2.6. Összefoglalás

- *Adatmodellek:* Az adatmodell egy jelölésmód egy adatbázis adatszerkezetének a leírására, beleértve az adatra vonatkozó megszorításokat is. Az adatmodell általában az adatokon végezhető műveletek leírására alkalmas jelölést is magába foglalja. A műveletek lehetnek lekérdezések, illetve adatmódosítások.
- *Relációs modell:* A relációk információt reprezentáló táblák. Az oszlopok fejlécében attribútumok vannak, minden attribútumhoz tartozik egy tartomány vagy adattípus. A táblasorokat soroknak hívjuk, és a reláció minden attribútumához tartozik a sornak egy komponense.

- *Sémák:* A relációnév a reláció attribútumaival együtt adja a relációsémát. A relációsémák összessége az adatbázisséma. Egy relációhoz vagy relációk összességéhez tartozó adatokat az adott relációsémához vagy adatbázissémához tartozó előfordulásnak nevezzük.
- *Kulcsok:* Egy fontos megszorítási típus a táblákra nézve, hogy ki tudunk jelölni egy attribútumot vagy attribútumhalmazt, amely egy kulcsot alkot a relációra nézve. Egy reláció semelyik két sora nem egyezhet meg a kulcsattribútumokon, de egy részhalmazukon igen.
- *Félig-strukturált adatmodell:* Ebben a modellben az adat fa- vagy gráfszerkezetbe van szervezve. Az XML a félig-strukturált adatmodellek egyik fontos példája.
- *SQL:* A relációs adatbázisrendszerek alapvető lekérdező nyelve az SQL. A jelenlegi szabványt SQL-99-nek nevezik. A forgalomban lévő rendszerek általában eltérnek a szabványtól, de nagyjából megőrzik azt.
- *Adatdefiníció:* Az SQL rendelkezik az adatbázisséma elemeit meghatározó utasításokkal. A `CREATE TABLE` utasítás lehetővé teszi egy tárolt reláció (nevezetesen tábla) sémájának megadását: az attribútumok, a hozzájuk tartozó típusok, az alapértelmezett értékek és a kulcsok meghatározását.
- *Sémamódosítás:* Az adatbázisséma részeit megváltoztathatjuk az `ALTER` utasítás segítségével. Ez a változtatás lehet a relációsémák egy attribútumának a hozzáadása vagy törlése, és lehetőséget ad egy attribútum alapértelmezett értékének megváltoztatására is. A `DROP` utasítás használható a reláció teljes sémájának vagy egyéb sémaelemnek a törlésére.
- *Relációs algebra:* Ez az algebra egy fontos lekérdező nyelv a relációs modellben. Alapvető műveletei: egyesítés, metszet, különbség, kiválasztás, vetítés, Descartes-szorzat, természetes összekapcsolás, théta-összekapcsolás és átnevezés.
- *Vetítés és kiválasztás:* A kiválasztás művelet egy olyan eredményt ad vissza, amely az argumentumában szereplő relációnak az összes olyan sorát tartalmazza, amely megfelel a kiválasztási feltételnek. A vetítés az argumentumában szereplő reláció nem kívánt oszlopait eltávolítja az eredményrelációból.
- *Összekapcsolások:* Két relációt kapcsol össze egy relációba a relációkban szereplő sorok összehasonlításával. A természetes összekapcsolásnál azon sorokat rakjuk össze, amelyek az összes közös attribútumon megegyeznek a két relációban. A théta-összekapcsolásnál azon sorokat fűzzük egybe, amelyek a théta-összekapcsolás kiválasztási feltételének eleget tesznek.
- *Megszorítások a relációs algebrában:* Többféle megszorítás is kifejezhető két relációs algebrai kifejezés közötti tartalmazásként vagy egy relációs algebrai kifejezés és az üres halmaz közötti egyenlőség segítségével.

3.8. Összefoglalás

- *Funkcionális függőségek:* A funkcionális függőség egy állítás, amely azt mondja, hogy ha egy reláció két sora megegyezik néhány megadott attribútumon, akkor megegyezik attribútumok egy másik halmazán is.

- *Relációk kulcsa:* Egy reláció szuperkulcsa egy olyan attribútumhalmaz, amely funkcionálisan meghatározza a reláció többi attribútumát. A kulcs olyan szuperkulcs, melynek egyik valódi részhalmaza sem szuperkulcs.

- *Funkcionális függőségek bizonyítása:* Számos olyan szabály van, amellyel beláthatjuk, hogy egy $X \rightarrow A$ funkcionális függőség fennáll minden olyan előfordulásban, amely megfelel bizonyos funkcionális függőségeknek. Az $X \rightarrow A$ igazolásához számoljuk ki X lezártját úgy, hogy a funkcionális függőségek felhasználásával addig bővítjük X-et, amíg nem tartalmazza A-t.

- *Minimális bázis funkcionális függőségek egy halmazához:* Bármely függőséghalmazhoz létezik legalább egy minimális bázis, amely az eredetivel ekvivalens funkcionális függőségekből álló halmaz (azaz egyik halmazból következik a másik). A függőségek jobb oldalain egyetlen attribútum áll, egyetlen szabály sem hagyható el az ekvivalencia megtartásával, és a bal oldalakról sem hagyható el egyetlen attribútum sem az ekvivalencia megtartása mellett.

- *Boyce–Codd normálforma:* Egy reláció BCNF-ben van, ha a nem triviális függőségek csak olyanok, melyekben egy szuperkulcs meghatároz egy vagy több más attribútumot. A BCNF legnagyobb előnye, hogy kiküszöböli azokat a redundanciákat, amelyeket a funkcionális függőségek jelenléte okoz.

