informatika fakultatea

facultad de informática

Informazioaren Kudeaketa Aurreratua

Arantza Irastorza Goñi

Lengoaia eta Sistema Informatikoak saila
Infomatika fakultatea

2016ko uztaila

XML-ren inguruko teknologia

1. gaia: XML dokumentuen kontsulta: XPath (egiteko)

2. gaia: XML dokumentuen eskemak: XMLSchema

XML dokumentu baten ***baliozkotzea*** dokumentu horrek eskema zehatz baten erregelak betetzen dituen egiaztatzen duen prozesua da. ***Baliozkotzailea*** helburu orokorreko programa bat da, eskema batekiko baliozkotzeak burutzen dituena. ***Ondo-eratutako*** dokumentua XML-ren estandarraren erregelak betetzen dituena da. ***Baliozko*** dokumentua eskema baten erregelak betetzen dituena.

Eskema batek zehazten du xml dokumentuan zein elementu mota ager daitezkeen, zein elementu agertu behar diren, beste batzuen parte bezala zein elementu ager daitezkeen, zein atributu ager daitezkeen edo agertu behar diren, zein erako balio egon daitezkeen edo egon behar diren atributu batean. XMLSchema estandarrak eskemen egitura nolakoa izan behar den zehazten du. Eskema bat, azken finean xml dokumentu bat da, *.xsd* hedapena duen fitxategia batean gordeko dena.

Eskema definitzeko identifikatzaile bat eta terminoen hiztegi bat zehaztuko dira. Identifikatzailea karaktere-kate bat besterik ez da, URI (*Uniform Resource Identifier*) estandarra jarraitzen duena, eta *izen-eremu* (*namespace*) deitu ohi zaio. Izen-eremuaren identifikatzailea bakarra da, eta izen-eremuarekin, eskeman definituriko termino bakoitza modu unibokoan identifikatuko dela ziurtatu behar da. Eskeman, hiztegia adierazteaz gain terminoen arteko murriztapenak ere definituko dira, xml dokumentuen egitura kontrolatzeko.

XML eskemak XML Schema estandarraren eskema erabiliz definitzen dira, *.xsd fitxategietan, eta XML dokumentuak XML eskemen arabera definitzen dira, *.xml fitxategietan. XML dokumentuaren goiburuan, dokumentuak jarraitzen duen XML eskemari buruzko erreferentzia adieraziko da, baliozkotzaileak bere baliozkotze prozesuan erabil dezan.

2.1 XML eskemaren edukia

Eskeman datu-motak (sinpleak nahiz konplexuak), atributuak eta elementuak (sinpleak nahiz konplexuak) definitzen dira.

Elementuen eta datu-moten definizioa hiru erara egin daiteke: (1) beste elementu edo datu-mota baten barnean *txertatuta*, (2) elementua aparte definitzen da eta gero beste elementu edo datu-mota baten definiziotik *erreferentzia* egiten zaio, eta (3) datu-mota aparte definitzen da eta gero elementuaren definizioan *mota erabili* egiten da. Azken bi aukerek elementuen eta datu-moten berrerabilpena ahalbidetzen dute.

Elementuak. *Sinpleek* ez dute ez atributurik ez beste azpielementurik, eduki moduan testua besterik ez dute; *konplexuek*, berriz, atributuak eduki ditzakete eta lau motakoak dira: hutsak direnak, testua bakarrik daukatenak, beste azpielementuak dauzkatenak eta testua eta beste azpielementuak nahasian dauzkatenak.

Datu-motak. *Izendunak* edo *anonimoak* izan daitezke. Lehenengoek izena dute, definizioan ezarrita, eta erreferentzia eginez beste elementu edo datu-mota baten definizioan erabil daitezke. Anonimoek ez dute izenik eta beste elementu edo datu-motan barnean txertatuta erabili behar dira. Hurrengo azpiatalean gehiago zehaztuko da.

2.2 Datu-motak: sinpleak eta konplexuak

Datu-mota sinpleak: (1) oinarrizko motak, (2) oinarrizko motetatik eratortzen diren mota sinpleak, (3) eratorritako mota sinpleetatik eratortzen diren mota sinpleak, (4) zerrendak edo bildura moduan lortutako mota sinpleak, (5) ID eta IDREF modukoak.

Datu-mota sinpleak elementuentzat eta atributuentzat definitzen dira, edukia besterik ez dute, atributurik eta egiturarik gabe. Oinarrizko moten arteko batzuk: string, boolean, double, date, time, etab. Datu-mota eratorriak definitzean, datu-mota horren balioek bete behar dituzten murriztapenak ezartzen dira. Oinarri bezala datu-mota bat (oinarrizkoa nahiz eratorria) hartzen da eta <restriction> klausularen bitartez murriztapenak ezarriko dira (adibidez, luzera mugatzeko, txantiloi bat ezartzeko, balio-zerrendaketa bat ezartzeko, etab.)

ID eta IDREF motak oro har batera erabiltzen dira. ID motarekin elementuaren balioa dokumentu osoan zehar bakarra izango dela murrizten da. IDREF motarekin elementuaren balioak ID motakoa den beste balio batekin bat etorri behar duela adierazten da. IDREFS-ek berdina adierazten du, baina balio zerrenda bat harturik. ID mota elementu baten balioari lotuta egongo da, ez elementu bikote edo hirukote bati (datu-baseen gakoekin ez bezala). Eta IDEREF-ek ez du elementu desberdinen artean desberdintzen, beraz ID motako edozein elementu erreferentzia dezake.

Datu-mota konplexuak: (1) atributuak edo azpielementuak dauzkaten mota konplexuak, (2) eduki sinplea duten mota eratorri konplexuak, (3) eduki konplexua duten mota eratorri konplexuak, (4) eduki konbinatua duten mota konplexuak.

