileari pasahitza aldatu zaio; orain, pasahitza 'loyo' izango da.

DBKSak ezagutzen dituen erabiltzaile guztien informazioa datu-baseko ALL_USERS ikuspegian islatzen da. Ikuspegi horren gainean DBAk SELECT * FROM ALL_USERS eginez gero, erabiltzaile bakoitzaren izena, sistemak banatzen dion identifikazioa eta sorrera-data erakutsiko dituen zerrenda lortuko da.

6.3.5. Rolak

Rol batean baimen jakin batzuk multzokatzen dira; horrela, erabiltzaileari baimenak banan-banan eman beharrean, baimen-multzo osoa sententzia bakar baten bidez eman ahal zaio. Kontuan hartu behar da, halaber, rola sortzen duen erabiltzaileak baimenduta egon behar duela rolak sortzeko. CREATE ROLE sententzia erabiltzen da horretarako. Adibidez:

CREATE ROLE erabili;

ERABILI izeneko rola sortu da.

GRANT SELECT, INSERT, DELETE ON erlazio2 TO erabili

ERLAZIO2 erlazioaren gainean SELECT, INSERT eta DELETE eragiketak egiteko baimena eman zaio ERABILI rolari.

GRANT CREATE SESSION, CONNECT TO erabili;

Lan-saioa sortzeko eta datu-basera konektatzeko baimena eman zaizkio ERABILI rolari.

GRANT ERABILI TO Ane;

Aneri ERABILI rola eman zaio. Beraz, datu-basera konektatzerakoan, SELECT, INSERT eta DELETE egin ahal izango du ERLAZIO2 erlazioaren gainean. Rolari baimen bat kentzeko, REVOKE sententzia erabiliko da. Adibidez:

```
REVOKE DELETE ON erlazio2 TO erabili;
```

ERABILI izeneko rolari ERLAZIO2 erlazioaren gainean DELETE egiteko baimena kendu zaio; beraz, rol hori esleituta zeukaten erabiltzaileek galdu egin dute baimen hori. Rolak ezabatzeko, DROP ROLE sententzia erabiltzen da. Hona hemen adibidea:

```
DROP ROLE erabili;
```

ERABILI izeneko rola ezabatu denez, bere erabiltzaile guztiek rolaren baimenak galduko dituzte.

Aipatu beharra dago erabiltzaile bati esleitutako rolak edo rolek barneratzen dituen baimenen informazioa datu-hiztegia islatzen dela. Aipatutako informazioa gordetzen duten *Oracle*ko ikuspegiak 6.2 taulan agertzen dira:

IKUSPEGIAREN IZENA	IKUSPEGIAK EMATEN DUEN INFORMAZIOA	IKUSPEGIAREN ZUTABEAK
ROLE_TAB_PRIVS	Elementuen gaineko baimenak barneratzen dituzten rolak	ROLE: Rolaren izena OWNER: Objektuaren (erlazioa) jabea TABLE_NAME: Erlazioaren izena COLUMN_NAME: Zutabearen izena PRIVILEGE: Baimena GRANTABLE: Baimena banatzeko baimena
ROLE_SYS_PRIVS	Sistemaren gaineko baimenak barneratzen dituzten rolak	ROLE: Rolaren izena PRIVILEGE: Baimena ADMIN_OPTION: Pribilegioa banatzeko baimena
USER_ROLE_PRIVS	Uneko erabiltzaileak jasotako rolak	USERNAME: Erabiltzailea GRANTED_ROLE: Jasotako rola ADMIN_OPTION: Pribilegioa banatzeko baimena DEFAULT_ROLE: Rol lehenetsia

6.2 taula: Hiztegia dauen rolen ikuspegiak.

6.3.6. Profilak

DBKSkon baliabideei mugak jartzeko erabiltzen dira profilak. *Oracle*en profilak sortzen dira baliabideen erabilerari murrizketak jarri nahi izanez gero. Ondoren, profil horiek erabiltzaileei banatu beharko zaizkie. Baliabideen gainean jar daitezkeen mugak, besteak beste, hauek dira:

- SESSIONS_PER_USER: erabiltzaile bakoitzak izan ahal dituen aldi bereko lan-saio askoren kopurua.
- CPU_PER_SESSION: lan-saio bakoitzeko PUZaren denbora murrizten du.
- CONNECT_TIME: lan-saio bakoitzeko denbora maximoa zehazten du.

- IDLE_TIME: jardun gabeko denbora maximoa zehazten du.
- FAILED_LOGIN_ATTEMPTS: zuzenak ez diren ekinaldien kopuru maximoa zehazten du.
- PASSWORD_LIFE_TIME: pasahitzaren baliozkotasun-epe maximoa zehazten du.

Lan-saioetan muga horietakoren bat gaintzen bada, lan-saioa moztu egingo da. Profilak sortzeko CREATE PROFILE sententzia erabiliko da. Demagun P1 profila sortu nahi dela:

```
CREATE PROFILE p1
LIMIT SESSIONS_PER_USER 1
CONNECT_TIME 45
IDLE_TIME 5;
```

Bestalde, ALTER USER sententzia erabiltzen da erabiltzaile bati profila esleitzeko. Adibidez:

```
ALTER USER Lorea
IDENTIFIED BY loyo
PROFILE p1;
```

LOREA erabiltzaileari P1 profila esleitu zaio. Profilak ezabatzeko, DROP PROFILE sententzia erabiltzen da. Adibidez, P1 profila ezabatzeko:

```
DROP PROFILE p1;
```

6.4. ATZIPEN KONTROLAREN AUDITORIA

Datu-baseak epealdi batean duen jarduera behatu behar izaten da askotan. Horretarako, DBAK hainbat informazio jasoko du: jardunaldiko tarte batean datu-basera konektatu den erabiltzaile-kopurua, zein erlazio (edo zeintzuk) eta nola erabili diren (irakurketa, idazketa...), zein ordenagailutan huts egin duten konexioek...

Behaketa hori denbora errealean egitea ia ezinezkoa da. Horregatik, ezinbestekoa egiten da DBKSak auditoriarako fitxategi bereziak sortu eta mantentzea. Bertan, datu-basean bideratutako hainbat ekin-tzaren informazioa jasoko da.

- Datu-baserako sarbideen auditorian, tarte zehatz batean konektatzeko egindako ahalegin guztiak (bai arrakasta izan dutenak, bai kale egindakoak) gordetzen dira.
- Ekintzen auditoriak: datu-baseko elementuetan aldaketak eragin dituzten ekintzak (CREATE, ALTER...) kontrolatzeko daude.
- Elementuak atzitzeko auditoria. Datu-baseko erlazioetan egindako ekintzak kontrola ditzake (SELECT, UPDATE, INSERT, DELETE).

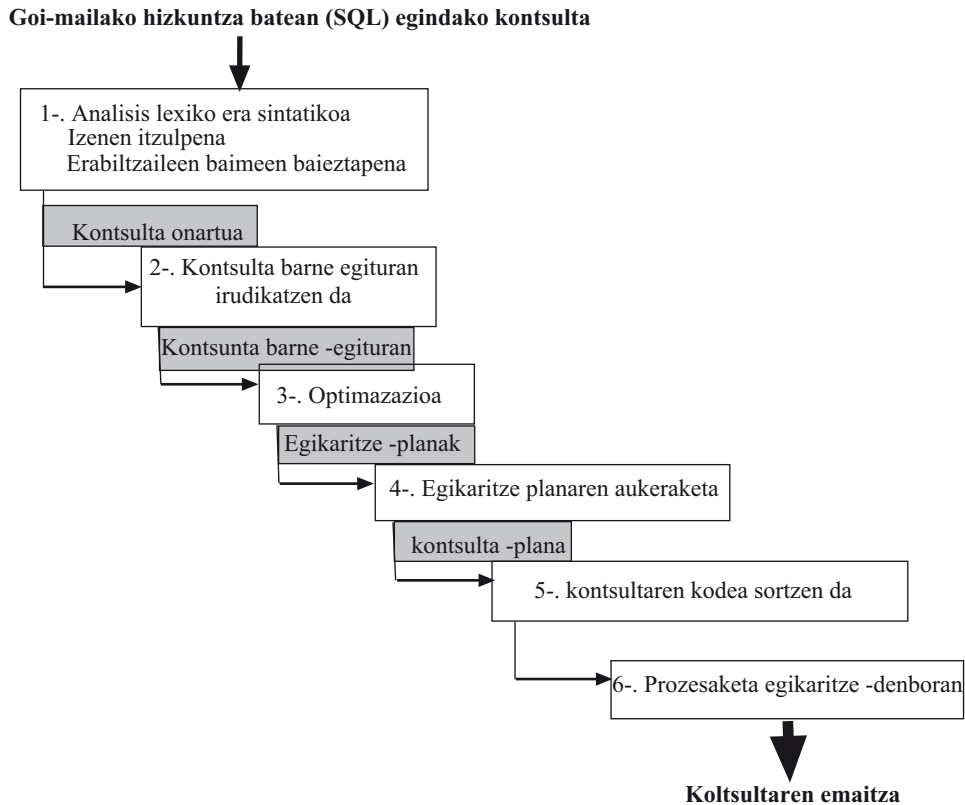
Kontrolatutako ekintza bakoitzeko informazioa gordeko duenez, auditoria-sistema erraz gainkargatzen da. Segurtasunaren eta errendimenduaren arteko konpromisoa lortu beharko da, errendimenduaren murrizketa orekatuta egon dadin segurtasunean lortutako hobekuntzekiko.

*Datu-baseen
optimizazioa*

7

7.1. KONTZEPTUA

Datu-baseen gainean etengabeen DDL-DML eragiketak egikaritzen dira eta une askotan eragiketa horiek guztiak paraleloan doaz. Sistema arin eta azkar joan dadin, beharrezkoa da sententzia horiek ahal den modurik optimizatuenean egikaritzea. Datu-basean egiten diren kontsultek hainbat fase bete behar dituzte, goi-mailako DMLa (SQL adibidez) erabiliz idazten direnetik egikaritzen diren arte. Optimizazio-prozesua da fase horietako pauso garrantzitsuenetako bat. Gainera, beste faseetan ere optimizazio-teknikak erabiliko dira ahal den neurrian (ikusi 7.1 irudia).



7.1 irudia. Datu-baseetako kontsultetan betetako faseak.

Lehenengo fasean, edozein programazio-lengoaia egiten duen bezala, idatzitako kontsultari analisi lexikoa eta sintaktikoa egiten zaizkio, kontsultak sintaxi egokia erabiltzen duela egiaztatzeko.

Gainera, uneko erabiltzaileak kontsulta hori egiteko baimenik duen edo ez egiaztatuko da, datu-basearen administratzaileak erabiltzaileen artean hainbat baimen-mota bana baititzake.

Fase horren ostean, barne-formatura bihurtuko da kontsulta. Adibidez, datu-base erlazionalen kasuan, goi-mailako lengoaia batean dagoen kontsulta aljebra erlazionalan edo kalkulu erlazionalan adieraziko da.

Ondoren, barne-egitura horretan dagoen kontsulta optimizatu eta egikaritze-planak sortuko dira. Egokiena aukeratu, lehenik kontsulta egiteko barne-kodea sortuko da eta, azkenik, kontsulta egikaritzeko da, emaitzak lortzeko.

DBKSaren erantzun-denbora hobetzea da kontsulten optimizazioaren helburu nagusia.

Sistemari dagokion optimizazio-prozesuak, barne-adierazpena, optimizazio-fasea bera eta egikaritze-planaren hautapena barneratzen ditu (ikusi 7.2 atala).

Testu honetan kontsulta hitza erabili da DML-DDL sententziak adierazteko. Ez da ahaztu behar optimizazioa eragiketa guztiei aplikatzen zaiela, ez soilik kontsultei (SELECT), nahiz eta orokorrean horiek izan datu-baseen inguruan gehien egikaritzen diren sententziak.

7.2. OPTIMIZAZIOAREN ARDURADUNAK

Optimizazioaz arduratuko dira:

- Datu-basearen diseinatzailea. Datu-basea diseinatzerakoan, hobeto da gero egingo zaizkion galdeketei buruzko ahalik eta informazio gehien izatea. Adibiderik garbiena desnormalizaziorako teknikek osatzen dute.
- Datu-basearen administratzailea. Erabiltzaile berezi honek dituen ahalmenei esker, erabaki asko har ditzake. Askotan, diseinatzerakoan kontuan hartu ez direnak konpontzeko gai da. Adibidez, datu-basearen erabileraren auditoria egitean, indize berrien beharrari erantzutea.
- Erabiltzaile arruntak. Datu-baseen sistema barruan mailarik apalena duten erabiltzaileak izan arren, beraien artean ere hainbat kasu daude. Adibidez, batzuek taulak, indizeak... sortzeko gaitasuna dute eta beste batzuek ez. Dena den, mailarik apalenerako erabiltzaileek ere kontsulta egiten duten moduaren arabera (indize batzuk erabiliz nahiz eta beraiek sortuak ez izan) sistema askoz arinago joatea lortuko dute.
- Datu-baseen sistemaren ardurapean dagoen optimizazioa. Aljebra erlazionalaren potentzia guztia aplikatzen zaio kontsulta bakoitzari, ahalik eta gehien optimizatzeko.

Jarraian garatuko dira sakonago puntu horiek.

7.2.1. Datu-basearen diseinatzailearen optimizazioak

Datu-basea diseinatzerakoan, gerora izan daitezkeen arazo asko saihestu daitezke. Horretarako, aldeztatik eduki behar da ahal den informazio gehien (ikus 9.3 puntua).

Aldeztatik hartutako informazio horrekin, diseinatzaileak honako hau egin behar du:

- **Ongi diseinatu, bai erlazioak, bai atributuen domeinuak.** Erlazioen eta ikuspegiaren diseinu egokiak asko hobetzen du sistemaren performantzia (errendimendua). Gainera, domeinuak ongi aukeratzen badira, datu-baseak disko gogorrean gutxiago okupatuko du. Aldi berean, segurtasun-tresna ere badira. Azken finean, sistemari etekin hobea aterako zaio. Murrizketak ere kontuan izan beharko dira eta, horretarako, CHECK motako klausulak daude. Murrizketen bitartez, kasu batzuetan gerta liteke optimizatu beharrean denbora alferrik galtzea (osotasuna bermatzeko); hala ere, epe luzean, murrizketarik ez erabiltzea baino hobea da.

```
CREATE DOMAIN Sexua AS CHAR (1) CHECK (VALUE IN ('G','E'))
```

SEXUA izeneko domeinu berria sortu da eta berak har ditzakeen balio bakarrak *G* (Gizonezkoa) edo *E* (Emakumezkoa) dira. CHECK erabili ez balitz, edozein hizkik beteko luke (CHAR (1)), baina CHECKek hizki horiek *G* ala *E* izatera behartzen du.

- **Indizeak ongi diseinatzea.** Hasieratik indizeak jartzen dira gako nagusien eta gako atzerritarren gainean. Adibide bat erabiliko da hobeto azaltzeko. Demagun hiru erlazio hauek daudela:

IKASLEAK (NA, Izena, Abizena, Herria, Telefonoa)

IKASGAIK (IkasKodea, Izena, IrakasleIzena)

IKASLEA-IKASGAIA (NA, IkasKodea, Nota)

Jo dezagun ikasle batek ikasgai asko dituela eta bakoitzeko, nota bat. Kasu horretan, IKASLEAK erlazioko gako nagusia 'NA' da eta indizatu egingo da; gauza bera gertatuko da IKASGAIK erlazioko 'IkasKodea' eremuarekin.

Baina zer gertatzen da IKASLEA-IKASGAIA erlazioarekin? Kasu horretan, gakoa 'NA'-'IkasKodea' bikotea da. Indizea bikote horri jarriko zaio, baina arazoa sortzen da ordenarekin. 'NA'-'IkasKodea' ala 'IkasKodea'-'NA' aukeratzea komeni da indize gisa? Indizeak sortzerakoan, ordena-aldaketa txiki horrek sistemako performantzia eragin handia izango du. Bien artean ongi aukeratzeko, diseinatzaileak informazio gehiago behar du. "Ikasgai jakin bateko notak" bezalako kontsultak sistemari nagusi badira, 'IkasKodea'-'NA' indizea hobetsiko da; kontsultak "ikasle jakin batek ateratu dituen notak" motakoak badira, hobe da 'NA'-'IkasKodea' indizea sortzea. Edo, agian, komeni izan daiteke bi indizeak sortzea.

Gainera, sistemari buruzko informazio gehiago badago, indize gehiago sortzea ere pentsa daiteke. Adibidez, ikasleak lehenik abizena eta izena sartzen ditu eta ondoren ateratzen zaizkio notak. Kasu horretan, mota horretako kontsulta asko egonez gero, beste indize bat sor daiteke, IKASLEAK erlazio horren gaineko bigarren indize bat sortuz. Indize hori 'Abizena' atributua edo 'Abizena'-'Izena' atributuen gainean sor daiteke. Adibidez:

```
CREATE INDEX AbizenaIndizea ON IKASLEAK (Abizena)
```

Eta "AbizenaIndizea" izeneko indizea sortuko da IKASLEAK erlazioko 'Abizena' atributuaren gainean.

Indize gehiago sortzeko prozesua ezin da mugarik gabe burutu. Oro har, erlazio bakoitzak ez ditu bi edo hiru indize baino gehiago izaten. Arazoa eguneraketetan sortzen da. Zenbat eta indize gehiago, erlazioko errendimendua bakoitza ezabatu edo txertatzerakoan, eragiketa gehiago egin behar dira eta horrek sistemaren errendimendua jaisten du. Kontsultak bizkortu / eguneraketak geldotu bi aldagaien erlazioen arteko orekak ezartzen du muga.

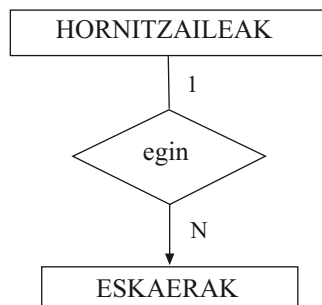
Salbuespen gisa, *Data Mining* egiteko datu-baseak ditugu. Eurotan kontsulta masiboak egiten dira, eta, orokorrean, txertatzeko edo ezabatzeke eragiketarik ez dagoenez, nahi izanez gero indize asko sor daitezke.

Esan bezala, indizeak sortzeko informazio gehiago behar da. Horregatik, askotan, datu-basearen diseinua egiterakoan, oraindik ezin izango dira indize guztiak aurreikusi. Datu-baseko administrariak beharrezko indizeak sortuko ditu aurrerago.

- **Eremu kalkulatuak erabiltzea.** Eremu kalkulatu datu-base batean jada agertzen diren beste eremu batzuekin sortutako eremua da. Erreduantzia dakar, baina batzuetan komenigarri gerta daitezke, DBKSak kontsulta egiterakoan behar duen denbora murrizten baitu.

Adibide batekin ikusiko da argiago (ikus 7.2 irudia). Demagun HORNITZAILEAK eta ESKAERAK erlazioak daudela, non hornitzaile bati eskaera asko egiten zaizkion. Datu horiekin diseinatutako erlazioetan, ESKAERAK erlazioan HORNITZAILEAK erlazioko gako nagusia egongo da, 1:N

erlazioa adierazteko. Ereku kalkulatu HORNITZAILEAK erlazioan sortutako eremu berria izango da. Ereku berriaren izena 'EskaeraKopurua' izan daiteke eta bertan hornitzaile horri egindako eskaera-kopurua agertuko da. Izatez eremu hori ez da beharrezkoa, SELECT COUNT (*) FROM eskaerak WHERE HorniKodea = "XXXXX" eginez datu berbera lortuko baita. Ereku kalkulatu hori aurretik izanda, denbora irabaziko da; baina, hori bai, eskaera gehitzailekoan, 'EskaeraKopurua' eremua eguneratu beharko zaio dagokion hornitzaileari.



7.2 Irudia. Hornitzaileak eskaera asko izan ditzake. E-R diagrama.

Hasieran (eremu kalkulatu gabe):

HORNITZAILEAK (HorniKodea, Izena, Abizena, Herria)

ESKAERAK (EskaeraKodea, HorniKodea, ProduktuIzena, Kopurua, Prezioa)

Ondoren (eremu kalkulatuak erabiliz):

HORNITZAILEAK (HorniKodea, Izena, Abizena, Herria, EskaeraKopurua)

ESKAERAK (EskaeraKodea, HorniKodea, ProduktuIzena, Kopurua, Prezioa)

Beste adibide honetan, BILTEGIA izeneko erlazioa dago, eta bertan, biltegiko produktuen zerrenda. 'ProduktuKopurua' eta produktuaren 'Prezioa' dira erlazioko bi eremu. Kasu kasu, baliteke komeni izatea aurreko bi balioen biderkadura soila den 'GuztiraBalioa' izeneko eremua sortzea.

Hasieran (eremu kalkulatu barik):

BILTEGIA (ProdKodea, ProdIzena, Kokapena, Kopurua, PrezioErosi, PrezioSaldu)

Ondoren (eremu kalkulatuak erabiliz):

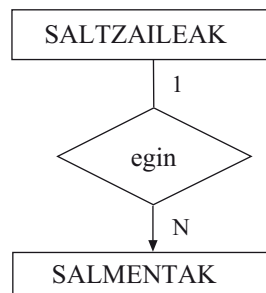
BILTEGIA (ProdKodea, ProdIzena, Kokapena, Kopurua, PrezioErosi, PrezioSaldu, GuztiraBalioa)

Ez da ahaztu behar teknika horrek dakarren arazoa. Datuen eguneraketak, txertaketak, ezabaketak... zailtzen dira, bi erlazio edo gutxienez bi eremu eguneratu behar baitira. Kontuz ibili behar da, bestela datu-basea azkenean ez da-eta errealitatearen ispilu izango. Kasu horietan, osotasuna mantentzeko prozedurak, *triggerak*, CHECK motako sententziak eta abar erabiliko dira.

- **Desnormalizazio-teknikak erabiltzea.** Datu-baseen diseinu egokia lortzerakoan bete beharreko pausoetariko bat eredu normalizatua lortzea da; hau da, erredundantzia minimoa duen eredu. Hori horrela den arren, baliteke desnormalizazio-prozesuren bat eraman behar izatea geroago, batzuetan informazioaren erredundantziak kontsulten optimizazioa baitakar. Beraz, diseinatzaileak normalizazioaren eta optimizazioaren arteko oreka lortu beharko du.

Eremu kalkulatuaren teknika ere desnormalizazio-teknika horren azpimultzo modura har daiteke, azkenean ideia bera baita.

Desnormalizazio-tekniken adibidetzat, demagun bi erlazio daudela (ikusi 7.3 irudia): SALTZAILEAK eta SALMENTAK. SALTZAILEAK erlazioan saltzaileen datuak agertzen dira eta SALMENTAK erlazioan, berriz, hilabetez hilabete azken urtean egin dituzten salmentak. Saltzaile bakoitzeko, gehienez, 12 errenkada egongo dira SALMENTAK erlazioan. Agian, desnormalizazioa aplikatzea komeniko da, eta saltzailearen beste 12 zutabe edo eremu berri sor daitezke azken 12 hilabeteetako datuak beraietan sartzeko. Datu-basearen performantzia hobetu egingo da eta SALMENTAK erlazioa ezabatuko da.



7.3 Irudia: Saltzaile batek salmenta asko egiten ditu. E-ER diagrama.

Hasieran (eremu kalkulatu gabe):

SALTZAILEAK (SaltzKodea, Izena, Abizena, Herria)

SALMENTAK (SalmenKodea, SaltzKodea, SalmentaKopurua)

Ondoren (eremu kalkulatuak erabiliz):

SALTZAILEAK (SaltzKodea, Izena, Abizena, Herria, EskaeraKopurua, SalMenUrt, SalMenOts, SalMenMar, SalMenApi, SalMenMai, SalMenEka, SalMenUzt, SalMenA-bu, SalMenIra, SalMenUrr, SalMenAza, SalMenAbe)

SALMENTAK erlazioa, orain, ez da behar.

7.2.2. Datu-basearen administratzailearen optimizazioak

Diseinatzaileak erabil ditzakeen optimizazio-teknika berberak aplikatu ditzake administratzaileak; hau da, domeinuen eremuak, indizeen diseinuak, eremu kalkulatuak eta desnormalizazio-teknikak. Jakina da diseinatzaileak izan ez duen informazioa duela administratzaileak. Indize berriak sortzeko edo zaharrak ezabatzeko aukera izango du, horretarako denbora errealean egindako estatistikak erabiliz. Diseinuan ageri izan ez diren domeinuak eta murrizketak ere ikusiko dira. Desnormalizatorako eta eremu kalkulatuertarako teknikak konplexuak izan daitezke administratzailearentzat, izatez, diseinuko zereginak dira eta. Adibidez, administratzaileak desnormalizazio-teknikak erabiltzen baditu, sistemako programatzaileei datu-basean aldaketak egon direla esan beharko die, horiek beraien programa berrietan eraldaketak aprobeitatu ditzaten. Kasu batzuetan, ikuspegiak erabiliz ezkuta daitezke eraldaketak, baina ez da beti posible izango. Batzuetan, desnormalizazio-teknikak ezin izango dira aplikatu prozedurak berridatzi gabe, datu-baseko programa zaharren bateragarritasuna dela eta.

Horiez gain, administratzaileak optimizaziorako dituen teknika propioak honako hauek dira:

- **Datu-basea purgatzea.** Datuetako batzuk jada ez badira eskatzen, hobe da artxibatzea. Ondorioz, erlazioak hustuago egongo dira (tuplo gutxiagorekin) eta bilaketak bizkorrago egingo dira.
- **Bezeroen eta zerbitzariaren arteko lan-kargaren banaketa.** Bezero-zerbitzari inguruneetan kontsultaren egikaritza banatu egin daiteke. Negozio-arauen kontrolaren banaketa modura ere ezagutzen da. Lanetako batzuk, hala nola datuen aurkezpena, bezeroan egiten dira. Beste lan batzuk, erabiltzailearen identifikazioa esaterako, zerbitzarian egiten dira. Baina negozio-arauen kontrola sistemako bi aldeetan egin daiteke. Bi aldeetako zeinetan egin, sistemaren azken performantzia izango du eragina; beraz, azken kostuko kalkulak egiterako orduan, kontuan izan beharko dira bai bezeroan egiten diren baliabideen kontsumoa, bai zerbitzarian egiten direnena, bai sareko baliabideena.
- **Konexioen iraupena mugatzea.** Programa gaizki egin bada edo erabiltzailea ordenagailu aurrean ezer egin gabe geratzen bada (Interneten maiz gertatzen da), baliabide asko erabili gabe mantentzen ditu konexiora lotuta. Beraz, geldirik dauden konexioen iraupena mugatu egin beha da. Iraupena ezberdina izango da datu-base bakoitzaren arabera; orokorrean, Interneten konexioen iraupena datu-basekoa baino luzeagoa izan ohi da erakundeetan.

7.2.3. Datu-basearen erabiltzailearen optimizazioak

Datu-baseetan erabiltzaile mota asko daude, administratzaileak GRANT moduko klausulak erabiliz emandako baimenen arabera. Atal honetan, erabiltzaile-mota bi bereiziko dira: programatzaileak eta erabiltzaile arruntak. Sakonago aztertuko ditugu ondoren.

Programatzaileek eragin handia dute performantzia. Administratzaileak eta diseinatzaileek erabilitako optimizazio-teknikak ez dira ia nabariturako programatzaileek egiten dituzten programetan, optimizazio horiek erabiltzen ez badira. Beraz, hobekuntza horien berri programatzaileei eman behar zaie. Adibidez, orain erlazio batzuek eremu kalkulatuak dituztela ohartu... Horiez gain, badira programatzaileek soilik erabiltzen dituzten optimizazio-teknikak:

- **Aurkezpenak mugatzea.** Erregistroen aurkezpena zenbaki txiki batera mugatzean datza, kontsultaren bukaeran LIMIT aukera erabiliz. Adibidez, ezagunak dira Interneteko bilaketetan izaten diren emaitzak. Milaka izaten dira, baina ez dira guztiak batera aurkezten; gehienetan, hamarnaka aurkezten dira.
- **Konexioen iraupena.** Konexioak ahal den laburren egin behar dira; beraz, ahal den neurrian, konexioa itxi ondoren egin behar dira datuen tratamendua eta aurkezpena. Horretarako, kontsultaren emaitzak gordeko dituzten aldagaiak erabiliko dira, datuen tratamendua eta aurkezpena egiteko.
- **Kontsulten pilaketa.** Elkarren artean zerikusirik ez duten kontsultak badaude, programaren hasieran pila daitezke, guztientzat konexio bakarra erabiliz (ikusi 7.1 testu-taula).

Programaren hasiera

Aldagaien identifikazioa: aldagaia1, aldagaia2, aldagaia3

...

Datu-basera konexioa egin.

Aldagaia1=Lehen Kontsulta

Aldagaia2=Bigarren Kontsulta

Aldagaia3=Hirugarren Kontsulta

...

Datu basetik deskonektatu.

Aurkezpena egin...

Datuak tratatu (aldagaia1, aldagaia2, aldagaia3)

Aurkezpen gehiago...

...

Programaren bukaera

7.1 testu-taula. Programen pausoak, hasieratik bukaerara.

Esan bezala, kontsultak pilatu eta beharrezkoak ez diren konexioak kendu egin behar dira. Konexio gutxi eta laburrak (denboraren ikuspuntutik) erabili nahi dira, nahiz eta batzuetan ezinezkoa gertatu. Bi baldintza horiek kontrajarriak dira eta zeresan handia dute sistemaren azken performantzian.

- **Cachea sortzea.** Datu-basetik hartu beharreko datuek eguneraketa askorik ez badute, *cache* batean sar daitezke. Adibidez, Interneteko bilatzaile baten kontsulta egitean milaka erantzun ematen ditu. Aurkezpenak mugatzeko teknika erabiliz (SQL sententzietan LIMIT erabilia, adibidez) lehen hamarrak soilik aurkeztuko dira. Hala ere, kontsultaren emaitza guztia *cache* batean edo, kasuan kasu, aldagai batean gorde daiteke. Hurrengo hamarrak eskatzean, ez da kontsulta berriz egikaritu behar.
- **Kontsultetan behar dena soilik hartzea.** Ahal den neurrian, ez da SELECT * erabili behar eta erlazioen arteko loturak ipintzea komeni da (WHERE Erlazioa1.Eremua1 = Erlazioa2.Eremua2). Egia da kasu askotan bestela ere emaitza egokiak lortuko direla, baina denbora luzea hartuko du.

Lan-karga handia dakarten kontsultekin, ahal izanez gero, HAVING eta GROUP BY moduko aukerak saihestu egin behar dira. Kontuz ibili behar da LIKE '%zerbait%' egitearekin ere. Batzuetan ez dago beste aukerarik, baina ez beti.

Erabiltzaile arruntaren ikuspuntutik optimizaziorako aukera gutxi dago. Euron optimizazio-teknika nagusia programak era egokian erabiltzean datza. Nahiz eta programak ahal diren optimizazio-teknika guztiak aplikatuz egin, erabilera ez egokiak performantzia jaisten dute askotan. Adibidez: zerbitzarira konektatuta gaudela kafe bat edatera joan edo ordenagailua bera itzali (aurretik lan-saioa itxi gabe itzaliz gero, zerbitzariarentzat irekita jarraitzen du). Bi kasu horietan konexioa ez dugu ixten eta zerbitzariko baliabideak hartuta gera daitezke. Zerbitzaria ongi administratua badago, denbora batean geldi dauden konexioak automatikoki itxiko ditu (ikusi administrariaren optimizazio-teknikak), baina bitartean programetik era ez egokian irteteagatik, performantzia galduko da.

7.2.4. Datu-basearen sistemaren ardurapeko optimizazioak

Datu-baseetarako eredu guztien artean DBKS erlazionalak dira optimizazio-prozesua modu formalenean tratatu duten lehenengokoak (oinarrian aljebra eta kalkulu erlazionalarekin). Erabiltzailearen kontsultak goi-mailako hizkuntzan egiten dira (SQL) eta datu-basean egiten diren eragiketen artean dagoen tarte oso handia da. Tarte horretan aplikatzen da sistemaren ardurapeko optimizazio-prozesua. Optimizazioaren ikuspuntutik, hortxe lortzen dira hobekuntzarik onenak.

Kontsulten optimizazioaren helburua DBKSaren erantzun-denbora hobetzea den arren, kasu gutxi batzuetan optimizazioa jorrazteko orduan, baliteke optimizazioak berak baliabide edo denbora gehiago behar izatea, optimizatu beharreko kontsultan lortuko den denbora-murrizketa baino. Horregatik, bi aspektu horiek izan behar dira kontuan:

- Alde negatiboa: optimizatorako prozedura berez beharko duen baliabide-kopuruan datza. Prozesu horrek ere bere denbora eta baliabideak behar ditu; izan ere, delako optimizazioak fase jakin batzuk barneratzen ditu: kontsulta barne-adierazpide batera itzuli, manipulazio aljebraikoak erabiliz barne-adierazpidearen bihurtzea egin (erregela logikoak erabilita), kontsulta egiteko behe-mailako prozeduren hautapena... Tarteko informazioa diskoan gordetzearen kostua, datuak bezerotik eskatu eta zerbitzaritik bidaltzearen komunikazio-kostua eta konputazio-baliabideen kostua dira kontuan izan beharreko aldagaiak.
- Alde positiboa: kontsulta optimizatu ostean, kontsultaren denbora murriztu egiten da. Kontsulta egiteko, beharrezkoak diren erlazioen tamaina sarritan txikitu egiten da optimizazio baten ondorioz. Hori dela eta, informazio bera lortzeko, irakurketa gutxiago egin behar dira, eta emaitza partzialak memoria printzipalean gordetzea posible da, diskora atzipenak murriztuz. Irakurri beharreko informazioa gutxitzen bada, bezeroen eta zerbitzariaren arteko komunikazioa txikiagoa izango da, eta sistemaren performantzia hobetu egingo da.

Bi aspektu horien azterketa egin ondoren, sistemak optimizazio-prozesua egitea merezi duen erabakiko du.