- *Veszteségmentes összekapcsolással rendelkező dekompozíció:* A dekompozíció egy másik hasznos tulajdonsága, hogy az eredeti reláció a felbontásban szereplő relációk természetes összekapcsolásával előállítható. Bármely dekompozíció visszaad legalább annyi sort, amennyivel elindultunk, de a gondatlanul elvégzett dekompozíció olyanokat is képes előállítani, amelyek nem voltak az eredeti relációban.

- *Függőségmegőrző dekompozíció:* A dekompozíció egy másik elvárt tulajdonsága, hogy minden, az eredeti relációban fennálló függőséget ellenőrizhessünk a felbontott relációkban fennálló függőségek ellenőrzésével.

- *Harmadik normálforma:* Néha a BCNF-dekompozíció elveszíti a függőségmegőrző tulajdonságát. A BCNF egy változata, a 3NF megenged olyan $X \rightarrow A$ függőségeket akkor is, ha X nem szuperkulcs, feltéve, hogy A egy kulcs része. A 3NF nem garantálja, hogy megszünteti a funkcionális függőségekből adódó redundanciát, de többnyire megteszi.

- *A chase:* Eldönthetjük, hogy egy dekompozíciónak megvan-e a veszteségmentes összekapcsolási tulajdonsága, ha kitöltünk egy táblázatot (tablót) – ami azon sorok halmaza, amelyek az eredeti reláció sorait reprezentálják. A táblázat kitöltését a fennálló funkcionális függőségek alkalmazásával végezzük úgy, hogy belátjuk egyes szimbólumpárok egyenlőségét. A dekompozíció veszteségmentes a megadott függőséghalmaz mellett akkor és csak akkor, ha a chase egy olyan sorhoz vezet, amely azonos azzal a sorral, amelynek a létezését feltettük a felbontás utáni relációk összekapcsolásakor.

- *3NF-szintetizáló algoritmus:* Ha vesszük a funkcionális függőségek egy minimális bázisát, beletesszük mindet egy relációba, és hozzáadjuk a kulcsokat, ha szükséges, akkor a kapott eredmény a 3NF-re bontás, amelynek megvan a veszteségmentes összekapcsolási és a függőségmegőrző tulajdonsága is.

- *Többértékű függőségek:* A többértékű függőség egy olyan állítás, amely azt mondja, hogy két attribútumhalmaz a relációban olyan értékeket tartalmaz, amelyek az összes lehetséges kombinációban előfordulnak.

- *Negyedik normálforma:* A többértékű függőségek szintén okozhatnak redundanciát a relációban. A 4NF hasonló a BCNF-hez, de kiküszöböli azokat a nem triviális többértékű függőségeket, amelyek bal oldala nem szuperkulcs. A relációk 4NF-re bonthatók anélkül, hogy információt veszítenénk.

- *Többértékű függőségek bizonyítása:* Funkcionális és többértékű függőségeket bizonyíthatunk függőségek egy halmaza alapján a chase-eljárással. Egy kétsoros tablóból indulunk ki, amely reprezentálja azt a függőséget, amelyet bizonyítani próbálunk. A funkcionális függőségeket szimbólumok egyenlővé tételére alkalmazzuk, míg a többértékű függőségeket a tablóhoz új sorok hozzáadására, amelyekben a megfelelő komponensek felcserélve szerepelnek.

4.11. Összefoglalás

- *Egyed-kapcsolat modell:* Az E/K-modellben egyedhalmazokat adunk meg, köztük lévő kapcsolatokkal, valamint attribútumokat ezen egyedhalmazokhoz és kapcsolatokhoz. Az egyedhalmazok elemeit egyedeknek nevezzük.

- *Egyed-kapcsolat diagramok:* Téglalapokat, rombuszokat és oválisokat használunk az egyedhalmazok, a kapcsolatok és attribútumaik ábrázolásához.

- *Kapcsolatok típusa:* Bináris kapcsolat lehet sok-sok, sok-egy vagy egy-egy típusú. Egy-egy kapcsolatban bármely egyedhalmaz egyede legfeljebb egy egyeddel lehet kapcsolatban a másik egyedhalmazból. Sok-egy kapcsolatban a „sok" oldalán álló egyedhalmaz minden egyede a másik oldalnak legfeljebb egy egyedével állhat kapcsolatban. A sok-sok típusú kapcsolatokra nincs a fentiekhez hasonló megszorítás.

- *A jó terv:* Az adatbázis-tervezés megköveteli, hogy a valóságot pontosan reprezentáljuk, megfelelő elemeket válasszunk (például kapcsolatokat, attribútumokat), és elkerüljük a redundanciát, azaz ugyanazt a dolgot ne vegyük kétszer, vagy valamit ne indirekt vagy bonyolult módon használjunk.

- *Alosztályok:* Az E/K-modellben ún. öröklési („az-egy" típusú) kapcsolattal adhatjuk meg azt, hogy egy egyedhalmaz speciális esete egy másiknak. Egyedhalmazokat e kapcsolatokkal hierarchikus szerkezetbe sorolhatunk, ahol a közvetlen leszármazott mindig speciális esete a szülőnek. Az egyedek komponensei tetszőleges részfát alkothatnak, melynek a gyökér mindig eleme.

- *Gyenge egyedhalmazok:* Ezek egyedei csak kapcsolódó egyedhalmazok attribútumainak segítségével azonosíthatók. A gyenge egyedhalmazok megkülönböztetésére a dupla keret speciális jelölést használjuk.