Datu-mota konplexuen egitura murriztapenen (<*all*>, <*sequence*> eta <*choice*>) eta errepikapenaren adierazleen (*minOccurs* eta *maxOccurs*) bitartez adierazten da. *maxOccurs* eta *minOccurs* adierazleek elementu bat zenbat aldiz gehienez eta gutxienez (hurrenez hurren) ager daitekeen adierazten dute. Beren defektuzko balioa 1 da. <*sequence*> murriztapenarekin azpielementuen ordena mugatzen da, definizioan aipatutakoa izan behara da, <*all*> murriztapenarekin ordenak ez du garrantzirik, dokumentuan azpielementu guztiak agertu beharko dira, baina ordenari garrantzia eman gabe. <*choice*> murriztapenarekin azpielementuen alternatibak definitzen dira, dokumentuan horietako bat agertu beharko da. <*sequence*> eta <*choice*> murriztapenak habiaratuta ager daitezke.

Datu-mota eratorriak definitzeko <*extension*> eta <*restriction*> murriztapenak erabiltzen dira. Lehenengoarekin datu-mota konplexuak lortzen dira, bigarrenarekin sinpleak nahiz konplexuak. Bi murriztapen horiekin, oinarri bezala, datu-mota sinpleak eta konplexuak erabil daitezke.

2.3 Datu-motak: ordezkapen mekanismoak

XML dokumentuan elementuaren edukia idazterakoan ***xsi:type*** klausularekin datu-motaren ordezkapena adieraz daiteke. Honekin, oinarri mota datu-mota horretatik eratorri den mota batengatik ordezkatzen dela adierazten da.

Ordezkapen aukerak murrizteko datu-mota definitzean hiru atributu erabil daitezke: *final*, *block* eta *abstract*. *final*-ekin eskematan datu-mota eratorrien definizioa mugatzen da, *final*

bezala definituriko datu-motak ezingo baitira eratorri. *block*-ekin instantziatan datu-mota eratorrien ordezkapena mugatzen da. Elementuetan ezartzen da: T motako elementu batean *block* ezarriz, instantziatan elementu horretan T-ren zein azpimota erabili ahal izango den mugatzen da. *abstract*-ek datu-mota eratorriak definitzera behartzen du, datu-mota abstraktuak ezin baitira instantziatu. Datu-mota abstraktua duten elementuak XML dokumentuan instantziatzean datu-mota eratorria erabili beharko da.

2.4 Datu-baseetakoen antzeko murriztapenak

Atributu bat: *nillable*, eta hiru murriztapen: *unique*, *key* eta *keyref*.

Elementu baten definizioan *nillable* atributua *true* balioarekin erabiliz gero, XML dokumentuetan elementu hori null balioarekin ager daitekeela adierazten da. *unique* murriztapenak elementu edo atributu baten balioa, bere bizi-eremuan, bakarra izan behar dela adierazten du. *key* murriztapenak elementu edo atributu baten balioa, berau hartzen duen elementuaren barnean, gakoa dela adierazten du. *keyref* murriztapenak adierazten du elementu edo atributu baten balioa *key* edo *unique* bezala definitutako elementu baten balioetako batekin bat etorri behar dela.

2.5 Eskemaren aldakortasuna

Eskeman elementu batzuen egitura oso antzekoa izan daiteke, aldakortasun hori definitzeko hiru aukera desberdin daude: choice murriztapena erabiltzea, elementuen ordezkapena erabiltzea eta datu-moten ordezkapena erabiltzea.

Elementuen ordezkapena erabiltzea (hots, elementua definitzean *substitutionGroup* atributua ezartzea, zeinek beste elementu bat, oinarri elementua, erreferentziatzen duen). Horrela elementu multzoak definitzen dira, eta hauek oinarri elementua erabil daitekeen toki guztietan erabili ahalko dira. Ordezko elementuaren mota oinarri elementuaren motatik eratorritakoa izan behar da. Ordezkapen erlazioa iragankorra da, baina ez du konmuztadura propietatea betetzen.

Datu-moten ordezkapena erabiltzea (hots, *xsi:type* atributuarekin). Oinarri mota bere datu-mota eratorrietakoren batekin ordezkatzen da.

Choice murriztapena erabiliko da, aldakortasuna definitzen duten elementuak semantikoki koherenteak ez direnean. Kontuan hartu behar da, aukera hau ez dela hedagarria izango, hau da, *choice* murriztapenari ezingo zaizkio gerora azpielementu gehiago gehitu. Elementuen eta datu-moten ordezkapenarekin elementuek nahiz datu-motek semantikoki erlazioa badute, nolabaiteko lotura dute, eta hedagarritasunerako aukera ematen dute: beste eskema batzuek elementu edo datu-mota multzoa hedatzeko aukera egongo da.

2.6 Eskemaren hedagarritasuna

any murriztapenak XML eskema nagusian erazagutu ez den elementu gehigarri bat XML dokumentuan sartzeko aukera ematen du. Horrela, dokumentuaren egileak eskemaren egileak aurreikusi ez duen elementu berri bat gehitu dezake.

anyAttribute murriztapenak helburu berdina du baina atributuekin.

2.7 Eskemen konplexutasunaren kudeaketa

Eskemak oso konplexuak izan daitezke, eta horregatik datu-mota eratorriak definitzeko aukerak eskemak modularizatzeko mekanismoa eskaintzen du. Horiekin batera

berrerabilpenerako mekanismoak ere beharrezkoak dira. Mekanismo horiek *include* eta *import* murriztapenak izango dira.

include erabilita, izen-eremu bera hainbat *.xsd fitxategitan barreiatuta dagoela adierazten da: izen-eremu horren elementuak eta eskemak fitxategi desberdinetan kokatzen dira, baina denek izen-eremu bera partekatzen dute. Eskema horretako XML dokumentuan izen-eremu bakar horri egingo zaio erreferentzia.

import erabilita izen-eremu batek beste izen-eremu batzuk erabiltzen dituela adierazten da. Izen-eremu edo eskema nagusiak besteen erreferentzia edukiko du, izen-eremuarena eta *.xsd fitxategiarena ere. Eskema horretako XML dokumentuan izen-eremuaren (eta bere *.xsd fitxategiaren) erreferentzia egingo da, eta berrerabilitako beste izen-eremuen aipamena ere gehituko da.