Optimizazioaren prozesua hobeto ulertzeko, adibide bat erabiliko dugu. Jo dezagun 5. gaiko 5.1, 5.2 eta 5.3 erlazioak daudela:

BEZEROA (NifBez, Izena, Abizena, Helbidea)

ALOKAIRUA (ZenbAlok, FilmKodea, Data, NifBez)

FILMA (FilmKodea, Izenburua, Kopurua, Prezioa)

Demagun 1.000 film eta 100.000 alokairu daudela eta bezeroaren NIFtzat 11111111L duen bezeroak alokatutako filmen kontsulta egin nahi dela:

```
SELECT Izenburua
```

```
FROM FILMA, ALOKAIRUA
```

```
WHERE FILMA.FilmKodea=ALOKAIRUA.FilmKodea
```

```
AND NifBez='11111111L'
```

Optimizazio gabe, erantzuna ondoren azaldutakoa litzateke: lehenik, FILMA eta ALOKAIRUA erlazioen arteko biderketa kartesiarra egingo da. $1.000 \times 100.000 = 100.000.000$ errenkada sortuko dira eta horrek, ziurrenik, memoria nagusian dugun espazioa beteko du. Beraz, erregistroak diskoan

gordeko dira, diskoa tarteko memoria gisa erabiliz (alegiatzko memoria kontzeptua). Bigarren pausoa, banan-banan erregistro horiek guztiak arakatu eta WHERE klausula betetzen duten erregistro-kopurua lortzea da. Bezero horrek 50 film alokatu dituela jotzen bada, lehengo 100.000.000 irakurketak egingo dira, azkenean, 50 errenkadarekin geratzeko. Azken pausoa 50 film horien izenburua lortzea; hau da, aukeratutako 50 errenkada horiei proiektzioa aplikatu eta izenburuak lortzea izango da; beraz, 50 irakurketa gehiago. Guztira egin diren irakurketak: $100.000.000 + 100.000.000 + 50 = 200.000.050$ irakurketa. Eta hori guztia errenkaden tamaina alde batera utzita (ez da gauza bera 500 byteko errenkada edo 125 bytekoa).

Orain optimizazio-prozesuak erabiltzen dituen sistema bat aztertuko dugu: lehenik, bezeroaren NIFtzat 11111111L duten alokairuak aukeratuko dira. 100.000 irakurketa egingo dira, baina azkenean 50 besterik ez dira geratuko. Bigarren pausoa, ALOKAIRUA erlaziotik lorturiko 50 errenkaden eta FILMA erlazioaren arteko biderketa cartesiarra egitea da: $50 \times 1.000 = 50.000$ errenkada bueltatuko ditu. Errenkada-kopuru hori memoria nagusian sar daiteke. Hirugarren pausoa, geratzen den WHERE klausulako zatia aplikatuko da. 50.000 irakurketa, azkenean 50 errenkadarekin geratzeko. Oraingoan ere azken pausoa 50 errenkada horiei filmen izenburua lortzeko proiektzioa aplikatzean datza. 50 irakurketa gehiago. Azkenean egindako irakurketa-kopurua: $100.000 + 50.000 + 50.000 + 50 = 200.050$ irakurketa.

Abantailak argi eta garbi ikusten dira, sistema optimizatuko irakurketa-kopurua ia 1.000 aldiz txikiagoa baita; gainera, eragiketa guztiak memoria nagusian egin daitezke.

Optimizazio-prozesuak modu orokorrean ikusi behar du DBKSaren funtzionatzeko prozesua. Datu-base erlazionalak egitura matematikoa euskarritzat duenez, optimizazio-prozesua optimizazio aljebraikoan datza. Honako fase hauek barneratzen ditu:

- 1) Goi-mailako lengoia idatzitako kontsulta itzuli egiten du aljebra erlazionalera edo kontsultazuhaitz modura ezagutzen den zuhaitz sintaktiko abstraktu.
- 2) Egikaritzeko orduan, aljebra erlazionalan edo zuhaitz sintaktikoan adierazitako kontsulta baino eraginkorragoa den bat aurkitu; horretarako, aljebra erlazionalerako erregelak eta transformazio logikoak erabiltzen dira. Estrategiaren pauso nagusiak honako hauek dira: hautapenerako eragiketak ahal bezain bizkor egin; egin daitezkeen proiektzioak aurreratu; hautapen eta proiektzio zerrenda bat dagoenean, banatu; lehenik, hautapen simple bat egin eta, ondoren, beste proiektzio simple bat. Hlburua ahal den neurrian erlazio bati dagozkion eragiketak lehenbailehen egitean datza. Beste pauso bat emaitza berberak emango dituzten azpikontsultak detektatzean datza, horrela konputazio-prozesu guztia ez dago errepikatu beharrik eta. Fase horretan lortuko den kontsultak (aurreragoko adibidea ikusi) hasierako kontsultak baino irakurketa gutxiago egingo ditu.
- 3) Aurreko fasean, kontsultari emandako optimizazio logikoa eta gero, informazioa biltegiturata dagoen moduaren arabera, hau da, barneko eskemaren arabera (erregistroak nola dauden taldekatuta, indizeak existitzen diren ala ez), kontsulta burutzeko behe-mailako prozedurak hautatuko dira (7.1 irudiko 4. fasea).
- 4) Kontsulta planifikatzailearen azken pausoa, behe-mailako prozeduren konbinazio posibleak aztertu ostean, kontsulta egiteko plana hautatzea da. Prozeduretako konbinazioen araberrako planak egingo dira. Atzipen-kostu txikiena duen plana aukeratuko da, honako aldagai hauek kontuan hartuz: PUZaren erabilera txikiena, sarrera/irteera eragiketa-kopuru txikiena eta datu-base banatuetan komunikazio-sarea gutxien erabiltzen duena. Arrazoiak erabiliz estrategia bat aukeratu nahi bada, DBKSak eskemako erlazio bakoitzeko erabilera-estatistikak izan behar

ditu; beraietan erlazio bakoitzak duen tuplo-kopurua, eremu-kopurua, atributuen domeinuei buruzko informazioa... etab. egongo da. (7.1 irudiko 4. fasea).

Fase horien guztien ostean, kontsultaren kodea sortzea besterik ez da geratzen, egikaritzean eskatutako datuak lortzeko.

Lau fase horiek hobeto ulertzeko, adibide bat garatuko dugu. Jo dezagun aurreko BEZEROA-ALOKAIRUA-FILMA datu-basea daukagula eta kontsulta berbera egin nahi dugula: NIFTzat 1111111L duen bezeroak alokatutako filmen izenburuak:

```
SELECT Izenburua
FROM FILMA, ALOKAIRUA
WHERE FILMA.FilmKodea=ALOKAIRUA.FilmKodea
AND NifBez='1111111L'
```

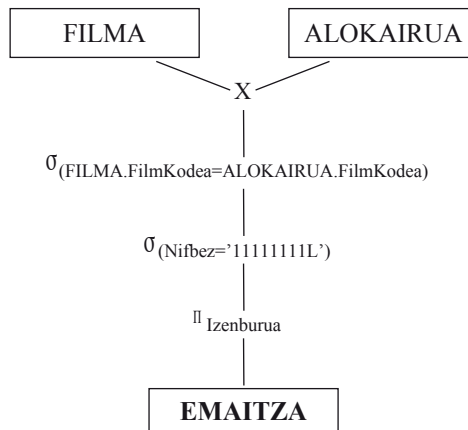
1. Kontsulta hori aljebra erlazionalera edo kontsulta-zuhaitzera pasatuko da. Aljebra erlazionalean honela geratuko da.

$$\pi_{\text{Izenburua}} (\sigma_{(\text{NifBez}='1111111L') \wedge (\text{FILMA.FilmKodea}=\text{ALOKAIRUA.FilmKodea})} (\text{FILMA} \times \text{ALOKAIRUA}))$$

Edo baliokidea den:

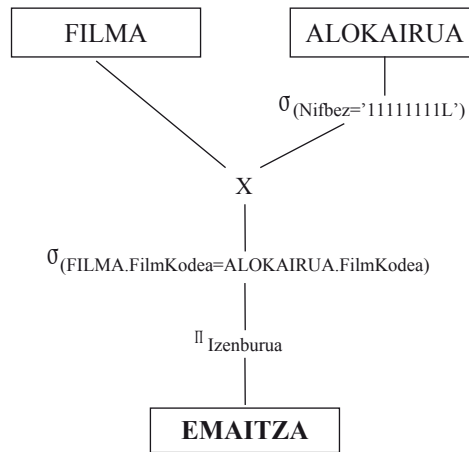
$$\pi_{\text{Izenburua}} (\sigma_{(\text{NifBez}='1111111L')} (\sigma_{(\text{FILMA.FilmKodea}=\text{ALOKAIRUA.FilmKodea})} (\text{FILMA} \times \text{ALOKAIRUA})))$$

Azken hori 7.4 irudian kontsulta-zuhaitz bitartez adierazi da.



7.4 irudia. Kontsulta-zuhaitz EZ optimoa.

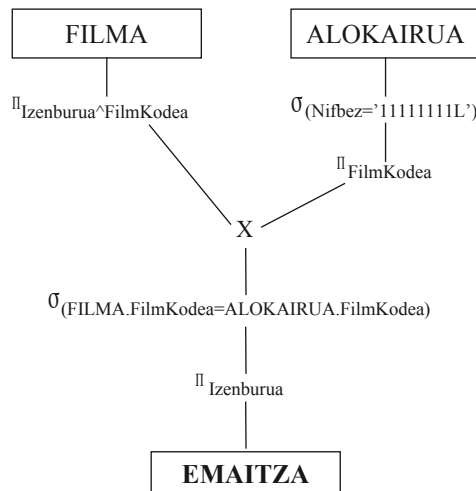
2. Aljebra erlazionaleko erregelak eta transformazio logikoak erabilia, optimizazio-prozesua 7.5 irudian eta bukaera 7.6 irudian agertzen den bezala emango da.



7.5 irudia. Kontsulta-zuhaitza optimizatzen. Hautapena ahal bezain lasterren egin da.

7.5 irudiko kontsulta-zuhaitzaren aljebra erlazionaleko adierazpena:

$$\pi_{\text{Izenburua}}(\sigma_{\text{FILMA.FilmKodea=ALOKAIRUA.FilmKodea}}(\text{FILMA} \times (\sigma_{\text{Nifbez='11111111L'}} \text{ALOKAIRUA})))$$



7.6 irudia. Kontsulta-zuhaitza optimizatuta. Proiekzioak aurreratu dira.

Eta 7.6 irudiko kontsulta-zuhaitzaren aljebra erlazionaleko adierazpena:

$$\pi_{\text{Izenburua}}(\sigma_{\text{FILMA.FilmKodea=ALOKAIRUA.FilmKodea}}((\pi_{\text{Izenburua}^{\text{FilmKodea}}}(\text{FILMA})) \times (\pi_{\text{FilmKodea}}(\sigma_{\text{NifBez='11111111L'}}(\text{ALOKAIRUA}))))))$$

Optimizatutako kontsulta horrek lehenago eskuz egindako adibideak baino emaitza hobeak izango ditu.

3. Adierazpen hori beteko duten behe-mailako prozedurak aukeratuko dira.

4. Azkena, sistemako aldagai batzuk kontuan hartuz, adibidez, PUZaren erabilera, sarrera/irteeren eragiketa-kopurua,... kontsulta egiteko planik onena hautatuko da.

Sistemak egin dezakeen beste optimizazio mota bat **optimizazio semantiko** modura ezagutzen da. Datu-basearen diseinua ongi eginda badago, edo lehenago administratzaileak edota disenaitzaileak

beren lana egin badute, erlazioetan CHECK motako murrizketak egongo dira. Kasu horietan, sistematik murrizketak begiratu beharko ditu kontsulta bera optimizatzen hasi aurretik, gerta baitaiteke kontsultan eskatutako datuetan murrizketak ezeztatuta izatea eta, beraz, ezer gehiago aztertu behar ez izatea, emaitzak ez baitu tuplorik izango. Adibidez:

Demagun soldata negatiboa duten langileen zerrenda eskatzen dela.

```
SELECT * FROM langilea WHERE Soldata < 0
```

Kontsulta ilogikoa iruditu arren, ongi dago eginda eta DBKSak erantzun egin beharko du. Datubasea ongi diseinatuta badago, LANGILEA erlazioko 'Soldata' eremuak CHECK murrizketa izango du: CHECK Soldata >= 0. Ondorioz, sistemak optimizazio semantikoari esker jada badaki aurreko kontsultaren emaitza hutsa dena eta ez ditu ez kontsultak ez prozesuak planifikatu eta egikaritu behar, horrek guztiorrek denbora aldetik dakarren hobekuntzarekin.

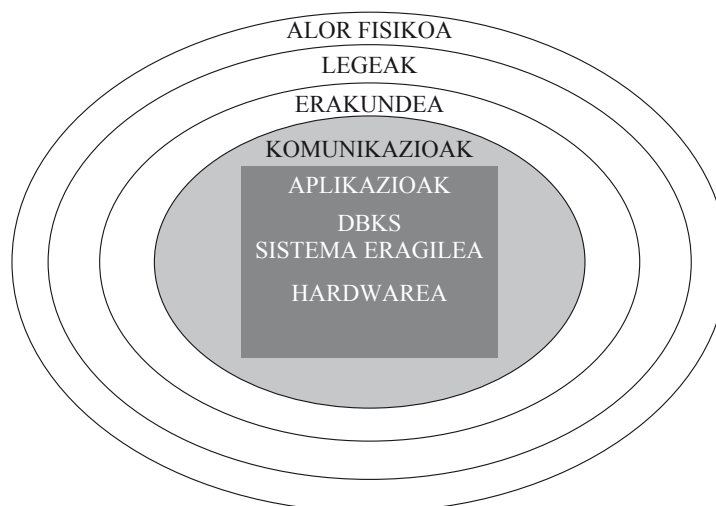
Hori dena dirudiena baino zailagoa da, sistemak bai kontsultak, bai murrizketak, ulertu egingo dituela jotzen baita. Gaur egun inteligentzia artifizialeko tekniken bilakaeraren ondorioz, gero eta gehiago erabiltzen da optimizazio semantikoa.

*Datuen segurtasunerako
teknikak eta prozedurak*

8

8.1. SEGURTASUNA: ZERTAZ ARI GARA?

Egun, euskarri informatikoetan dauden datuak gizartearen oinarri bihurtu dira. Are gehiago, hori dela eta, informazioaren gizartean bizi garela esaten da. Baina euskarri informatikoetan dauden datuek akats-mota asko izan ditzakete: datuen babesa egiterakoan, akats fisikoak, logikoak, gizakiak sortutakoak (nahita edo nahi gabe)... izan behar dira kontuan. Akats guzti horiek datuak eraldatzen dituztenez, informazio horrek zuen balioa galdu egiten da informazioa gordetzerakoan. Segurtasuna maila askotan zaindu behar da eta maila bakoitzak bere babes-neurriak ditu (ikusi 8.1 irudia).



8.1 irudia. Babes-neurriek kontuan hartzeko mailak.

8.1 irudian ikusten diren mailak ordenagailu arrunt baten mailen berdinak dira. Maila gehiago ere defini daitezke: komunikazioen gainetik beste aplikazio-maila bat, edo sistema eragilea bera ere zaitu... Edozein zatiketa eginda ere, zati bakoitzak bere segurtasun-arazoak izango ditu, eta batek huts egiten badu, sistema osoko segurtasuna kolokan geratuko da. Are gehiago, mailarik ahulenak emango du sistema osoari dagokion segurtasunaren maila.

Lehen maila **hardwarea** bera da. Sistema babestea beharrezkoa da. Horretarako daude RAID³ diska-sistemak, ordenagailu bat baino gehiago zerbitzu bera ematen, etenik gabeko elikadura (UPS⁴)... Hau da, orokorrean sistemako gailu fisiko guztiak (hardwarea) bikoiztu edo hirukoiztuz, ataletakoren bat beti martxan ibiliko dela ziurtatzea da helburua.

Horri lotuta beste gai bat agertzen da: zer den hobe, zenbait txipek (mikroprozesagailuek adibidez) ezkutuko identifikazio-zenbakia izatea ala ez izatea. Zenbaki horren bitartez “gaiztakeria” egiten duena identifikatuko litzateke. Beharrezkoa da ala alferreko giza-kontrol neurria? Puntu horretan argi ikusten dira segurtasunaren bi aldeak: sistemaren segurtasuna eta erabiltzailearen segurtasuna/anonimotasuna.

Bigarren mailan **sistema eragilea** legoke, segurtasuna ongi kudeatzeko oinarritzko softwarea baita. Ez dago inora joaterik alegiazko memoria kudeatzeko sistema gaizki badabil edo *driverak* edo kontrolatzaileak era egokian instalatuta ez badaude, esaterako.

³ RAID: Bere hasierako ingeleseko esanahia *Redundant Array of Inexpensive Disk*’ geroago *Redundant Array of Independent Disk*’ bihurtu da. Euskaraz ‘Diska Merkeen Multzo Erredundatea’ zena orain ‘Diska Independenteen Multzo Erredundatea’ da. Gazteleraz lehenik *Conjunto redundante de discos baratos*’ zen eta orain, *Conjunto redundante de discos independiente*’.

⁴ UPS: *Uninterruptible Power Supply*. Euskaraz EGE (Etenik Gabeko Elikadura). Gazteleraz SAI (*Sistemas de Alimentación Ininterrumpida*).

Kontuan hartu behar da sistema eragilea oso ongi instalatuta egon eta ongi ibili arren, egunak joan ahala, beste edozein softwaretan bezala, akatsak aurkituko zaizkiola. Horrek ez du esan nahi sistematik funtzionatzeari utziko dionik, denborak aurrera egin ahala segurtasun-zuloak agertuko direla baino. Sistema eragile guztiek dauzkate segurtasun-akatsak eta akatsok ongi kudeatzea da segurtasun-arduradun on edo txar baten arteko aldea. Segurtasun-akatsen berri izaten denean, lehenbailehen konpontzeko (edo beraien kaltea txikitzeko) neurriak hartu behar dira. Orokorrean, partxeak edo eguneraketak instalatzea nahikoa izaten da, baina sarritan zerbitzu batzuk desgaitzeko aholkua ere ematen da, segurtasun-zuloak aurkitu eta konponketarako partxeak plazaratzen diren bitartean arazorik ez edukitzeko.

3. mailan, hau da, **DBKSreanean**, har daitekeen erabakietariko bat datuak gorde aurretik zifratzean datza. Ez da ahaztu behar zifratze-prozesuak denborazko kostua duela eta, orokorrean, datua irakurtzen ematen den denbora, askotan, gehiegizkoa dela. Kasu horietan ongi aztertu beharko da (eta dagokionarekin adostu) irakurketatze/idazketa denboraren eta zifratzeko erabiliko den algoritmoaren arteko erlazioa. Algoritmoa ahula bada, ez du gogor zifratuko, baina azkarra izan daiteke, eta agian zifratze-maila horrekin nahikoa da. Demagun, karaktere bakoitzari dagokion ASCII kodeari unitate bat gehitu edo kentzen dion algoritmoa. Sinplea eta azkarra denez, berehala bihurtuko du testua ulergaitz. Hala ere, erraz antzeman daiteke zein izan den enkriptatzeko erabili den teknika. Amaitzeko, bai DBKS mailan, bai sistema eragilearen mailan, erabiltzaileek izan ditzaketan baimenak kudeatzea ere beste segurtasun-tresna bat da.

Aplikazioetan, segurtasuna arazo larria da. Orokorrean programatzaileek ez dituzte segurtasun-arazoak kontuan izaten, batez ere lana egiteko asti askorik ez dagoenean. Egun, hasi da kontzeptu hori pixkanaka aldatzen, baina oraindik bide luzea dago egiteko. Arazo horren ondorioz, SQL INJECTION⁵ edo BUFFER OVERFLOW⁶ eta antzeko erasoak izaten dira.

Komunikazioetan segurtasuna oso garrantzitsua da, gero eta garrantzitsuagoa. Gaur egun kokapen informatiko gehienak sare bitartez konektatzen dira. Horrek sekulako arazoa sortzen du. Kablea entzunez komunikazioen espioitza egin (*sniffing*), sarean dagoen ordenagailu bat atzitu edo ezgaitu (*Denial of Service* edo *DoS*)... Kalte handia egin daiteke sarearen bitartez. Ondorioz, babes-neurriek ere parekoak izan beharko dute. SSH (*Secure SHell*), SSL (*Secure Sockets Layer*), sarea trafikoz betetzea informazioaren transferentzia noiz egiten den ez jakiteko, zerbitzuen konfigurazio egokia, erabili behar ez diren zerbitzuak ezgaitu segurtasun-zuloak ekiditeko, sistema eragilea eguneratu... horiek dira sarean segurtasunaren ikuspuntutik aholku komenigarriak. Interneten hedakuntzarekin, sarean dema guztia ordenagailu gehienetan sistema eragilearekin bat eginda dator. Ia ezinezkoa da banaketa teorikoa errealitate bihurtzea, sistema eragile gehienek sareak kudeatzeko oinarrizko tresnak izaten baitituzte; beraz, sistema eragileen mailako segurtasunaz esandakoa ere berretsi egin behar da puntu honetan. Maila horretan izaten dira arriskurik handienak: baimenik ez duen erabiltzaileak sistemako datuak atzitu ahal ditu, bere lorratza ezkutatzeko gai izan daiteke, erakundetik kanpokoa izan daiteke, eta abar.

Hurrengoa **erakunde maila** da. Hau da, enpresa barruko legea. Enpresa barruan ere lege batzuk bete behar dira. Arau batzuk idatzita daude eta beste batzuk organigrama jarraituz lortzen dira: nor diren buruzagiak, zer baimen daukan bakoitzak datuak ikusteko, atzitzeko, programazioan/analisian

⁵ SQL INJECTION: atzealdean dagoen SQL sententzia susmatzean datza. Ondorioz, datu batzuk (sarrerako datu batzuk) aldatzen dira eta DBKSak espero ez den erantzuna ematen du, ezkutuko informazioa lortuz.

⁶ BUFFER OVERFLOW: adibidez, aplikazio batek gehienez 20 hizki izango dituen izen bat eskatzen badu, eta 200 hizki sartzen bazaizkio, bi aukera daude. Aplikazioa ongi egin badago, lehen 20 hizkiak hartuko dira, besteak baztertu eta normal-normal aurrerantz jarraituko du. Gaizki egin badago, aplikazioa eten egingo da eta, kasu batzuetan, erasotzaileak makinaren kontrola har dezake.

Bi kasu horiek arlo kontzeptualean konpontzeko errazak dira: erabiltzaileak sartzen dituen datuak tratatu aurretik, datuak prozesatu egin behar dira, aplikazioak dituen baldintzak betetzen dituen ala ez jakiteko. Adibidez: adinak zenbaki positiboa izan behar du, izenaren barruan hizki arrarorik (' , " , l , .) ez du egon behar, tamainak neurtu...

erabiltzen diren politikak... Demagun enpresa batean datu-base bakarria dagoela. Zentzuzkoa ematen du, adibidez, soldata eta bestelako datu pertsonalei buruzko informazioa erabiltzaile guztien esku ez egoteak.

Azkenaurreko segurtasun-maila **tokian tokiko legeria** da. Herrialde guztiek dituzte bete beharreko legeak, baita segurtasun informatikoari buruzkoak ere. Datuek babes-neurriak bete behar dituzte (LOPD), Interneten gune bat kokatzeko arauak (LSSI eta ondorengoak)

Azken maila **alor fisikoak** emango du. Lehenago ere aipatu dugu baliabideak bikoiztu eta hirukoiztu egiten direla sistemak fidagarria izan dadin. Horrela, sistema bakarrek izan ditzaketan arazoak ekiditen dira. Beste konponbide erraz bat segurtasun-kopiak egitea da. Derrigorrezkoak dira, azkenean beti gertatzen baita ustekaberen bat: tenperatura gehiegi igota sistema laguntzailearen matxura, suteak, lapurretak, lurrikarak... Sistema bertan behera geratuko litzateke. Hori kontuan izanda, neurriak askoz lehenago hartu behar dira: sistema lehenbailehen berriz martxan jartzeko planak, simulazioak... egin behar dira. Segurtasun-kopiak maiz egitea agintzen duen politika izatea ere funtsezkoa da sistema ongi babestuta izateko. Gainera, kopia horien kudeaketa aproposa egitea ere komeni da: eraikuntza berean ez gorde (lurrikarak), suaren aurkako kaxatan gorde (suteak),...

Maila horiek guztiak izan arren, kasu guztietan ez du zentzurik segurtasunerako neurri berberak hartzeak. Aukeraren kostua ere kontuan izan behar da. Hiru langileko enpresa txiki baten eta multinazional baten segurtasun-arazoak ez dira berdinak. Bakoitzak bere datuen garrantzia neurtu eta dituen ahalmenen eta gaitasunen arabera jokatu beharko du.

Ikusi den bezala, segurtasunaz aritzean, ezkutuan dauden beste arazo pare bat agertzen da. “Segurtasun” hitza nori/zeri aplikatzen zaion arabera, batzuetan datuen segurtasunaz ari delako eta beste batzuetan, berriz, erabiltzailearen segurtasunaz.

Lehen arazoa informazioa bera da. Egun, euskarri eta tresna informatikoak direla medio, pertsona bakoitzeko datu asko pilatzen da hainbat tokitan. Kutxazainetan, dendetan, interneten nabigatzearen gelditzen diren *cookie*etan, posta elektronikoa eta abarretan lortatzeko asko uzten dira. Horrek, berez, ez dauka garrantzi handirik, baina orain merkatu berriak hasi dira zabaltzen. Informazio hori guztia erosi eta saldu egiten da eta, ondorioz, pertsona bati buruzko datu asko pila daiteke. Kasu askotan, datuak ongi erabiliko dira, baina edonork edozelan erabil ditzake. Puntu horretan berriro egiten da topointimitatearen arazoarekin eta anonimotasun ezarekin.

Bigarrena, intimitatearen arazoa da. Non dago segurtasunaren muga? Non geratzen dira gizabakoaren eskubideak sare bateko trafikoa aztertzen ari garenean? Langileek lan-ordutegian idatzitako posta elektronikoa irakur daiteke? Egitez, egin daiteke. Zailagoa da etikaren edo legeriaren arabera horretarako eskubiderik dagoen ala ez zehaztea. Kasu batzuetan legeria horraino heldu da, baina ez guztietan, beti egongo dira zirrikituak eta. “*Big Brother*” mundurantz goaz? Alor horretan galdera etiko-filosofiko ugari zabaltzen da.

Gai honetan ez gara horretaz guztiaz arduratuko; izan ere, sistema eragilearen segurtasunak liburu asko idazteko ematen du. Hemen datuen segurtasunaz soilik hitz egingo da. Zehatzagoak izateko, datuez ere ahaztu egingo gara eta DBKSen segurtasunaz eta beraiek gordetzen dituzten datuez arituko gara.

8.2. DBKS-AK ETA SEGURTASUNA

DBKSkon administratzailea (DBA) izango da datuak kudeatzeko garaian ardura guztiak dituen erabiltzaile bakarria. DBAk baimen guztiak dauzka eta beraren esku dago segurtasunerako politika guztia. Administratzaileak erabiltzaile arruntek izaten ez dituzten goi-mailako aginduak erabil ditzake

eta, beraien bitartez, erabiltzaile kontuak sortzeko, kontuoi pribilegio batzuk emateko edo kentzeko, eta segurtasun-mailak kudeatzeko gaitasuna izango du.

Datu-baseko administratzaileak lehenik erabiltzaileak sortuko ditu eta, gero, erabiltzaile bakoitzak objektu bakoitzeko zein pribilegio duen zehaztu beharko du (datu-basea erabili, datu batzuk ikusi, beste batzuk eraldatu...).

DBKS baten datuen atzipena kontrolatzeko hainbat elementu daude. Sistemak duen lehen eginkizuna erabiltzaileak identifikatu eta kautotzea da. Horretarako, hainbat modu dago:

- Izena eta pasahitza
- Hardware bitartez identifikatu: giltzak, txartelak...
- Ezaugarri biometrikoak erabiliz: ahotsa, atzamarrak, begiak...
- Erabiltzailearen gaitasunak, jakintzak, ohiturak...
- Aldez aurretik lehenetsitako informazioa

Segurtasunaren kontzeptuaren barruan hiru atal garrantzitsu daude: konfidentzialtasuna (baimenik ez duten erabiltzaileei daturik ez erakustea), erabilgarritasuna (datuak edozein mementotan eskuragarri izatea) eta osotasuna (datuak ez direla faltsutu ziurtatzea).

8.2.1. Konfidentzialtasuna

Baimenik ez duten erabiltzaileei daturik ez erakustea da helburua. Horretarako, DCL sententziak erabili beharko ditu DBAk. Zeregin horixe izango da, segurtasunaren ikuspuntutik, DBAk duen lanik garrantzitsuenetako bat. Horretarako, GRANT eta antzeko sententziak, rolak... erabiliko ditu DBAk. Hori guztiari administrazioari buruzko gaien (6. gaien) landu da. Hemen testuaren barne-koherentzia mantentzeko aipatuko dugu, baina sakonago jorratu barik.

8.2.2. Erabilgarritasuna

Erabilgarritasunaren kontzeptuarekin, lehenago esan bezala, datuek edozein mementotan eskuragarri egon behar dutela adierazi nahi da.

Edozein akats gertatzen denerako, oso garrantzitsua da datu-basea egoera sendoan geratuko dela bermatzea. Horretarako, "transakzio" izeneko instrukzio-multzoa definitzen da. Transakzioko instrukzio guztiak (transakzioa bera) batera doaz,;hau da, edo guztiak egikarituko dira edo bat ere ez; horri **atomizitate** propietatea deitzen zaio.

Hasieran, transakzioa egikaritu aurretik, datu-basea egoera sendoan dagoela pentsa daiteke; transakzioa bukatzen denean, datu-baseak egoera sendoan jarraitu beharko du. Horri **sendotasun** propietatea deritza. Bien bitartean (transakzio erdian) datu-basea memento batzuetan egoera ez-sendoan egon daiteke. Transakzioek eduki beharko dituzten ezaugarri nagusiak ACID izenez ezagutzen dira eta honako hauek dira:

- Atomizitatea (*Atomicity*). Transakzio osoa egingo da edo ez da ezer egingo. Ezin da zati bat bakarrik egikaritu.
- Sendotasuna (*Consistency*). Transakzioek datu-basea egoera sendoan hartu eta beste egoera sendo batean utzi behar dute.

- Isolamendua (*Isolation*). Transakzio batek egindako eraldaketa guztiak ez dira aurkeztuko transakzioa bukatu aurretik. Bukaeran, aldaketa guztiak batera plazaratuko dira.
- Iraunkortasuna (*Durability*). Transakzioa ongi bukatzen denean, emaitzak datu-basean gordeko dira.

Hau da, transakzioa osatzen duten instrukzio guztiak batera doaz beti, atomizitate deritzon ezaugarria betez. Horrela, edo guztiak egikarrituko dira edo bat ere ez. Ezaugarri horrekin batera isolamenduarena ere badator, transakzioaren emaitzak ez baitira bukatu arte agertuko, emaitza guztiak plazaratuz.

Bestalde, transakzioa egikaritzeak ezin du datu-basearen sendotasuna apurtu. Datu-basea egoera sendoan hartu eta beste egoera sendo batean utzi behar du. Eta, azkenik, transakzioa ongi bukatzean, emaitzak datu-basean gordeko dira, iraunkortasunaren ezaugarria bermatuz. Hona hemen adibide bat.

```
BEGIN TRAN
```

```
--transakzioa hasiko da
```

```
    UPDATE produktuak SET Prezioa=28
```

```
WHERE ProduktuIzena='piperrak';
```

```
-- 'piperrak' izeneko produktuaren prezioa eguneratuko da
```

```
    UPDATE produktuak SET Prezioa=43
```

```
WHERE ProduktuIzena='porruak';
```

```
-- 'porruak' izeneko produktuaren prezioa eguneratuko da
```

```
COMMIT TRAN;
```

```
--egindako eguneraketak datu-basera pasatuko dira.
```

UPDATE sententzia biak bakarra balira bezala egikarrituko dira. Horrela, transakzio erdian, lehenengo UPDATEa egikaritu eta gero sistema geldituko duen arazorik balego, sistema berriro martxan hasten denean, transakzio baten erdian egongo da. Orduan, edo bigarren UPDATEa egikarrituko da, edo lehenengo UPDATEa desegingo da, baina ez da erdibidean geratuko.

Beste kasu bat litzateke logika aldetik sententziaren batek akatsen bat ematearena. Kasu horretan, transakzioa ez da geldituko eta, nahiz eta lehenengo sententziak akatsa eman, bigarren sententzia egikaritu egingo da. Mota horretako akatsak detektatzeko, hona hemen adibide aproposa:

```
DECLARE @Akatsa int
```

```
--akatsen bat gertatuz gero, akats horren kodea gordeko duen aldagaia
```

```
--deklaratu da.
```

```
BEGIN TRAN
```

```
--transakzioa hasi da
```



```

UPDATE produktuak SET Prezioa=28
WHERE ProduktuIzena='piperrak';
    -- 'piperrak' izeneko produktuaren prezioa eguneratu da
    SET @Akatsa=@@ERROR
    --akatsen bat gertatuko balitz, kodea @Akatsa aldagaian gordeko da eta
    --akatsa tratatzeko instrukzio-multzora egingo da salto
IF (@Akatsa <>0) GOTO AkatsaTratatu
--lehenengo UPDATE sententziak akatsik ematen ez badu orduan
--bigarren UPDATEA egikaritzera pasatuko da
UPDATE produktuak SET Prezioa=43
WHERE ProduktuIzena='porruak';
    --'porruak' izeneko produktuaren prezioa eguneratu da
    SET @Akatsa=@@ERROR
    --akatsen bat gertatuko balitz, kodea @Akatsa aldagaian gordeko da eta
    --akatsa tratatzeko instrukzio-multzora joango da
IF (@Akatsa <>0) GOTO AkatsaTratatu
--UPDATE bietatik batek ere ez badu akatsik ematen, egindako
--eguneraketak datu-basera pasatuko dira.
COMMIT TRAN;
Akatsa Tratatu:
BEGIN
    PRINT 'Akatsen bat gertatu da eta transakzioa abortatuko da'
    --erabiltzaileari jakinarazi zaio--
    ROLLBACK TRAN
    --transakzioan egindako eguneraketak desegin egingo dira. Datu-basea
    --ezer egin izan ez bagenu bezala geratuko da
END

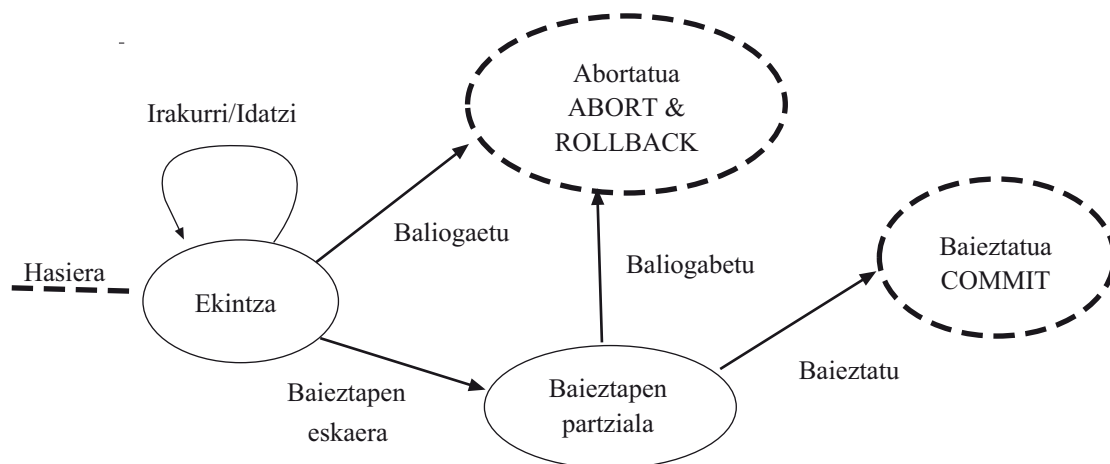
```

Ezaugarri horiek kontuan hartuta, transakzio ugari badaude, lortzen diren emaitzak transakzio horiek era sekuentzialean egikaritzeko balira bezalakoak izango dira. Hau da, bukatu ez den transakzio batek ezin dizkio emaitzak erakutsi era konkurrentean egikaritzen ari den beste transakzio bati.