- *Egyedhalmazok átírása relációkká:* Az egyedhalmazhoz megfeleltetett relációban az egyedhalmaz minden attribútumához tartozik egy attribútum. Kivéve, ha az E gyenge egyedhalmaz, amelyhez tartozó relációban olyan attribútumoknak is benne kell lenniük, amelyek más egyedhalmaz kulcsattribútumai és az E egyedek azonosítását segítik.

- *Kapcsolatok átírása relációkká:* Egy E/K-kapcsolat átírásával keletkezett reláció tartalmazza a kapcsolatban részt vevő minden egyes egyedhalmaz kulcsattribútumait. Ha egy kapcsolat valamely gyenge egyedhalmaz támogató kapcsolata, akkor nem szükséges hozzá külön relációt létrehozni.

- *Öröklési hierarchiák átírása relációkká:* Az egyik lehetőség az, hogy az egyedeket a hierarchia különböző egyedhalmazai szerint komponensekre bontjuk, és minden egyedhalmazhoz létrehozunk egy-egy relációt az összes szükséges attribútummal. A második lehetőség az, hogy az egyedhalmazok összes lehetséges részhalmazához (részfájához) létrehozunk egy-egy relációt, így minden egyedhez egy sor tartozik abban a relációban, amely pontosan azon egyedhalmazoknak felel meg, amelyhez az egyed hozzátartozik. A harmadik lehetőség az, hogy csupán egyetlen relációt hozunk létre, melyben nullértékeket szerepeltetünk azon attribútumokon, amelyek egy adott egyedet reprezentáló sorban nem értelmezettek.

- *UML (Unified Modeling Language – Egységesített Modellező Nyelv):* Az UML-ben osztályokat és az osztályok közötti társításokat írunk le. Az osztályok az E/K-modell egyedhalmazainak megfelelői, a társítások pedig az E/K bináris kapcsolataihoz hasonlók. A sok-egy kapcsolatok speciális fajtái az aggregációk és a kompozíciók; ezek a relációvá való átalakításkor adnak segítséget.

- *UML-osztályhierarchiák:* Az UML-ben az osztályoknak lehetnek alosztályai, ezek öröklik a szuperosztály tulajdonságait. Az osztály alosztályi rendszere lehet teljes vagy részleges, diszjunkt vagy átfedő.

- *Az UML-diagramok relációkká konvertálása:* A módszer hasonló ahhoz, amit az E/K-modellekhez használtunk. Az osztályokból relációkat képezünk, a társításokból a társított osztályok kulcsaiból álló relációt hozunk létre. Az aggregációk és a kompozíciók a kapcsolat „sok" végéhez létrehozott relációkba épülnek bele.

- *ODL (Object Definition Language – Objektum Definíciós Nyelv):* Adatbázisok sémájának objektumorientált stílusú formális leírására alkalmas nyelv. Osztályokat definiál, amelyek háromféle tulajdonsággal, nevezetesen attribútumokkal, metódusokkal és kapcsolatokkal rendelkezhetnek.

- *ODL-kapcsolatok:* ODL-ben a kapcsolatoknak binárisaknak kell lenniük. A kapcsolatot nevük reprezentálja, amelyet a két kapcsolatban levő osztályban egymás inverzeiként kell definiálni. A kapcsolat lehet sok-sok, sok-egy, vagy egy-egy típusú, attól függően, hogy a kapcsolatban lévő párok típusa egyszerű objektumként vagy objektumok halmazaként van deklarálva.
- *Az ODL típusrendszere:* Az ODL lehetővé teszi típusok konstruálását. Az osztálynevekkel és atomi típusokkal (mint például integer típus) kezdve a következő típuskonstruktorokat használhatjuk: struktúraképző, halmaz-, multihalmaz-, lista-, tömb- és szótárkonstruktorok.
- *Kulcsok az ODL-ben:* a kulcsok használata az ODL-ben nem kötelező. Megengedett, hogy egy vagy több kulcsot deklaráljunk, de minthogy az objektumoknak van objektumazonosítójuk, amely nem tartozik a (deklarációban felsorolt) tulajdonságaik közé, az ODL-t megvalósító rendszer különbséget tud tenni két objektum között akkor is, ha minden tulajdonságértékük azonos.
- *Az ODL-osztályok relációkká konvertálása:* A módszer megegyezik az E/K és az UML esetében használt módszerrel, kivéve amikor az osztálynak komplex típusú attribútuma is van. Ekkor előfordul, hogy az eredményreláció normalizálatlan lesz, és emiatt szét kell bontanunk (dekomponálni kell). Szükségessé válhat új attribútum létrehozása is az objektum azonosítójának reprezentálása céljából, ez fog kulcsként is szolgálni.
- *Az ODL-kapcsolatok relációkká konvertálása:* A módszer megegyezik az E/K-kapcsolatok esetében használt módszerrel, azzal a különbséggel, hogy először az ODL-kapcsolatokat az inverzeikkel összepárosítjuk, és e kapcsolatpár számára egyetlen relációt hozunk létre.

5.5. Összefoglalás

- *Relációk multihalmazokként:* A forgalomban lévő adatbázisrendszerekben a relációkat multihalmaznak tekintik, amelyben ugyanazon sor többször is előfordulhat. A relációs algebra halmazokon értelmezett műveletei kiterjeszthetőek multihalmazokra is, de van néhány algebrai azonosság, amely sérülni fog.

- *A relációs algebra kiterjesztései:* Az SQL lehetőségeinek megfelelően néhány műveletet be kell vezetnünk a relációs algebrába. A relációk rendezése például egy ilyen művelet. Ugyanilyen példa a kiterjesztett vetítés, amely során az oszlopokon történő számítások is elvégezhetők. A csoportosítás, az összesítés és a külső összekapcsolás is hasonlóan szükséges műveletek.