3. gaia: Murriztapenak XML dokumentuetan: Schematron

Negozio arau edo erregela bat komunitate edo erakunde baten ardurapean ezartzen dena da. Komunitateak du eskubide osoa erregela definitzeko, eta behar duenean, aldatzeko ere. Erregela hori area zehatz batean prozedura edo portaera bideratzeko behar diren legeak edo printzipioetako bat izango da. Erakundearen ardurapean egote horrek garrantzia du, legediak eta estandarrak, esaterako, ez baitira negozio arautzat hartuko: erakundeak edo komunitateak ez du legedi edo estandar horiek moldatzeko eskubiderik izango.

XML arloan negozio arauak inplementatzeko, XML Schema estandarrak aurreko gaian ikusitako hainbat aukera ematen ditu: elementuen egitura eta ordena, elementuen agerpen kopurua, datu-motak murriztea, etab. Beste era batera esanda, XML Schema estandarra gramatikan oinarritzen da. Baina elementu balioen inguruan murriztapen semantikoak, hots, datuen arteko erlaziotan oinarritutako murriztapenak ezarri behar badira, XML Schema motz geratzen da. Horretarako *Schematron* dago (ISO/IEC 19757 estandarra).

Schematron XML dokumentuetako datuen gainean *asertzioak* definitzeko aukera ematen duen lengoaia irekia da. Asertzio horiek, beraien baldintzak ezartzeko xPath adierazpenak erabiliko dituzte. Erabilpen hauek izango dituzte: datuen arteko elkar-mendekotasunak (hots, *co-murriztapenak*) egiaztatzea, datuen kardinalitatea egiaztatzea eta egiaztapen algoritmikoak ezartzea.

3.1 Schematron eskema

Schematron eskema bat XML dokumentu bat da, *.sch* hedapena duen fitxategi batean gordeta. Schematron eskematan erabiltzen den aurrizkia izen-eremurako ***sch:*** izan ohi da.

<*assert*> elementua erabiltzen da murriztapena ezartzeko. Elementuaren edukia lengoaia naturalean idatziriko esaldia izango da, baldintzaren inguruan behar diren azalpenak emateko. Elementuaren *test* atributuak XPath adierazpena edukiko du, betearazi nahi den erregela daukana. Elementuaren edukia asertzioaren egiaztapena false den kasuan bakarrik erakutsiko da, errore mezuaren parte bezala.

<*rule*> elementua testuinguru berdina partekatzen duten asertzioak multzokatzeko erabiltzen da. Bere *context* atributuak erregela XML dokumentuaren zein testuingurutan aplikatuko den zehazten du. Testuingurua dokumentuan parekatzen bada, aipaturiko egoera dokumentuan aurkitzen bada, asertzioak egiaztatuak izango dira. Baliozkotzaileak erregela barnean definituriko asertzio bakoitza baliozkotuko du eta huts egiten duen asertzio bakoitzeko errorea altxaraziko du.

<*pattern*> elementua helburu bera duten erregelak multzokatzeko erabiltzen da, eta <*phase*> elementua *pattern* elementuak multzokatzeko. Horrela asertzioak zatika edo pausoka baliozkotzeko aukera egongo da. Baliozkotzean, fase zehatz bat besterik ez baliozkotzea eska daiteke.

3.2 Baliozkotze prozesua

XML dokumentu instantzia baten baliozkotzean elementu bakoitzetik pasatzen da, dokumentuan dauden orden berean. Elementu bakoitzeko patroi aktiboak bilatzen dira (fasekako aukera aktibatu bada) eta horien barnean elementuak betetzen duen

testuingurua edo baldintza zehazten duten erregelak. Patroi bakoitzean, elementuarekin parekatzea egiten duen lehen erregela altxarazten du, eta erregela barneko asertzio guztiak prozesatzen, egiaztatzen dira. Beraz, elementu bat patroi baten hainbat erregeletako testuinguru baldintzarekin parekatuko balitz, egiaztapena patroiko lehen erregelarekin bakarrik burutuko litzateke.

3.3 Arauen definiziorako hainbat aholku

Schematron-eko erregelak definitzeko orduan kontuan hartu behar dugu baliozkotzearen emaitzak XML-n edo teknologian adituak ez diren erabiltzaileek jasotzen dituztela. Beraz honako gomendioak kontuan hartu beharko lirateke:

- Baliozkotzean errorea gertatuz gero, asertzioan idazten den testua da erabiltzaileak jasoko duen komunikazioa, horregatik testu hori problemaren domeinuko terminologia erabiliz idaztea gomendatzen da. Asertzioak gure esparruko negozio-arauak dira, kontratu moduko bat izan daitezke, horregatik testua erabat zehatza eta argia izatea komeni da, eta aldi berean erabiltzailea errespetuz tratatu behar da. Mezua nork jasoko duen kontuan hartu behar da: erregela ezagutu arren uneren batean hanka sartu duen erabiltzailea, ala erregela ez dakien bat eta bere berri jaso behar duen erabiltzailea. Idazkera estiloa desberdina izango da kasu bakoitzean.
- Erakusten den mezuan, baldintza betetzen ez duen elementuaren inguruko datuak ematea komeni da, mezua ahalik eta hobetoen uler dadin.
- Egokia ikusten denetan, *assert* elementuaren *see* atributuaren bitartez, araua justifikatzen duen materialari esteka gehitu daiteke.

4. gaia: XML dokumentuen kontsultak: XQuery

XQuery kontsulta lengoaia da, dokumentu bateko atributuak nahiz elementuak aukeratzeko, hainbat dokumentuko nodoak konbinatzeko, datuak aldatzeko, emaitzari atributu edota elementuak gehitzeko, balio berriak kalkulatzeko, etab. Laburtuz, XML munduko SQL lengoaia izango litzateke.