• Transakzioen faseak

Transakzio batean hainbat fase bereizten dira (ikusi 8.2 irudia):

- Hasiera. Transakzioa hasiko da.
- Ekintza. Transakzioak datu-basean gauza ditzakeen ekintzak: irakurketa edo idazketa. Fase horretan abortatzeko agindu zuzena jaso daiteke.
- Baieztapen partziala. Eraldaketa guztiak (irakurketak eta idazketak) bukatu dira; baina oraindik datuak behin-behinekoak dira, konkurrentzia dela-eta arazoak izan baitaitezke.
- Baieztapena. Transakzioa ondo bukatu da. Aldaketak behin betiko bihurtuko dira. COMMIT.
- Baliogabetzea edo abortatzea. Azkenean ez da baieztapenik jaso; beraz, egikaritutako eraketak desegin eta datu-basea hasierako egoerara itzuliko da. ROLLBACK. Baliogabetutako transakzioa ere bukatutako transakzioa da.



8.2 irudia. Transakzioen faseak.

Ikusten denez, transakzioa bi modutan buka daiteke: dena ongi eta, ondorioz, egindako eguneraketak datu-basean gordeko dira (COMMIT prozesua); edo transakzioan zehar akatsen bat gertatzen bada, datu-basea hasieran zegoen egoerara (transakzioa egikaritu aurretik) bueltatu beharko da; ondorioz, transakzioa abortatzen da (ABORT prozesua) eta bitartean egindako eguneraketa guztiak desegin egin beharko dira (ROLLBACK prozesua).

Horrek guztiorrek zera dakar, transakzioa aldatzen ari den datuen zenbait balio gorde egin beharko direla: transakzioa hasi aurretik zegoen balioa batetik, eta transakzioan zehar datu horrek hartu dituen balioak bestetik (azken balioa baino ez da gordeko). Eguneraketarako ariketa ongi bukatzeak ez du ezer esan nahi, transakzioa bukatu arte ez baitago jakiterik ongi joan den ala ez. Orduan gordeko dira datu-basean transakzioak zituen eraketetako datuak (eguneraketak, sortu diren datu berriak, ezabatutako datuak...) atomizitate propietateak zehazten duen moduan.

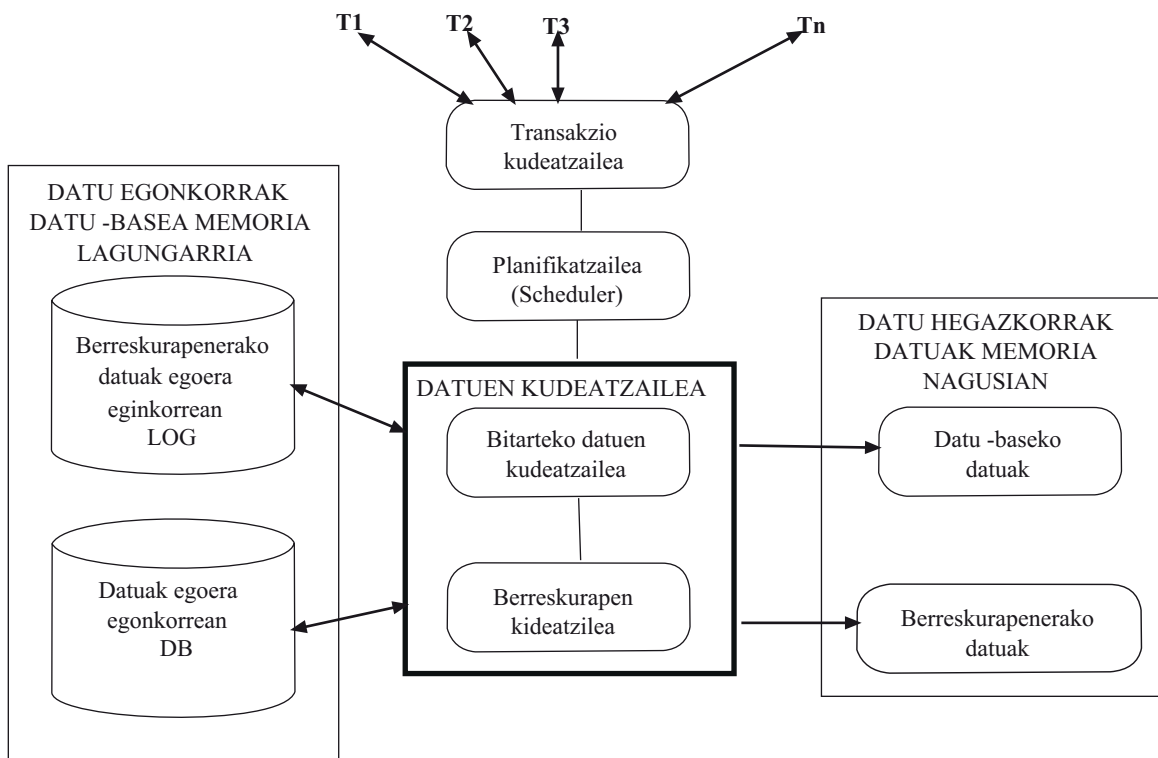
• Transakzioen kudeaketa

Garbi dago hori dena kudeatzeko DBKSak lan handia egin beharko duena. Horretarako, oso lagungarri izango du bitakora-lanak egingo dituen LOG fitxategia. Datu-basean grabatu aurretik, datu guztiak LOG fitxategian gordeko dira eta fitxategi hori disko gogorrean (memoria lagungarrian). Transakzioarekin aurrera jo aurretik, LOG fitxategiak memoria lagungarrira pasatu beharko dira eta,

ondoren, datu-baseko erregistroak eguneratuko dira. Ez du balioko eragiketa memoria nagusian egin bada. Azken pausoa, datu-basearen eguneraketa, bai egin daiteke memoria nagusian bakarrik (aurrerago memoria lagungarrian gordeko da).

LOG fitxategian transakzioak desagiteko (hutsen bat gertatzen bada) edo berregiteko, honelako informazioa gordetzen da:

- Transakzioa identifikatzen duen zenbakia
- Eraldaketa egin den ordua
- Eraldatu den erregistroa identifikatuko duen zenbakia, gakoa alegia
- Egindako ekintza-mota
- Lehen zegoen balioa
- Balio berria
- Informazio gehigarria



8.3 irudia. DBKS baten atalak.

Arazorik ez egoteko, memoria nagusian dagoen erregistro baten balioa aldatzen bada, datu-basean idatzi aurretik LOG fitxategian idaztera behartzen da (horrela, transakzio osoa bukatu bitartean arazoren bat gertatzen bada, balio guztiak berriro aurreko balioetara bueltatzeko gai izango gara). Horri *LOG write-ahead* protokoloa esaten zaio.

Hala ere, sistema eragilea, segmentazioa, orrikatzea, alegiazko memoria, sarrera-irteera *bufferak*... eta sistema arinago joateko pentsatuta dauden beste hainbat prozedura dela eta, askotan ez da diskoan (memoria lagungarrian) nahi denean idazten eta, zerbait gertatuz gero, datuak eta aldaketak ez dira memoria egonkorrean egongo; beraz, galdu egingo dira. Datuen kudeatzaileari dagokio hori ez gertatzearen ardura, zehatzago esanda, berreskurapenen kudeatzaileari (ikusi 8.3 irudia).

Datu-base hegazkorra deitzen zaio memoria hegazkorrean dagoen datu-base zatiari. Datuen kudeatzaileak hartuko ditu datuak. Datu horiek beti memoria hegazkorrean egongo dira; beraz, beranduagoko prozesu batek arduratu beharko du aldaketak memoria egonkorrean gordetzeaz.

8.3 irudiko osagaiek honako funtzio hauek betetzen dituzte:

- Transakzioen kudeatzailea: erabiltzailearen transakzioek behar dituzten aurreprozesatze-eragiketak egiteaz (START, READ, WRITE, COMMIT eta ROLLBACK) arduratzen da.
- Planifikatzailea edo *scheduler*: transakzioen kudeatzaileak eskatzen dituen eragiketen ordena kontrolatzen du. Transakzioen fluxuak sortzen dituen ekintzen serializazioaz arduratzen da. Hiru instrukzio nagusi ditu: EXECUTE, eragiketa datuen kudeatzaileari emateko; REJECT, eragiketa albo batera uzteko eta, ondorioz, transakzioa atzera botatzeko; eta DELAY, eragiketa itxarotilara batean lagatzeko. Geroago, ilara horretatik hartu eta EXECUTE edo REJECT eragiketak egingo ditu.
- Datuen kudeatzailea: transakzioek izaten dituzten eragiketak egikaritzea du helburu. Horretarako, datuak memoria nagusian edo memoria lagungarrian dauden jakin eta bien arteko harremana kudeatu beharko du. Bi osagai ditu:
 - Bitarteko datuen kudeatzailea: datuak memoria lagungarritik nagusira eta kontrako noranzkoan mugitzeaz arduratzen da. Bitarteko erregistroen kudeaketa ere bere lana da.
 - Berreskurapen-kudeatzailea: datu-base egonkorrean baieztatu diren transakzio guztien eraldaketak eta ezeztatu diren transakzioen eraldaketa bat bera ere ez egoteaz arduratuko da.

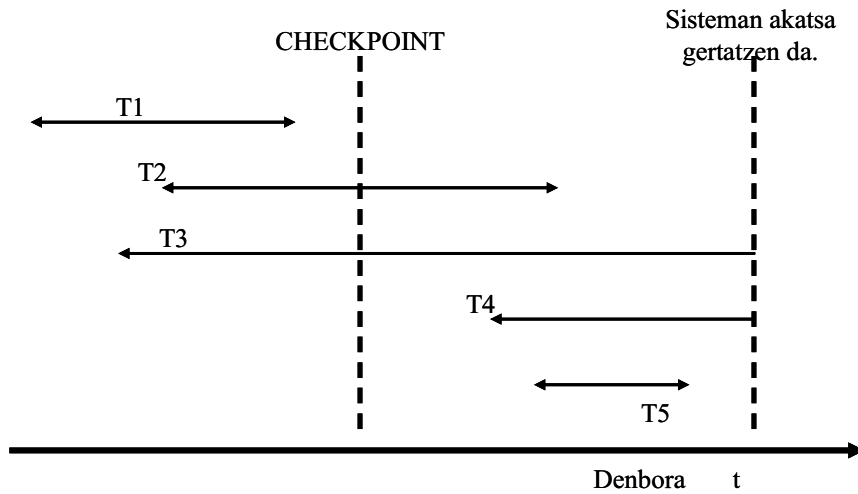
Akats guztiak ekiditea ezinezkoa denez, konpontzeko edo, behintzat, azken unean genuen egoerara heltzeko, LOG fitxategia behar da. LOG fitxategia goitik behera tratatzeko denbora luzea behar denez, egiaztapen-puntuak edo berreskurapen-puntuak (CHECKPOINT) erabiltzen dira. puntu horiek aldiro egikarrituko dira. Hona hemen beraien funtzioak:

- Bitarteko gunean dauden LOG fitxategiko datuak memoria egonkorrera eramatea (datu-baseko datuekin bezala, LOG fitxategiko bitarteko guneak egongo dira).
- Aurreko eragiketa ongi egin dela adierazteko, LOG fitxategia erregistro batekin markatzea.
- Datu-baseko bitarteko gunean dauden datuak memoria egonkorrera eramatea.
- Abiarazteko dagoen fitxategiaren berreskurapen-erregistroaren helbidea gordetzea. Esaterako, datu-basea berriro abiaraztean, CHECKPOINT horretatik abiaraziko da eta erregistro horretan azalduko da noraino dagoen eguneratuta.

• Berreskurapena

Memoria hegazkorreko edukia galtzen denean, sistemak LOG fitxategiari begiratzen dio eta, ondorioz, zein transakzio desegin (osatu ez direlako) eta zein berregin jakingo du. Gogoratu, nahiz eta transakzioa eginda egon, emaitzak ez zirela datu-base egonkorrean gordeta egongo akatsa gertatu zen unean. Berreskurapen-mota horri UNDO/REDO deitzen zaio (ikusi 8.4 irudia). Sistema martxan jartzen denean, abiarazteko dagoen fitxategitik abiapuntua hartuko da eta LOG fitxategia soilik puntu horretatik aurrera egikarrituko da. Ikusten denez, T1 egiaztapen-puntuaren (CHECKPOINT) aurretik dago; beraz, jadanik eginda. Baina T2 eta T5 berregin egin beharko dira egiaztapen-puntuaren ostean amaitu direlako; eta T3k eta T4k datu-basean egin dituzten eragiketak ezabatu egin beharko dira, bukatu gabe daudelako.

Datu-base bakoitzak berreskurapena antolatzen duen moduaren arabera beste 3 berreskurapen-mota ere existitzen dira NO UNDO/REDO, UNDO/NO REDO eta NO UNDO/NO REDO. Horietariko batzuk ikusiko dira ondoren.



8.4 irudia. Sistema, akatsa gertatu denean, transakzioak egikaritzen ari da.

Badaude LOG fitxategia behar ez duten berreskurapen-technikak ere. Horietako bat *shadow paging* edo ezkutuko orrikatzea da. Sistema eragileak orrikatzeko erabiltzen duen estrategiaz baliatzen da. Hasieran, orrikatze-taula bat beharrean bi izango dira. Lehena jatorrikoa izango da, eta bigarrena (kopia) transakzioak bere aldaketa guztiak egiteko erabiliko du (edo alderantziz, lehenengoa kopia, eta bigarrena, jatorrikoa). Azkenean, transakzioa onesten bada (COMMIT), egin behar den gauza bakarra sistema eragileari orria alda dezala esatea da. Hortik aurrera, hasieran kopia zena jatorriko bihurtuko da eta horretarako sistema eragilearen mailan orrikatze-taularen erakuslea aldatu beharko da. Transakzioa ezeztatzen bada (ROLLBACK), ez da ezer egingo. Bi aukera horietan (COMMIT eta ROLLBACK), bukaeran behar ez den memoria berriro erabilgarri bihurtzeko zabor-biltzaile teknikak (*garbage collector*) erabiliko ditu. Berreskurapen-tekника horri NO UNDO/NO REDO izena ematen zaio.

DBKSaren atalen batek akats larri bat duenean, berreskurapena luzeagoa da. Horretarako, lehenik segurtasun-kopia (BACKUP) erabili beharko da eta, ondoren, LOG fitxategi osoa tratatu beharko da.

Kasu horiek aztertuz, berreskurapenak izan behar duen ezaugarrietako batez jabetuko gara: zer gertatzen da berreskurapena egiterakoan berriro akatsen bat gertatzen bada? Ezer ere ez, berriro kasuaren arabera dagokion pausotik hasten baita. Izan ere, REDO eragiketa behin eta berriro egikaritu arren, emaitza berberera heldu behar da. Gauza bera gertatuko da UNDO eragiketekin. Horri **idenpotentzia** propietate deritzo.

Beste irizpide baten arabera, bi berreskurapen-estrategia daudela esan daiteke:

- *In place*: transakzioko datuak, datu-basera bertara doaz eta jatorrizko balioak zapaltzen dituzte. Behar izanez gero, LOGean dauden balioak erabiliz bukaeran UNDO egingo da. UNDO/NO REDO teknika, adibidez, mota horretakoa litzateke. UNDO/REDO teknika ere mota horretakoa da; berreskurapena egiteko LOG fitxategiaren erabileraren arabera teknika bata edo bestea izango da aurrean dagoena.
- *Out of place*: jatorrizko balioak beti mantenduko dira; kopiak eraldatuko dira. NO UNDO/NO REDO teknika adibidez.

Kalte gehien egin dezakeen errore-mota ezinbesteko errore edo errore larria delakoa da, eta LOG fitxategiak galtzen direnean gertatzen da. Ez dago konponbiderik; datu-basea ezingo da zegoen egoerara bueltatu. Arazorik ez izateko, LOG edo bitakora-fitxategia (eta, nahi izanez gero, datu-basea bera ere) bikoiztu egiten da. Segurtasun-kopiak egiten zaizkio... Teknika horiei *mirroring* (ispilu), *duplexing* (bikoizketa)... deritze.

Laburbilduz, SQL sententziak erabiliz, erabilgarritasuna kudeatzeko bi instrukzio daude: COMMIT eta ROLLBACK.

8.2.3. Osotasuna

Osotasunaren helburu da datu-basea babestea sendotasunik izango ez duten datuek egingo dituzten eragiketetatik. Horregatik, osotasunaz aritzean, datuen zuzentasuna, baliagarritasuna eta zehaztasuna ere hartzen dira kontuan. Beraz, DBKSaren osotasunaz arduratzen den azpisistemaren lana eragiketa ez-egokiak detektatu eta zuzentzea izango da.

DBKSkos osotasun-azpisistemak funtzio hauek beteko ditu:

- Definitutako arauen koherentzia egiaztatu.
- Transakzioak kontrolatu eta osotasunari egiten dizkioten bortxaketak detektatu.
- Osotasunaren bortxaketa gertatzen denean, komeni diren eragiketak egikaritu.

Osotasuna bi eragiketa-motaren ondorioz gal daiteke: batetik, semantikoki sendotasunik ez duten eragiketen ondorioz, eta bestetik, datu-basean memento berean gertatzen diren eragiketen ondorioz (konkurrentzia).

Zenbait eragiketa egiterakoan, osotasun semantikoarekin arazoak izatea gauza normala da. Atributuen eta domeinuen balioekin gertatzen dena da adibide garbia. Atributu batek domeinu jakin bateko balioa hartu beharko du (0tik 100era bitarteko zenbakia, adibidez); transakzioa bukatzean, domeinu horretatik kanpo dagoen balioaren bat hartzen badu (-1 edo 102, adibidez) akatsa gertatzen denaren seinale da, eta transakzioa ezeztatu egin beharko da. Hori guztiori kontrolatzeko, DBKSak zenbait tresna ditu eta garrantzitsuenetako bat, datuen deskripzio gisa, datu-baseko hiztegian gordetzen diren arauak dira.

Osotasunari buruzko arauak datu-baseko hiztegian zentralizatuta edukitzeak zenbait abantaila eskaintzen ditu:

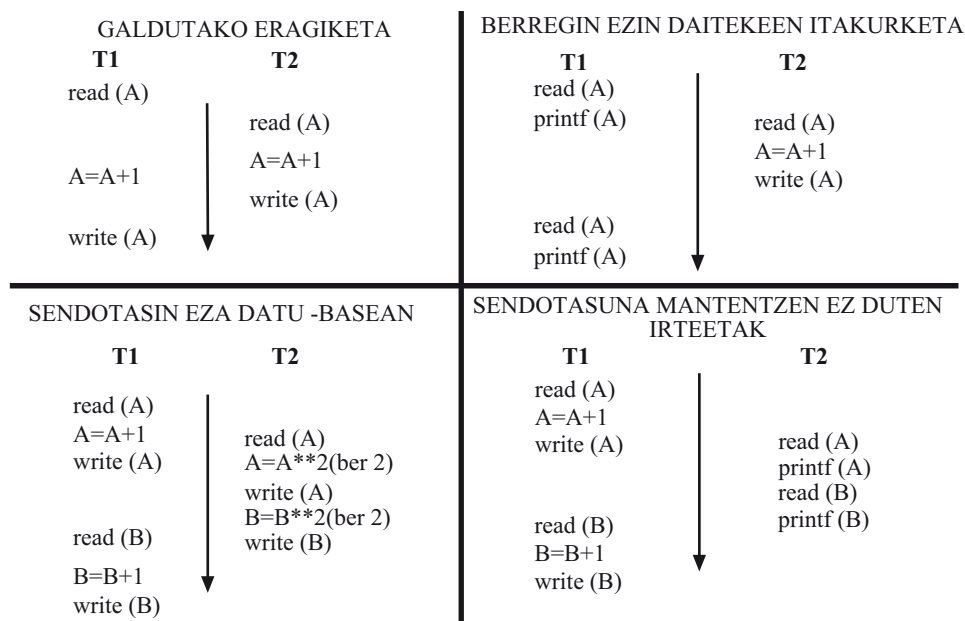
- Errazagoak dira ulertu eta aldatzeko; ondorioz, mantentze-lana erabat errazten da.
- Sendotasun ezak hobeto detektatzen dira.
- Osotasuna hobeto babesten da, inork ezingo duelako datu-basea funtsik gabeko egoera batera eramango duen programarik idatzi.

DBKSan konkurrentziaren **ondorioz** gertatzen diren osotasun-arazoak kudeatzeko teknikak askoz ere zailagoak dira. Jakina, arazo horiek sortzeko datu-baseak memento berean prozesu eta erabiltzaile bat baino gehiago kudeatzeko gai izan beharko du.

Konkurrentziaren ondorioz gerta daitezkeen akats-motak 8.5 irudian ikus daitezke. Ondorengo hauek dira:

- Galdutako eragiketak. Kasu horretan, bi transakziok datu bera aldi berean eraldatzen dute, baina eraldaketetako bat galdu egingo da.

- Berregin ezin daitekeen irakurketa. Transakzio batek bi aldiz irakurriko du datu bera eta balioa ez da berdina izango.
- Datu-basean sendotasunik eza. Transakzio batek bere ibilian zehar bi aldagairen balioa hartzen du. Balioetako bat ongi egongo da, baina besteak ez du lehenengoarekiko erlazioa mantenduko, bigarren transakzio batek aurretik eraldatu baitu. Adibidez: A, B, C... zenbaki osoak dira (2, 4, 6...). T1 transakzioaren helburua, segida 2tik hasi beharrean 3tik hastea da eta horregatik +1 egingo du, emaitza 3, 5, 7... izateko. Bitartean T2k ber bi egin nahi du, emaitza 4, 16, 36... izan dadin. Onargarria litzateke T2ren emaitza 9, 25, 49... balitz (T1 osoa egikaritua izan delako) edo T1en emaitza 10, 26, 50... balitz (lehenik T2 osoa eta gero T1 egikarituko balitz), baina T2 osorik egikaritzen bada justu T1ek $2+1=3$ egin ondoren irteten den emaitza, hau da, T1 = 3, 17... eta T2 = 9, 16... gaizki dago.
- Sendotasuna mantentzen ez duten irteerak. Bi transakzioek datu berberak erabiltzen dituzte, baina ez dira balio berdinak inprimatzen/gordetzen.



8.5 irudia. Konkurrentziaren ondorioz izan daitezkeen osotasun-arazoak.

Transakzioan mementoko sendotasunik eza gerta daiteke eta, horren ondorioz, transakzioen artean konkurrentzia badago, datu-basean sendotasunik eza ager daiteke. Konkurrentziarik ez badago, arazorik ez dela egongo jakinda, orduan non dago konponbidea? Konponbidea serializazioa da. Transakzioak bata bestearen atzetik serie batean egikaritzen badira ez da arazorik izango, baina orduan zertarako nahi da konkurrentzia? Serializazioarekin sistemari atera dakioken etekina asko jaitziko delako.

Hori bai, ez da beharrezkoa planifikazioa seriean izatea serializazioa bermatzeko. Serializazioa bermatzen duten planifikazio paraleloak egon daitezke. DBKSaren helburuetako bat, ahal den konkurrentzia-maila altuena lortzea da, betiere serializazioa kontuan izanda. Hori guztia kudeatzeko hainbat teknika daude, baina bereziki bitan banatzen dira:

- Baikorak: serializazioa betetzen den edo ez jakiteko transakzioaren bukaerara arte itxaroten dute. Orokorrean transakzioek irakurketarako fase bat izaten dute; ondoren, baieztapenarena eta, gehienetan, hirugarren fase batean, emaitzak idazten dira. Baieztapenaren fasean egiaztatzen da serializazioa betetzen den ala ez.

- Ezkorrak: transakzioen hasieratik behartzen dute serializazioa.

Teknika baikorrek erabiliz datu-basearen paralelismoari etekin hobea ateratzen zaio, bukaeran transakzio gehiago ezeztatu (ROLLBACK) behar izaten bada ere. Teknika ezkorrekin kontrakoa gertatzen da.

Konkurrentzia kontrolatzeko erabiltzen diren zenbait teknika:

- blokeorako teknikak
- denbora-markak
- bertsio ugariko denbora-markak
- transakzio habiaratuak
- transakzio luzeak
- koordinaziorako transakzioak

Blokeorako teknika: ezkorra da. Transakzioek erabili behar dituzten elementuak blokeatzen dituzte; ondorioz, beste erabiltzaile batzuek ezin izango dituzte elementu horiek erabili. Horrela, datu-basearen sendotasuna mantenduko da. Blokeo-mailari **bikorkadura** deitzen zaio. Blokeoak atributu mailan, tuplo mailan, bloke mailan, taula mailan eta datu-base mailan eman daitezke. Blokeo-maila zenbat eta txikiagoa izan (atributu < tuplo < bloke < ...) datu-basea kudeatzeko zailagoa izango da, baina konkurrentzia handiagoa onartuko du: elementu bat atzitzerakoan, datu-base guztia blokeatzen bada, ez da aldiberekotasunik izango, memento bakoitzean transakzio bat bakarrik egikarituko baita. Beraz, kudeaketa oso erraza da: baimen guztiak transakzio horri eman eta beste transakzioek, itxaron dezatela.

Bi blokeo mota daude: irakurtzeko blokeoa eta idazteko blokeoa. Elementu bati irakurtzeko blokeoa jartzen bazaio, transakzioak elementu horren balioa aldatuko ez duelako egiten da. Transakzioak elementua irakurri baino ez du egingo, inoiz ez idatzi. Irakurtzeko blokeoan beste transakzioek ere elementu hori erabili ahal izango dute eta elementu beraren gainean irakurketa gehiago ipini ere bai. Idazteko blokeoa, ordea, beste kontu bat da. Transakzio batek elementu bati idazteko blokeoa ezartzen badio, elementuaren balioa aldatzera doalako da (edo elementua ezabatzeraz). Beraz, beste transakzioek ezin izango dute elementu hori atzitu eta blokeoa noiz askatu zain geratuko dira.

Asma daitekeen bezala, horrek guztiak elkar-blokeoa (*deadlock*) ekar dezake. Elkar-blokeoa transakzio bi edo gehiago elkarri itxaroten daudenean gertatzen da. Bata besteak duen elementu baten zain daudenean, elkar blokearazten dute. Elkar-blokeoa garaiz detektatzen ez bada, azkenean datu-base osoa blokeatuko da. Elkar-blokeoa ez da soilik datu-baseekin gertatzen, sistema eragileek ere horrelako arazo asko izaten dituzte eta. Ikerketa asko egin dira gaiaren inguruan eta hainbat konponbide daude. Horietako batzuk elkar-blokeoa eragoztera edo saihestera bideratuta daude, behar den guztia hasieran hartuz edo baliabideak ordena jakin batean hartuz.

Elkar-blokeoa gertatzen denean, konponketa bakarra dago: blokeatutako transakzioetako bat ezeztatzea. Ondorioz, ezeztatu den transakzio horrek blokeatuta zeuzkan baliabideak askatuko dira eta agian korapiloa askatuko da, blokeatutako beste transakzio batek, orain, falta zuen baliabidea har baitezake eta, ondoren, jarraitu duen transakzioa buka daitekeelako.

Denbora-marken teknika (*timestamp*): ezkorra da. Blokeorako teknikek gune kritikoan lehenago sartutako transakzioa lehenesten dute (lehenago eskatzen duenari ematen zaio blokeoa). Denbora-teknikek, berriz, lehenago hasitako transakzioa hobesten dute. Transakzio bakoitzari hasten denean

hasierako denbora-marka bat lotzen zaio. Bigarren hasitako transakzioak eraldatutako balioa ezin izango du lehenengo transakzioak irakurri. Kasu horretan, bigarren transakzioa desegingo da eta lehenengoak jarraituko du aurrerantz, hasieran zegoen balio zaharrarekin. Teknika horrekin ez da elkar-blokeorik gertatzen.

Bertsio ugariko denbora-marken teknika: aurrekoaren antzerakoa da, baina datu beraren hainbat bertsio daudela suposatzen du (eta bertsio bakoitzean transakzio bakarra erabiliko da). Teknika hori *Inprise* etxearen (*Borland*) *Interbase* datu-baseek erabiltzen dute, baina ez da oso erabilia.

Transakzio habiaratuen teknika: transakzio handiak azpitransakzio txikiagoetan zatikatzean datza. Azpitransakzio horiek modu konkurrentean edo aldi berean egikaritu ahal izango dira (edo berriro zatikatu...) eta bikorkadura txikiagoarekin. Ondorioz, datu-basearen errendimendua hobetuko da. Beste transakzio batzuk gorengo transakzioa soilik ikusiko dute, transakzio arrunta balitz bezala; beraz, teknika hori beste teknikekin nahas daiteke, bai onerako zein txarrerako.

Transakzio luzeen teknikarekin ere antzerako zerbait gertatzen da. Denboran zehar oso luzeak diren transakzioen kudeaketaz arduratzen da. Azkenean, helburua hobeto kudeatu eta errendimendu handiagoa emango duten azpitransakzio txikiagoetan zatitzean datza.

Azken teknika, egun gehien erabiltzen den **koordinaziorako transakzioena** da. Transakzio horien helburua datu-baseak kudeatzen dituen baliabideen atzipena koordinatzea da. Bertsioen kontrol-mekanismo batean oinarritzen da. Erabiltzaileak blokeoa zenbait taldetan multzotzeko aukera ematen du eta horrela isolamendu-maila ugari lortzen dira.

SQL sententziak erabiliz osotasuna kudeatzeko zenbait instrukzio daude:

- Osotasun semantikoaz ari garenean: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, CREATE DOMAIN, CREATE ASSERTION...

Adibidez:

```
CREATE DOMAIN Kodeak AS NUMBER (6);
CREATE TABLE LANGILEAK (
  LangZenb Kodeak,
  LangIzena VARCHAR2 (40),
  Soldata NUMBER (4),
  DepartamZenb Kodeak NOT NULL,
  CONSTRAINT LangZenbBaldintza PRIMARY KEY (LangZenb),
  CONSTRAINT DepartamBaldintza FOREIGN KEY (DepartamZenb)
  REFERENCES DEPARTAMTAULA (DepartamZenb),
  CONSTRAINT IzenEzBerdinak UNIQUE (LangIzena),
  CONSTRAINT SoldataTartea CHECK (Soldata BETWEEN 300 AND 9990));
```

Eragiketa horrekin, lehenengo, KODEAK izeneko domeinua sortzen da. Ondoren LANGILEAK izeneko erlazioa sortzen da eta CONSTRAINT sententziak erabiliko dira arauak jartzeko. Erlazioak lau atributu ditu. Lehena, “LangZenb” gako nagusia izango da (PRIMARY KEY); bigarrena, ‘LangIzena’, bakarra izango da, hau da, bi langilek ez dituzte izen eta abizen berdinak izango (UNIQUE); hurrengo atributuak, ‘Soldata’, 300 eta 9990 bitartean egon beharko du (CHECK); eta azken

atributuaren ('DepartamZenb') balioak DEPARTAMTAULA izeneko erlazioan dagoen "DepartamZenb" atributuetakoren baten berdina izan behar du (FOREIGN KEY ... REFERENCES ...).

- Konkurrentziaz ari garenean: SQL estandarrak ez dakar datuak blokeatzeko sententziarik. Hori bai, *Oracle*k adibidez, LOCK dauka.

LOCK TAULAIZENA IN BlokeoMota MODE

Datu-base banatuak

9

9.1. SARRERA

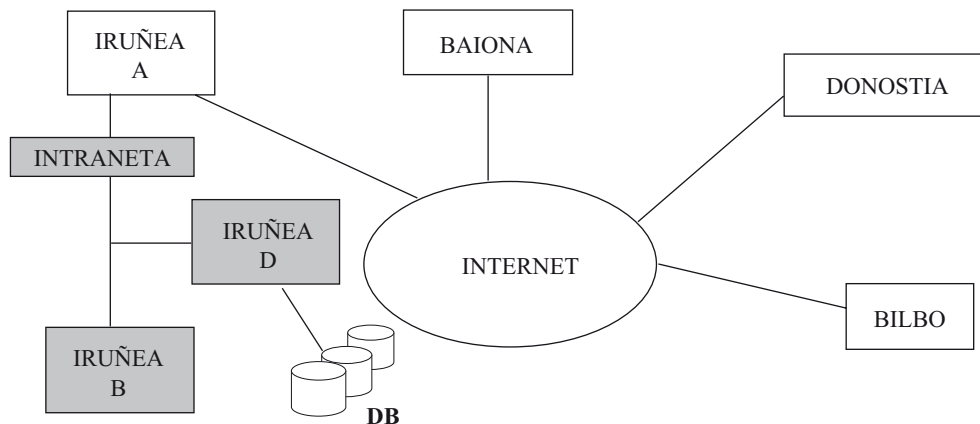
Laurogeita hamarreko hamarkadaren hasieran bi gertaera garrantzitsu izan ziren. Batetik, datu-baseen ugalketa eta, bestetik, ordenagailu-sareen garapena. Bion uztartetatik jaio ziren datu-base banatuak (DBB). Mota horretako datu-baseetan, era logikoan erlazionaturiko datuak kokapen ez-zentralizatueta egoten dira. Datuak banatuta egotean, sareak erabili beharko dira hainbat puntutan (kokapen) dauden datuak hartzeko.