- *Csoportosítás és összesítés:* Az összesítés összegzi egy reláció egyik oszlopát. A tipikus összesítési műveletek az összegzés, az átlag, a számlálás, a minimum és a maximum. A csoportosítás lehetővé teszi számunkra, hogy a sorokat bizonyos attribútumuknak vagy attribútumaiknak az értéke vagy az értékei alapján felosszuk. A csoportokra pedig összesítés(eke)t végezhetünk.

- *Külső összekapcsolás:* Két reláció külső összekapcsolása az érintett relációk összekapcsolásával kezdődik, majd bármelyik relációból származó lógó soroknak (azaz olyan sorok, melyeket nem sikerült más sorral összekapcsolni) a másik reláció értékeinek null értékekkel kitöltött változatának az eredményrelációhoz történő hozzáadásával folytatjuk.

- *Datalog:* Ezen logikai formalizmus lehetővé teszi a relációs modellbeli lekérdezések leírását. Egy Datalog-szabály részcélokból tevődik össze, azaz egy fejpredikátum vagy reláció a törzs feltételei által van definiálva.

- *Atomok:* Mind a fej, mind a részcélok atomok. Egy atom tartalmazhat egy (esetleg negált) predikátumot valahány argumentummal. A predikátumok reprezentálhatnak relációkat vagy aritmetikai összehasonlításokat (például $>$).

- *IDB- és EDB-predikátumok:* A tárolt relációkra hivatkozó predikátumokat EDB- (extenzionális adatbázis) predikátumoknak vagy relációknak nevezzük. A többi, szabályokkal meghatározott predikátumokat pedig IDB- (intenzionális adatbázis) predikátumoknak nevezzük. Az EDB-predikátumok nem szerepelhetnek a szabály fejében.

- *Biztonságos szabályok:* Általában a Datalog-szabályokról feltesszük, hogy biztonságosak, azaz hogy a szabályokban szereplő összes változó előfordul a törzs valamely nem negált relációs részcéljában. A biztonságosság biztosítja, hogy ha az EDB-relációk végesek, akkor az IDB-relációk is azok lesznek.

- *Relációs algebra és Datalog:* Minden olyan lekérdezés, amely felírható relációs algebrában, kifejezhető Datalogban is. Ha a szabályok biztonságosak és nem rekurzívak, akkor ugyanazt a lekérdezéseket tartalmazó halmazt kapjuk, mint a relációs algebránál.

6.7. Összefoglalás

- *SQL:* Az SQL a relációs adatbázis-kezelő rendszerek legfontosabb lekérdező nyelve A jelenlegi szabvány az SQL-99 vagy SQL3. A kereskedelmi rendszerek lényegében kielégítik ezt a szabványt.
- *Select-from-where lekérdezések:* Az SQL-lekérdezések legáltalánosabb alakja select-from-where alakú. Segítségével több reláció szorzatára (`FROM` záradék) alkalmazhatunk egy feltételt (`WHERE` záradék), és levetíthetjük a kívánt komponensekre (`SELECT` záradék).
- *Alkérdések:* A select-from-where lekérdezéseket alkérdésként is használhatjuk egy másik lekérdezés `WHERE` záradékában vagy `FROM` záradékában. Az `EXISTS`, `IN`, `ALL` és `ANY` operátorok logikai értékű feltételeket fejezhetnek ki a `WHERE` záradékban használt alkérdések eredményeként keletkezett relációkkal kapcsolatban.
- *Halmazműveletek relációkon:* Képezhetjük relációknak vagy relációkat definiáló lekérdezéseknek egyesítését, metszetét és különbségét a `UNION`, `INTERSECT` és `EXCEPT` kulcsszavak használatával.
- *Összekapcsolási kifejezések:* Az SQL-ben megtalálhatók olyan kifejezések, mint a `NATURAL JOIN`, amelyeket relációkra lekérdezésként vagy lekérdezésekben a `FROM` záradékban relációdefiniálásokra lehet alkalmazni.
- *Nullértékek:* Az SQL biztosít egy speciális értéket, a `NULL`-t, amely olyan komponensek helyén szerepel a relációkban, melyeknek nem tudjuk a konkrét értékét. A `NULL`-ra vonatkozó aritmetikai és logikai műveletek különböznek a szokásostól. Egy `NULL` érték összehasonlítása bármely értékkel, legyen az akár maga a `NULL` is, `ISMERETLEN` logikai értéket eredményez. Ez a logikai érték úgy viselkedik a logikai kifejezésekben, mintha az `IGAZ` és a `HAMIS` között félúton helyezkedne el.
- *Külső összekapcsolások:* Az `OUTER JOIN` operátor összekapcsolja a relációkat, de az eredmény tartalmazza azokat a sorokat is, amelyek nem kapcsolódnak össze a másik relációból egy sorral sem; ezeknek a soroknak az ismeretlen komponensei `NULL` értékekkel töltődnek ki.
- *A relációk multihalmazmodellje:* Az SQL a relációkat multihalmazoknak és nem halmazoknak tekinti. Az ismétlődéseket megszüntethetjük a `DISTINCT` kulcsszó segítségével, míg az `ALL` kulcsszó biztosítja, hogy az ismétlődések ne legyenek kitörölve olyan esetben, amikor az lenne az alapértelmezés.

- *Összesítések:* A reláció egy oszlopában megjelenő értékeket összesíthetjük a `SUM`, `AVG` (átlag), `MIN`, `MAX` és `COUNT` kulcsszavak segítségével. A sorokat az összesítés előtt csoportosíthatjuk a `GROUP BY` kulcsszavakkal. Bizonyos csoportokat kitörölhetünk az eredményből a `HAVING` kulcsszó segítségével.