Lengoaia funtzionala da, kontsulta bakoitza adierazpen baten bitartez adierazten da eta adierazpenak habiaratu egiten dira. Irakurterraza izateko diseinatuta dago, ez da XML notazioa erabiltzen, baina adierazpen luzeagoak ateratzen dira. XQuery adierazpen baten sarrera eta irteera XML datu ereduaren instantziak dira

4.1 XQuery adierazpenen osagai garrantzitsuenak

XQuery adierazpenen osagai garrantzisuenak honakoak dira:

- *Oinarrizko adierazpenak*: aldagaiak, literalak, funtzio deiak eta parentesiak (lehentasunak kontrolatzeko).
- *Bide-izen adierazpenak*: xml zuhaitz batean nodoak lokalizatzen, kokatzen ditu, eta nodo horiek itzultzen ditu dokumentuan dauden orden berean.
- *Sekuentzia*: hainbat item dituen bilduma ordenatua da, non item bakoitza balio atomikoa edo nodo bat izan daitekeen. Sekuentzia hutsa izatea posible da, eta sekuentziak ez dira habiaratzen. Adibidez, hiru kideko sekuentzia: (3, 45, 10).
- *Adierazpen aritmetikoak*: batuketa, kenketa, biderkaketa, zatiketa eta modulurako eragileak daude.
- *Konparaketa adierazpenak*: bi balio konparatzeko. Konparaketarako lau aukera: (1) balio bakunak konparatzea, (2) orokorra (‘existitzen da’ semantika duten konparaketak, eragileak sekuentziak izan daitezke. Adb. *$book1/author = "Atxaga"* konparaketa horrek true itzultzen du *$book1*-en *author* azpielementuren baten balioa *“Atxaga”* bada), (3) nodoak beraien artean konparatzea eta (4) ordenaketa konparaketa, bi eragileak nodo sinpleak izan behar dira edo sekuentzia hutsak (Adb. *//purchase[item="28"] << //sale[item="33"]* adierazpen honek true itzuliko du ezker aldean identifikaturiko nodoa eskuin aldetik identifikaturiko nodoa baino lehen badago, dokumentuaren ordenean.
- *Adierazpen logikoak*: *and* edo *or* eragile logikoak dituztenak.
- *Eraikitzaileak*: kontsulta baten barnean XML egiturak sor ditzakete, elementuak, atributuak, iruzkinak, prozesatze aginduak sortzeko eraikitzaileak daude. Adibide honetan *<book isbn="{$i/@booknum}" />* , parentesi arteko adierazpena ebaluatuta, atributuaren balioa eraikitzen da. Beste adibide honetan elementuen eta atributuaren eraikitzaileak erabiltzen dira, eraikitzaile konputatuak: ***element*** *book { **attribute** isbn { "34-0060229357" }, **element** author { **element** first { "Bernardo" }, **element** last { "Atxaga" } } }*.
- *FLWR adierazpenak*: iterazioak burutzeko eta bitarteko emaitzak aldagaietan jasotzeko. Adierazpen hauek askotan bi dokumentu edo gehiagoren arteko konbinaketak burutzeko eta datuak berrantolatzeko erabilgarriak dira. ‘FLWR’ akronimoa *for*, *let*, *where*, *return* gako hitzetatik dator, FLWR adierazpenetan aurkitzen diren lau klausulak.

- *Ordenaketa adierazpenak*: sekuentzia bateko itemen ordena kontrolatzeko. Orden gorakorrean edo beherakorrean ordena daitezke.
- *Baldintzazko adierazpenak*: *if*, *then*, *else* gako hitzetan oinarritzen dira.
- *Adierazpen kuantifikatuak*: kuantifikatzaile existentziala eta unibertsala duten adierazpenak, *some* eta *every* gako hitzekin. Adierazpenaren emaitza true edo false da.
- *Datu-motak*: *instance of*, *typeswitch*, *cast*, *treat* eragileekin datu-motaren egiaztapenak eta eraldaketak burutu daitezke.
- *Validate*: eragile honekin emandako argumentua adierazitako eskemaren definizioarekin egiazta daiteke, XML Schema-k ezartzen duen eskema baliozkotzearen prozesua aplikatuz.

4.2 FLWR adierazpenak

Bide-izen (*path*) adierazpenak zuhaitzaren barnean nodoak lokalizatzen ditu, eta nodoen sekuentzia itzultzen du, dokumentuan duten ordena jarraituz. XPath lengoiaren notazioa jarraitzen du, eta XML dokumentu zehatz bat adierazteko *doc()* funtzioa erabiltzen da.

FLWR adierazpenek FOR ... LET ... WHERE ... RETURN ... egitura jarraitzen dute, *order by* klausula ere egon daitekeenez (FOR ... LET ... WHERE ... ORDER BY .... RETURN ...), FLWOR adierazpenak ere esaten zaie. FOR eta LET adierazpenek nodoen sekuentziak itzultzen dituzte, XML dokumentuaren ordena mantenduz. WHERE adierazpenak aurreko sekuentzien gainean iragazkia ezartzen du eta aipaturiko predikatua betetzen ez dituen elementuak atzera botatzen ditu. Sekuentzian predikatua betetzen duen item bakoitzari RETURN adierazpena aplikatzen zaio, eta irteera elementuen zerrenda ordenatua lortzen da. RETURN adierazpen horretan elementu eta atributu eraikitzaileak erabiliko dira, nahi den egitura osatzeko, egitura hori XML arauekin baliozkoa izan beharko da.

FOR eta LET adierazpenak desberdin tratatzen dira. Biek nodo sekuentzia bat lortzen dute, baina jarraian (WHERE eta RETURN adierazpenetan) FOR-eko sekuentziako nodoak indibidualki tratatzen dira, eta, aldiz, LET-eko nodoak denak batera tratatzen dira, sekuentzia bere osotasunean.

Elementu eraikitzailearen bi aukera daude: (1) {..} erabiliz, adibidez, *<bib> { ... "XQuery adierazpen bat" ... } </bib>*, honek 'bib' etiketa duen elementua sortzen du, edukia barneko adierazpenak esaten duena delarik; (2) ***element*** eragilea erabiliz, adibidez, element author { "John Smith"}, honek 'author' etiketa duen elementua sortzen du, edukia John Smith delarik.