Horren guztiaren kudeaketa Datu-base Banatuak Kudeatzeko Sistemak (DBBKS) egiten du.

Bi dira datu-base banatuen garapenean eragina duten faktoreak: erabiltzaileek hainbat kokagunetan lan egiteko duten beharra, eta hori posible egiteko teknologiak eskaintzen dituen aukera berriak.

9.2. DATU-BASE BANATUEN SISTEMAK

Datu-base banatua datu-base askoren multzoa da. DBBKS batek kontrolatuko du multzoa. Erabiltzaileak atzipen bat egiten duenean, datuak ordenagailu berean bilduta baleude bezala jokatzeko du, eta DBBKSa arduratuko da datuak erabiltzailea dagoen tokitik edo, sarearen bitartez, beste tokiren batetik hartzeaz.

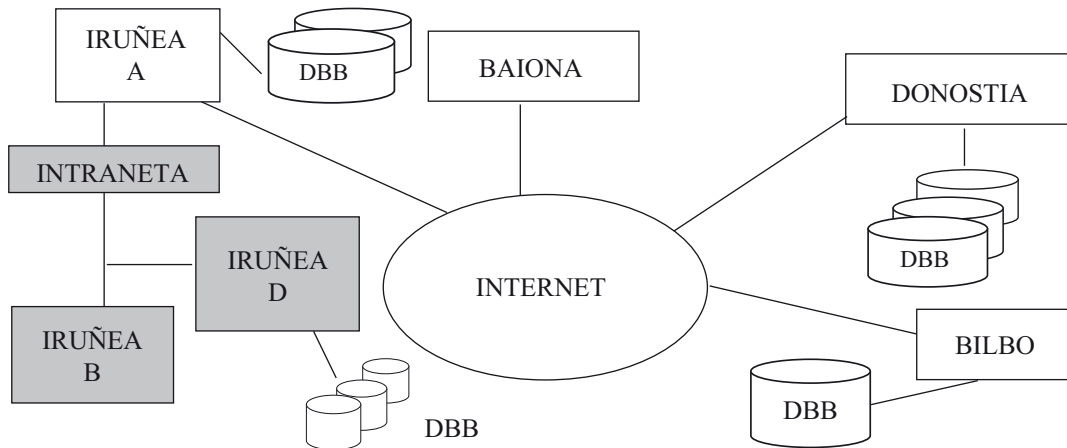


9.1 irudia. Sare bitartez atzi daitekeen datu-baseen sistema zentralizatuaren adibidea. Bezero/zerbitzari eredua. Ez da DBB sistema, datuak Iruñean baino ez daudelako.

Ordenagailuek hainbat tamaina eta funtzio izan ditzakete. Hala ere, baldintza batzuk bete behar dituzte:

- Memoria nagusia ez konpartitzea eta Datu-base Banatua Kudeatzeko Sistema bakarra izatea.
- Ordenagailu bakoitzak DBBKsaren zati bana edukitzea. Gutxienez, datu-baseak bi kokapenetan egon behar du banatuta; bestela, bezero/zerbitzari sistema klasiko baten aurrean aurkituko baikinake (ikusi 9.1 eta 9.2 irudiak).

Datu-base banatu bateko ordenagailuei **nodo** edo **adabegi** deritze.



9.2 irudia. Datu-base banatuaren adibidea. Datuak hainbat kokapenetan daude, ez bakar batean.

Beraz, DBBKSa sare baten bitartez konektatuta (LAN, MAN edo WAN) dagoen nodo-multzoa baino ez da. Gaur egun, IP sareen garapenarekin, konexioa egiteko aukerak Internet bidezkoak, Intranet bidezkoak edo mistoak dira. Sistema horretara datu-base banatuaren zatirik ez duten ekipoak ere konekta daitezke.

Nodo bakoitzak atzipen-mota bi egin ditzake:

- Lokalak: nodoan bertan dauden datuak atzitzen dira.
- Orokorrak: nodoan bertan ez dauden datuak atzitzen dira.

Banaketa egiteak abantailak eta desabantailak dauzka. Hauek dira abantaila garrantzitsuenak:

- Datuen erabilera konpartitua eta kontrolaren banaketa: nodo bakoitzak bertan dauden datuen inguruko kontrola du, nahiz eta administratzaile orokorraren kontrolpean egon. Kontrol hori ezberdina izan daiteke beharren edo tamainaren arabera.
- Nodo batean hutsegiteren bat gertatzen bada, sistemak funtzionatzen jarrai dezake eta, neurri egokiak hartu baldin badira, baita hutsegitetik berreskuratu ere.
- Kontsulten prozesua bizkortzea: normalean erabiltzen dituzten datuak erabiltzaileek hurbil badituzte, kontsultak bizkortu egingo dira. Horrez gain, kontsultak hainbat nodotan dauden taulei badagozkie, prozesua arintzeko azpikontsulta paraleloak egin daitezke.
- Eskalagarritasun ziurtatua: hedatzeko aukera handia dago, prozesadore zentral batekiko mendekotasunik ez dagoelako. Demagun 9.2 irudian delegazio berri bat sortzen dela Gasteizen, zentraleko datuak edukiko lituzkeena.

Desabantaila garrantzitsuenak, berriz:

- Softwarearen garapena garestitzea.
- Datuak errepika daitezkeenez, konkurrentziaren eta berreskurapearen mekanismoak kontrolatzea zailagoa da.
- Kontsultak hobetzeko algoritmoen zailtasuna handiagoa da, datuak sakabanatuta egon daitezkeelako. Komunikaziorako kostuak ere kontuan hartu behar dira.
- Hainbat nodok parte hartzen dutenean, erantzun-denborak altuagoak izango dira.
- Osotasuna kontrolatzea askoz zailagoa da.

Datu-base banatuak kudeatzeko sistemak kontuan izan behar dituen puntu azpimarragarrienak hauek dira:

- Kotsulten prozesaketa: datu-basea hainbat lekutan banatuta dagoenez, kontsulta optimoa egiteko, ahalik eta operazio gutxien egiteko modua asmatu behar da. Arazo hori konpontzeko, ez dago irtenbide zehatzik, eta egon daudenak gutxi gorabeherakoak dira. Gainera, kontuan hartu behar da kontsulta egiterakoan informazioa garraiatzeak hartzen duen balioa.
- Konkurrentziaren kontrola: datu-basean atzipenak batera gerta daitezke. Hori gertatzen denean, konkurrentzia kontrolatu behar da, erabiltzaileek egiten dituzten operazioek elkarrengan ez eragiteko. Datu-base banatuetan kontrola zailagoa da eta, nahiz eta kontrol hori egiteko erabiltzen diren algoritmoak datu-base zentralizatuetan erabiltzen direnen antzerakoak izan, sistema banatuek dituzten berezitasunak kontuan hartzeko aldaketa batzuk dituzte.
- Fidagarritasuna: Datu-base banatuetan transakzioek leku bati baino gehiagori eragin diezaiokete. Ondorioz, parte-hartzaile guztiek COMMIT edo ROLLBACK batera egiten dutela bermatu behar da.

9.3. DATU-BASE BANATUEN DISEINUA

Datu-base banatuak datu-base erlazionaletan oinarritzen dira. Jarraian, DBBen diseinuaz arituko gara. Diseinu hori egin ondoren, ezarketa DBKS hierarkikoen bitartez, sare-motakoen bitartez edo DBKS erlazional bat erabiliz egingo da. Hori bai, orokorrean DBBa eredu erlazionala jarraituz ezarriko da.

Datu-baseen sistemaren diseinua egiterakoan, hiru dimentsio hartzen dira kontuan bereziki:

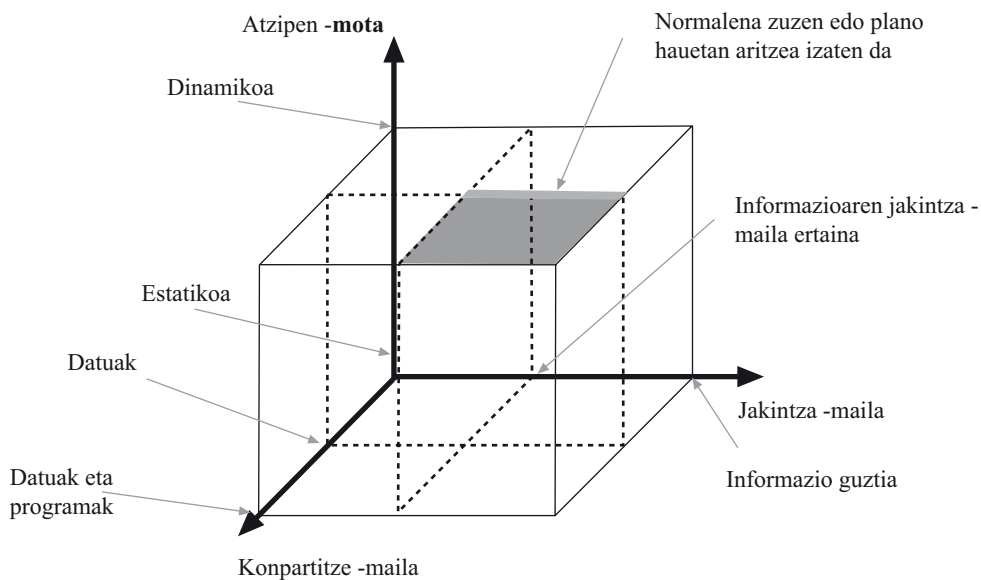
1. Datuak zenbat konpartitzen diren:
 - a. Ezer ez. Aplikazio bakoitza ordenagailu bakar batean dago eta bertan ditu beharrezko datuak.
 - b. Datuak konpartitu, baina programak ez. Datuak sarean ibiliko dira baina ordenagailu bakoitzak programaren kopiak egikarituko ditu.
 - c. Datuak eta programak konpartitu. Konpartitu egiten dira bai datuak, bai programak.

Kasu honetan, bigarren konpartitze-mailan kokatuko dugu gure burua.

2. Datuak atzitzeko moduak:
 - a. Estatikoa. Datu berberak atzitzen dira beti (SQL sententzia berdinak egikaritzen dira).
 - b. Dinamikoa. Orokorrean, DBBKS batean atzipen dinamikoa erabiliko da. Erabiltzaileek, orokorrean, kontsulta bakoitzean hainbat datu eskatuko dituzte.
3. Atzipen-ezaugarriei buruz zenbatekoa den ezagutza.

Datu-basearen diseinatzaileek ez dakitenean erabiltzaileek datu-basea nola atzitzeko duten. Aukera teorikoa da, baina errealitatean oso zaila egingo litzateke datu-base banatua horrela diseinatzea. Gainera, gutxienez zerbait jakiten da gehienetan. Behintzat, bilaketarako indize nagusiak jakiteko beste.

- a. Egoerarik onena erabiltzaileek datu-basea nola atzitzeko duten zehatz-mehatz jakiten denean gertatzen da.



9.3 irudia. Datu-basea diseinatu aurretik egin behar den sailkapena.

Hiru ezaugarri horien menpe diseinatzea zaila izaten da (ikus 9.3 irudia). Konpartitzerik ez dagoen kasuan izan ezik, beste kasu guztietan datu-base zentralizatueta agertzen ez diren arazoak ere azalduko dira. 9.3 irudiarri begiratuz jakin daiteke orokorrean atzipen-mota dinamikoa erabiliko dela, datuak (marra zuzena) edo datuak eta programak banatuko direla eta erabiltzaileek egingo dituzten kontsultei buruzko informazio asko (erdia baino gehiago) izango dela.

Aurreko gaitan landu dena erabat laburbilduz, datu-base zentralizatuak diseinatzerakoan, honako puntu hauek hartu behar dira kontuan:

- Eskema kontzeptualaren diseinua eta eskema logikoa.
- Datu-basearen diseinu fisikoa; hau da, eskema logikoa biltegietara itzuli, bihurtu, egokitu eta datu-basera atzipena egiteko nolako metodoak beharko diren zehaztea.

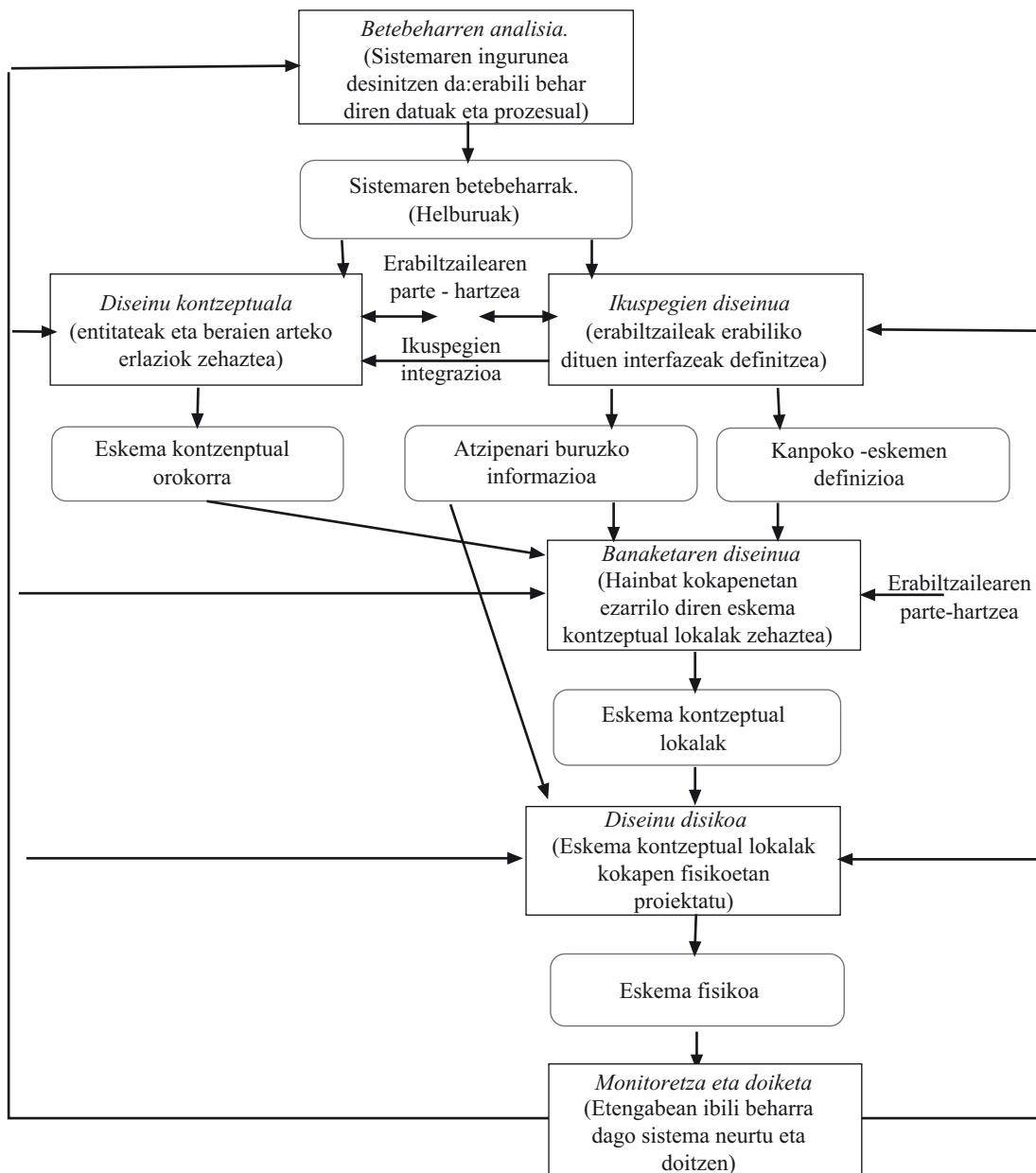
Eta datu-base banatueta, beste puntu bereizgarri hauek:

- Zatiketaren diseinua: erlazioak nola zatitzen diren (horizontalean, bertikalean edo modu mistoan).
- Zatiak banaketa: zatiak non kokatzen diren.

Diseinuaren arazo nagusia informazioa zenbait lekutan banatzea da; hau da, informazioa nola zati-titu eta sareko hainbat puntutan zati horiek nola banatu.

Kontuan hartu beharko da informazioa errepikatuta dagoen ala ez; horrela bada, sendotasuna nola mantendu aztertu beharko da.

Datu-base banatueta diseinua egiterakoan, bi estrategia-modu izaten dira: goranzkoa (*Bottom-Up*) eta beheranzkoa (*Top-Down*). Batera erabil daitezkeen estrategiak dira eta diseinua egiterakoan, ohikoa izaten da proiektuaren hainbat puntutan estrategiak ere ezberdinak izatea. Adibidez: goranzko estrategia datu-base txiki asko handiago batean bildu nahi direnean erabiltzen da. Kasu horretan, eskema kontzeptual lokalak hartu eta beraien gainean lan egingo da eskema kontzeptual orokorra lortzeko. Beheranzko estrategia orokorra zerotik hasten denean erabiltzen da. 9.4 irudian, datu-baseen diseinuan ohikoak diren pausoak erakusten dira, baina datu-base banatueta dituzten ezaugarriak gehituta.



9.4 irudia. Beheranzko estrategia erabiliz diseinatzerakoan jarraitu behar diren pausoak.

Datuen banaketa egiterakoan, honako hauek hartu behar dira kontuan:

- Prozesaketa lokalaren maximizazioa: datuak erabiltzen dituzten aplikazioetatik ahalik eta hurbilen kokatzea.
- Lan-zamaren banaketa: ahal den paralelismorik handiena lortu nahian, datuen prozesaketa hainbat ordenagailutan banatzea. Banaketa badago, ez dago prozesaketa lokalik; beraz, tarteko irtenbideak aurkitu behar dira.
- Erabilgarritasunaren kostua: bi ikuspuntutatik azter daiteke: alde batetik, datuak garraiatzeko sarea erabiltzearen kostua (konexioak balio duena, prezioa) eta, beste alde batetik, denboraren kostua, hau da, datuak heldu arte itxaroten igarotzen dena.

9.3.1. Zatiketa eta kokapena

Informazioa gordetzeko orduan, banaketa naturala baino unitate txikiagoak bilatzen dira. Erlazioa bertikalean begiratzean, atributuz osaturik dagoela ikus daiteke; eta horizontalean begiratu, tuploz osatuta dagoela.

Zatiketak egiteko orduan, horizontalean edo bertikalean egin daitezke. Zatiketa horizontalak tuploen inguruan egiten du lan, erlazioa azpirlazioetan banatuz. Horietako azpirlazio bakoitza jatorrizko tuploen azpimultzoek osatzen dute. Bertikalean gauza bera gertatzen da, baina atributuekin. Horiez gain, badaude zatiketa mistoak ere, zeinetan biak nahasten diren gelaxkako zatiketa egiteko. Lehenik, zatiketa bertikala garatzen da eta, ostean, aurrekoaren zatiekin horizontala, edo alderantziz.

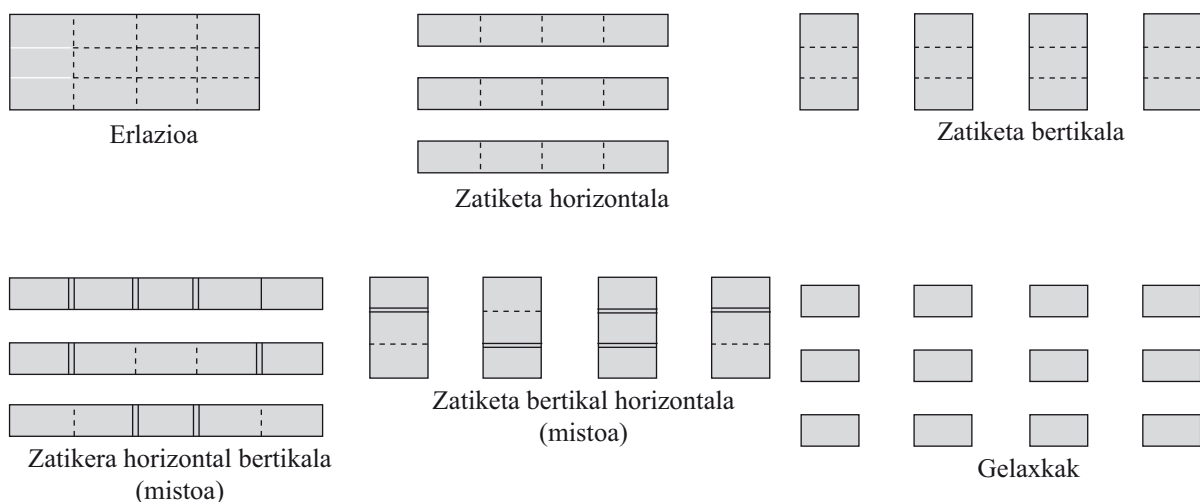
Zatiketa egiterakoan, baldintza-arau batzuk bete behar dira:

- e erlazioa e_1, e_2, \dots zatitan deskonposatzen da eta osoa dela esaten da, e-n dagoen elementu bakoitza gutxienez e_i batean baldin badago.
- e erlazioa e_1, e_2, \dots zatitan deskonposatzen da eta e_i -etatik erlazioa birsor daiteke.

Zatiketa eta datuen banaketa egiterakoan, derrigorrezkoa da hainbat informazio biltzea; batez ere:

- Datu-basearen antolaketa logikoa: eskema erlazional osoa, gako nagusiak, gako alternati- boak,...
- Aplikazioen kokapena: horren arabera, antolaketa bat edo beste erabiliko da.
- Aplikazioen atzipen-ezaugarriak: aplikazio bakoitzak zein datu irakurriko dituen, egunera- tuko dituen... Sarearen topologia eta ezaugarriak ere garrantzitsuak dira.
- Leku bakoitzeko ezaugarriak: leku bakoitzean softwarea eta hardwarea zein den.

DBBKS batean datu-basea nola eta non zatituko den erabaki behar da. Hasiera batean ez dela errepikapenik egongo pentsatu behar da. Bestela, errepikapenek eta horren ondorioek are gehiago zailduko dute prozedura. Dena den, abiapuntua datu-base erlazionalaren diseinua egitea da, eta be- raren gainean erabakiko da leku ugari horien artean erlazioak nola banatu. 9.5 irudian ikus daitezke zatiketa-mota posibleak.



9.5 irudia: Datu-base banatuetan egon daitezkeen zatiketa-motak.

Demagun 9.1 taula aztertu nahi dela, zeinetan ospitale batean dauden gaixoei buruzko datuak dauden.

GAIXOAK⁷ (GaixoKodea, EskualdeKodea, Gela, JaiotzaData, NA, IzenAbizenak, Helbidea, Herria).

GaixoKodea	EskualdeKodea	Gela	Jaioteguna	NA	IzenAbizenak	Helbidea	Herria
10	DEBA_G	123A	2004/01/5	12121212Y	Fernandez Azkue, Patxi	X1	ELGETA
12	DEBA_B	411B	1993/06/9	32323232Z	Axpe Azpilikueta, Alain	Y1	EIBAR
6	DURANGALDEA	239A	1997/12/3	12344567Z	Lopez Perez, Aitor	Z1	IURRETA
9	DEBA_G	121C	2000/03/22	71924521A	Iturri Olañeta, Maitane	X2	BERGARA
2	DEBA_G	423B	1984/01/31	19345421C	Maguregi Casas, Felisa	Z2	BERGARA
14	LEA ARTIBAI	391A	1998/07/7	72834521H	Sanchez Garcia, Luisa	C2	MARKINA
3	LEA ARTIBAI	213A	1999/11/6	42244224J	Martin Apalategi, Eneko	B2	LEKEITIO
17	BILBAO	415A	1955/12/27	44325677B	Rodriguez Perez, Maite	D3	BASAURI
11	BILBAO	111B	1975/05/7	72344327G	Calderon Zalla, Irene	X5	BILBAO
18	URDAIBAI	415B	1993/06/4	89001233R	Marcos Zubia, Iñaki	Z4	GERNIKA
.....

9.1 taula. GAIXOAK erlazioa.

- Zatiketa horizontala: erlazioko tuploen azpimultzoa da. Baldintza bat jarriko da azpimultzoak egiteko. Demagun, Deba ibaiaren inguruari buruzko ikerketa egiteko, ingurutik galderak egiten direla.

```
SELECT * FROM gaixoak
```

```
WHERE EskualdeKodea IN (DEBA_G, DEBA_B)
```

Ondorioz, X kokagune horretarako zatiketa horizontal hau egin daiteke (ikusi 9.2 taula):

GaixoKodea	EskualdeKodea	Gela	Jaioteguna	NA	IzenAbizenak	Helbidea	Herria
10	DEBA_G	123A	2004/01/5	12121212Y	Fernandez Azkue, Patxi	X1	ELGETA
12	DEBA_B	411B	1993/06/9	32323232Z	Axpe Azpilikueta, Alain	Y1	EIBAR
9	DEBA_G	121C	2000/03/22	71924521A	Iturri Olañeta, Maitane	X2	BERGARA
2	DEBA_G	423B	1984/01/31	19345421C	Maguregi Casas, Felisa	Z2	BERGARA
.....

9.2 taula. GAIXOAK erlazioaren zatiketa horizontala.

```
Z_DEBA=S (EskualdeKodea IN ('DEBA_G','DEBA_B')) (GAIXOAK)
```

- Zatiketa bertikala: erlazio batean dauden atributuen azpimultzoa da. Adibidez: kokagune batean gaixoen NA, izena eta herria soilik nahi dira.

⁷ 9.1 taulan ez da gakoak finkatu oraindik, zein den ez bageneki bezala jokatu dugulako. Adibidea aurrera doan heinean zehaztuko dugu. Ez ahaztu, halaber, erlazioak ezinbestean izan behar duela gakoaren bat.

SELECT NA, IzenAbizenak, Herria FROM gaixoak

Ondorioz, kokagune horretarako zatiketa bertikal hau egin daiteke:

NA	IzenAbizenak	Herria
12121212Y	Fernandez Azkue, Patxi	ELGETA
32323232Z	Axpe Azpilikueta, Alain	EIBAR
12344567Z	Lopez Perez, Aitor	IURRETA
71924521A	Iturri Olañeta, Maitane	BERGARA
19345421C	Maguregi Casas, Felisa	BERGARA
72834521H	Sanchez Garcia, Luisa	MARKINA
42244224J	Martin Apalategi, Eneko	LEKEITIO
44325677B	Rodriguez Perez, Maite	BASAURI
72344327G	Calderon Zalla, Irene	BILBAO
89001233R	Marcos Zubia, Iñaki	GERNIKA
.....

9.3 taula. GAIXOAK erlazioaren zatiketa bertikala.

$$Z_PERTSONAK_1 = \Pi_{(NA, IzenAbizenak, Herria)} (GAIXOAK)$$

Zatiketa egiterakoan, kontuz ibili behar da, ordea. Gogoratu edozein zatiketarik hasierako erlazioa birsortzeak posible izan behar duela. Gaixo guztien zerrenda lortu nahi da, beraien datu guztiekin. Beraz, GAIXOAK erlazioko gako nagusia zein den jakin behar da zatiketa egin aurretik, beste kokagune batzuetako gaixoen datu guztiak bildu ahal izateko. Gako nagusia 'NA' balitz, ez legoke arazorik eta dena ongi legoke. Baina, zer gertatzen da gako nagusia 'GaixoKod' bada? Zatiketa, orduan, ez dago ondo, ezin baita hasierako erlazioa birlortu. Hori, edo gako alternatiboekin ari garela esan behar zaio sistemari. Gako alternatiboen kontua alde batera utziz, hasierako galderari erantzuteko beharko diren datuak honako hauek izango dira:

$$Z_PERTSONAK_2 = \Pi_{(GaixoKodea, NA, IzenAbizenak, Herria)} (GAIXOAK)$$

GaixoKodea	NA	IzenAbizenak	Herria
10	12121212Y	Fernandez Azkue, Patxi	ELGETA
12	32323232Z	Axpe Azpilikueta, Alain	EIBAR
6	12344567Z	Lopez Perez, Aitor	IURRETA
9	71924521A	Iturri Olañeta, Maitane	BERGARA
2	19345421C	Maguregi Casas, Felisa	BERGARA
14	72834521H	Sanchez Garcia, Luisa	MARKINA
3	42244224J	Martin Apalategi, Eneko	LEKEITIO
17	44325677B	Rodriguez Perez, Maite	BASAURI
11	72344327G	Calderon Zalla, Irene	BILBAO
18	89001233R	Marcos Zubia, Iñaki	GERNIKA
.....

9.4 taula. GAIXOAK taularen zatiketa bertikala GAIXOAK taulako gako nagusia 'GaixoKodea' balitz.

Kasu horretan, GAIXOAK erlazioa lortzeko:

$$Z_PERTSONAK_2 \mid x \mid \Pi_{(GaixoKodea, EskualdeKodea, Gela, JaiotzaData, Helbidea)} (GAIXOAK)$$

- Zatiketa mistoa: zatiketa bat egin eta gero, bestea aplikatzea da kontua. Bertikal-horizontala edo horizontal-bertikala izango da. Hasierako erlazioari batura eta biderkadura aplikatzen aterako da. Adibidez: Deba eskualdeko gaixoen izen-abizenak eta herria jakin nahi dira beste kokagune batean.

```
SELECT IzenAbizenak, Herria FROM gaixoak
WHERE EskualdeKodea IN (DEBA_G, DEBA_B)
```

Ondorioz, X kokagune horretarako zatiketa misto hau egin daiteke (ikusi 9.5 taula):

IzenAbizenak	Herria
<i>Fernandez Azkue, Patxi</i>	<i>ELGETA</i>
<i>Axe Azpilikueta, Alain</i>	<i>EIBAR</i>
<i>Iturri Olañeta, Maitane</i>	<i>BERGARA</i>
<i>Maguregi Casas, Felisa</i>	<i>BERGARA</i>
.....

9.5 taula. SELECT IzenAbizenak, Herria FROM gaixoak WHERE EskualdeKodea IN (DEBA_G, DEBA_B).

Baina ez, oraingoan garaiz konturatu gara hau SELECTaren emaitza dela, baina ezin dela kokagune batean ipini, bestela ezinezkoa izango delako hasierako erlazioa birsortzea (lehengo arazoa, baina, oraingoan, areagotua, gako nagusia eta gako alternatiboa, biak falta direlako). Beraz, nahiz eta galderari erantzuna emateko aurreko SELECT agindua ongi egon, kokapen berrian erlazioa sortzeko erabiliko genukeen SELECTa beste hau litzateke (gako nagusia 'GaixoKod' dela onartuz):

```
SELECT GaixoKodea, IzenAbizenak, Herria FROM gaixoak
WHERE EskualdeKodea IN (DEBA_G, DEBA_B)
```

Eta ondorioz, emaitza 9.6 taulako izango da:

GaixoKodea	IzenAbizenak	Herria
<i>10</i>	<i>Fernandez Azkue, Patxi</i>	<i>ELGETA</i>
<i>12</i>	<i>Axe Azpilikueta, Alain</i>	<i>EIBAR</i>
<i>9</i>	<i>Iturri Olañeta, Maitane</i>	<i>BERGARA</i>
<i>2</i>	<i>Maguregi Casas, Felisa</i>	<i>BERGARA</i>
...

9.6 taula. GAIXOAK taularen zatiketa mistoa gakoak kontuan izanda.

Ondorengo bi kasu hauetako emaitza berdina izango da, baina lortzeko jarraitzen diren pausoak, ezberdinak.

- Zatiketa horizontal-bertikala: Deba eskualdeko gaixoen (zatiketa horizontala) izen-abizenak, herria eta kodea (zatiketa bertikala).

$$Z_DEBAKO_PERTSONAK_1 = \Pi_{(GaixoKodea, IzenAbizenak, Herria)} (\sigma_{(EskualdeKodea \text{ IN } ('DEBA_G', 'DEBA_B'))} (GAIXOAK))$$

- Zatiketa bertikal-horizontala: gaixoen izen-abizenak, herria, eskualdea eta kodea (zatiketa bertikala), non Deba eskualdekoak diren (zatiketa horizontala).

$Z_DEBAKO_PERTSONAK_2 = \sigma_{(EskualdeKodea \text{ IN } ('DEBA_G', 'DEBA_B'))}$

$(\Pi_{(GaixoKodea, IzenAbizenak, Herria, EskualdeKodea)}(GAIXOAK))$

Zehatzagoak izateko, emaitza ez litzateke guztiz berdina, bigarren kasuan zutabe bat gehiago izango baikenuke ('EskualdeKoda').

- Gelaxkako zatiketa: zatiketa mistoa azken muturreraino eramaten denean erabiltzen da. Kasu horretan, zati bakoitzak hasierako erlazioan gelaxka bakarrean dagoen informazioa izango du (hori bai, erlazioa osatzeko beharrezkoak diren gakoak ere behar dira). Orokorrean, zatiketa hori egin ondoren, zenbait gelaxka elkartzen dira sareko aplikazioen eta kokaguneen beharren arabera, gelaxkako zatiketa osoaren errendimendua ez baita oso handia oro har. Helburua zatiketa espezifikoa egitea izaten da, beharren arabera, sistemari etekin handiagoa ateratzeko.

9.3.2. Errepikapena eta kokapena

Zatiak zein lekutan banatzen diren zehaztu behar da. Zatia leku batean baino gehiagotan banatzen bada, errepikatuta dagoela esaten da.

Informazioa errepikatzeak kontsulten paralelismoa areagotzea ahalbidetzen du. Horrez gain, fidagarritasuna gehitzen du. Bestalde, eguneratzea, konkurrentziaren kontrola eta berreskurapena zaildu egiten dituela ere esan behar da.

Beraz, errepikapena egiteko irizpide ugari erabil daitezke:

- Errepikapenik ez egotea.
- Errepikapen osoa egitea; hots, leku guztietan datu-base osoa egotea.
- Errepikapen partziala.