- *Módosító utasítások:* Az SQL biztosítja a lehetőséget a sorok módosítására egy relációban. Beszúrhatunk sorokat az `INSERT`, törölhetünk a `DELETE` és módosíthatunk az `UPDATE` kulcsszavak segítségével.

- *Tranzakciók:* Az SQL lehetővé teszi utasítások tranzakcióba szervezését. Egy tranzakció véglegesíthető vagy hatása visszavonható (abortálható). A visszavonást kezdeményezheti az alkalmazás (ekkor a célja a módosítások visszavonása), vagy maga a rendszer abból a célból, hogy az atomosságot és az elkülönítést biztosítani tudja.

- *Elkülönítési szintek:* Az SQL a tranzakciók futtatásához négy különböző elkülönítési szintet biztosít, amelyek a legszigorúbbtól a leggyengébbig rendre a következők. Sorba rendezhető (egy ilyen szinten futó tranzakció úgy viselkedik, mintha vele egy időben nem futna másik tranzakció), ismételhető olvasást biztosító szint (egy ezen a szinten futó tranzakcióban egy lekérdezés során beolvasott összes sort újból megkapjuk a lekérdezés megismételt végrehajtásakor), véglegesített olvasás szintje (ezen a szinten a párhuzamosan futó módosító tranzakcióknak először le kell futniuk, és a véglegesített módosításokat látja a tranzakció). A negyedik, leggyengébb konzisztenciát biztosító szint nem támaszt megszorításokat azzal kapcsolatban, hogy a tranzakció más tranzakciók által elvégzett milyen módosításokat láthat.

7.6. Összefoglalás

- *Hivatkozásiépség-megszorítások:* Előírhatjuk, hogy egy attribútum vagy egy attribútumhalmaz értékeinek elő kell fordulnia a reláció vagy egy másik reláció valamelyik sorának megfelelő attribútumában vagy attribútumaiban. Ezt a relációséma megadásakor a `REFERENCES` vagy a `FOREIGN KEY` kulcsszóval adhatjuk meg.

- *Attribútumra vonatkozó* `CHECK` *feltételek:* Egy attribútum értékeire vonatkozó megszorítást előírhatunk úgy, hogy a `CHECK` kulcsszót és az ellenőrizendő feltételt megadjuk a relációsémában az attribútum deklarációja után.

- *Sorra vonatkozó* `CHECK` *feltételek:* Egy reláció soraira megadhatunk ellenőrizendő feltételeket úgy, hogy a reláció deklarációja után megadjuk a `CHECK` kulcsszót és a feltételt.

- *Megszorítások módosítása:* Sorra vonatkozó `CHECK` feltételt törölhetünk vagy hozzáadhatunk a táblához, a táblára vonatkozó `ALTER` utasítással.

- *Önálló megszorítások:* Önálló megszorítást az adatbázisséma részeként adhatjuk meg. A deklarációban adjuk meg az ellenőrizendő feltételt. A feltétel vonatkozhat az adatbázisséma egy vagy több relációjára, és hivatkozhat a reláció egészére – például összesítő függvények esetén – vagy az egyes sorokra.

- *Az ellenőrzések kezdeményezése:* Az önálló megszorításokat a rendszer akkor ellenőrzi, amikor a feltételben szereplő relációk valamelyike megváltozik. Az attribútumra és a sorra vonatkozó `CHECK` feltételeket a rendszer csak akkor ellenőrzi, ha az az attribútum vagy az a reláció, amire a feltétel vonatkozik, beszúrás- vagy módosításutasítás hatására változik meg. Így ez utóbbi megszorítások megsérülhetnek, ha alkérdéseket is tartalmaznak.

- *Triggerek:* Az SQL-szabvány a triggereket is tartalmazza, amelyek megadhatnak bizonyos eseményeket, amelyek életre hívják őket. Ilyen esemény lehet a beszúrás, a módosítás vagy a törlés. Amikor a trigger kiváltódik, a rendszer egy feltétel teljesülését ellenőrzi, amelynek igaz volta esetén egy megadott, SQL-utasításokból álló sorozat hajtódik végre.

8.6. Összefoglalás

- *Virtuális nézettáblák:* A virtuális nézettábla egy definíció; annak definíciója, hogy a nézettáblát logikailag hogyan lehet megkonstruálni az adatbázis tárolt tábláiból vagy más nézettáblákból. A nézettáblák ugyanúgy lekérdezhetők, mint a tárolt táblák, de az SQL átalakítja a lekérdezést úgy, hogy az a nézettábla definiálásánál használt tárolt táblákon fog lefutni.
- *Módosítható nézettáblák:* Az egy relációból felépített virtuális nézettáblák módosíthatók abban az értelemben, hogy a nézettáblába sorokat szúrhatunk be, törölhetünk és módosíthatunk ugyanúgy, mintha tárolt tábla lenne. Az ilyen műveleteket az adatbázisrendszer a nézettábla definíciójában szereplő alaptáblára vonatkozó ekvivalens műveletekké alakítja át.

- *Instead-of-Trigger („helyette működő trigger"):* Az SQL lehetővé teszi, hogy a nézettáblákra speciális triggert alkalmazzunk. Amikor nézettáblát módosító utasítás kerül végrehajtásra, akkor helyette az instead-of-trigger az alaptáblákon a triggerben megadott utasításokat hajtja végre.

- *Index:* Az SQL-szabvány ugyan nem tartalmazza, de a kereskedelmi rendszerek általában biztosítják az indexek definiálásának lehetőségét, és ezek az indexek meggyorsítják az egyes lekérdezéseket és módosításokat, amelyek keresési feltételében egy adott indexelt attribútumhoz tartozó, adott érték vagy értéktartomány szerepel.