FLWR adierazpenetan erabil daitezkeen funtzio batzuk:

- ***data()***: nodoaren edukia itzultzen du. Nodo egituratua bada, hostoetako datuen kateamendua da lortzen dena.
- ***distinct-values()***: eduki bera duten nodoak ezabatzen ditu eta nodoen edukia da itzultzen dena. Horretarako *data()* funtzioa erabiltzen du.
- ***deep-equal(a,b)***: bi aldagaien arteko konparaketa, biek nodo egituratuak dituzte. Funtzioak true itzuliko du baldin eta bi aldagaiek balio berdinak badituzte (eta orden berean), zehazkiagoa: atributu berdinak eta deep-equal baldintza betetzen duten azpinodoak badituzte (azpinodoak orden berean egon behar dira).

- **<<**: dokumentuaren ordenean oinarrituz, nodoen arteko konparaketa. *a* << *b* adierazpenak true itzuliko du, dokumentuaren ordenean a nodoa b nodoa baino lehenago azaltzen bada.

5. gaia: XQuery adierazpenak Java-n (egiteko)

6. gaia: XML dokumentuak eta DB erlazionalak: XSQL

Datu-baseen arloan erlazionala oso oihartzun handiko eredua da, oso hedatua eta bere inguruan milaka aplikazio dituena. XML dokumentuak ere datuak biltegiratzeko eta datuen trukaketarako oso tresna egokiak dira. Horregatik, bi egiturak batera erabiltzeko edo modu osagarrian tratatzeko aukera ematearren, SQL estandarrak XML-rekin lan egiteko hainbat funtzio gehitu zituen. ANSI/ISO SQL estandarraren 14. atala da eta SQL/XML deitzen zaio (edo XSQL ere). Oraclek, IBMk, Microsoftek, etab.ek erabiltzen dute.

Estandarrean, alde batetik, funtzioak gehitu ziren datu erlazionaletatik XML dokumentuak sortu ahal izateko, eta bestetik, XMLType datu-mota gehitu zen taula erlazionaletan XML dokumentuak txertatu ahal izateko eta XML motako datuak tauletan kontsultatu eta eguneratu ahal izateko, beste edozein motako datuak bezala.

6.1 SQLrako XML funtzioak

Jarraian SQL kontsultetan erabil daitezkeen funtzio batzuk azaltzen dira, hauek datu erlazionaletatik xml dokumentuak edo zatiak lortzeko erabiliko dira:

- ***XMLAgg()***: GROUP BY erako kontsultetatik lortutako datuak taldekatzen ditu.
- ***XMLAttributes()***: SQL kontsultak itzultzen dituen XML elementuetan atributuak sortu edo txertatzen ditu.
- ***XMLConcat()***: hainbat XML balio kateatzen ditu.
- ***XMLElement()***: balio erlazional bat XML elementuan bihurtzen du, itxura honekin: *<elementuizena>balioa</elementuizena>*.
- ***XMLForest()***: balio erlazionalen zerrenda batetik XML elementuen zerrenda bat sortzen du.
- ***XMLNamespaces()***: XML elementu batean izen-eremua (*namespace*) erazagutzen du.
- ***XMLRoot()***: XML balio bati erro informazioa gehitzen du (hots xml prozesatze agindua VERSION eta STANDALONE propietateekin, adibidez, *<?xml version="1.0" standalone="yes"?>*.
- ***XMLSerialize()***: XML balio bat karaktere kate moduan serializatzen du.

6.2 XML dokumentuak taula erlazionaletan

XMLType datu-motaren bitartez datu-base erlazionaletan XML datuak txertatu ahal izango dira. Horrela, XML dokumentuak XMLType motaren instantzia bezala errepresenta daitezke. Datu-motak hainbat funtzio ditu XML edukiarekin lan egin ahal izateko (sortu, indizea definitu, kontsultatu, etab.).

XMLType datu-mota bi eratara erabil daiteke:

- Taula osoari aplika diezaiokegu, horrela taulako tupla bakoitzak XML dokumentu bat eduki dezan. Taula hauei *XML taula* dei diezaiekegu. Adibidea: *CREATE TABLE CustomerOrders OF XMLTYPE;*

- Taulako zutabe bati aplika diezaiokegu, horrela tuplako zutabe bat izango da XML dokumentua biltegiratuko duena. Adibidea: *CREATE TABLE Warehouses (id NUMBER(4), spec XMLTYPE, name VARCHAR2(30), location NUMBER(4));*

XML dokumentuekin kontsultak egiteko hainbat funtzio daude:

- ***XMLType()*** funtzio eraikitzaileak XMLType motako balio bat sortzen du, ondo eratutako XML balioa izatea egiaztatzen du.
- ***OBJECT_VALUE*** adierazpenak XML dokumentu baten edukia eskuratzen du. XML taulekin erabiliko da tuplako dokumentuaren edukia eskuratzeko, XML dokumentu osoa berreskuratzen da.
- ***getStringVal()***: XMLType motako instantziaren balioa String moduan itzultzen du.
- ***getCLOBVal()***: XMLType motako instantziaren balioa CLOB moduan itzultzen du. (CLOB (Character Large Object), karaktere kate handietarako datu-mota).
- ***getSchemaUrl()***: sarrerako dokumentua XML eskema batean oinarritzen bada, XML eskema horren URLa itzultzen du.
- ***existsNode(x: xmltype, s: string)***: XML dokumentu batean (x) bide-izen adierazpen batek (s) zehaztutako baldintza betetzen duen nodorik ba ote dagoen adierazten du. 1 edo 0 itzultzen du.
- ***extract(x: xmltype, s: string)***: XML dokumentu batean (x) bide-izen adierazpen batek (s) zehaztutako baldintza aplikatzen da eta lortutako XML zatia (XMLType instantzia moduan) itzultzen du. Bide-izena absolutua edo erlatiboa izan daiteke.
- ***extractValue(x: xmltype, s: string)***: XML dokumentu batean (x) bide-izen adierazpen batek (s) zehaztutako baldintza aplikatzen da eta lortutako nodoaren balio eskalarra itzultzen du. Bide-izena absolutua edo erlatiboa izan daiteke. Emaitza nodo bakarrekoa izan behar da (hots, elementu bat, atributu bat, edo testu nodo bat).