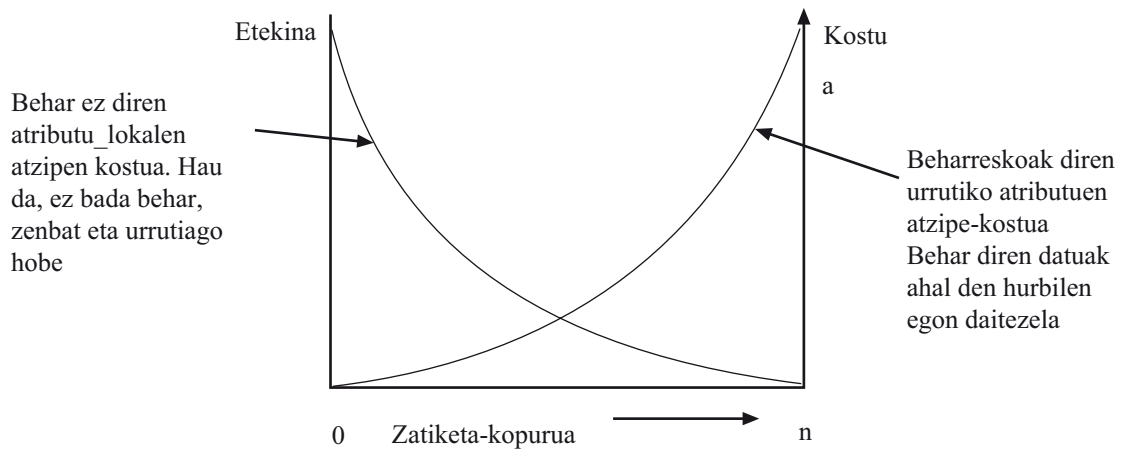
Arau orokor modura, kontsulten kopurua eguneratze-eragiketen kopurua baino handiagoa denean egiten da zatien errepikapena (ikusi 9.6 irudia).

Banaketa egiteko behar den informazioa lau mailatan zati daiteke: datuei buruzko informazioa, aplikazioei buruzka, sare-sistemari buruzkoa eta ordenagailuei buruzkoa.

Sare bidez lotutako kokapenak eta aplikazioak kontuan hartuz aukeratu behar dira kokapenak (ikusi 9.6 irudia). Irizpide bi hartzen dira kontuan:

- Gutxieneko kostua. Bai diruaren ikuspuntutik (sare telematikoen kostua), bai datuak atzitzerakoan sortzen diren zailtasunaren eta atzerapenen ikuspuntutik.
- Etekina.

Kostu minimoa eta etekin maximoa izango litzateke aukerarik onena, baina kontuan hartu behar da bata hazi ahala bestea jaitsi egiten dela. Haien arteko oreka bilatu behar da.



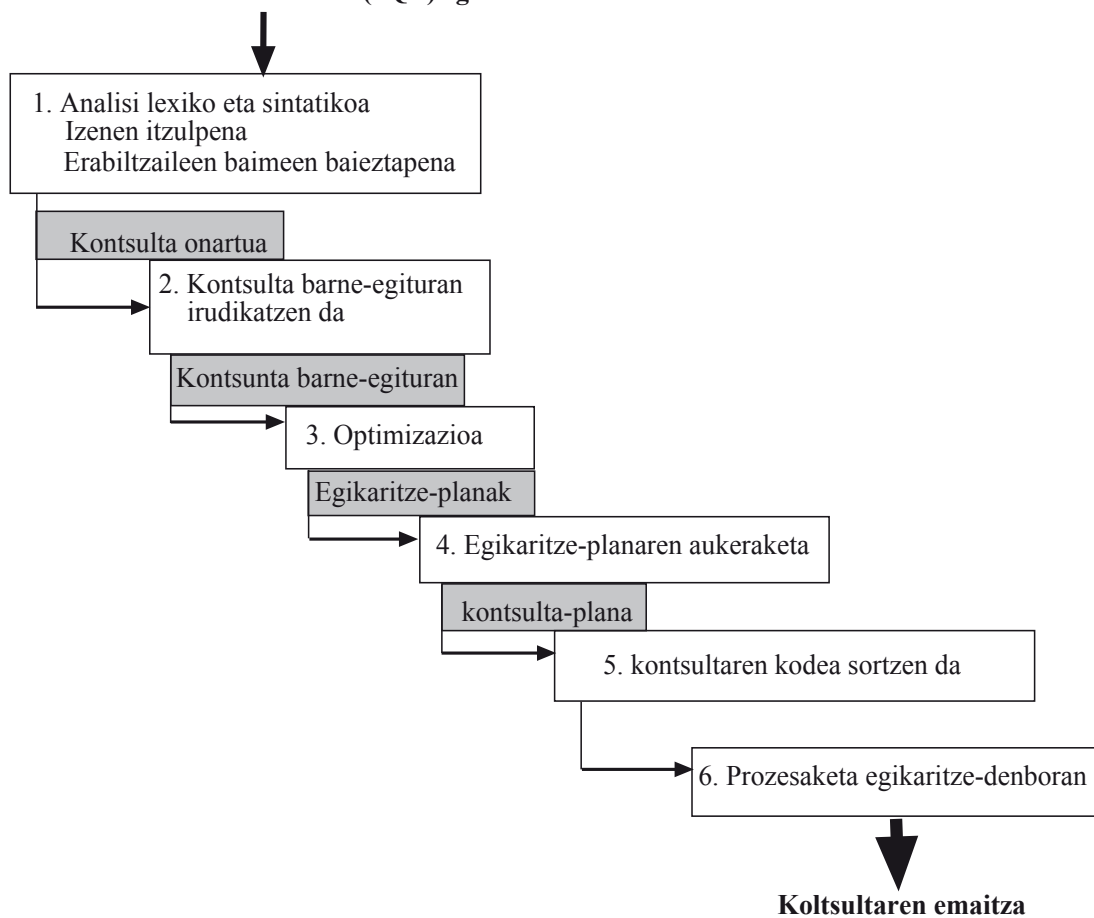
9.6 irudia. Atributuen kostuen arteko erlazioa.

9.4. KONTSULTEN PROZESAKETA

DBKSetan, kontsultak egiteko zenbait fase bete behar dira: analisia, barne-egitura, optimizazioa, egikaritze-planak...

Kontsulten optimizazioen helburua DBBKSan erantzun-denborak hobetzea da (ikusi 9.7 irudia).

Goi-mailako hizkuntza batean (SQL) egindako kontsulta



9.7 irudia: Datu-base banatuetako kontsulten faseak

Atal honetan ez dira ohiko pausoak landuko. Hori guztiori, DBKS ez-banatueta kontsulten optimizazioaz hitz egiten den gaian landuta baitago.

Egun, kontsulten optimizazioa ikerkutzako arlo garrantzitsua da. Kontsultak optimizatzeko, bi eremuren inguruan lan egiten da: erabiltzailearen optimizazioan eta sistemaren optimizazioan.

Sistemek, eragiketak erregistroz erregistro egiten dituztenean (sare eredu eta eredu hierarkikoa), estrategia erabilgarri bakarra erabiltzailearen optimizazioa da.

Sistema erlazioaletan, eragiketak goi-mailakoak dira. Ondorioz, optimizazioaz sistema bera arduratzen da. Erabiltzaileak zer egin nahi duen ulertzen du eta modurik onenean erantzun; hala ere, erabiltzaileak ere optimizazio-lanak egiten baditu, sistemaren etekina asko hobetuko da⁸.

Kontsulten prozesaketa datu banatueta askoz zailagoa da zentralizatueta baino, faktore berri eta ezberdin askok parte hartzen dutelako. Dena den, kontsulta banatua kontsulta lokal askoren batura baino ez da.

9.4.1. Gardentasuna eta kontsulten prozesaketa

Diseinatzaileak, datu-basea banatzerakoan, honako irizpide hauek izango ditu kontuan, kokapen guztietan datu-base osoa egongo balitz bezala lan egin ahal izan dezan.

- Datu-base banatuen hainbat kokagunetan errepikatuta dagoen elementu batek izen bera izan behar du kokagune guztietan. Gainera, izen bakoitzak bakarra izan beharko du datu-base guztian; hau da, datu-baseko elementuek izen ezberdinak eta bereiziak izan behar dituzte. Hona hemen ezaugarri hori bermatzeko erabiltzen diren irtenbideak:
 - Izenak emateko orduan arduradun zentralizatu erabiltzea.
 - Leku bakoitzari dagokion aurrizkia erabiltzea.
 - Erabiltzaileek aliasak/ezizenak erabiltzea.
- Erabiltzaileak ez dauka zertan jakin datu-basea nola banatzen den; hau da, datuak modu eraginkorrean aurkitu behar ditu.
- Datuen kopiak egitea ez da erabiltzailearen ardura izango. Are gehiago: ez dauka zertan jakin kopia horiek existitzen diren ala ez; sistemak erabakiko du zer kopiatu behar den, une bakoitzean zein kopia erabili...
- Leku bakoitzak datu berriak sortzeko ahalmena izango du.

Datu-base banatueta kontsulten prozesaketetan berezitasun garrantzitsuak daude. Kontuan hartu behar dira sare-komunikazioari dagozkion datuak, prozesaketa hainbat lekutan paraleloan eman daitekeela edo kokapen bakoitzak duen prozesaketa-ahalmena eta berari dagokion kostua ezberdina izan daitekeela.

⁸ Informazio gehiago 7. gaian.

9.4.2. Datuen transferentziak

Kontsulta banatuak egiteko orduan, tarteko emaitzak eta bukaerakoak sare bitartez bidaltzen dira. Hori ondo antolatuz gero, kostuak txikitu egin daitezke. Horretarako, algoritmo bereziak erabiltzen dira.

9.7 eta 9.8 tauletakoa adibideak erabiliko dira, OSPITALEAK eta GAIXOAK erlazioak hain zuzen ere:

OSPITALEAK (OspiKodea, Ospilzena, GelaKop)

GAIXOAK (NA, Ospikodea, IzenAbizenak, Helbidea, Herria, JaiotzaData)

OSPITALEAK erlazioa.

- 200 erregistro (ikusi 9.7 taula).

OspiKodea	Ospilzena	GelaKop
10	Mendaro	70
30	Txagorritxu	105
50	Gurutzeta	400
70	Galdakao	150
90	Arrasate	50
...

9.7 taula. OSPITALEAK.

- Erregistro bakoitzak 50 karaktere: ‘OspiKodea’ 5 hizki, ‘Ospilzena’ 40 hizki eta ‘GelaKop’ 5 hizki.

GAIXOAK erlazioa:

- 100.000 erregistro.
- Erregistro bakoitzak 124 karaktere: ‘NA’ 9 hizki edo karaktere, ‘IzenAbizenak’ 40, ‘Helbidea’ 40, ‘Herria’ 20, ‘JaiotzaData’ 10 eta ‘Ospikodea’ 5.

Hiru kokagune baleude, K1ean OSPITALEAK erlazioa legoke, K2an GAIXOAK erlazioa eta K3an, galderak egiteko tokia.

NA	IzenAbizenak	Helbidea	Herria	JaiotzaData	Ospikodea
12121212Y	Fernandez Azkue, Patxi	X1	ELGETA	2004/01/5	10
32323232Z	Axpe Azpilikueta, Alain	Y1	EIBAR	1993/06/9	30
12344567Z	Lopez Perez, Aitor	Z1	IURRETA	1997/12/3	50
1924521A	Iturri Olañeta, Maitane	X2	BERGARA	2000/03/22	30
19345421C	Maguregi Casas, Felisa	Z2	BERGARA	1984/01/31	10
72834521H	Sanchez Garcia, Luisa	C2	MARKINA	1998/07/7	70
42244224J	Martin Apalategi, Eneko	B2	LEKEITIO	1999/11/6	30
44325677B	Rodriguez Perez, Maite	D3	BASAURI	1955/12/27	30
72344327G	Calderon Zalla, Irene	X5	BILBAO	1975/05/7	10
89001233R	Marcos Zubia, Iñaki	Z4	GERNIKA	1993/06/4	90
.....	

9.8 taula: GAIXOAK.

Kontsulta posible bat izan liteke gaixo baten izena emanda non dagoen (ospitalearen izena) eta ospitale horrek zenbat gela dituen jakitea. Erantzuna 45 bytekoa izango da (40 karaktere ospitalearen izenarenak eta 5 gela-kopuruarenak).

Hiru estrategia erabil daitezke:

1. GAIXOAK eta OSPITALEAK erlazioak galdera egin duen kokagunera bidali eta han egin eragiketa. Alde batetik 200 x 50 gehi beste alde batetik 100.000 x 124 byte bidaltzen dira.
2. GAIXOAK erlazioa OSPITALEAK erlazioa dagoen kokagunera eraman (hau da, GAIXOAK erlazioa K2tik K1era eraman), han biderketa egin eta emaitza K3ra eraman: 100.000 x 124 alde batetik eta 45 bestetik.
3. OSPITALEAK erlazioa GAIXOAK erlazioa dagoen kokagunera eraman (hau da, OSPITALEAK erlazioa K1etik K2ra eraman, han biderketa egin eta emaitza K3ra eraman: batetik 200 x 50, bestetik 45.

Lehentasuna datuen bolumena txikitzea denez, hirugarren estrategia da egokiena. Bolumena txikiagoa denez, datuak garraiatzearen kostua txikituko da.

Kontsultak optimizatzerako orduan, estrategia pare bat erabil daitezke: paralelismoa errazten dutenak eta semikonposizioa.

Paralelismoa errazten duten estrategien adibidetzat, demagun honako biderketa natural hau: $E = E1 \times E2 \times E3 \times E4$ eta erlazio bakoitza L leku (kokapen) ezberdin batean gordetzen dela. Emaitza $L1$ era eraman behar da.

- $E1$ $L2$ ra eraman eta $e1 \times e2$ kalkulatu da.
- $E3$ $L4$ ra eraman eta $e3 \times e4$ kalkulatu da.
- $L2$ an eta $L4$ an emaitzak eman ahala, $L1$ era eramaten dira.
- Emaitzak heldu ahala, azken eragiketa egiten da: $E1 \times E2 \times E3 \times E4$.

Semikonposizioaren estrategia konplexuagoa da; baina kasu batzuetan estrategia sinpleak baino hobea. Jo dezagun $E1$ eta $E2$ erlazioak ditugula $L1$ eta $L2$ lekuetan eta emaitza ($E = E1 \times E2$) $L1$ dagoen tokian nahi dela.

- $L1$ en ELAG izeneko erlazio laguntzailea erabiliko da eta bertan $E1$ eta $E2$ ko atributu komunen (berdinen) proiektzioa gordeko da. $ELAG = \Pi E1 \cap E2(e1)$
- ELAG $L2$ ra bidaliko da.
- $L2$ an ELAG2 izeneko erlazio laguntzailea erabiliko da. $ELAG2 = ELAG \times E2 = \Pi E1 \cap E2(e1) \times E2$. Ondorioz, $E2$ ko atributuez osatuta dagoen $E1 \times E2$ biderketaren zatia lortuko da.
- $L1$ era bidaltzen da ELAG2.
- $E = E1 \times ELAG2 = E1 \times \Pi E1 \cap E2(E1) \times E2$ egin eta emaitza lortzen da, $E1 \times \Pi E1 \cap E2(E1)$ eginez $E1$ lortzen baita; beraz, azkenean $E = E1 \times E2$ egiten da.

Estrategia horrek etekin hobek ematen ditu $E2$ ko tuplo gutxik parte hartzen dutenean. Helburua erlazioa bidali aurretik tuplo-kopurua gutxitzea da. $L1$ etik $E1$ erlazioak konposizioa egiteko erabiliko dituen zutabea(k) bakarrik bidaltzen d(ir)a $L2$ ra, $L2$ n $E1$ en zati horrekin konposizioa egin eta emaitzak $L1$ era bidaltzen dira. $L2$ tik $L1$ era konposizioa egin ondoren, geratutako $E2$ ko zutabeak bidaltzen dira. Estrategia horrekin datuen transferentzia bat gehiago egiten da ($L1$ etik $L2$ ra), baina $E2$ ko tuplo

gutxik parte hartzen badute, aurretik galdutako denbora-galera hori E2 osoa L1era bidaltzea baino hobe izango da.

Semikonposizioa $E1 \times E2 = \Pi E1(E1 \times E2)$ modura idazten da aljebra erlazionalean eta goiko adibidean ELA2 kalkulatzekoan egiten da.

Bi estrategia horiek ongi aplikatzeko, sistemak informazio gehigarria behar du (estatistikak), bestela erlazio askorekin ez luke jakingo E1 L2ra bidali edo alderantziz egin. Gainera, bi estrategietan kokagune bakoitzean optimizazioak gehi daitezke (ikusi 7. gaia).

9.5. KONKURRENTZIAREN KONTROLA

Sistema zentralizatueta konkurrentziaren kontrolerako erabiltzen diren tekniken hedapena da.

Datu-base banatuak zati errepikaturik ez badu, sistema ez-banatueta erabilitako blokeorako mekanismoen modu hedatua erabil daiteke. Leku bakoitzak blokeo-kudeatzaile lokala izango du, leku horretan dauden datuentzako transakzio-eskaerak administratzeko. Transakzio batek Ln dagoen D datua blokeatu nahi badu, leku horretan dagoen blokeo-kudeatzaileari mezua bidaliko dio, D datua blokeatzeko eskatuz. D datuak blokeo bateraezina badu, eskaera atzeratu egingo da, egin ahal izan arte, eta orduan blokeo-kudeatzaileak L lekura mezua bidaliko du blokeoa egin daitekeela esanez. Eskatzen duenak eskaera onartu den edo ez jakiteko *timeout* edo itxarote-denbora maximoa zehaztu behar du. Datua duen lekuak ere jakin egin behar du zenbatekoa den denbora hori. Erantzunik ez badago, eskatu duenak blokeoa ezeztatu zaiola jakingo du.

DBBetan teknika horiek ezartzea nahiko erraza da (batez ere zatiek kopiak ez dauzkatenean), baina blokeoen kudeaketa datu-base zentralizatueta baino konplexuagoa da.

Zatiak hainbat kokapenetan kopiatuta daudenean, konkurrentzia-kontrola zailagoa da. Hiru dira erabiltzen diren teknikak:

9.5.1. Blokeoaren koordinatzaile bakarra (lehentasunezko lekua)

Kasu honetan, blokeo-kudeatzaile zentralizatu erabiliko da, L lekuan kokatuta. Leku hori egokitasunaren arabera aukeratuko da.

Blokeoa behar duten transakzio guztiek aipatutako kokapenera jo beharko dute lehenengo. Han erabakiko da blokeoa noiz onar daitekeen eta, mezua baten bidez, eskatzaileari komunikatuko zaio. Transakzioek edozein lekutan irakur dezakete datua edo datuaren kopia. Irakurri beharrean idatzi egin nahi bada, eragiketa leku guztietan egin beharko da.

Teknika hori blokeorako kontrol zentralizatuetaoen hedapena baino ez da; hori bai, kasu honetan, datuak banatuta egongo dira.

Teknika horrek bi eragozpen ditu:

- Ahulunea lehentasunezko kokagunea/lekua da. Eroriz gero, sistema bertan behera geratzen da. Hobetzeko, eta badaezpada, bigarren leku nagusiren bat ere erabil daiteke.
- Lehentasunezko lekuan datuen joan-etorria izugarria da, eta prozesua erabat geldotzen da.

9.5.2. Blokeoaren koordinatzaile banatua (lehentasunezko kopia)

Kasu honetan, blokeoaren kudeaketa zenbait lekuren artean banatzen da. Datuaren kopietako bat lehentasunezkoa izango da eta bertan egongo da datu horren blokeoa koordinatzearen ardura. Hori bai, beste datu bat blokeatzen bada beste leku batean, bertan izango da haren kontrolaren ardura.

9.5.3. Gehiengo bidezko blokeo-koordinatzailea

9.5 puntuaren hasieran ikusitakoaren aldaera bat da. Leku bakoitzak bertako datu guztien blokeoak kontrolatzen dituen blokeo-kudeatzaile lokala du. Transakzio batek blokeoa eskatzen badu, datu hori gordeta dagoen lekuen erdian baino gehiagoren erantzuna behar du. Kudeatzaile bakoitzak, autonomia osoz, mezu bat igorri, blokeoa baimentzeko aukera du. Kokapenen gehiengo baimena lortzen denean, transakzioa hasi duen lekuak mezua igorriko du datuaren kopia duten leku guztietara; horrela, datua bere esku duela adieraziko du. Kasu horretan ere gehienezko denbora erabili behar da; denbora hori igaro eta gero, ez badira kokapenen gehiengo oniritziak jasotzen, transakzioa ezeztatu egingo da.

Teknika hori oso konplexua da eta sarean trafiko handia sortzen du. Era berean, elkar-blokeoak ere sor daitezke.

9.6. BERRESKURAPENA

Sistemak erabiltzaileek egiten dituzten eskaerak prozesatzen ditu. Datu-base banatua kudeatzeko sistemak, gelditu gabe, kokagune jakin batek huts egiten duenean ere, erabiltzaileen eskaerak prozesatzen jarraitu beharko du. Fidagarritasunak gaitasun hori neurtzen du. Sistema gelditzen denean ere, hutsegitea konpondu ostean, datu-baseak normaltasunez lanean jarraitzea ahalbidetu behar da.

Datu-basea kudeatzeko sistema guztietan bilatzen da fidagarritasuna. Denbora zehatz batean sistemak hutsegiterik ez egitea da fidagarritasuna. Beste helburu bat zera da: datu-baseak beti erantzunen bat ematea erabiltzaileari, onerako zein txarrerako. Adibidez, galdera bat egiten bazaio datu-baseari, horren erantzuna ematea, edo bestela, zergatik posible ez den izan azaltzea, edo blokeatu dela esatea...

Sistema batean egon daitezkeen hutsegite-motak zein diren jakitea oso garrantzitsua da:

- Programaren logikari atxikitutakoak: datu desegokiak sarreran, aurkitu ezin den informazioa... Arazoak konpontzeko transakzioa bertan behera uzten da.
- Biltegitratzearekin zerikusirik ez duten kanpoko hutsegite logikoak/fisikoak: sistema eragilean hutsegiteak, DBBKSarenak...
- Biltegitratzeko azpisistemaren hutsegiteak.
- Komunikazio-arloko hutsegiteak. Hainbat kokagunetako ordenagailuak ondo aritu arren, sarean hutsegiteak badira, sistema geldituta gera daiteke. Sistema banatuen hutsegite propioak dira.

Konkurrentziaren kontrola eta berreskurapenaren funtsa ACID propietateak dira. DBBKSren zat oso zaila da atomizitatea bermatzea, transakzioetan hainbat lekuk parte har dezaketelako. Hori konpontzeko, DBBKSetan sistemaren leku bakoitzak bi azpisistema izaten ditu:

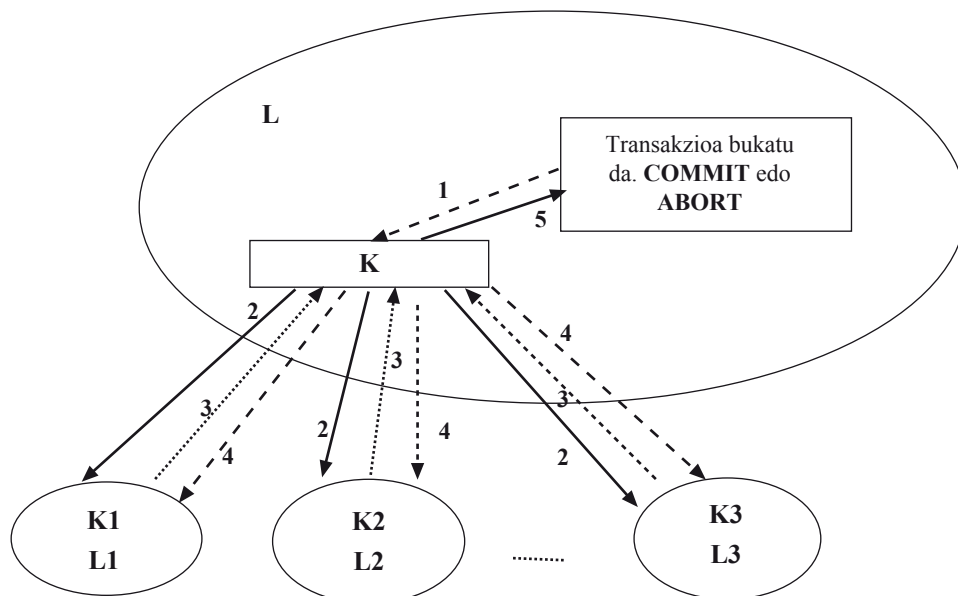
- Transakzioen administratzailea: sistema zentralizatueta dagoenaren antza du eta leku horretan dauden datuak erabiltzen dituzten transakzioak kontrolatzen ditu. Berreskurapenerako bitakora mantentzeaz eta konkurrentzia kontrolatzeko eskeman parte hartzeaz arduratzen da.
- Transakzioen koordinatzailea, leku horretan egikaritzen diren transakzioen koordinazioaz arduratzen da; hots, transakzioak abiarazteaz, transakzioak zatitzeaz (azpitransakzioetan banatzen ditu, bakoitza dagokion kokapenean egikaritzeko) eta transakzioen bukaeren koordinazioaz.

T transakzioen atomizitatea bermatzeko, transakzioan parte hartu duten leku guztiek azken emaitza bera izan behar dute, edo egikaritua edo abortatua. Hori lortzeko, ACPak erabiltzen dira, hau da, konpromiso atomikoko protokoloak.

9.6.1. Bi faseko konpromisoa (*two-phase commit, 2PC*)

2PC honek transakzio banatua ondo egikaritu dela baieztatzeko balio du. Transakzioaren azken urratsa ematen denean, koordinatzaileak protokoloa abiarazten du. L lekuak/kokapenak T transakzioa hasi badu, horren bukaera egokiaz arduratuko da; beraz, K koordinatzailearen lana bere gain hartuko du eta 2PC protokoloa T transakzioarentzat martxan jarriko du.

T transakzioa L lekuan hasi da eta bertako (Lko) koordinatzailea K da. Bukatu dutela esanez leku guztien baieztapena jasotzen denean, 2PC protokoloa abiarazten da (ikusi 9.8 irudia). Transakzioa bukatzen denean, koordinatzaileari jakinarazten zaio (1); DBBKS zenbait gunerekin konektatu eta beraien egoeraren berri emateko eskatzen die koordinatzaileak (2). DBBKSaren kokaguneen erantzunen arabera (3), erabaki bat hartzen da eta erabaki horren berri kokagune guztiei (4) eta transakzioari berari ematen zaio (5).

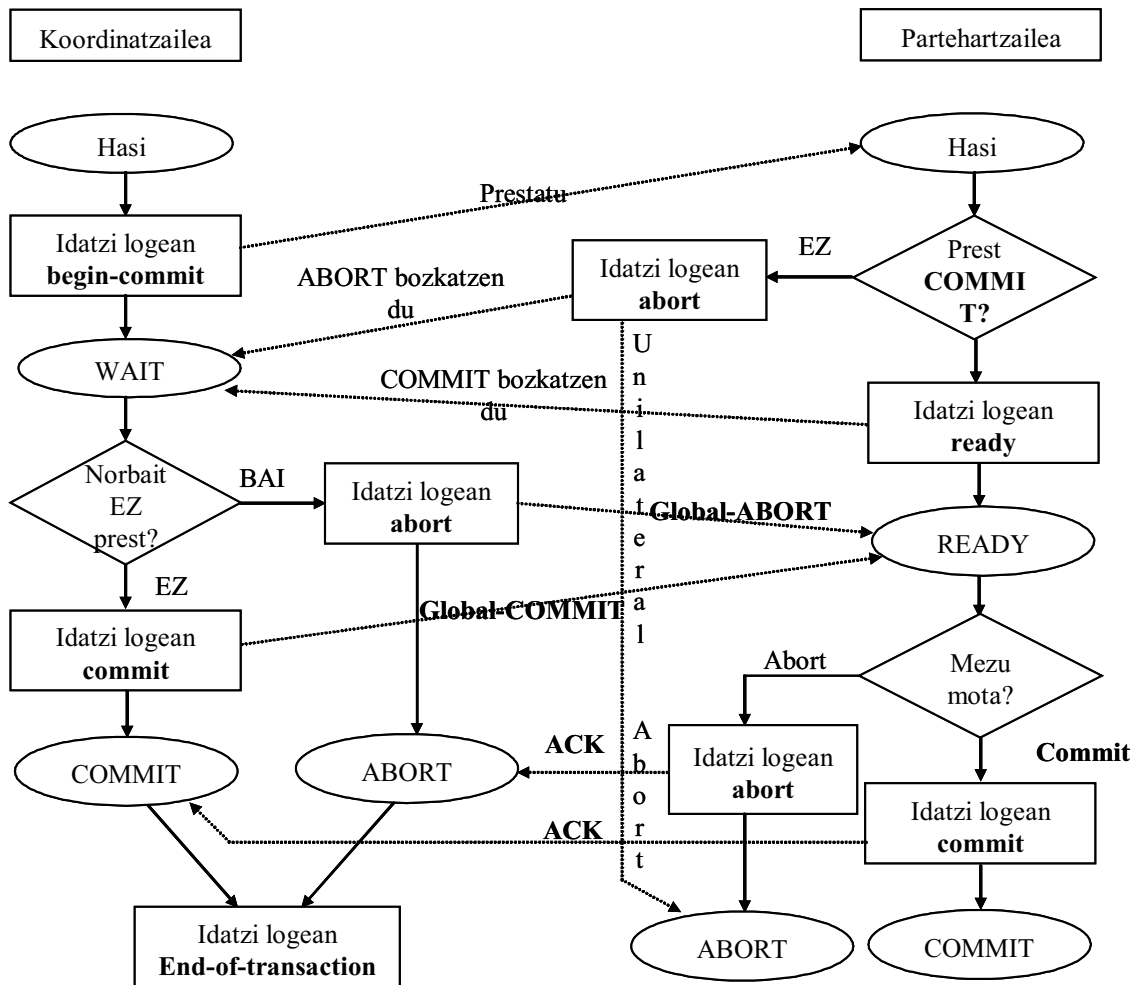


9.8 irudia. 2PC batean DBKsek jarraitzen dituzten pausoak.

K-k koordinatutako T transakzioaren prozesaketan L lekuak huts egiten badu, berreskurapena Lren bitakorako (log fitxategia) informazioaren menpe egongo da. Bitakorak COMMIT, ABORT edo READY bat izango du. K-k leku guztien emaitzak jasoko ditu eta ondorioak ateratu: adibidez, leku batek ABORT bat bidali badu, transakzio guztia ezeztatuko da; beraz, beste kokapen guztiei (beste partaideei) ezeztatzeke esango zaie, ROLLBACK/ABORT egiteko esanez (ikusi 9.9 irudia).

K koordinatzaileak T transakzioaren 2PC protokoloa egikaritzerakoan huts egiten badu (*timeout* moduan detektatuko da), transakzioaren nondik norakoa (egoera) zenbait L lekuk erabaki beharko dute. Batzuetan, koordinatzailearen hutsegitearen ondorioz, beste lekuek ez dute informazio nahikorik izango transakzioarekin jarraitu ala ez jakiteko, koordinatzailea berreskuratu arte. Berreskurapena asko atzeratzen bada, DBBKsaren blokeo orokorra ekar dezake.

K koordinatzaileak COMMIT edo ABORT egoeretan *timeout* bat detektatzen badu, ez du jakingo nodo guztietan COMMIT edo ABORTen prozedurak bukatu dituzten. Erantzuten ez badute, koordinatzaileak COMMIT edo ABORT mezuak behin eta berriro bidali beharko dizkie leku parte-hartzaileei, eta euron erantzunaren zain egongo da. Komunikazio-sarea jausten bada edo beste edozein arrazoiengatik mezuak galtzen direnean gerta daiteke.



9.9 irudia. 2PC protokoloan ematen diren pausoak.

2PC protokoloak gehiegizko lana dakar sistema erdi eta oso banatuetan, eta itxaron-denbora luzeak eragiten ditu. Aldakuntza berriak ekarri ditu egoera horrek eta beste metodo batzuk agertu dira. Aldakuntza horietariko batzuk:

- Hutsegite ustea (PrA, *Presumed Abort*): huts egiten duten transakzioen kostua murrizteko diseinatu da. Koordinatzaileak ez du gordetzen huts egindako transakzioari buruzko informaziorik, ezta parte hartu dutenen erantzun-mezurik itxaroten ere (baieztatutako transakzioak bai gordetzen dira bitakoran). Ondorioz, bestelako baieztapenik ezean, hutsegitetzat jotzen da.

- Baieztapen ustea (PrC, *Presumed Commit*): teknika baikorra da: transakzio guztiak onera helduko direla jotzen du. Aurrekoaren berdina, baina hutsegiteen lekuan, transakzio baieztatuak kokatuz.

9.6.2. Hiru faseko konpromisoa (*three-phase commit, 3PC*)

2PCren desabantaila nagusia blokeoaren protokoloa da. Koordinatzaileak huts egiten badu, bali-teke koordinatzailea berreskuratu arte parte hartzen duten beste lekuak itxarote-ziklo batean sartzea. Luzatu egin daiteke eta bitartean baliabideak blokeatuta daude transakzioa bukatu edo desegin arte.

2PCaren diseinu alternatiboa egiteko orduan, honako helburu hauek ezarri ziren:

- Huts egiten duten lekuek ezin dute mantendu errekurtsorik blokeatuta.
- Huts egiten duten leku guztiek, behin bere onera bueltaturik, transakzioarekiko emaitza bera lortu behar dute.
- Huts egiten ez duten leku guztiek transakzioarekin bat etorri behar dute onerako (*global commit*) edo txarrerako (*global abort*).

Honela funtzionatzen da: Izan bedi T transakzioa, L lekuan hasita eta K koordinatzaile-lanak egiten dituen; horrek leku guztietatik Tren eragiketa guztiak bukatu direla esanez mezuak hartzen ditue-nean eta biltegitratze egonkorrean baieztatu eta grabatuta daudenean, hots, baieztapen orokorraren faltan baino ez dagoenean, orduan 3PC protokoloa abiarazten da.