- *Indexek megválasztása:* Amíg az indexek a lekérdezéseket felgyorsítják, ugyanakkor az adatbázis módosításait lelassítják, ugyanis a módosított relációhoz tartozó indexeket is módosítani kell. Így az indexek kiválasztása komplex probléma, az adatbázison végrehajtott tipikus utasítások függvénye.

- *Az index automatikus kiválasztása:* Egyes adatbázisrendszerek az indexek automatikus kiválasztására alkalmas eszközöket kínálnak. Ezek felmérik az adatbázison végrehajtott tipikus lekérdezéseket és módosításokat, és értékelik a különböző létrehozható indexek költséghatását.

- *Tárolt nézettáblák:* Ahelyett, hogy a nézettáblát az alaptáblákon végrehajtott lekérdezésként kezelnénk, a nézettáblát definiáló lekérdezést egy újabb tárolt tábla definiálására is használhatjuk, ezen táblában tárolt értékek az alaptáblákban tárolt értékek függvényei.

- *A tárolt nézettáblák karbantartása:* Amint az alaptábla megváltozik, akkor a rá hivatkozó, olyan tárolt nézettáblákat is megfelelően módosítani kell, amelyek értékei a módosításban érintettek. A tárolt nézettáblák sok általános típusánál a módosítások növekményesen is végrehajthatók anélkül, hogy a teljes nézettáblát újra elő kellene állítani.

- *A lekérdezések átírása tárolt nézettáblák lekérdezésévé:* Azok a feltételek, amelyek mellett a lekérdezések átírhatók úgy, hogy a tárolt nézettáblákat (is) felhasználják, bonyolultak. Ezzel együtt, ha a lekérdezésoptimalizáló képes ilyen átírásra, akkor az automatikus tervezési eszköz figyelembe veheti a tárolt nézettáblák létrehozásával elérhető hatékonyságnövekedést, és így automatikusan kiválaszthatja a létrehozandó tárolt nézettáblákat.

9.8. Összefoglalás

- *Háromrétegű architektúrák:* A nagy mennyiségű felhasználói interakciót támogató, webes elérésű, nagyméretű adatbázis-telepítések általában három szintet különböztetnek meg a folyamatokra nézve: webszerverek, alkalmazásszerverek, illetve adatbázisszerverek. Egy szinten belül több folyamat is lehet aktív, ezek a folyamatok egy vagy több processzorra lehetnek szétosztva.

- *PHP:* Egy másik népszerű rendszer a hívásszintű felület megvalósítására a PHP. Ezt a nyelvet a HTML-dokumentumokba ágyazva találhatjuk meg. Ez a nyelv lehetővé teszi a weblapok és adatbázisok közötti kommunikációt.

- *Kliens-szerverrendszerek:* A szabvány szerint egy SQL-kliens hozzákapcsolódik egy SQL-szerverhez, kettejük között létrehozva egy kapcsolatot, valamint egy munkafázist (műveletek sorozatát). Egy munkafázis alatt általában egy modul kerül végrehajtásra, és a végrehajtás alatt álló modult SQL-ágensnek nevezzük.

- *Adatbázis-környezet:* Egy SQL-adatbázis-kezelő telepítésekor kialakul egy új SQL-környezet. Egy környezeten belül az egyes adatbáziselemek (pl. a relációk) csoportosítva lesznek (adatbázis)sémákba, katalógusokba és klaszterekbe. A katalógus sémák gyűjteménye, a klaszter pedig az elemeknek az a legbővebb gyűjteménye, amelyet még egy felhasználó elérhet.

- *A típuseltérés problémája:* Az SQL adatmodellje jelentősen különbözik a hagyományos programozási nyelvek által támogatott adatmodellektől. Ezért az SQL és a befogadó nyelven írt programok közötti adatcsere olyan közös elérésű (osztott) változókon keresztül történhet, amelyek a program SQL nyelvű részeiben a sorok komponenseinek felelnek meg.

- *Beágyazott SQL:* Az SQL-lekérdezések és -módosítások kifejezésére egy általános lekérdező felület használatánál gyakran hatékonyabb, ha hagyományos nyelven írunk programokat, és azokba ágyazzuk be az SQL-utasításokat. Egy előfordító fordítja le a beágyazott SQL-utasításokat a befogadó nyelv megfelelő függvényhívásaira.

- *Kurzorok:* A kurzor egy olyan SQL-változó, amely egy reláció egy sorára hivatkozik. Az SQL és a befogadó nyelven megírt program közötti adatcsere úgy történik, hogy a kurzorral egyenként végigmegyünk egy reláció sorain, az egyes sorok tartalmát beolvassuk közös elérésű változókba, és a befogadó nyelven megírt programmal feldolgozzuk azokat.

- *Dinamikus SQL:* Az SQL-utasításoknak befogadó nyelvű programba való ágyazása helyett lehetőség van arra, hogy a nyelv karakterlánc-változóiban hozzunk létre SQL-utasításokat. Ezeket az SQL-rendszer fogja értelmezni és végrehajtani.

- *Tartósan tárolt modulok:* Az adatbázisséma részeként létrehozhatunk eljárásokat és függvényeket tartalmazó gyűjteményeket. Ezeket egy speciális nyelven írjuk meg, amelyben megtalálhatók az ismert vezérlési szerkezetek és az SQL-utasítások is.

- *Hívásszintű felület:* Van egy szabványos függvénykönyvtár, amelyet SQL/CLI-nek vagy ODBC-nek hívunk, amelyet bármely C-programhoz hozzá lehet szerkeszteni. Ezek a függvények a beágyazott SQL-hez hasonló lehetőségeket nyújtanak, de nincs szükség a használatukhoz előfeldolgozóra.