XML dokumentuekin eguneraketak egitean dokumentuaren zati zehatz bat alda daiteke edo bestela XML dokumentu osoa ordezka daiteke. Eguneraketa horiek burutzeko hainbat funtzio daude:

- ***updateXML(x: xmltype, s: string, b: string)***: XML dokumentu batean (x) bide-izen adierazpen batek (s) zehaztutako baldintza betetzen duen nodoa aurkitu eta bere balioa aldatzen du, b jarriz.
- ***insertChildXML(x: xmltype, s: string, c: string, b: string)***: XML dokumentu batean (x) bide-izen adierazpen batek (s) zehaztutako baldintza betetzen duen nodoa aurkitu eta hor XML zati bat zintzilikatzen du (b kateak adierazitakoa), azpi-nodo edo atributu bezala. Azpinodo edo atributu horren etiketaren izena c-k adierazitakoa izango da.
- ***insertXMLbefore(x: xmltype, s: string, b: string)***: XML dokumentu batean (x) bide-izen adierazpen batek (s) zehaztutako baldintza betetzen duen nodoa aurkitu eta bere aurretik b kateak adierazten duen XML zatia txertatzen du.
- ***appendChildXML(x: xmltype, s: string, b: string)***: XML dokumentu batean (x) bide-izen adierazpen batek (s) zehaztutako baldintza betetzen duen nodoa aurkitu eta hor b kateak adierazitako XML zati bat zintzilikatzen du. Edozein eratako nodo izan daiteke.
- ***deleteXML(x: xmltype, s: string)***: XML dokumentu batean (x) bide-izen adierazpen batek (s) zehaztutako baldintza betetzen duen nodoa ezabatzen du.

Aurreko funtzioak erabili ordez, DB hornitzaile batzuen (esaterako, Oracleren) azken bertsiotan XQuery eta berarekin lotutako ***XMLQuery()*** funtzioa erabiltzea bultzatzen dute. ***XMLQuery(s: string)*** funtzioak XQuery adierazpen bat (s) egikaritzen du eta lortutakoa itzultzen du. *PASSING exp* klausularekin, XQuery eragiketa exp adierazpenak esandako XML dokumentuaren gainean aplikatuko dela zehazten da. *RETURNING CONTENT* klausularekin erantzuna XML dokumentu bat izango dela zehazten da. XMLQuery() funtzioa SQL kontsulten SELECT klausulan erabiliko da. Adibidez,

```
SELECT w.w_name,
       XMLQUERY(
           'for $i in //Warehouse/Building
            where contains($i/text(), "Donosti")
            return  <whNumber>{$i/../@whNo} </whNumber>'
       PASSING w.w_spec RETURNING CONTENT).getStringVal()
         "result"
  FROM Warehouses w;
```

Eredu Objektu-Erlazionala

7. gaia: Eredu Objektu-Erlazionala

Erlazionala eredu oso zabaldua izanik ere, baditu bere mugak ere. Alde batetik, kontuan hartu behar da biltegiratzen diren balioen datu-motak aurredefinituta daudela eta ezin direla aldatu, eta eskalarrak eta balio bakarrekoak direla. Ezin da balio egituraturik gorde. Domeinuko balioen eta datu-basean biltegiratutako balioen artean dagoen alde edo diferentzia semantikoa zenbaitetan handia da. Bestetik, datu horiekin lan egiteko eragiketak ere aurredefinitutakoak dira (CRUD eragiketak), erabiltzaileak ezin ditu bere eragiketa propioak definitu. Eta bukatzeko, tuplak identifikatzeko mekanismoa, gakoa, atributuen balioetan oinarritzen da. Beraz kontzeptuen edukiak eta identifikazioa nahastu egiten dira. Eta honek zenbait arazo sortzen ditu: (a) edukia ez dela errepikatzen ziurtatu behar da, (b) entitate errepresentatzerakoan bere deskribapen partziala hartzen da askotan, eta identifikaziorako nahikoa ez izatea gerta daiteke, eta (c) gakoa osatzen duten atributuen edukia aldatuta objektuaren identifikazioa aldatzea ekartzen du.

Arazo edo muga horiei aurre egiteko, Eredu Objektu-Erlazionalak erabiltzaileek beren datu-mota propioak definitzeko aukera ematen du, datu-mota egituratuak eta bere eragiketa eta funtzio propioak izango dituztenak. Gainera identifikatzailearen kontzeptua ezartzen du, datu-baseko objektu bakoitza identifikatzeko erabiliko den karaktere-katea, eta garrantzitsuena dena, edukiaren independentea.

7.1 Datuen egituraren definiziorako aukerak

Eredu Objektu-Erlazionalak hiru datu-mota berri eskaintzen ditu: objektuak, bektoreak eta taula habiaratuak. Jarraian deskribatzen dira.

Objektuen datu-mota. Objektua egituratuta dago, hots, atributuen bitartez definitzen da, eta horiekin lan egiteko metodoak ere definitzen dira. Adibidez, *create type Project_t* ***as object*** *(title varchar2(30), unescoCode number(4), description varchar2(30));*

Bektoreen (varray) datu-mota. Bektorea datu-mota bera duten elementuen bilduma ordenatua da. Adibidez, 3 zenbaki biltegiratzen dituen bektore egitura definitzeko: *create type Preferences_t* ***as varray(3) of*** *number(4);*. Bektorearen aukera, taula habiaratuarekin alderatuz, hobea da: elementu kopuru finko bat bakarrik biltegiratu behar denean, edo elementuen atzipena ordenean egin behar denean, edo bilduma bere osotasunean, balio bat bezala, atzitu eta maneiatu behar denean.