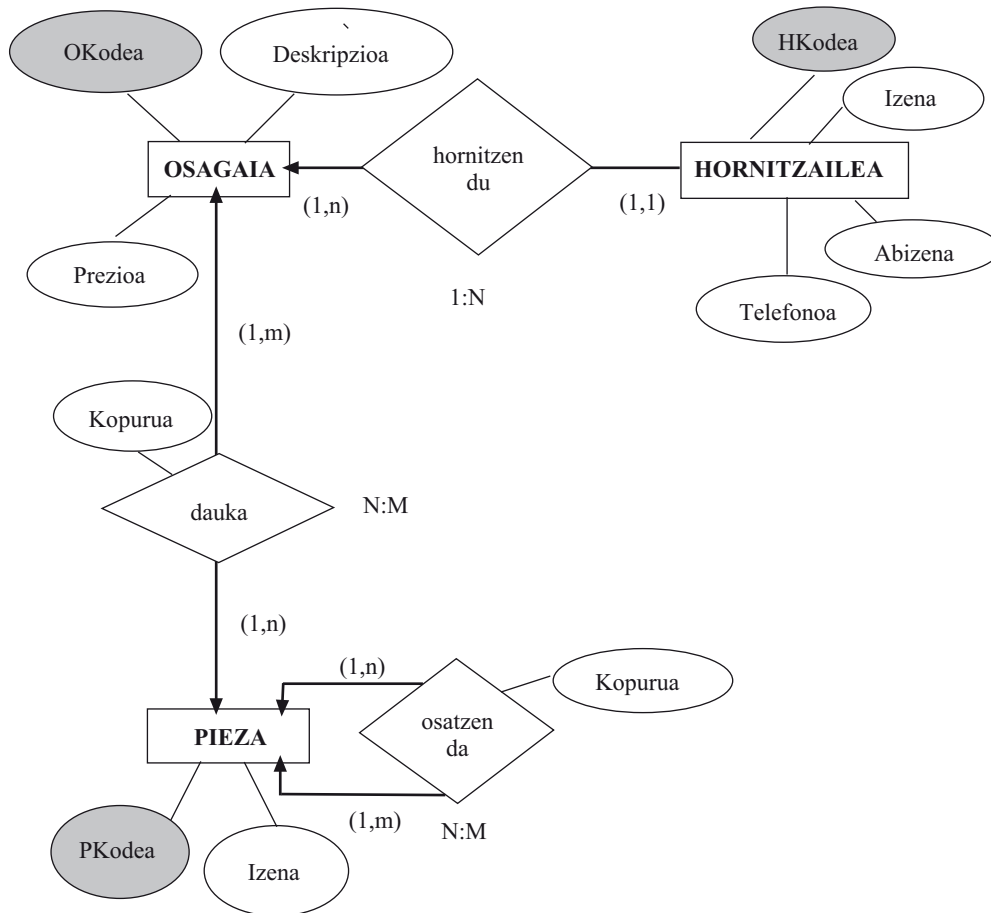
Hutsegite batetik berreskuratzean edo *timeout* bat detektatzerakoan, edozein L lekuk, bere bitakora aztertu beharko du hutsegitea izan zenean egikaritzen ari ziren transakzio guztien nondik norakoa zehazteko.

Leku parte-hartzaile batek koordinatzailearen falta detektatzen badu (*timeout* baten bitartez), hutsegiteak koordinatzeko protokoloa abiaraziko du. Protokolo horrek koordinatzaile berri baten hautaketa eragingo du. Huts egin zuen koordinatzailea berreskuratzen denean, parte-hartzaile soilaren maila baino ez du hartuko, eta ez da koordinatzailea izango.

ARIKETAK

1. ARIKETA

Ondoko E/R ereduari dagokion eredu erlazionala atera:



Eredu erlazionala

HORNITZAILEA (HKodea, Izena, Abizena, Telefonoa)

OSAGAIA (OKodea, Deskripzioa, Prezioa, **HKodea**)

DAUKA (OKodea, PKodea, Kopurua)

PIEZA (PKodea, Izena)

OSATZEN_DA(PKodea, PKodea2, Kopurua)

2. ARIKETA

Ospitale bateko datu-basea diseinatu nahi dugu, bertako erizainak, gaixoak eta sendagileak kudeatzeko, besteak beste.

Ospitaleak solairu-kopuru jakina dauka eta solairu bakoitzari gaixotasun zehatz bat dagokio. Erizain batek solairu askotan lan egin ahal izango du, eta solairu bakoitzean betebeharrak izango ditu. Betebeharrak hori kode eta deskripzio baten bitartez islatzen dira; solairua, ordea, solairuaren kodearekin eta gaixotasunaren izenarekin. Gainera, solairu berean erizain askok lan egiten dute.

Solairu bakoitzean gelak daude, bakoitza bere zenbakiarekin. Zenbaki hori bakarra izango da ospitalean.

Beste alde batetik, sendagile batek gaixo asko bisita ditzake eta gaixo bakoitza sendagile askok bisitatu izan daiteke. Bisita bakoitzeko data gordetzea interesatzen zaigu, gaixo bakoitzak egunean bisita bakarra izan baitezake. Sendagileei buruz, beraien izena, abizenak, espezialitatea, gizarte-segurantzako zenbakia, bizi diren herriaren izena eta kodea (herri bik izen bera izan baitezakete) eta, azkenik, bizi diren probintziaren izena eta kodea gordetzea interesatzen zaigu.

Gaixoari dagokionez, izen-abizenak, NA, bizi den herriaren izena eta kodea, eta bizi den probintziaren izena eta kodea gordetzen da. Aipatu beharra dago, gaixo bat ingresatuta dagoen aldi osoan, beti logela berean egon ohi dela; halere, horrek ez du esan nahi ingresatzen den bakoitzean logela horretan egon behar duenik. Hori dela eta, gaixo baten ospitalizazio bakoitzeko ingresuko data, altako data eta arrazoia gordetzea interesatzen da.

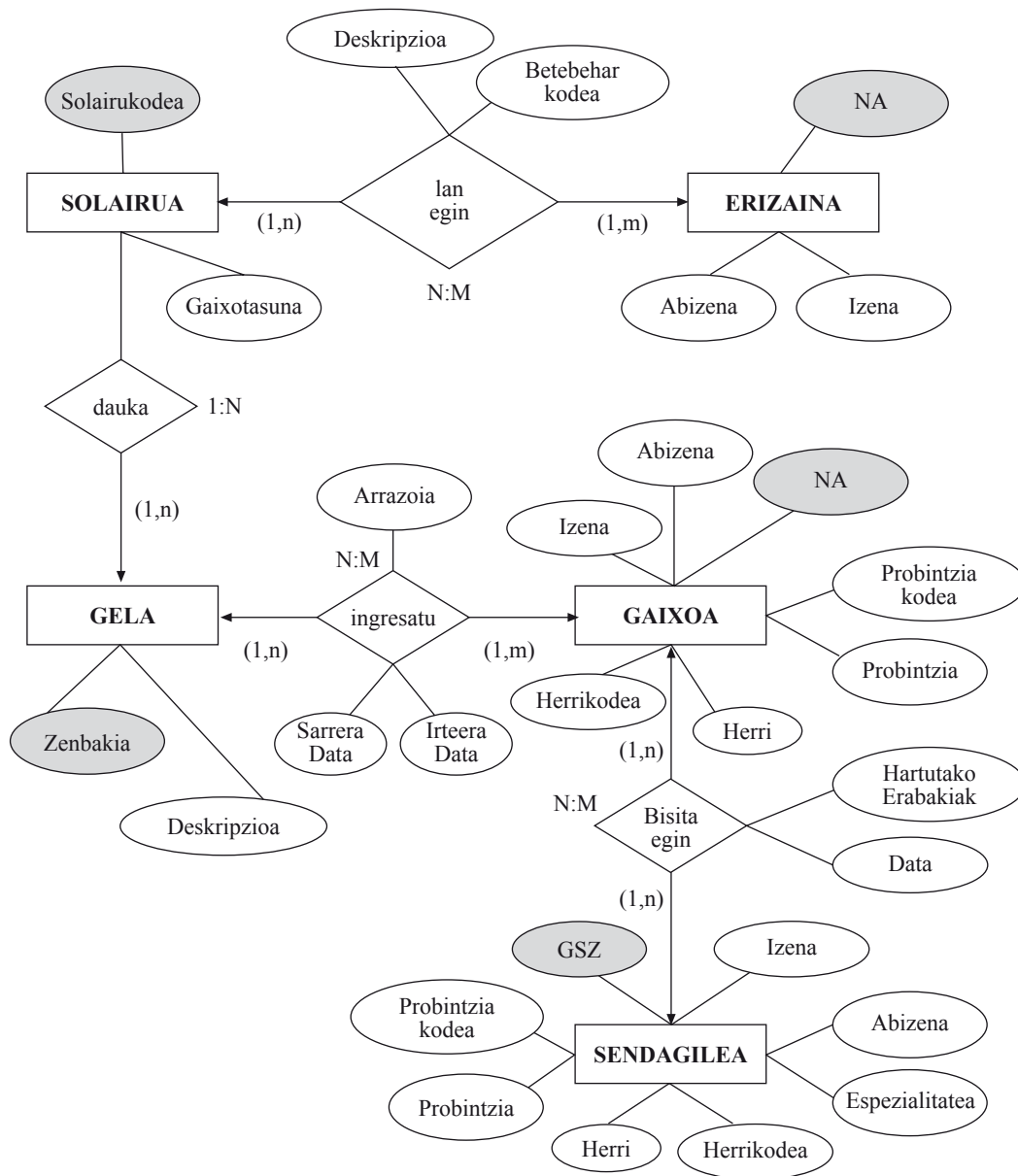
Datu-base horren gainean gehien egingo diren kontsulten artean, honako hauek daude, besteak beste:

- Gaixo baten NA zenbakia izanda, horrek izandako ingresu guztien sarrera-eguna, alta-eguna eta arrazoia.
- Sendagile baten izena eta abizenak izanda, duela 15 egun bisitatutako gaixoen zerrenda eta bakoitzaren ingresuaren arrazoia.
- Gaixotasun baten deskripzioa izanda, gaixotasun horretaz arduratzen diren solairuetan lan egiten duten erizainen izen-abizenak eta solairu horretan betetzen duten betebeharraren deskripzioa.
- Gelaren zenbakia izanda, horri dagokion solairua.
- Gaixo baten izen-abizenak edo NA izanda, ingresatuta dagoen gela.

Datu-base horrentzat, lortu:

- E/R eredu.
- Eredu erlazionala.
- 3. forma normaleraino normalizatutako eredu erlazionala.
- Agirre Ortundua sendagileak, 2006ko urtarrilaren 3an bisitatutako gaixoak.

E/R eredu



Eredu erlazionala

SOLAIRUA (SolairuKodea, Gaixotasuna)

ERIZAINA (NA, Izena, Abizena)

LANEGIN (SolairuKodea, NA, BetebeharKodea, Deskripzioa)

GELA (Zenbakia, **SolairuKodea**, Deskripzioa)

INGRESATU (Zenbakia, NA, SarreraData, Arrazoa, IrteeraData)

BISITAEGIN (NA, GSZ, Data, HartutakoErabakia)

SENDAGILEA (GSZ, Izena, Abizena, Espezialitatea, Herria, HerriKodea, Probintzia, ProbintziaKodea)

GAIXOA (NA, Izena, Abizena, Herria, HerriKodea, Probintzia, ProbintziaKodea)

Eredu erlazioanala

Normalizazio-prozesua

1FNa: multzorik ez da ageri; beraz, erlazio guztiak daude lehenengo forma normalean.

2FNa: mendekotasun funtzional partzialak ezabatzen dira. Adibideko mendekotasun funtzional partzialak honako hauek dira:

LANEGIN erlazioan:

BetebeharKodea → Deskripzioa

‘Deskripzioa’ atributua bakarrik ‘BetebeharKodea’ atributuaren mendekoa da eta ez gakoa osatzen duten atributu guztien mendekoa. Hori dela eta, gakoarekiko mendekotasun funtzional partziala dago eta egoera hori saihesteko:

LANEGIN (SolairuKodea, NA, BetebeharKodea)

BETEBEHARRA (BetebeharKodea, Deskripzioa)

Eredua 2FNean honela geratzen da:

SOLAIRUA (SolairuKodea, Gaixotasuna)

ERIZAINA (NA, Izena, Abizena)

LANEGIN (SolairuKodea, NA, BetebeharKodea)

BETEBEHARRA (BetebeharKodea, Deskripzioa)

GELA (Zenbakia, **SolairuKodea**, Deskripzioa)

INGRESATU (Zenbakia, NA, SarreraData, Arrazoa, IrteeraData)

BISITAEGIN (NA, GSZ, Data, HartutakoErabakia)

SENDAGILEA (GSZ, Izena, Abizena, Espezialitatea, Herria, HerriKodea, Probintzia, ProbintziaKodea)

GAIXOA (NA, Izena, Abizena, Herria, HerriKodea, Probintzia, ProbintziaKodea)

3FNa: mendekotasun iragankorrak ezabatuko dira. Adibideko mendekotasun iragankorrak honako hauek dira:

GAIXOA eta SENDAGILEA erlazioetan:

GSZ → HerriKodea → Herria.

GSZ → ProbintziaKodea → Probintzia

NA → HerriKodea → Herria

NA → ProbintziaKodea → Probintzia

‘Herria’ atributua atributu iragankorra da eta ‘HerriKodea’ atributu nagusia. Era berean, ‘Probintzia’ atributu iragankorra da eta ‘ProbintziaKodea’ atributu nagusia. Hori guztia dela eta, mendekotasun iragankorrek daude, eta egoera hori saihesteko:

GAIXOA (NA, Izena, Abizena, HerriKodea, ProbintziaKodea)

SENDAGILEA (GSZ, Izena, Abizena, Espezialitatea, HerriKodea, ProbintziaKodea)

HERRIA (HerriKodea, Herria)

PROBINTZIA (ProbintziaKodea, Probintzia)

Azkenik, ereduak 3FNean horrela gelditzen da:

SOLAIRUA (SolairuKodea, Gaixotasuna)

ERIZAINA (NA, Izena, Abizena)

LANEGIN (SolairuKodea, NA, BetebeharKodea)

BETEBEHARRA (BetebeharKodea, Deskripzioa)

GELA (Zenbakia, **SolairuKodea**, Deskripzioa)

INGRESATU (Zenbakia, NA, SarreraData, Arrazoia, IrteeraData)

BISITAEGIN (NA, GSZ, Data, HartutakoErabakia)

GAIXOA (NA, Izena, Abizena, **HerriKodea**, **ProbintziaKodea**)

SENDAGILEA (GSZ, Izena, Abizena, Espezialitatea, **HerriKodea**, **ProbintziaKodea**)

HERRIA (HerriKodea, Herria)

PROBINTZIA (ProbintziaKodea, Probintzia)

SQL sententzia

Luis Ortundua sendagileak 2006ko urtarrilaren 3an bisitatutako gaixoak

```
SELECT * FROM Gaixoa, BisitaEgin, Sendagilea
```

```
WHERE BisitaEgin.Data='2006/01/03'
```

```
AND Sendagilea.Izena='Luis'
```

```
AND Sendagilea.Abizena='Ortundua'
```

```
AND Gaixoa.NA=BisitaEgin.NA
```

```
AND BisitaEgin.GSZ=Sendagilea.GSZ;
```

3. ARIKETA

Jatetxe bateko jatorduak eta sukaldariak kudeatzeko sortu nahi den datu-base baten diseinurako, honako informazio hau eman digute:

- Plater bakoitzeko, kodea, izena eta deskripzioa gordetzen dira datu-basean.
- Jatordu bakoitza plater ugariz osatua dago, eta jatordu bakoitzeko eguna, ospakizuna eta prezioa gordetzen dira. Egun bakoitzeko, jatordu bakarra ematen da jatetxean.
- Plater bakoitzak, jatorduko, funtzio bat izango du. Plater bera jatordu batean sarreran joan daiteke eta beste batean, bigarreanean.
- Plater bakoitzak osagai ugari izango ditu. Osagai bakoitzeko, kodea eta izena gordeko dira. Plater bakoitzari dagokionez, osagai bakoitzeko beharrezkoa den kantitatea eta erabilera (txikitua, birrindua...) dira interesgarriak. Plater batek osagai bera erabilera-mota askotan izatea posible da. Adibidez, patata frijitua eta patata-purea.
- Jatetxean sukaldari ugarik egiten du lan. Bakoitzeko, kodea, izena, abizena, jatorrizko herrialdea eta herrialdearen kodea gordeko dira.
- Sukaldari batek plater asko presta ditzake; era berean, plater bera hainbat sukaldarik presta dezakete. Hori horrela izanik, plater bakoitza zer sukaldarik eta zer egunetan prestatu duen gorde nahi da.

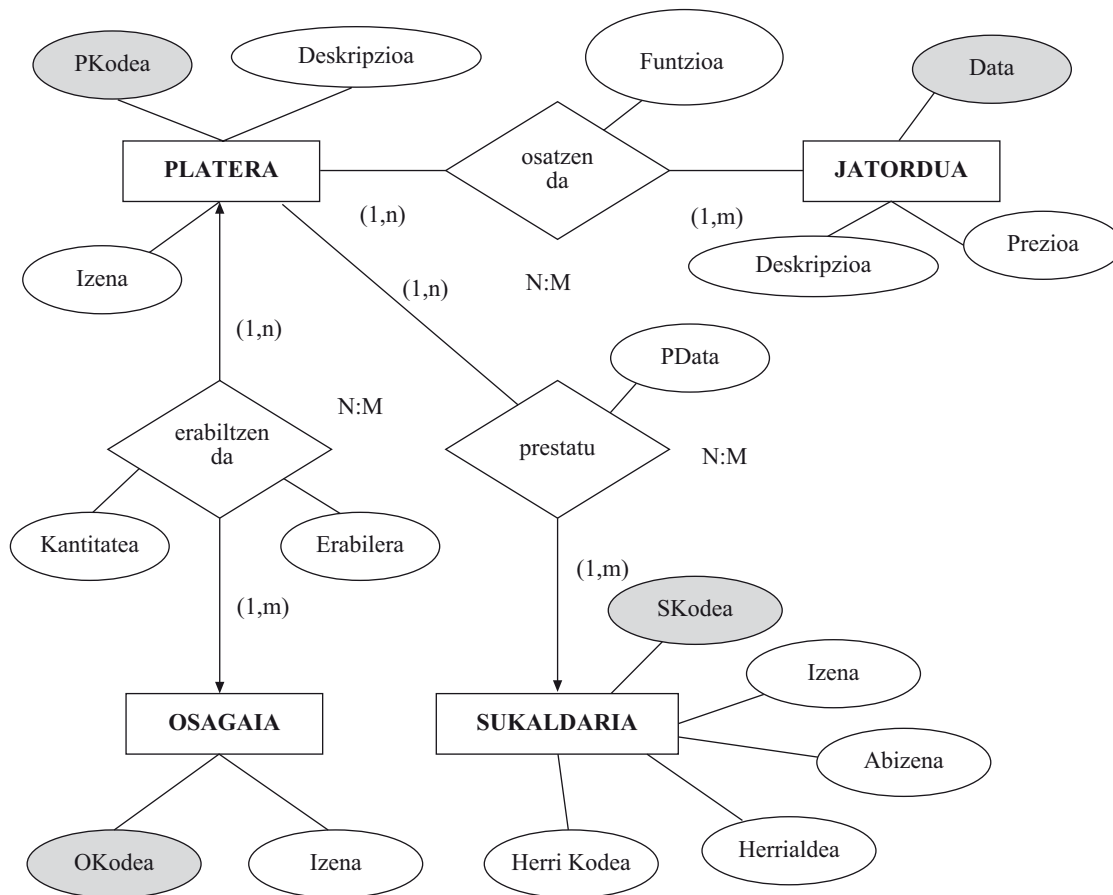
Datu-base horretan gehien egingo diren kontsulten artean, honako hauek daude:

- Egun jakin batean, jatetxe horretan emandako jatorduaren ospakizuna eta zerbitzatutako plateren deskripzioa atera, plater bakoitzak betetzen duen funtzioa adieraziz.
- Herrialde baten kodea adieraziz, bertako sukaldarien izena eta kodea bistaratu.
- Plater baten osagaiak, bakoitzaren kantitatea eta erabilera-mota bistaratu.

Datu-base horrentzat, diseinatu:

- E/R eredu
- Eredu erlazionala
- 3. forma normaleraino normalizatutako eredu erlazionala

E/R eredu:



Eredu erlazionala

JATORDUA (Data, Prezioa, Deskripzioa)

OSAtzENDA (Data, PKodea, Funtzioa)

PLATERA (PKodea, Izena, Deskripzioa)

ERABILTZENDA (PKodea, OKodea, Erabilera, Kantitatea)

OSAGAIA (OKodea, Izena)

SUKALDARIA (SKodea, Izena, Abizena, HerrialdeKodea, Herrialdea)

PRESTATU (PKodea, SKodea, PData)

Normalizazio-prozesua

1FN_a: Multzorik ez da ageri; beraz, erlazio guztiak daude lehenengo forma normalean.

2FN_a: Mendekotasun funtzional partzialik ez da ageri; beraz, erlazio guztiak daude bigarren forma normalean.

3FN_a: Mendekotasun iragankorrek ezabatzen dira. Adibideko mendekotasun iragankorra honako hau da:

SUKALDARIA erlazioan:

Skodea → HerrialdeKodea → Herrialdea.

‘Herrialdea’ atributua iragankorra da, eta ‘HerrialdeKodea,’ atributu nagusia. Egoera hori saihesteko:

SUKALDARIA (SKodea, Izena, Abizena, HerrialdeKodea)

HERRIALDEA (HerrialdeKodea, Herrialdea)

Azkenik, eredia 3FNean honela geratuko da:

JATORDUA (Data, Prezioa, Deskripzioa)

OSATZENDA (Data, PKodea, Funtzioa)

PLATERA (PKodea, Izena, Deskripzioa)

ERABILTZENDA (PKodea, OKodea, Erabilera, Kantitatea)

OSAGAIA (OKodea, Izena)

SUKALDARIA (SKodea, Izena, Abizena, HerrialdeKodea)

HERRIALDEA (HerrialdeKodea, Herrialdea)

PRESTATU (PKodea, SKodea, PData)

4. ARIKETA

Ondoren agertzen den erlazioa 3FNean jarri:

FILMA (Izenburua, Urtea, Aktorea, Paperak, Zuzendaria, Nazionalitatea)

Goian aipatutako erlazioa islatzeko, hona hemen A.1.1 taula:

Izenburua	Urtea	Aktorea	Paperak	Zuzendaria	Nazionalitatea
<i>Banketxeko Lapurreta</i>	2001	<i>Arantza Gartzia</i>	<i>Lapurra, amona</i>	<i>Txus Ormaetxea</i>	<i>Espainiarra</i>
<i>Banketxeko Lapurreta</i>	2001	<i>Ismael Loma</i>	<i>Bankuko langilea, polizia</i>	<i>Txus Ormaetxea</i>	<i>Espainiarra</i>
<i>Banketxeko Lapurreta</i>	2001	<i>Luis Urkidi</i>	<i>Atezaina</i>	<i>Txus Ormaetxea</i>	<i>Espainiarra</i>
<i>Amaren historia</i>	2006	<i>Agurtzane Mendikute</i>	<i>Ama</i>	<i>Amaia Agirre</i>	<i>Frantsesa</i>
<i>Amaren historia</i>	2006	<i>Ane Jaio</i>	<i>Alaba, irakaslea</i>	<i>Amaia Agirre</i>	<i>Frantsesa</i>
<i>Amaren historia</i>	2006	<i>Jon Ugarte</i>	<i>Senarra, sendagilea</i>	<i>Amaia Agirre</i>	<i>Frantsesa</i>

A.1.1 taula. FILMA erlazioa

Normalizazio-prozesua

1FN_a: Multzoak ageri dira ‘Paperak’ atributuan, esan bezala, aktore batek paper asko izan baititzaie film berean. Film bakoitzeko, aktoreak eta euron paperak bilduko dituen erlazioa sortuko da multzoak saihesteko. Datu-basea 1FNean horrela geratuko litzateke:

FILMA(Izenburua, Urtea, Zuzendaria, Nazionalitatea)

PAPERA(Izenburua, Aktorea, Papera)

2FN_a: Mendekotasun funtzional partzialik ez da ageri, atributu guztiak gako nagusiaren mendeko baitira.

3FN_a: Mendekotasun iragankorrak ezabatzen dira. Adibideko FILMA erlazioan ageri den mendekotasun iragankorra honako hau da:

Izenburua → Zuzendaria → Nazionalitatea.

‘Nazionalitatea’ atributua atributu iragankorra da eta ‘Zuzendaria’, atributu nagusia. Egoera hori saihesteko, ZUZENDARIA deritzon erlazioa sortuko da:

FILMA (Izenburua, Urtea, Zuzendaria)

ZUZENDARIA (Zuzendaria, Nazionalitatea)

Datu-base horren eredu erlazionala, normalizazio-prozesua aplikatu ondoren, honako hau litzateke:

FILMA (Izenburua, Urtea, **Zuzendaria**)

ZUZENDARIA (Zuzendaria, Nazionalitatea)

PAPERA (Izenburua, Aktorea, Papera)

5. ARIKETA

Ondoren agertzen den erlazioa 3FNean jarri:

SALMENTAK (Merkatua, Kalea, PostaKodea, Herria, Postua, Jabea, NA, Fruta, Prezioa)

Goian aipatutako erlazioaren adibidea dugu A.1.2 taula:

Merkatua	Kalea	PostaKodea	Herria	Postua	Jabea	NA	Fruta	Prezioa
M1	Kale Handi	48256	Ondarroa	P1	Ane	78852635K	Laranja	1.5
M1	Kale Handi	48256	Ondarroa	P1	Ane	78852635K	Sagarra	1.3
M1	Kale Handi	48256	Ondarroa	P1	Ane	78852635K	Gerezia	1.2
M1	Kale Handi	48256	Ondarroa	P2	Gorka	78852635K	Gerezia	1.4
M1	Kale Handi	48256	Ondarroa	P3	Luis	25045632B	Gerezia	1.2
M2	Ostendi	48250	Abadiño	P12	Jon	12568632J	Platanoa	1.9
M3	Traña	48250	Abadiño	P15	Jon	12568632J	Sagarra	1.5

A.1.2 taula. MERKATUA erlazioa.

Normalizazio-prozesua

1FN_a: Multzorik ez da ageri; hau da, balore guztiak atomikoak dira. Beraz, erlazioa 1FNean dago.

2FN_a: Honako mendekotasun funtzional partzial hauek ageri dira:

Merkatua → Kalea

Merkatua → PostaKodea

Merkatua → Herria

Merkatua, Postua → Jabea

Merkatua, Postua → NA

Egoera hori saihesteko MERKATUA eta JABEA erlazioak sortuko dira:

SALMENTAK (Merkatua, Postua, Fruta, Prezioa)

MERKATUA (Merkatua, Kalea, PostaKodea, Herria)

JABEA (Merkatua, Postua, Jabea, NA)

3FN_a: Mendekotasun iragankorrak ezabatu behar dira. Adibideko MERKATUA eta JABEA erlazioetan ageri diren mendekotasun iragankorrak honako hauek dira:

Merkatua → PostaKodea → Herria

Merkatua, Postua → NA → Jabea

‘Herria’ atributua iragankorra da eta ‘PostaKodea’, atributu nagusia. Era berean, ‘Jabea’ iragankorra da eta ‘NA’, nagusia. Egoera hori saihesteko, HERRIA eta IZENA erlazioak sortuko dira:

MERKATUA (Merkatua, Kalea, PostaKodea)

HERRIA (PostaKodea, Herria)

JABEA (Merkatua, Postua, NA)

IZENA (NA, Jabea)

Datu-base honen eredu erlazionala, normalizazio-prozesua aplikatu ondoren, honako hau litzateke:

SALMENTAK (Merkatua, Postua, Fruta, Prezioa)

MERKATUA (Merkatua, Kalea, PostaKodea)

HERRIA (PostaKodea, Herria)

IZENA (NA, Jabea)

JABEA (Merkatua, Postua, NA)

6. ARIKETA

Ondoren agertzen den erlazioa 3FN_{ean} jarri:

MENUAK (MenuKodea, Platera1, Platera2, Postrea, Edaria, AdinTartea, Eskola, Jantokia, Kalea, PostaKodea, Herria, Probintzia, Data)

Goian aipatutako erlazioa islatzeko A.1.3 taula erakusten da:

MenuKodea	Platera1	Platera2	Postrea	Edaria	AdinTartea	Eskola	Jantokia	Kalea	PostaKodea	Herria	Probintzia	Data
M1	P1	P44	flana	ura	haurra gaztea	E1	J1 J2	Axpe	48500	H1	Bizkaia	06/03/15
M2	P23	P56	fruta	ura	haurra gaztea heldua	E1	J1 J2	Axpe	48500	H1	Bizkaia	06/03/16
M3	P1	P68	flana	esnea	gaztea	E2	J78	Antso	48200	H2	Bizkaia	06/03/15
M1	P1	P44	flana	ura	haurra gaztea	E11	J6	Urre	20125	H67	Gipuzkoa	06/03/15
M1	P1	P44	flana	ura	Haurra gaztea	E75	J1	Goikoa	01568	H89	Araba	06/05/22
M2	P23	P56	fruta	ura	haurra gaztea heldua	E1	J1 J2	Axpe	48500	H1	Bizkaia	06/05/26
M3	P1	P68	flana	esnea	gaztea	E2	J78	Antso	48200	H2	Bizkaia	06/06/22

A.1.3 taula. MENUAK erlazioa.

Normalizazio-prozesua:

1FNa: Multzoak ageri dira ‘AdinTartea’ eta ‘Jantokia’ atribuetan, menu bera adin askorentzat baliagarria izan baitaiteke eta eskola batek jantoki bat baino gehiago izan baititzake. Menuak, Adinak, Eskolak eta Jantokiak islatuko dituzten erlazioak sortuko dira multzoak saihesteko. Datu-basea horrela geratuko litzateke 1FNean:

MENUAK(MenuKodea, Platera1, Platera2, Postrea, Edaria)

ADINA (MenuKodea, AdinTartea)

ESKOLA ((MenuKodea, Eskola, Kalea, PostaKodea, Herria, Probintzia, Data)

JANTOKIA (Eskola, Jantokia)

2FNa: Honako mendekotasun funtzional partzial hauek ageri dira:

Eskola → Kalea

Eskola → PostaKodea

Eskola → Probintzia

Eskola → Herria

Egoera hori saihesteko MENUKODEA izeneko erlazioa sortuko da:

BANAKETA (MenuKodea, Eskola, Data)

ESKOLA (Eskola, Kalea, PostaKodea, Herria, Probintzia)

3FNa: Mendekotasun iragankorrak ezabatuko dira. Adibideko ESKOLA erlazioan ageri diren mendekotasun iragankorrak honako hauek dira:

Eskola → PostaKodea → Herria

Eskola → PostaKodea → Probintzia

Mendekotasun iragankor horiek saihesteko, HERRIA deritzon erlazioa sortuko da:

ESKOLA (Eskola, Kalea, PostaKodea)

HERRIA (PostaKodea, Herria, Probintzia)

Datu-base horren eredu erlazionala, normalizazio-prozesua aplikatu ondoren, honako hau litzateke:

MENUAK(MenuKodea, Platera1, Platera2, Postrea, Edaria)

ADINA (MenuKodea, AdinTartea)

JANTOKIA (Eskola, Jantokia)

BANAKETA (MenuKodea, Eskola, Data)

ESKOLA (Eskola, Kalea, PostaKodea)

HERRIA (PostaKodea, Herria, Probintzia)

7. ARIKETA

Jo dezagun datu-base baten eredu erlazionala dela honako hau:

LANGILEA (LangileKodea, Izena, Soldata, NagusiKodea, Betebeharra, SarreraData, Komisiao, **Zenbakia**)

SAILA (Zenbakia, Izena, Herria)

Ondoren aipatzen diren emaitzak lortzeko beharrezkoak diren SQL sententziak idatzi:

- a) 1.000 eta 2.000 euroren arteko soldata ez duten langileen izena eta soldata bistaritzen dituen sententzia.

```
SELECT Izena, Soldata
```

```
FROM langilea
```

```
WHERE Soldata NOT BETWEEN 1000 AND 2000;
```

- b) 10 edo 20 zenbakia duen sailean lan egiten duten langileen izena eta sailaren zenbakia bistaritzen dituen sententzia, emaitza izenez alfabetikoki ordenatuz.

```
SELECT Izena, Zenbakia
```

```
FROM langilea
```

```
WHERE Zenbakia IN (10, 20)
```

```
ORDER BY Izena;
```

- c) A hizkia izeneko hirugarren hizkitzat duten langile guztiak bistaratu.

```
SELECT Izena
FROM langilea
WHERE Izena LIKE ' _ _A%';
```

- d) Bi langile mota bistaratu: Lehenengo motakoak izenean bi L dituzten langileak eta 30 zenbakidun sailean lan egiten dutenak dira; bigarren motakoak, 37 kodea duen nagusiarentzat lan egiten dutenak.

```
SELECT Izena
FROM langilea
WHERE Izena LIKE '%L%L%'
AND Zenbakia=30
OR NagusiKodea=37;
```

- e) Argazkilariak eta editoreak bistaratu, baina soldata 1.000, 1.500 edo 3.500ekoa ez dutenak.

```
SELECT Izena, Betebeharra, Soldata
FROM langilea
WHERE Betebeharra IN ('Argazkilaria', 'Editorea')
AND Soldata NOT IN(1000, 1500, 3500);
```

- f) Komisiao irabazten duten langileen izena, betebeharra, sailaren zenbakia eta sailaren izena bistaratu.

```
SELECT L.Izena, L.Betebeharra, L.Zenbakia, S.Izena
FROM langilea L, saila S
WHERE L.Zenbakia=S.Zenbakia
AND L.Komisioa IS NOT NULL;
```

- g) Donostian lan egiten duten langileen izena, betebeharra, sailaren zenbakia eta sailaren izena bistaratu.

```
SELECT L.Izena, L.Betebeharra, L.Zenbakia, S.Izena
FROM langilea L, saila S
WHERE L.Zenbakia=S.Zenbakia
AND M.Herria='Donostia';
```

- h) Betebehar bera duten langileen kopurua bistaratu.

```
SELECT L.Betebeharra, count (*)
FROM langilea
```

GROUP BY Betebearra;

- i) Nagusi bakoitzeko, soldata txikiena duen langilea bistaratu. Soldatak 1.200 euro baino handiagoa izan beharko du eta ez du balioko nagusi gabeko langileen soldatarik. Emaizta, soldataren arabera ordenatu.

```
SELECT NagusiKodea, MIN (soldata)
```

```
FROM langilea
```

```
WHERE NagusiKodea IS NOT NULL
```

```
GROUP BY NagusiKodea
```

```
HAVING MIN (Soldata)>1200
```

```
ORDER BY MIN (Soldata) DESC;
```

- j) "T" hizkia abizenean duen langile baten sail berean lan egiten duten langileen kodea eta izena bistaratu.

```
SELECT LangileKodea, Izena
```

```
FROM langilea
```

```
WHERE Zenbakia IN (SELECT Zenbakia FROM langilea WHERE Izena LIKE '%T%')
```

- k) Komisiao duen langile baten soldata berdina eta sailkide diren langile guztien izenak, sailen zenbakiak eta soldatak bistaratu.

```
SELECT Izena, Zenbakia, Soldata
```

```
FROM langilea
```

```
WHERE (Soldata, Zenbakia) IN (SELECT Soldata, Zenbakia FROM langilea WHERE Komisiao IS NOT NULL)
```

- l) Langilearen kodetzat 23, izena *Unai Oregi*, soldata 1.320, nagusi modura 78 kodea duen langilea, betebearra *Editorea*, sarrera-data 2006/03/23, komisiao 10, eta 20 sailean lan egiten duen langile baten datuak txertatu LANGILE erlazioan.

```
INSERT INTO langilea (LangileKodea, Izena, Soldata, NagusiKodea, Betebearra, SarreraData, Komisiao, Zenbakia)
```

```
VALUES (23, 'Unai Oregi', 1320, 78, 'Editorea', '23/03/2006', 10, 20)
```

- m) Langile kodetzat 45, izena *Miren Lizundia*, soldata 1.280, nagusitzat 82 kodea duen langilea, betebearra *Aholkularia*, sarrera-data 2006/02/22, komisiao 13, eta 30 sailean lan egiten duen langile baten datuak txertatu LANGILE erlazioan

```
INSERT INTO langilea
```

```
VALUES (45, 'Miren Lizundia', 1280, 82, 'Aholkularia', '22/02/2006', 13, 30)
```

- n) Aurreko langileari izena aldatu eta *Jon Ortuondo* ezarri.