- *JDBC:* A Java-adatbázis-kapcsolat (Java Database Connectivity) hasonló a CLI-hez, egy Java-osztályok kollekciója, amelynek a segítségével csatlakozhatunk egy adatbázishoz.

10.8. Összefoglalás

- *Jogosultságok:* Biztonsági követelmények teljesítése érdekében egy SQL-rendszer sokféle jogosultságot biztosít az egyes adatbáziselemeken, mint például: a relációk lekérdezési (olvasási), beszúrási, törlési vagy módosítási jogosultságai, valamint megszorításokban a relációnak más relációkból való hivatkozásának jogosultsága, és a trigger létrehozásának a joga.

- *Engedélyezési diagramok:* Az egyes jogosultságokat azok tulajdonosai adhatják tovább egy másik felhasználónak vagy az általános felhasználónak (`PUBLIC`). Továbbadható az engedélyezési képesség is, és az így megszerzett jogosultságokat azok megszerzői tetszésük szerint átruházhatják másokra is. A jogosultságok akár vissza is vonhatók. Az engedélyezési diagram hasznos szemléltető eszköz. Segítségével könnyen számon tartható az, hogy egy adott adatelemre vonatkozóan ki milyen jogosultságokkal rendelkezik, és kitől szerezte ezeket a jogosultságokat.

- *Rekurzív lekérdezések SQL-ben:* Az SQL-ben a relációk rekurzívan is megadhatóak, azaz saját környezetükkel. Sőt több reláció is definiálható kölcsönösen rekurzívan.

- *Monotonitás:* Az SQL-rekurzióban szereplő negációknak és összesítéseknek monotonnak kell lenni, azaz egy sor hozzáadása egy relációhoz nem eredményezheti sorok törlését egy másik relációból (önmagát is beleértve). Vagy másként, egy relációt nem szabad se direkt módon, se indirekt módon (vagyis saját negáltjával vagy saját összesítésével) önmagával definiálni.

- *Objektumrelációs modell:* Egy választási lehetőség az ODL-hez hasonló tiszta objektumorientált adatbázismodellel szemben a relációs adatmodell kiterjesztése lehet a legfőbb objektumorientált lehetőségekkel. Ezek a kiterjesztések tartalmazzák a beágyazott relációkat, azaz egy reláció attribútumainak összetett típusát (a relációt is mint típust). Egyéb kiterjesztések eljárások megadását is megengedik egy típusra, illetve azt is lehetővé teszik, hogy egy sorból egy másikra tudjunk hivatkozni hivatkozási típus használatával.

- *Felhasználói típusok SQL-ben:* Az SQL objektumrelációs adottságai az UDT (vagy *user-defined type*) körül forognak. Ezeket a típusokat megadhatjuk attribútumaik és más, a tábladeklarációknál is használt információk listájával. Sőt metódusokat is deklarálhatunk az UDT-khez.

- *UDT-típussal rendelkező relációk:* Egy reláció attribútumainak megadása helyett egy relációt deklarálhatunk úgy, hogy egy UDT-típussal rendelkezik. Ebben az esetben a sorai csak egy komponensből állnak, és ez a komponens egy UDT-objektum lesz.

- *Hivatkozási típus:* Egy attribútum típusa lehet egy hivatkozás egy UDT-re. Az ilyen attribútumok tulajdonképpen a szóban forgó UDT-objektumra mutató pointerek.

- *Objektumazonosítás UDT-ben:* Egy UDT-típussal rendelkező reláció létrehozásánál egy olyan attribútumot is deklarálunk, amely a sorok OID-ját (objektumazonosítóját) szolgáltatja. Ez a komponens egy hivatkozás a hozzá tartozó sorra. Ehhez az „OID" oszlophoz, az objektumorientált rendszerekkel ellentétben, a felhasználó is hozzáférhet, habár ez az információ nem túl sokatmondó.

- *Egy UDT komponenseinek elérése:* Az SQL megfigyelő- és változtató függvényeket nyújt egy UDT minden egyes komponenséhez. Ezen függvények visszaadnak, illetve megváltoztatnak egy megfelelő attribútumértéket, amikor meghívjuk őket a hozzájuk tartozó UDT-objektumra.

- *Az UDT rendezési függvényei:* Az objektumok összehasonlíthatósága miatt (az olyan SQL-műveletek esetén, mint a `DISTINCT`, `GROUP BY` vagy az `ORDER BY`) kellenek olyan UDT-eszközök, amelyekkel olyan függvényt tudunk megvalósítani, amellyel le tudjuk írni, hogy a szóban forgó objektumok megegyeznek-e vagy az egyik nagyobb-e a másiknál.

- *OLAP:* Az online analitikus feldolgozás olyan összetett lekérdezéseket jelent, amelyek az egész adatbázist vagy annak egy nagy részét foglalják le ugyanabban az időben. Gyakran egy elkülönített adatbázist, vagyis adattárházat, hoznak létre az ilyen lekérdezések futtatására, miközben az aktuális adatbázist csak rövid távú tranzakciók futtatására használják. (OLTP – On-Line Transaction Processing – Online Tranzakciós Feldolgozás)

- *ROLAP és MOLAP:* Ha OLAP-alkalmazás céljából építünk adattárházat, gyakran hasznos, ha az adatra egy kocka képében gondolunk, ahol a kocka dimenziói mentén az adatot más és más megvilágításban látjuk. Az

  olyan rendszer, amely ezt az adatmodellt támogatja, vagy relációs szempontból tekint a kockára (ROLAP- vagy relációs OLAP-rendszerek), vagy az adatkockára specializálódik (MOLAP- vagy többdimenziós OLAP-rendszerek).