Taula habiaratuen datu-mota. Taula habiaratua mota bereko hainbat elementuren bilduma da, ordenatu gabea. Elementu kopurua mugarik gabea da. Adibidez, *create type Project_list_t* ***as table of*** *Project_t;*. Taula habiaratuak zutabe bakarra du, eta definizioan zehaztutako motako elementuak hor txertatuko dira. Taula habiaratuak, tupla txertatu, eguneratu edo ezabatzeko, beste taula erlazional arruntak bezala maneiatzen dira. Taula habiaratuaren aukera hobea izango da bildumaren gainean kontsulta eraginkorrak burutu nahi direnean, elementu kopuru arbitrarioa biltegiratu behar denean, edo txertatze/eguneratze/ezabatze eragiketak asko burutu behar direnean.

Aurreko datu-mota bakoitza definitzean, inplizituki, motaren izen bera izango duen metodo eraikitzailea ere sortuko da. Motaren balioak sortzeko metodo hori erabili beharko da. Horrela, aurreko adibideekin honako eragiketak burutu ahal izango dira: ***Project_t*** *('web*

design', 3543, 'The target is describing …')); ***Preferences_t*** *(3456, 1234, 3498);* ***Project_list_t*** *(Project_t ('web design', 3444, 'The target is ...'), Project_t ('mongodb intro', 5644, 'The introduction of ...'), ...);*

Lehen esan bezala, eredu honen ezaugarri garrantzitsu bat objektuen identifikaziorako, edukiaren ordez, identifikatzaileak erabiltzea da. Objektuen identifikatzaileentzat ***REF datu-mota*** erabiltzen da. Tupla objektu baten erreferentzia edo erakusle logikoa da. Horrela, atributu baten balioak identifikatzaileak direla adierazteko REF gakoa erabiliko da, adibidez, *create type Student_t(name varchar2(20),..., herProj* ***ref*** *Project_t);*. Honekin *herProj* atributuan *Project_t* motako objektu baten identifikatzailea gordeko dela adierazten da (identifikatzailea, ez objektua bera).

Proposatu diren datu-motak domeinu entitateen deskribapena egiteko balio dezakete, eta gainera, behar bezala konbinatuta, entitaten arteko erlazioak adierazteko ere: 1:N eta M:N erlazioak. 1:N motako asoziazioak adierazteko bi aukera daude, *edukian oinarrituta* edo *erreferentzian oinarrituta*. Lehenengoarekin datu-mota bateko objektuak beste motako objektuetan txertatzen dira, adibidez: *create type Student_t(.... hisProj Project);*. Bigarrenarekin, datu-mota bateko objektuen identifikatzaileak dira beste motako objektuetan txertatzen direnak, adibidez: *create type Student_t(.... hisProj ref Project);*. M:N erlazioak definitzeko orduan taula habiaratuak erabili beharko dira, datu-mota bateko objektuen identifikatzaileak biltegiratzen dituzten taula habiaratuak.

7.2 Objektuen biltegiratzea

Objektuak biltegiratzeko bi aukera daude:

- Taula erlazionaletan: objektuak taulako beste balioekin batera biltegiratzen dira. Objektuak zutabe baten balioak dira, eta horregatik ***zutabe objektu*** deitu ohi zaie. Adibidez, *create table Person (id number(3), ...,* ***herProj Project_t****);*.
- Objektu tauletan: taula hauek objektuak bakarrik biltegiratzen dituzte. Tupla bakoitza objektu bat errepresentatzen du, horregatik objektu hauei ***tupla objektu*** esan ohi zaie. Objektu hauek *OID (objektuaren identifikatzailea)* dute. Taula hauek maneiatzeko orduan bi eratara egin daiteke: (1) zutabe bakarreko taula moduan hartuta, eta taulan objektuak daudela kontuan hartuz objektuei-bideratutako eragiketak burutuz; (2) zutabe anitzeko taula moduan hartuta, objektu motaren atributu bakoitza zutabe bat hartzen duela ulertu eta eragiketa erlazionalak burutuz. Taula mota honen adibide bat: *create table Project* ***of Project_t****;*. Objektu mota bereko hainbat objektu taula defini daitezke.

Lehen aipatu den *OID (objektuaren identifikatzailea)* objektu tauletan biltegiratzen diren tupla objektuen identifikatzaileak dira. Horrela, tupla objektu horiek erreferentziatu ahal izango dira, beste tupla objektutatik edo taula erlazionaletatik (horretarako definizioan REF mota erabili beharko da). OID identifikatzaile horiek *systemak berak automatikoki sortutakoak* izan daitezke (adibidez, Oracle-k 16 biteko luzera ezartzen dio, eta indizea definitzen dio, bilaketak azkartzeko), edo bestela taularen oinarrizko gakoan oinarritutakoak izan daitezke (honela adieraziko litzateke: *create table (...) object identifier is primary key*;

Bektoreak taulan bertan txertatuta biltegiratzen dira, taula habiaratuak, berriz, aparteko taula batean, biltegiratze taula deituko duguna. Adibidez, *create type project_list_t as table of project_t;* definitu ondoren, taula bat honela definitzen bada: *create table student (name varchar2(20), ..., hisProj project_list_t);* orduan *hisProj* atributu horren balioak (taula habiaratuak direnak) fisikoki non biltegiratuko diren adierazi behar da, zein biltegiratze taula erabiliko den. Hortaz, taularen definizioa honela osatu behar da: *create table student*

(name varchar2(20), ..., hisProj project_list_t) nested table hisProj store as proi_taula; eta *proi_taula* sistema eragileko fitxategi bat da, *Project_list_t* motako balio guztiak (*hisProj* atributukoak) biltegiratuko dituena.