UPDATE langilea

SET Izena='Jon Ortuondo'

WHERE LangileKodea=45;

o) Ezabatu aurreko langilea

DELETE

FROM langilea

WHERE LangileKodea=45;

p) LANGILEA eta SAILA erlazioak sortzeko DDL sententziak idatzi, A.1.3 eta A.1.4 tauletan agertzen den informazioaren arabera:

Atributua	Mota	Berezitasunak
<i>LangileKodea</i>	<i>Number(4)</i>	<i>Gako primarioa</i>
<i>Izena</i>	<i>Varchar2(20)</i>	
<i>Soldata</i>	<i>Number(10)</i>	
<i>NagusiKodea</i>	<i>Number(4)</i>	
<i>Betebeharra</i>	<i>Varchar2(10)</i>	
<i>SarreraData</i>	<i>Date</i>	<i>1990/1/1etik 2006/3/31ra bitartean egon behar du.</i>
<i>Komisioa</i>	<i>Number(4)</i>	
<i>Zenbakia</i>	<i>Number(3)</i>	<i>Gako atzerritarra</i>

A.1.3 taula. LANGILEA erlazioa.

Atributua	Mota	Berezitasunak
<i>Zenbakia</i>	<i>Number(3)</i>	<i>Gako primarioa</i>
<i>Izena</i>	<i>Varchar2(10)</i>	
<i>Herria</i>	<i>Varchar2(10)</i>	

A.1.4 taula. SAILA erlazioa

Lehenengo, SAILA erlazioa sortuko da, LANGILEA erlazioko 'Zenbakia' atributua SAILA erlazioko 'Zenbakia' atributuarekin erlazionatuta dago eta. Hau da, LANGILEA erlazioa SAILA erlazioarekin 'Zenbakia' atributuaren bitartez erlazionatzen da.

CREATE TABLE saila (

Zenbakia NUMBER(3),

Izena VARCHAR2(10),

Herria VARCHAR2(10),

PRIMARY_KEY Zenbakia);

CREATE TABLE langilea (

LangileKodea NUMBER(4),

Izena VARCHAR2(20),
 Soldata NUMBER(10),
 NagusiKodea NUMBER(4),
 Betebeharra VARCHAR2(10),
 SarreraData DATE,
 Komisiao NUMBER(4),
 Zenbakia NUMBER(3),
 CHECK (SarreraData BETWEEN '1990/1/1' AND '2006/03/31')
 PRIMARY_KEY LangileKodea,
 FOREIGN_KEY Zenbakia REFERENCES SAILA);

- q) SAILA taulan, 'Herria' atributuarentzat luzera 30 hizkikoa jarri

```

ALTER TABLE saila
MODIFY (Herria VARCHAR2 (30));
  
```

- r) SAILA taulan, 'ProbintziaKodea' atributua gehitu. Atributu hori NUMBER(3) motakoa izango da.

```

ALTER TABLE saila
ADD ProbintziaKodea NUMBER (3);
  
```

- s) Komisiao irabazten duten langile guztiekin ikuspegi bat sortu.

```

CREATE VIEW komisio-langile AS
(SELECT * FROM langilea
WHERE Komisiao IS NOT NULL);
  
```

- t) SAILA taula ezabatu

```
DROP TABLE saila;
```

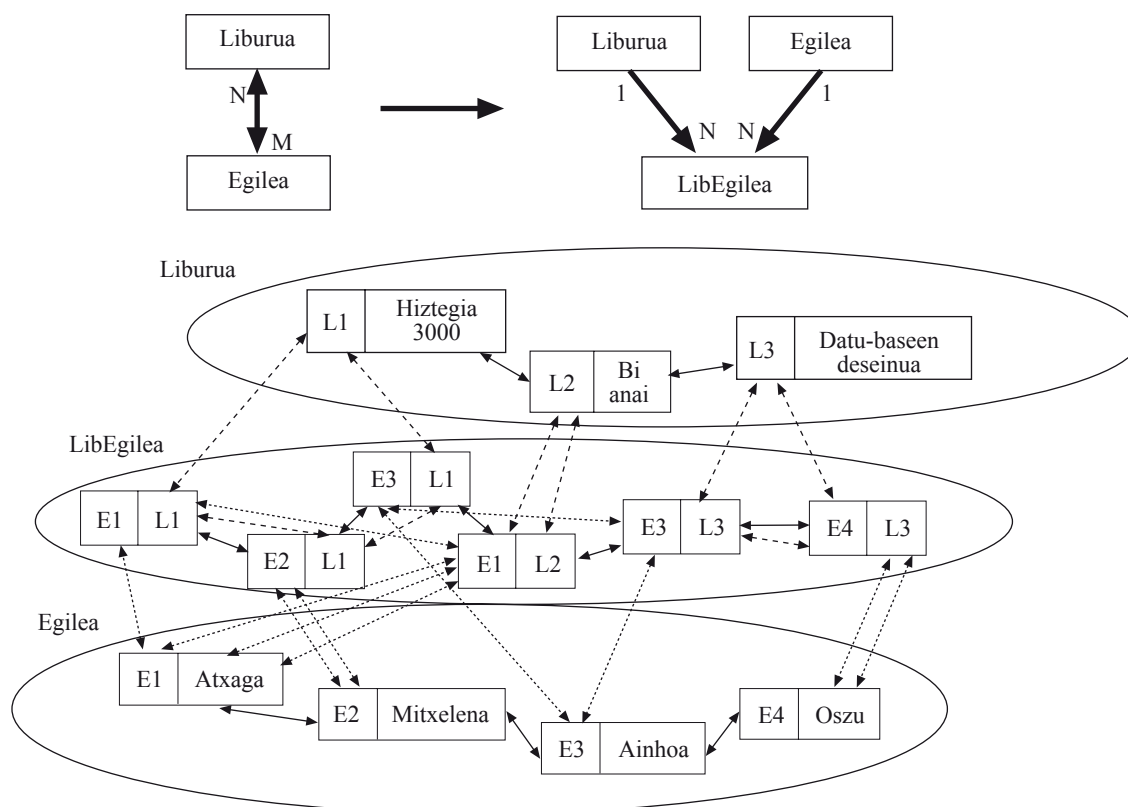
- Sare ereduako datu-baseak

Sare ereduako datu-basea esteken bidez lotutako hainbat motatako erregistro-multzo osatuta dago.

Erregistro bakoitza atributu-multzoa da. Atributuek domeinu bateko balio bakarra dute. Esteka batek bi erregistro elkarlotzen ditu. Erregistroek eta berauen arteko estekak sare itxura hartzen dute.

ERANSKINA

1. SARE EREDUKO DATU-BASEAK



E2.1 irudia. Sare ereduko datu-base baten egitura eta agerraldiak. Hiru liburu daude (*Hiztegia*, *Bi anai* eta *Datu-baseen diseinua*). Kolore urdina duten estekari jarraituz gero, *Hiztegia* idatzi dutenak izango ditugu. Kolore berdeko estekak *Bi anai* liburuiko egileak izango ditu, eta kolore gorriko estekak, *Datu-baseen diseinua* izeneko liburuak. Kolore ezberdinez baina puntukako marrekin autore bakoitzaren liburuak jarraitu ahal izango dira. Kolorezko baina etengabeko marrek taula ezberdinetako errenkadak lotzen dituzte: LIBURUA, LIB-EGILEA eta EGILEA.

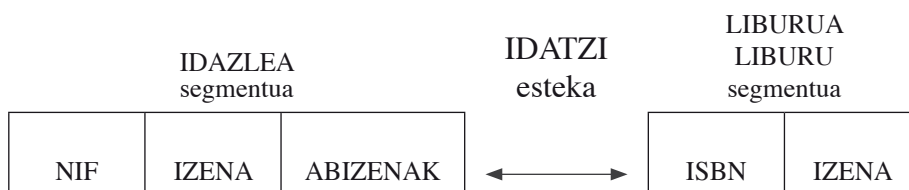
Hona hemen sare ereduko datu-baseen sistema komertzial batzuk: IDS/II, IDMS, DBMS-11, DMS1100.

E.2.1. Sare ereduko datu-base baten diseinua: datuen egitura-diagrama

Datu-base erlazioetan Entitate-Erlazio diagrama erabiltzen bada, sare ereduan, berriz, datuen egitura-diagrama erabiltzen da. Datuen egitura-diagrama datu-baseko diseinu kontzeptuala grafikoki adierazteko tresna da. Horretarako, honako osagai hauek erabiliko dira:

- Laukizuzenak: erregistro-multzoak adierazten ditu. Adibidez, IDAZLEA.
- Marrak: erregistro-multzoen arteko erlazioak adierazten ditu.

Datuen egitura-diagramak Entitate-Erlazio diagramaren helburu berbera du; hau da, sare-motako datu-baseko egitura logiko orokorra zehaztea. Ikusi E2.2 irudia.



E2.2 irudia. Sare ereduko egitura-diagrama baten adibidea

Ondoren aztertuko dugu kasuz kasu zelan geratzen diren Erlazio-Entitate diagramako erlazioak egitura-diagraman.

- Batetik batera (1:1) erlazioa:

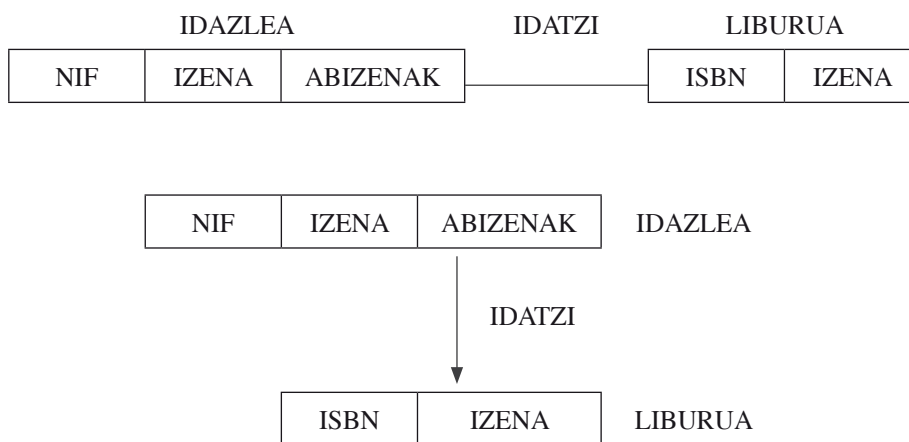
Ager daitekeen kasurik errazena. Marra soil batez adierazten da. Kasu horretan, idazle batek liburu bakarra idatz dezake. Erlazioaren beste aldetik ere, liburu baten idazlea bakarra da. Ikusi E2.3 irudia



E2.3 irudia. 1:1 erlazioaren egitura-diagrama.

- Batetik askotara (1:N):

Kasu horretan, idazle bakoitzak liburu bat baino gehiago idatz ditzake, baina liburu baten idazlea bakarra izango da. Marrak noranzko bakar batean dauka geziburua; hain zuzen, N erregistro dituen entitateari begira egon behar du geziburuak. Dagokigun kasuan, LIBURUA erregistro-multzoari.



E2.4 irudia. 1:N erlazio baten egitura-diagrama.

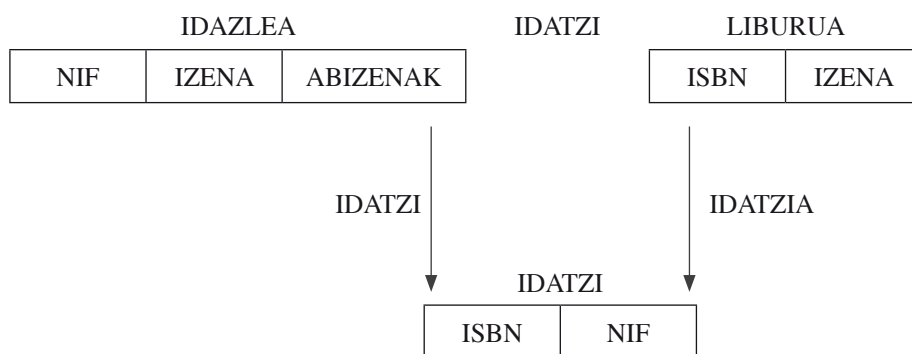
- Askotatik askotara (N:M):

Kasu horretan, idazle batek liburu bat baino gehiago izan ditzake eta gainera, liburu batek, zenbait egile. Marrak noranzko bietan dauka geziburua.



E2.5 irudia. N:M erlazio baten egitura-diagrama

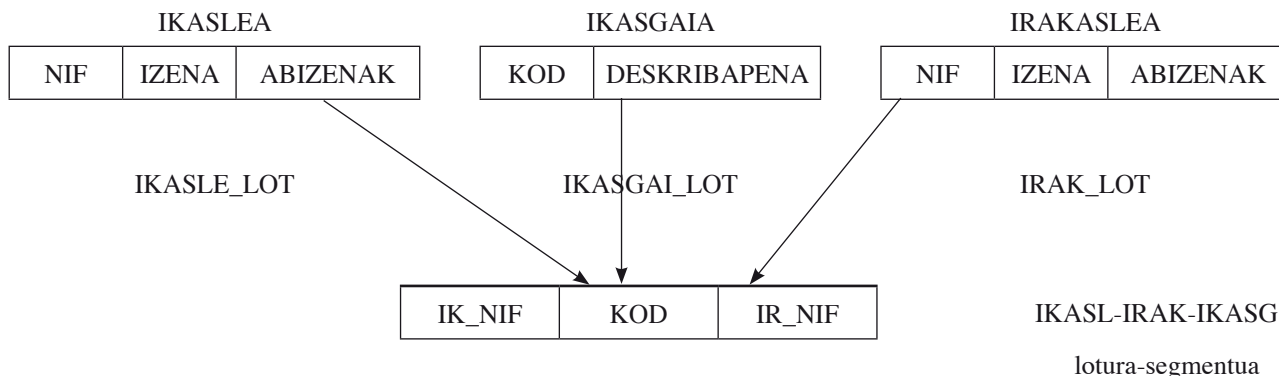
Erlazio hori islatzeko, lotura-erregistro bat erabiltzen da. 1:Nko bi erlazioren bidez gauzatzen da N:M erlazioa.



E2.6 irudia. N:M erlazioa egitura-diagramaren inplementazioa.

Hiru entitateren edo gehiagoren arteko erlazioa adierazteko, erregistro berria (lotura-erregistroa) sortuko da eta erregistro horri zuzenduko zaizkio erlazioak, 1:N edo kasuan kasuko kardinaltasunarekin.

Adibide moduan, IKASLEA-IRAKASLEA-IKASGAIA erlazioa erabiliko da. Kasu horretan, erregistro-multzoak binaka hartuz gero, beraien arteko erlazioak N:M izango liriteke; hots, ikasle bat ikasgai askotan egon daiteke matrikulatuta eta ikasgai batean ere, hainbat ikasle; irakasle batek ikasgai asko irakats ditzake eta ikasgai bat irakasle askok irakatsia izan; eta azkenik, ikasle batek irakasle batzuk izango ditu eta irakasle batek hainbat ikasle. Hori guztia 3 graduko erlazio bakar baten bidez isla daiteke, ondorengo irudi honetan erakusten den bezala.



E2.7 irudia. Bikoa baino gradu handiagoko erlazioen inplementazioa.

Horrela, lotura-erregistroko agerraldi bakoitzak zera jasoko du: ikasle bakoitzeko, matrikulatu izan den ikasgai bakoitzean zein irakaslek irakasten dion. Nahi izanez gero, erlazio horri 'Nota' atributua gehitu ahal zaio; lotura-erregistroan jasoko litzateke.

• **CODASYL eredia**

CODASYL txostena 1973. urtean sare ereduko inplementazio orokor eta praktiko gisa onartu zen. Hiru mailako ANSI/SPARC arkitektura jarraitzen du. ANSI/SPARCeke maila kontzeptuala, kanpoko eta barnekoa, CODASYL eredian, eskema, azpieskema eta barneko eskema deitzen dira.

Sareko eredu teorikotik honako murrizketa hauek ditu CODASYL ereduak:

- CODASYL ereduaren 1:N erlazioak baino ez dira onartzen. N:M erlazioak saihestu, eta 1:1 erlazioak 1:N motako erlazio bihurtu dira.
- N:M erlazioak lotura-erregistroen eta 1:N erlazioen bidez inplementatzen dira.

CODASYL ereduaren oinarritzko osagaiak hauexek dira:

- Erregistro mota (RECORD): kontzeptu edo objektu bera adierazten duen erregistroen agerraldi-multzoa. Adibidez: IDAZLEA erregistro-motan agerraldi hauek egongo dira: Atxaga, Delibes, Cervantes, García Márquez.
- Multzo mota (SET): Bi erregistro moten arteko erlazioari deritzo. Erregistro-moten arteko loturei **multzo** deitzen zaie. Multzo bat erregistro-jabe (aita) batez eta bat edo zenbait erregistro-kidez (kume) osatuta dago. Adibide modura, ikusi E2.4 irudia.
- Erregistroaren jabea (aita edo OWNER): 1:N erlazio bateko 1 aldeari dagokion erregistro-mota, beste erregistro-motako zeinagerraldirekin erlazionatuta dagoen esango digu. Adibidez: IDATZI multzoan, IDAZLEA erregistroaren jabeak zenbait liburu eduki ditzake.
- Erregistroaren kidea (kumea edo MEMBER): 1:N erlazio bateko N aldeari dagokion erregistro-mota, multzo bereko beste erregistro-motako zein agerraldi bakarrekin erlazionatuta dagoen esango digu. Adibidez: IDATZI multzoan LIBURUA erregistro-kidea idazle batekin dago lotuta soilik.

Erregistro mota batek multzo-motaren araberako zeregina izan dezake. Orduan, erregistro mota bera, jabea edo kide izan daiteke, zenbait multzotan. Adibidez, egunkari batean hainbat kazetarik idatz dezake eta, aldi berean, kazetari bakoitzak hainbat artikulu.



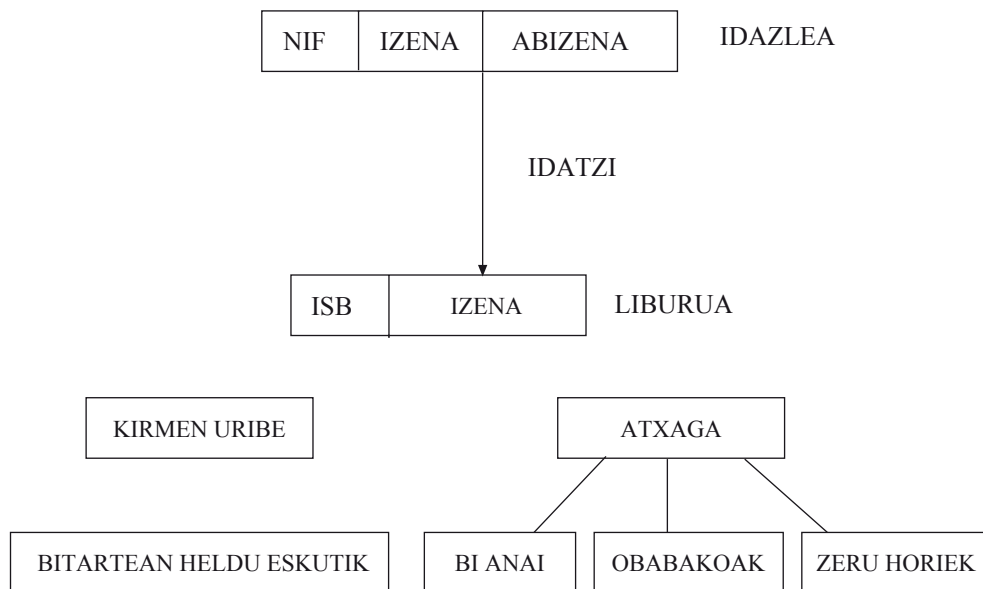
E2.8 irudia. KAZETARIA erregistroak, jabe zein kide papera betetzen duen adibidea.

Datu-baseen sistemak erregistro-kideak eta jabeak sortzeko eta berreskuratzeko aginduak hornitu behar ditu. Sintaxia honako honen antzekoa litzateke:

SET NAME Idatzi

OWNER IS Idazlea

MEMBER IS Liburua



E2.9 irudia. Multzo baten definizioa, egitura-diagrama eta agerraldi batzuen adibidea.

• Programaren lan-eremua

Datu-basea atzitzen duten aplikazioek datuak maneiatzeko eremu bat kudeatzen dute. Eremu horrek honako atal hauek ditu:

- Erregistro-txantiloiak: datu-baseak erregistro-mota bakoitzeko memorian gune bat mantentzen du, berorren bidez erabiltzaileek eta programek informazioa eskuragarri izateko.
- Uneko erakusleak: biltegitarte periferikoetako datuak maneiatzeko erakusleak dira. Euron artean daude, alde batetik, uneko erregistro-motako erakuslea (erregistro-mota bakoitzeko bat dago), uneko multzoko erakuslea (multzo bakoitzeko bat) eta uneko exekuzio-unitateko erakuslea (mota axola gabe, berriki atzitutako erregistroa seinatzen du). Eta beste alde batetik, DB_STATUS, azken eragiketaren emaitza adierazten duena (0 denean eragiketa ondo burutu dela adierazten du) eta DB_RECORD_NAME, atzitutako erregistro-motaren izena.

• Datuak Definitzeko Lengoaia (DDL)

CODASYL ereduko definizio-lengoaia ez da modu berean gauzatu aplikazio komertzialetan; hori dela eta, sintaxi orokorra erabiliko da ondorengo hauetan.

- Datu-basearen eskema (datuen egitura-diagrama) definitzeko agindua:

SCHEMA IS <eskema-izena>.

Behin baino ez da agertuko, eskemaren deskripzioaren hasieran.

- Erregistro mota definitzeko:

RECORD NAME IS <erregistro-motaren izena>

<eremuaren izena> TYPE IS <eremu-mota> <tamaina>

non datu motak TYPE IS bidez definitzen diren. Adibidez:

RECORD NAME IS Idazlea

NIF TYPE IS character (9)

Izena TYPE IS character (25)

Abizenak TYPE IS character (50)

- Multzoak zehazteko:

SET NAME IS <multzo-izena>

OWNER IS <erregistro-izena>

MEMBER IS <erregistro-izena>

Adibide baten bitartez errazago ikusten da. Hona hemen IDAZLEA-LIBURUA multzoa (erlazioa):



E2. 10 irudia. IDAZLEA-LIBURUA 1:N erlazioa.

SCHEMA IS NireLiburuak

RECORD NAME IS Idazlea

DUPLICATES ARE NOT ALLOWED FOR NIF /*eremuen gaineko baldintzak */

NIF TYPE IS character (9)

Izena TYPE IS character (25)

Abizenak TYPE IS character (25)

RECORD NAME IS Liburua

DUPLICATES ARE NOT ALLOWED FOR ISBN

ISBN TYPE IS character (15)

Izena TYPE IS character (50)

Urtea TYPE IS integer

SET NAME IS idatzi

/* idazle erregistroa lotura-erregistroarekiko multzoa) */

OWNER IS Idazlea

ORDER IS SORTED BY DEFINED KEYS

/* gako eremuekin ordenatzen da multzoa */

MEMBER IS Liburua

ORDER IS SORTED BY DEFINED KEYS

RETENTION IS MANDATORY

/* ezin daitezke egon erregistro kideak jaberik gabe */

• **Datuak Manipulatzeko Lengoia (DML)**

Datu-baseak kudeatzeko sistema erlazionaletan, erlazio osoa hartzen da prozesaketan. Halaber, sare ereduak kudeatzeko sistemetan, erregistroak banaka prozesatzen dira. Gainera, maneiatzeko lengoia hori beste lengoia batean murgilduta egon ohi da (COBOLen, gehienetan).

Hauek dira agindu batzuk:

- Nabigatu (FIND)
- Berreskuratu (GET)
- Txertatu (STORE)
- Ezabatu (ERASE)
- Aldatu erregistroa (MODIFY)
- Konektatu (CONNECT)
- Deskonektatu (DISCONNECT)
- Konektatu eta deskonektatu eragiketa berean (RECONNECT)

Beste eragiketa batzuk: Ireki edo itxi gunea (OPEN/CLOSE), datu-basean esteka gorde (KEEP), multzo bateko erregistroak ordenatu (ORDER) eta transakzioa burutu ala desegin (COMMIT/ROLLBACK).

Eragiketa batek, egoeran eta errore-adierazleetan ere aldaketak eragin ditzake. Adibidez, multzo baten amaierara heldu den jakin daiteke adierazleok erabiliz. Kasu horretan, DB_STATUS≠0 izango da amaierara heltzerakoan.

• **Datuak atzitzeko aginduak**

Oharra: Ondorengo adibide hauetan COBOLen txertatutako DML aginduak agertuko dira.

FIND eta GET:

Bi agindu horien bidez datuak berreskuratzen dira:

- FIND: erregistroa aurkitu eta erakusleak gaurkotzen ditu.
- GET: erakusleen bidez seinalatutako erregistroa dagokion uneko exekuzioko txantiloian kopiatzen du.

Lehenengo erregistroa aurkitu behar da (FIND), gero berreskuratzeko (GET).

FIND USING:

Erregistroen bidez atzipena lortzeko, FIND aginduak aldaera bi ditu:

```
FIND ANY <erregistro-mota> [ USING <eremu-zerrenda> ]
FIND DUPLICATE <erregistro-mota> [ USING <eremu-zerrenda> ]
```

1. adibidea:

```
MOVE "Edorta" TO Idazlea.Izena IN Idazlea
FIND ANY Idazlea USING Idazlea.Izena
GET Idazlea
PRINT (Idazlea.Izena, Idazlea.NIF, Idazlea.Abizenak)
```

Adibide horrekin *Edorta* izeneko lehenengo idazlearen datuak inprimatuko genituzke. Baina *Edorta* izeneko idazle guztiak nahi izanez gero, FIND DUPLICATE agindua da beharrezkoa.

2. adibidea:

```
MOVE "Edorta" TO idazlea.Izena IN Idazlea
FIND ANY Idazlea USING Idazlea.Izena
WHILE DB_STATUS = 0 DO
BEGIN
GET idazlea
PRINT (Idazlea.Izena, Idazlea.NIF, Idazlea.Abizenak)
FIND DUPLICATE Idazlea USING Idazlea.Izena
END
```

FIND DUPLICATE agindua begizta baten barruan sartu behar da, datu-base guztia zeharkatzeko. Ikuspegiaren hasieran, DB_STATUS adierazlea aztertu behar da erregistro gehiago dagoen jakiteko.

FIND FIRST eta FIND NEXT:

Multzo baten erregistro-kideak aurkitu nahi izanez gero, FIND agindua erabiliko da:

```
FIND FIRST <erregistro-mota> WITHIN <erregistro-multzoa>
```

Multzoko lehenengo erregistroa aurkitzen du. Multzoko hurrengo erregistroak aurkitzeko:

```
FIND NEXT <erregistro-mota> WITHIN <erregistro-multzoa>
```

3. adibidea: Idazlearen eta liburuaren erlazio honetan, idazle jakin baten liburuak bistaratuko dira. Kontuan izan, kasu honetan, liburu batek idazle bakarra duela.



E2. 11 irudia. IDAZLEA-LIBURUA 1:N erlazioa.

```

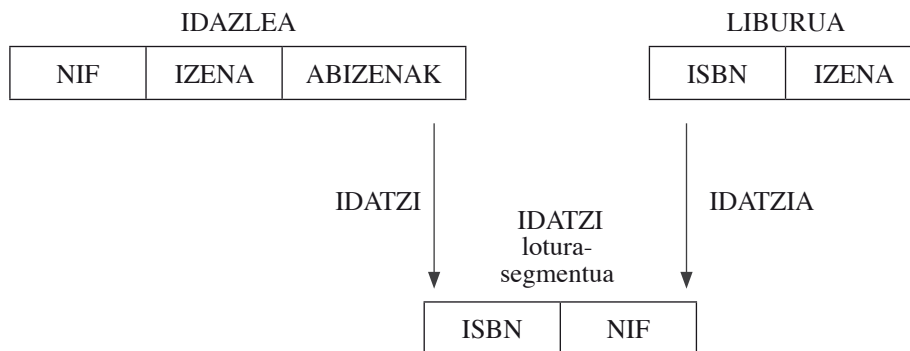
MOVE "Edorta" TO Idazlea.Izena IN Idazlea
FIND ANY Idazlea USING Idazlea.Izena
PRINT (Idazlea.Izena, Idazlea.Abizenak "idazleak ondorengo liburu hauek ditu")
FIND FIRST Liburua WITHIN Idatzi
WHILE DB_STATUS = 0 DO
BEGIN
GET Liburua
PRINT (Liburua.Izena)
        FIND NEXT Liburua WITHIN Idatzi
END

```

4. adibidea: idazlearen eta liburuaren erlazio honetan idazle jakin baten liburuak bistaratuko dira. Hemendik aurrera, adibideko erlazioa N:M bihurtu da. Kontuan izan, kasu honetan, liburu batek idazle asko izan dezakeela.



E2. 12 irudia. IDAZLEA-LIBURUA egitura diagramaren eskema



E2. 13 irudia. IDAZLEA-LIBURUA N:M erlazioaren inplementazioa.

```

MOVE "Edorta" TO Idazlea.izena IN Idazlea
/* Lehengo eta behin "Edorta" idazlearen erregistroa aurkituko da */
FIND ANY Idazlea USING Idazlea.Izena
PRINT (Idazlea.Izena, Idazlea.Abizenak "idazleak ondorengo liburu hauek ditu")
/* Edorta erregistroak "idatzi" multzoan dituen erregistro kideak aurkituko dira. Lotura-erregistroa "IDAZ_LIB" deitzen da*/

```

```

FIND FIRST Idaz_Lib WITHIN Idatzi
WHILE DB_STATUS = 0 DO
BEGIN
    /* idatzitako liburuak eskuratzeko lotura-erregistroaren “idatzia” multzoaren jabea berreskuratu behar da*/

FIND OWNER WITHIN Idatzia

GET Liburua

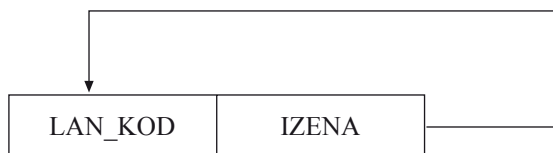
PRINT (Idazlea.Izena, Liburua.Izena)

FIND NEXT Idaz_Lib WITHIN Idatzi

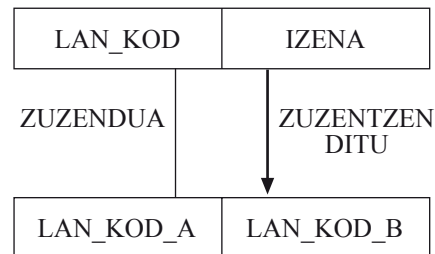
END

```

5. adibidea: langile eta langile-zuzendariaren erlazioa oinarritzat hartuta, *Josu* langilearen menpekoak bistaratuko dira.



E2.14 irudia. Entitate-erlazioko erlazio erreflexiboa.



E2.15 irudia. Erlazio erreflexiboaren implementazioa.

```

MOVE “JOSU” TO Langilea.Izena IN Langilea

/* Lehenengo eta behin “Josu” langilearen erregistroa aurkituko da */

FIND ANY Langilea USING Langilea.Izena

/* Orain erregistro horren “zuzentzen ditu” multzoan dauden erregistroak aurkituko dira. Lotura-erregistroa “LOT_LANGILEA” deitzen da.*/

FIND FIRST Lot_Langilea WITHIN ZuzentzenDitu

WHILE DB_STATUS = 0 DO

BEGIN

/* Menpeko langileak eskuratzeko lotura-erregistroaren “ZuzentzenDitu” multzoaren jabea berreskuratu behar da*/

FIND OWNER Langilea WITHIN Zuzendua

GET Langilea

PRINT (Langilea.Izena)

```


FIND NEXT Lot_Langile WITHIN ZuzentzenDitu

END

STORE:

STORE aginduak honako egitura hau du:

STORE <erregistro-mota>

- Adibidea

MOVE “Edorta” TO Idazlea.Izena IN Idazlea

MOVE “Penades Zapiain ” TO Idazlea.Abizenak IN Idazlea

MOVE “78.123.567” TO Idazlea.NIF IN Idazlea

STORE Idazlea

Ez da ahaztu behar, gero erregistroa multzo bati konektatu behar diogula.

MODIFY:

Lehenengo eta behin, erregistroa lan-eremura eraman behar da; hau da, FIND baten bidez aurkitu, gero GET berreskuratzeko. Azkenik MODIFY aginduaren bidez gaurkotuko da. *Oharra: FIND agin-
dua gaurkotzeko erabiliz gero, FOR UPDATE gehituko zaio adibideari.*

MODIFY aginduak honako egitura hau du:

MODIFY <erregistro-mota>

- Adibidea

MOVE “Edorta” TO Idazlea.Izena IN Idazlea

FIND FOR UPDATE ANY Idazlea USING Idazlea.Izena

GET Idazlea

MOVE “Urresti Ibarra” TO Idazlea.Abizenak IN Idazlea

MODIFY Idazlea

ERASE:

Kasu honetan, ere, FIND FOR UPDATE aginduaren bidez aurkitu behar da erregistroa, gero ezabatzeko.

ERASE aginduak honako egitura hau du:

ERASE <erregistro mota>

- Adibidea

MOVE “Edorta” TO Idazlea.Izena IN Idazlea

FIND FOR UPDATE ANY Idazlea USING Idazlea.Izena

GET Idazlea

ERASE Idazlea

Oharra: RETENTION IS MANDATORY erabili izan bada erlazioa definitzerakoan (eta horrela izan da kasu honetan), idazle horren liburu guztiak ere ezabatuko dira.

• Erregistroak multzoetara konektatzeko edo deskonektatzeko aginduak

CONNECT:

Ez da nahikoa erregistroa sortzearekin; dagokion multzoarekin konektatu behar da. Horretarako, CONNECT agindua dago, egitura honekin:

CONNECT <erregistro mota> TO <multzo-izena>

- Adibidea

Liburu berria sortu eta gero “Edorta” idazleari konektatu behar zaio. Kasu horretan, liburu batek idazle bakarra izan dezake; ondorioz, LIBURUA eta IDAZLEA erregistroak multzo bakarrarekin lotzen dira.

MOVE “Datu-baseak” TO Liburua.Izena IN Liburua

MOVE “12345678” TO Liburua.ISBN IN Liburua

STORE Liburua

MOVE “Edorta” TO Idazlea.Izena IN Idazlea

FIND FOR UPDATE ANY Idazlea USING Idazlea.Izena

CONNECT Liburua TO Idatzi

RECONNECT:

Erregistroa multzo batetik deskonektatu eta beste batera konektatu nahi izanez gero, RECONNECT agindua erabiliko da. Hauxe da dagokion egitura:

RECONNECT <erregistro-mota> TO <multzo-izena>

Demagun aurreko adibidean erratu egin garela eta *Datu-basea* liburuaren egilea *Josu* dela. Liburu hori ezabatu eta berria sortu eta konektatu ordeztu, liburua benetako egileari konektatuko diogu, aurrekoetik deskonektatuz. Beste modu batera esanda, liburua IDATZI multzoan beste jabe bati lotuko diogu.

MOVE “Datu-baseak” TO Liburua.Izena IN Liburua

FIND ANY Liburua USING Liburua.Izena

GET Liburua

MOVE "Josu" TO Idazlea.Izena IN Idazlea

FIND ANY Idazlea USING Idazlea.Izena

GET Idazlea

/* uneko liburu erregistroa idatzi uneko erregistroari lotuko zaio*/

RECONNECT Liburua TO Idatzi

Oharra: RECONNECT egiteko derrigorrezkoa da erregistroa konektaturik egotea. Hau da, ezin da egin DISCONNECT eta jarraian RECONNECT, bien artean CONNECT bat tartekatzen ez bada.

DISCONNECT:

Erregistroa multzo batetik deskonektatu nahi izanez gero, DISCONNECT agindua erabiliko da. Hauxe da egitura:

DISCONNECT <erregistro-mota> TO <multzo-izena>

Adibidea: Aurreko adibidearen kontrakoa egingo da. Hau da, lehenengo *Datu-baseak* liburuaren jabea aurkitu eta, ondoren, liburua eta idazlea lotzen duen IDATZI multzotik deskonektatu.

MOVE "Datu-baseak" TO Liburua.Izena IN Liburua

FIND FOR UPDATE ANY Liburua USING Liburua.Izena

GET Liburua

/* *Datu-baseak* liburuaren jabea aurkitu*/

FIND OWNER WITHIN Idatzi

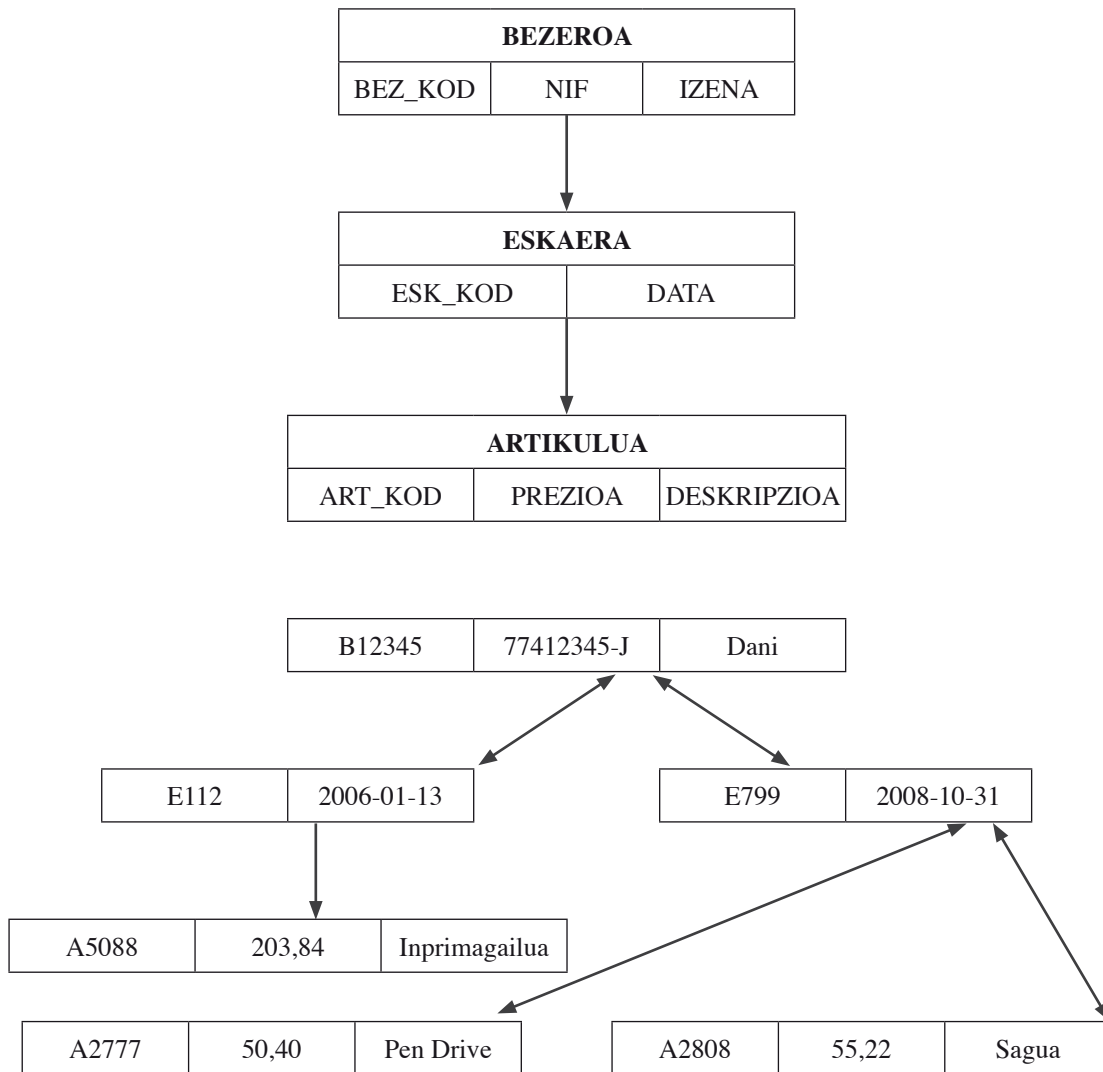
DISCONNECT Liburua TO Idatzi

Oharra: kasu honetan, erregistroa ezabatzeke geratzen da; hori beste prozedura bat izango litzateke.

2. EREDU HIERARKIKOA

Segmentuz (erregistroz) eta estekaz osaturiko datu-basea da. Sare ereduaren antzekoa da, baina, sare ereduaren ez bezala, estekak modu hierarkikoan lotzen dira. Horregatik, zuhaitz-eredu ere deitzen da, alderantzikatutako zuhaitzaren antzeko egitura eratzen delako.

Aipatu egin behar da eredu hierarkikoa arrazionalizazioaren eta estandarizazioaren ondorioa baino ez dela eta IBMk 70eko hamarkadaren hasieran erabilitako IMS base hierarkikoaren hedapenaren ondorioa. Horrela, gainerako datu-base hierarkikoentzat estandar bilakatu zen IMS, berorren egitura zein ezaugarriak oinarritzat hartu baitziren.



E3.1 irudia: Eredu hierarkikoa.

Eredu horren ezaugarrien artean honako hauek nabarmentzen dira:

- Datu-basea zuhaitz-multzo batez antolatzen da.
- Datu-base fisikoko segmentu motaren (DBST) agerraldi bat hainbat segmentuz osatzen da. Segmentuek luzera finkoa dute. E3.1 irudian ikus daiteke hori: beheko zuhaitzak DBST agerraldi bat islatzen du (Daniri dagokiona). Bertan hainbat segmentu daude. Adibidez, A2808 - 55,22 – Sagua artikulu bati dagokion segmentua litzateke.
- Segmentu bakoitzak atributu diren eremuak dauzka.
- Hierarkian edo zuhaitz-egituran goren dagoen segmentuari **sustrai** deitzen zaio.
- Beheragoko mailetakoko beste segmentu batzuekin lotuta dagoen segmentua **aita** izango da. Segmentu bat ezin daiteke izan bere buruaren aita.
- Segmentu batek ezin du aita (jabe) bat baino gehiago izan.
- Segmentu batek lotuta dituen besteak **kumeak** dira. Sustrai ez diren segmentu guztiek aita bat dute eta aita diren segmentu guztiek zenbait kume izan ditzakete.

- **Datu-base hierarkiko baten diseinua**

Datu-base erlazionaletan Entitate/Erlazio diagrama erabiltzen bada, eredu hierarkikoan, berriz, **datuen egitura-diagrama** erabiltzen da. Datuen egitura-diagrama datu-baseko diseinu kontzeptuala grafikoki adierazteko tresna da. E3.2 irudian ikus daitekeenez, honako osagai hauek erabiliko dira:

Laukizuzenak: segmentu motak adierazteko erabiltzen dira.

Marrak: segmentu moten arteko erlazioak adierazten dituzte.



E3.2 irudia. Egitura-diagramaren adibidea.

Aita eta kumeen arteko erlazioek 1:1 eta 1:N erakoak izan behar dute, non aita batek zenbait kume izan ditzakeen eta kume batek, aita bakarra. Jarraian, egitura-diagramak nola implementatzen diren ikusiko da. Konparazioa errazteko, E2 eranskinean (sare ereduari dagokiona) erabilitako segmentu-motetan oinarrituko gara.

- Batetik batera (1:1) erlazioa:

Ager daitekeen kasurik errazena da. Marra soil batez adierazten da. Kasu honetan, idazle batek liburu bakarra idatz dezake. Erlazioaren beste aldetik ere, liburu baten idazlea bakarra da. Ikusi E3.3 irudia.



E3.3 irudia. 1:1 erlazioaren egitura-diagrama.

- Batetik askotara (1:N):

Kasu honetan, idazle batek liburu bat baino gehiago idatz ditzake, baina liburu baten idazlea bakarra da. Marrak noranzko bakar batean dauka geziburua; hain zuzen, geziburuak N segmentu duen entitateari zuzenduta egon behar du. Dagokigun kasuan LIBURU segmentu-multzoari.



E3.4 irudia. 1:N erlazioaren adibide baten egitura-diagrama eta implementazioa.

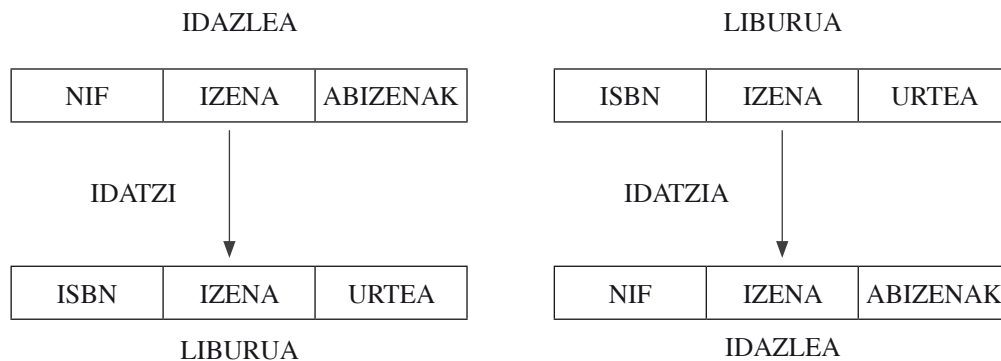
- Askotatik askotara (N:M):

Kasu honetan, idazle batek liburu bat baino gehiago izan dezake eta gainera, liburu batek zenbait egile. Marrak noranzko bietan dauka geziburua. Ikusi E3.5 irudia.



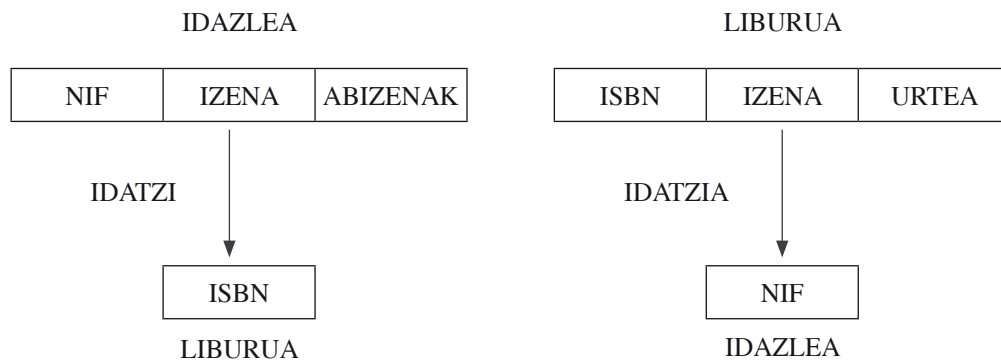
E3.5 irudia. N:M erlazioaren adibide baten egitura-diagrama.

Erlazio hori implementatzeko, bi erlazio erabiltzen dira, biak 1:N motakoak. Ikusi E3.6 irudia.



E3.6 irudia. N:M erlazioaren egitura-diagramaren implementazioa.

Ikus daitekeenez, irtenbide honek erredundantzia asko sortzen du. Adibidea jarraituz gero, liburuaren izena eta urtea alferrik errepikatzen dira IDATZI erlazioan eta gauza bera gertatzen da IDATZIA erlazioan idazlearen izen-abizenekin. Horrexegatik, errealitatean E3.7 irudiko hobekuntza aplikatzen da erredundantzia ahal den neurrian gutxitzeko, bi erlazioetan “N” aldeko segmentuak atributu bakar batera murriztuz (gakoa izango da, hain zuzen ere):



E3.7 irudia. N:M erlazio baten egitura-diagramari dagokion inplementazioaren hobekuntza.

• Programaren lan-eremua

Oharra: Puntu hau eranskin honetan eta sare eredu eranskinean guztiz berdinak dira, baina berriro idatzi dugu eranskinen irakurketa independentea errazteko.

Datu-basea atzitzen duten aplikazioek datuak maneiatzeko eremu bat kudeatzen dute. Eremu horrek honako atal hauek ditu:

- Segmentu-txantiloia: datu-baseak segmentu-mota bakoitzeko memorian gune bat mantentzen du, berorren bidez erabiltzaileek eta programek informazio eskuragarria izateko.

- Uneko erakusleak: biltegitratze periferikoetako datuak maneiatzeko erakusleak dira. Euron artean daude, alde batetik, uneko segmentu-motako erakuslea (segmentu-mota bakoitzeko bat dago), uneko multzoko erakuslea (multzo bakoitzeko bat) eta uneko exekuzio-unitateko erakuslea (mota axola gabe, berriki atzitutako segmentua seinalatzen du). Eta beste alde batetik, DB_STATUS, azken eragiketaren emaitza adierazten duena (0 denean eragiketa ondo egin dela adierazten du) eta DB_RECORD_NAME, atzitutako segmentu motaren izena.

- **Datuak Definitzeko Lengoaia (DDL)**

Eredu hierarkikoan Datuak Definitzeko Lengoaia (DDL) azaletik baino ez da ikusiko, ez baita liburu honen helburua eredu horren ikerketa sakona egitea, batez ere dagoeneko ez delako erabiltzen. Ondorengo hauetan, IBMren IMS sistemaren aginduak agertuko dira, hura izan baitzen arrakastatsuenetakoa. IMS da IBMren datu-baseak kudeatzeko sistema. Hedapen zabala lortu zuen 70eko hamarkadan. Osagai nagusi bi ditu: DCBa datu-basearen egitura definitzeko eta PCB datu-basearen ikuspegia.

Eredu hierarkikoan, datu-base fisiko batek zuhaitz-egitura baten agerraldi osoak dauzka. Datu-base fisiko hori eta beraren eta pilatzearen arteko korrespondentzia datu-baseko deskripzioan gordetzen dira (DBD). DBDa eskema kontzeptualaren baliokidea da.

Beste alde batetik, erabiltzaileak ez du zuzenean lan egiten eredu fisikoaren kontra; aitzitik, kanpoko bista bat edo batzuk erabiliz aritzen da. Bista datu-basearen zati bat da (erabiltzaileak une horretan dakusana) eta bista bakoitzeko eta base fisikoaren arteko erlazioa programarekiko komunikazio-bloke batez (PKB) definitzen da. Erabiltzaile baten PKB guztien multzoari Programaren Espezifikazio Blokea esaten zaio (PEB, PSB 'Program Specification Block' ingelesez) eta kanpoko eskemaren baliokidea da.

Badira ensanbladoreko makro batzuk IMSrako DDLa osatzen dutenak. Beraien artean hauek daude DBDa definitzeko:

- DBD: datu-baseari izena emateko erabiltzen da. Horrez gain, beste ezaugarri batzuk ere definitzen ditu (atzipen-modua, besteak beste: HSAM...). Adibidez:

DBD=NireLiburutegia

- SEGM: segmentu mota definitzeko. BYTES erabiltzen da tamaina zehazteko (gure kasuan $15+50+8=73$). Berton adierazten da segmentu-mota horren aita ere. Adibidez:

SEGM NAME=Liburua, PARENT=Idazlea, BYTES=73

SEGM NAME=Idazlea, BYTES=49

- FIELD: segmentu moten atributuak definituko ditu. Adibidez

FIELD NAME= ISBN, BYTES=15, START=1

FIELD NAME= Izena, BYTES=50, START=16

FIELD NAME=Urtea, BYTES=8, START=66

FIELD NAME= NIF, BYTES=9, START=1

FIELD NAME= Izena, BYTES=10, START=10

FIELD NAME=Abizenak, BYTES=30, START=20

Eta PKBa definitzeko, berriz, ondorengo hauek:

- PCB: PKBko lehenengoa izango da. Berton, DBNAME datu-basearen izena jartzeko erabiltzen da eta KEYLEN, klabeak kontuan izanik, zuhaitzeko adar luzeenaren tamaina jartzeko (kasu honetan, NIFak gehi ISBNak okupatzen dutena; hau da, 9 gehi 15, ez dago beste adarrik eta).

PCB TYPE=DB, DBNAME=NireLiburutegia, KEYLEN=24

- SENSEG: PCBko segmentu motak zehazteko. NAMEk izena adierazten du, PARENTEk aita zein den eta PROCOPTek prozesu-aukerak zein diren (G: irakurri, D: ezabatu, I: gehitu, R: aldatu, A: denak, eta K: gakoari soilik atzipena). Adibidez:

SENSEG NAME=Idazlea, PROCOPT=G

SENSEG NAME=Liburua, PARENT=Idazlea, PROCOPT=G

- **Datuak Manipulatzeko Lengoaia (DML): IMSren DL/I**

Eredu hierarkikoan segmentuak banaka prozesatzen dira eta manipulatzeko lengoaia beste lengoaia batean murgilduta egoten da. Segmentuen zuhaitza irakurtzeko, gehien erabilitako ordena hau da: erro segmentutik kumeetara eta ezkerreko segmentutik eskuinetarako segmentuetara. Segmentu bat ezabatuz gero, kume dituen segmentu guztiak ere ezabatuko dira. IMS sistemaren DML lengoaia DL/I da.

DL/I agindua egikaritu ondoren, gune bat geratzen da eskuragarri programarentzat, non emaitzari eta datu-basearen egoerari buruzko informazioa jasotzen den (adibidez, ikuspegi baten amaiera horri esker ohartarazten da). Gune horrek **egoera-kode** du izena. Berton agertzen diren balio batzuk:

- GA: segmentu eta mailaz aldaketa
- GB: datu-basearen amaiera
- GP: errorea kokamenduan
- GE: ez dago adierazitako baldintzak betetzen dituen segmenturik
- GK: segmentu-aldaketa, maila baten barruan
- AC: segmentu-izen ezezaguna
- AK: atributu-izen ezezaguna
- LC: errorea kargatzerakoan, segmentua ez dago sekuentzia hierarkikoan

DML aginduen laburpena:

- Berreskuratu (GET)

- Txertatu (INSERT)
- Ezabatu (ERASE)
- Aldatu (REPLACE)

Agindu baten egitura edo sintaxia honako hau da:

AGINDUA <segmentu-mota> [WHERE <baldintza>]

WHERE atalak sare eredian FIND aginduaren zeregina egiten du.

• Segmentuak atzitu

Ohar modura, komeni da aipatzea zuhaitzean bilaketak egiterakoan IMSk erabilitako algoritmoa “aurre orden” izeneko delat. Hau da, lehenengo sustraia, gero ezkerreko azpizuhaitza eta amaieran, eskuinekoa. GET UNIQUE aginduak (GU) datu-baseko segmentuak atzitzeko balio du. Datu-baseko hasieratik era sekuentzialean hasiko da bilatzen eta aginduak baldintza betetzen duen lehenengo segmentua aurkituko du.

Adibidea: *Edorta* idazlearen *Datu-baseak* izeneko liburua aurkitzeko:

GU Idazlea (Izena='Edorta')

Liburua (Izena='Datu-baseak')

Hurrengo segmentua irakurtzeko GET NEXT dago (GN). Segmentuak adieraziz gero, maila baxue-
netakoa baino ez da itzuliko. Adibidez, 2007. urtean argitaratuko liburuak kausitu nahi izanez gero:

GU Idazlea

Liburua (Urtea='2007')

Segmentuaren trataerako lengoia ostalariaren aginduak

PERFORM UNTIL “ez aurkitua”

GN Liburua

Segmentuaren trataerako lengoia ostalariaren aginduak

END-PERFORM

Badago GET NEXT UNIQUE aita baten azpian hurrengo segmentua aurkitzeko. Adibidez, *Edorta* izeneko idazlearen liburu guztiak bilatzeko:

GU Idazlea (Izena='Edorta')

Liburua

Segmentuaren trataerako lengoia ostalariaren aginduak

PERFORM UNTIL “ez aurkitua”

GNP Liburua

Segmentuaren trataerako lengoia ostalariaren aginduak

END-PERFORM

- **Segmentuak txertatu**

Segmentua txertatzerakoan, aita datu-basean sortuta egongo da aurretiaz. INSERT agindua (ISRT) erabiltzen da. Adibidez, *Edorta* idazlearen beste liburu baten agerraldi bat txertatzeko:

ISRT Idazlea (Izena='Edorta')

Liburua (ISBN='86-00634-2', Izena='ORACLE', Urtea='2007')

- **Segmentuak aldatu**

Horretarako, lehenengo, segmentua aurkitu, gero, datuak aldatu, eta azkenik, REPLACE (REPL) aginduarekin aldaketak datu-basean gaurkotu behar dira. GET HOLD agindua erabili behar da aldatu edo ezabatu ahal izateko; hiru era ezberdin ditu: GHU (*get hold unique*), GHN (*get hold next*) eta GHNP (*get hold next within parent*).

Adibidea: *Edorta* idazleari NIFa aldatzeko .

GHU Idazlea (Izena = 'Edorta')

NIFa aldatzeko lengoia ostalariaren aginduak

REPL

Glosarioa

ACID: Atomicity, Consistency, Isolation, Durability. Transakzioen ezaugarriak.

ACP: transakzioen atomozitatea bermatzeko protokoloa.

Agerraldi: erregistro mota bateko erregistro zehatz bat. Adibidez, IKASLE izeneko erregistroan, *Aitor Sarasola Egurrola*.

Alegiazko memoria: ordenagailuko memoria zentralak eta memoria lagungarri azkarrek (normalean diskokoek) osaturiko multzoa, azken horiek memoria zentralaren hedapen moduan erabiltzen direnean. Sistema eragileek kudeatzen dute programatzaileek alegiazko memoria osoa memoria zentrala balitz bezala erabil dezaten, instrukzioak edo informazioa memoria zentrallean edo lagungarrian dauden arduratu gabe.

Algoritmo: problema baten ebazpenerako eman behar diren urratsen deskripzio formalak.

ANSI (American National Standards Institute): estandareak gainbegiratzeaz arduratzen den EBBBko erakundea

Atributu: erlazio bateko zutabea.

CODASYL: 1959 urtean sorturiko enpresa informatikoen erakundea, COBOL lengoia asmatu zuena.

COMMIT: transakzioa ondo bukatu delako baieztapena ematen duen agindua.

Data mining: datuen pilaketatik ezagupena lortzeko bideraturiko teknikak.

Datu-baseen sistema: datu-baseak eta datu-baseen kudeaketarako sistemak osatutako multzoari deritza.

Datu-hiztegi: biltegi daitezkeen datuei buruzko informazio-bilduma da.

DB: datu-basea, denbora-tarte batean existitzen den datu-bilduma egituratua.

DBA (Data Base Administrator): datu-basearen administratzailea, datu-basea administratuko du eta pribilegio-mailarik altuenaren jabe izango da.

DBB: datu-base banatua. Mota horretako datu-baseetan, logikoki erlazonaturiko datuak zenbait kokapenetan egoten dira.

DBBKS: datu-base banatua kudeatzeko sistema. Datu-basea banatua era automatizatuan kudeatzeko behar den aplikazio-multzoa.

DBD (Data Base Description): eredu hierarkikoan, eskema kontzeptualaren baliokidea da.

DBKS: datu-basearen kudeaketa guztiz automatizatua egiteko behar den aplikazio-multzoa.

DCB: IMSko datu-basea definitzeko DDLaren atala.

DCL (Data Control Language): datu-basea kontrolatzeko lengoia. Erabiltzaile ugariko sistemetan datuen segurtasuna mantentzeko, atzipen-murrizketak ezartzeko, konkurrentzia kudeatzeko eta abar kudeatzeko erabiltzen da.

DDL (Data Definition Language): datuak definitzeko lengoia. Abstrakzio-maila bakoitzeko, datu-basea osatzen duten elementuen definizioa egiten da, hala nola eredu erlazionalean, erlazioak osatzen dituen eremuak, gakoak eta abar.

Disparadore: ikusi *trigger*.

DML (Data Manipulation Language): datuak manipulatzeko lengoia, hau da, datuak txertatzeko, aldatzeko...

E/R: entitate/erlazioak adierazteko laburdura.

Erlazio: sare motako datu-baseetan eta hierarkikoetan, taulek elkarren artean dauzkaten loturak zehazteko erabiltzen dira. Erlazionaletan, berriz, taula bera da.

Esteka: lotura.

Gako atzerritarra: bi erlazio erlazionatzen direnean, lehenengo erlazioan gako nagusia osatzen duen atributu-multzoa bigarren erlazioan ere agertzen denean, gako atzerritar esaten zaio multzo horri.

Gako nagusi: datu-basearen diseinua egiterakoan, tuplo bakoitza era bakarrean definituko duen atributua edo atributu-multzoa da.

Ikuspegi: erabiltzaile bakoitzari beharizanen arabera erakusten zaion informazioa.

Indize: atzipenak azkartzeko, erregistroei dagozkien gako ordenatuen zerrenda.

Intranet: intranet bat erakunde barruko sare lokala da (LAN). Internet bidezko erremintak eskaintzen ditu: kontsulta, mezularitza, biltegiatzea eta beste hainbat funtzio.

Konfidentzialtasun: zenbaiten artean geratuko den segurtasunaz egiten dena.

Konkurrentzia: une berean egiten diren eragiketen multzoa.

LAN (Local Area Network): sare lokal, tokiko sare.

MAN (Metropolitan Area Network): hiriko edo metropoli-barrutiko eremu geografikoa hartzen duen sarea.

Memoria lagungarri: mementoan behar ez den informazioa gordetzen du, eta hemen dagoen informazioa erabiltzeko memoria nagusira pasatu behar da. Memoria nagusia baino motelagoa baina eduki handiagokoa izaten da.

Memoria nagusi: eragiketa bat egiteko behar diren datuak eta instrukzioak gordetzen ditu. Ezaugarri nagusien artean ditu zuzeneko atzipena, kapazitate ez oso handia eta nanosegundo gutxi batzuetako atzipen-denbora.

Osotasun: atal guztiez hornituta dagoenari dagokion izaera.

PCB: IMSko datu-basearen ikuspegia definitzeko erabiltzen den DDLaren atala.

PKB: bista bakoitzeko eta base fisikoaren arteko erlazioa definitzen duen agindu-multzoa. Kanpoko eskemaren baliokidea.

PSB (Program Specification Block): eredu hierarkikoan, erabiltzaile baten PKB guztien multzoa. Kanpoko eskemaren baliokidea da.

RAID (Redundant Array of Independent Drives): disko gogor batean baino gehiagotan datuak gorde eta errepikatzen dituen ordenagailurako datu-gordailu eskema da.

Rol: baimen jakin batzuen multzoa, erabiltzaileari baimenak banan-banan eman beharreak, sententzia bakar baten bidez emateko sortzen dena.

ROLLBACK: transakzioa bertan behera gelditzen bada, egindako aldaketak desegiteko agindua.

Segmentu: eredu hierarkikoan, erregistro-motaren baliokidea.

Sendotasun: nahi ez den eran aldatzeko arriskurik gabe mantentzeko izaera duena.

SQL (Structured Query Language): datu-base erlazionalak kudeatzeko erabiltzen den lengoia estandarra.

Transakzio: instrukzio-multzoa. Transakzioa osatzen duten instrukzioak, guztiak, edo bat ere ez egikariturako dira.

Trigger: gertaera konkretu eta baldintza batzuen menpe egikaritzen den agindu-multzoa.

Tuplo: erlazio bateko errenkada.

UPS (Uninterruptible Power System): etenik gabeko elikadura.

WAN (Wide Area Network): hedadura zabaleko sarea.

Bibliografia

- BELL, D., GRIMSON, J.: *Distributed Database Systems*, Addison-Wesley, London, Erresuma Batua, 1992.
- CANIVELL, Verónica: *Bases de Datos Relacionales*, Apuntes Universidad de Deusto, Espainia, 1992.
- COPELAND, George P., MAIER D.: *Making SMALLTALK a database system*, ACM SIGMOD Conference on the Management of Data, Boston, EEBA, 1984.
- DE MIGUEL, A., PIATTINI, M.: *Fundamentos y modelos de bases de datos*. 2. edizioa, Rama, Madril, Espainia, 1999.
- GUERRA, M.: *Introducción práctica al diseño de SGBD*, Ed. Anaya Multimedia, Madril, 1987.
- HERNANDEZ ORALLO, J.: *La disciplina de los sistemas de bases de datos. Historia, situación actual y perspectivas*, <http://www.dsic.upv.es/~jorallo/docent/BDA/DisciplinaBD.pdf> , 2002
- KORTH, H., SILBERSCHATZ, Abraham: *Fundamentos de bases de datos*. 3. ed, Mc Graw-Hill, Madril, Espainia, 1998.
- LAMB, C., LANDIS, G., ORENSTEIN, J., WEINREB D.: *Readings in Database Systems (The Objectstore System)*, The Morgan Kauffman Series in Data Management Systems, Morgan Kauffman, San Francisco, EEBA, 1998.
- LAPORTA, P.: *Estructura de la informacion*, Ed. McGraw-Hill, Madril, Espainia, 1992.
- LURUEÑA JIMÉNEZ, Sonia: *Historia de la informática*,
< <http://www.um.es/docencia/barzana/IATs/PDF/IATS04.pdf> >, 2005.
- NAVATHE, S.B., KARLPALEM, K., MINYOUNG, R.: *A mixed Fragmentation Methodology for Initial Distributed Database Design*, <<http://www.cs.ust.hk>>, 1997.
- NIELSEN, S.: *Lexicographical Basis for an Electronic Bilingual Accounting Dictionary. Theoretical considerations*, <http://www.sprog.asb.dk/sn/lexicographicalbasis.htm>, 2002.
- ÖZSU, M.T., VALDURIEZ, P.: *Principles of Distributed Database Systems*, Ed. Prentice Hall, New Jersey, EEBA, 1991.
- RAMOS, M.J., RAMOS, A., MONTERO, F.: *Desarrollo de aplicaciones en entornos de 4ª generación y con herramientas case*, Ed. McGraw-Hill, Madril, Espainia, 2000.
- GARCÍA BLANCO, S.M., MORALES RAMOS, E.: *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión*, Ed. Paraninfo, Madril, Espainia, 2003.
- SHANKLAND, S.: *Study: Open-source databases going mainstream*, <<http://news.com.com>>, 2004
- SHEPERD, J.A., HARANGSRI, B., CHEN, H.L., NGU A.H.H.: *A Two-Phase Approach to Data Allocation in Distributed Databases*, Fourth International Conference on Database Systems for Advanced Applications, World Scientific Press, Singapur. <<http://www.cse.unse.edu.au>>, 1995-1996.
- SKEEN, D.: *A Formal Model of Crash Recovery in a Distributed System*, IEEE Transactions on Software Engineering, 9 Bol., 3 Zba, May, 219-228, <<http://www2.computer.org>>, 1983.

SOL, Selena: *Introduction to Databases for the Web: Pt. 1*, <<http://www.databasejournal.com>>, 1998.

PIATTINI, M., CALVO-MANZANO, J.A., CERVERA, J., FERNÁNDEZ, L.: *Análisis y diseño detallado de aplicaciones informáticas de gestión*, Ra-Ma, Madril, Espainia, 1996.