7.3 Motaren deskribapenean metodoak definitzeko aukerak

Datu-mota deskribatzean egitura ezartzeaz gainera, mota horretako balioekin lan egiteko behar diren metodoak ere defini daitezke. Datu-mota definitzean, inplizituki, bere ***metodo eraikitzailea*** ere sortuko da, motaren izen bera duena eta motak dituen atributu adina parametro dauzkana. Metodo honek definituriko datu-motakoa den objektu bat itzuliko du erantzun gisa. Balioen hasieraketa kontrolatzeko-edo datu-motaren definizioa egiten duen erabiltzaileak bere metodo eraikitzaile propioak defini ditzake.

Beharren arabera, datu-motarekin kide metodoak eta metodo estatikoak ere defini daitezke.

- ***Kide metodoak***. Metodo operazionalak ere deitzen zaie. Instantzien, objektuen gainean deitzen dira, sintaxi honekin: *objektua.metodoa()*. Objektuaren egoera, ezaugarriak erabili ohi dituzte. Metodo hauen artean garrantzia berezia dute ordenaketarako diren metodoak, MAP eta ORDER gako hitzekin bereizten direnak. Azken metodo hauek objektuen arteko alderaketak, konparaketak, egiteko erabiliko dira.
- ***Metodo estatikoak***. Datu-mota erabiliz deitzen dira, ez objektuen gainean. Sintaxi honekin: *datuMotarenIzena.metodoa()*. Metodo orokorrak dira, ez dute objektu zehatz baten balioak edo edukia behar.

Datu-mota baten definizioan ordenaketa metodo bakarra deskriba daiteke, MAP ala ORDER. MAP erako kide metodoak objektuak balio eskalarretara mapeatzen ditu eta objektuak ordena ditzake ardatz eskalarrean duten posizioaren arabera. Oro har metodo hauek objektuaren atributuen araberako kalkuluak egiten dituzte, itzuli beharreko balioa sortzeko. ORDER metodoak zuzenean bi objekturen balioak konparatzen ditu. ORDER metodoen aukera hartu beharko da konparaketaren semantika MAP metodo bat erabiltzeko konplikatuegia denean.

Metodoak deitzeko modua.

Datu-motaren definizioan metodoak deitzerakoan zein baimen erabiliko diren zehatz daiteke. Bi aukera daude *current user* ala *definer*, azken hau da defektuzkoa. Definizio baten adibidea, *create type Project_t* ***authid current user*** *as object (...);*. Datu-motaren definizioan *current user* ezarrita, metodoen egikaritzapenean deia burutzen duen erabiltzailearen baimenak erabiltzen dira. *definer* ezarrita, berriz, metodoen egikaritzapenean datu-mota definitu zuen erabiltzailearen baimenak erabiltzen dira.

7.4 Datuen maneiurako aukerak

Eredu erlazionaleko ohiko eragiketak erabil daitezke (insert, update, delete). Objektu bat sortzea egokitzen denean dagokion metodo eraikitzailea erabili beharko da, adibidez, *create table Project of Project_t;* bezala definitutako taula baterako *insert into Project values (Project_t(...));* egikaritu beharko litzateke.

Kontsultetan nahiz aldatze eragiketatan objektuak erreferentziatzeko ***ref()*** funtzioa erabiliko da, objektu baten OID eskuratzeko. Bestalde, objektuaren OID jakinda, ***deref()*** funtzioa erabiliko da berak erreferentziatzen duen objektua bera eskuratzeko. Adibidez,

select ref(p) from Project p; kontsultak Project taulan biltegiratutako objektu guztien OID eskuratzen ditu.

Taula habiaratuekin lan egiteko ***table()*** edo ***the()*** funtzioak daude (baliokideak dira). Funtzio hauen argumentu bezala SQL kontsulta bat pasako da, taula habiaratua eskuratzen duena. Funtzio horiek itzulitako objektuaren gainean ohiko SQL eragiketak aplikatu ahal izango dira. Adibidez, *create type Project_t as object (title varchar2(30), unescoCode number(4), description varchar2(30)); create type project_list_t as table of project_t; create table student (name varchar2(20), ..., hisProj project_list_t);* definitu ondoren, *hisProj* atributuaren balioan (taula habiaratuan) proiektu berri bat txertatzeko honako agindua burutu beharko litzateke: *insert into **the**(select hisProj from student where name='Jon')* values (Project_t('design', 3321, 'the target of...'));*

7.5 Datuen hierarkia

Eredu Objektu-Erlazionalak *heredentzia edo espezializazio bakuna* onartzen du, hots azpiklase batek superklase bakarra izan dezake. Espezializazioa datu-mota mailan definitzen da. Datu-mota zehatz bat espezializatzeko aukera eman nahi bada, bere definizioan ***not final*** gako hitzak gehituko dira, adibidez, *create type Project_t as object (title …) not final;*. Beste datu-mota baten azpimota definitzeko ***under*** klausula gehituko da, adibidez *create type BigProject_t under Project_t (budget ...);*. Azpimotak supermotaren atributuak eta metodoak heredatzen ditu, eta gainera atributu eta metodo berriak gehitu ditzake, edo heredatutako metodoak berdefinitu.

is_of() funtzioarekin objektu baten datu-mota zehatza egiaztatu daiteke. Adibidez, *select value(a) from Project a where value(a**)** is of (bigproject_t);* kontsulta honek *Project* taulan gorde diren *Project_t* motako objektuen artean zehazki *BigProject_t* azpimotakoak direnak bakarrik eskuratzen ditu.

treat() funtzioak objektu aipaturiko azpimotakoa balitz bezala tratatzen ditu eta bere atributuetarako atzipena egiteko aukera ematen du. Adibidez, *select treat(value(a) as BigProject_t).budget from Project a;* kontsultan proiektuen aurrekontua eskuratzen da. Atributu hori *BigProject_t* datu-motakoak diren objektuena bakarrik denez, eta errorea eman ez dezan, *treat()* funtzioa aplikatu da *Project_t* motakoak izanik *BigProject_t*-ekoak ez direnek (eta ondorioz *budget* atributua ez dutenek) *budget* balio bezala *null* erakuts dezaten.

BigData

8. gaia: BigData: sarrera (egiteko)