- *Csillagsémák:* Egy csillagséma esetén minden adatelemet (például egy árucikk eladását) egy relációban, a ténytáblában tárolunk, a dimenziók különböző értékeinek az értelmezéséhez (például az 1234 azonosítójú cikk milyen típusú termék?) szükséges információkat pedig az egyes dimenziókhoz tartozó dimenziótáblákban.
- *Kockaművelet:* Egy speciális művelet, a kocka a ténytábla dimenzióinak lehetséges részhalmazai mentén előösszesítést végez. Az így kibővített táblázat kicsit több helyet foglal, mint az eredeti ténytábla, de segítségével nagymértékben csökkenthető az OLAP-lekérdezések futási ideje.
- *Adatkockák SQL-ben:* A lekérdezésünk eredményét egy adatkockába is rakhatjuk a csoportosítási utasítást követő `WITH CUBE` segítségével. Az adatkocka egy részét is előállíthatjuk, ha az előző helyett `WITH ROLLUP` kifejezést írunk.

11.5. Összefoglalás

- *Félig-strukturált adat:* Ebben a modellben az adatot egy gráf reprezentálja. A csomópontok objektumszerűek vagy attribútumok értékei, és címkézett élek kapcsolják össze őket mind az attribútumértékekkel, mind a más, velük relációban álló objektumokkal.

- *XML:* A kiterjesztett leíró nyelv (Extensible Markup Language) a World-Wide-Web Consortium által létrehozott szabvány, amely képes a félig-strukturált adatot lineárisan ábrázolni.

- *XML-elem:* Az elemek egy nyitó jelölővel kezdődnek (például `<valami>`) és egy záró jelölővel fejeződnek be (például `</valami>`) és minden ami ezek között van. Előfordulhat benne bármilyen szöveg, illetve beágyazott elemek, tetszőleges mélységben.

- *XML-attribútum:* A jelölőkben megjelenhetnek tulajdonságérték-párok. Ezek az attribútumok további információkat szolgáltatnak azokról az elemekről, amelyekben elhelyezzük őket.

- *Dokumentumtípus-definíció (Document Type Definition):* A DTD egy egyszerű nyelvtan XML-elemek és attribútumok definiálására, így egy körülbelüli sémát ad azon XML-dokumentumoknak, amelyek használják az adott DTD-t. Egy elemről megadhatjuk, hogy gyermekelemek sorozatát tartalmazza, továbbá előírhatjuk, hogy ezek az elemek pontosan egyszer, többször, maximum egyszer vagy tetszőlegesen sokszor fordulhatnak elő a szülőben. Egy elemhez továbbá definiálhatunk kötelező vagy opcionális attribútumokat is.

- *Azonosítók és hivatkozások DTD-ben:* Annak érdekében, hogy a nem faszerkezetű gráfokat is tudjuk ábrázolni, a DTD lehetőséget biztosít `ID` és `IDREF(S)` típusú attribútumok definiálására is. Egy elemnek adhatunk egyedi azonosítót, amelyre hivatkozhatunk egy másik elemből, amellyel össze szeretnénk kapcsolni.

- *XML-séma (XML Schema):* Ez egy újabb módszer arra, hogy egyes XML-dokumentumokhoz sémát definiáljunk. Az XML-sémadokumentumok maguk is XML-ben íródnak, de egy olyan névtér jelölőit használják, amely a World-Wide-Web Consortium szabványa.

- *Egyszerű típusok az XML-sémában:* Az XML-sémában definiálva vannak egyszerű típusok, mint például az egész vagy a szövegtípus. További egyszerű típusokat hozhatunk létre úgy, hogy e típusok értékkészletét korlátozzuk tartomány megadásával vagy a lehetséges értékek felsorolásával.

- *Összetett típusok az XML-sémában:* Összetett típusokat definiálhatunk elemek sorozataként, melynek minden tagjáról megadhatjuk az előfordulásainak minimális, illetve maximális számát.

- *Kulcsok és idegen kulcsok az XML-sémában:* Elemek vagy attribútumok egy halmazáról megadhatjuk, hogy bizonyos elemeken belül egyediek legyenek. Más elem- vagy attribútumhalmazról pedig megadhatjuk, hogy értékei csak olyanok lehetnek, amelyek kulcsként előfordulnak bizonyos más típusú elemekben.

12.4. Összefoglalás

- *XPath:* Ez a nyelv egy egyszerű módja az XML-adatokkal kapcsolatos lekérdezések kifejezésére. Utak írhatók le vele a jelölők szekvenciájával a dokumentum gyökerétől indulva. Az út egy attribútumban és egy elemben is végződhet.

- *Az XPath-adatmodell:* Minden XPath-érték tételek szekvenciája. Egy tétel vagy egy egyszerű érték vagy egy elem. Egy elemhez egy nyitó XML-jelölő, és a neki megfelelő záró jelölő tartozik, továbbá minden, ami e kettő között található.

- *Tengelyek:* Ahelyett, hogy lefelé haladnánk a fában, egy másik tengelyt is követhetünk, amely ugrásokat tartalmazhat valamely leszármazottra, szülőre vagy testvérre.

- *XPath-feltételek:* Az út bármely lépése olyan feltételhez köthető, amely egy logikai értékű kifejezés. Ez a kifejezés szögletes zárójelek között jelenik meg.
- *XQuery:* Ez a nyelv az XML-dokumentumok lekérdező nyelvének egy elterjedt formája. Az XPath-adatmodellt használja. Az XQuery egy funkcionális nyelv.
- *FLWR-kifejezések:* Az XQuery számos lekérdezése tartalmaz let, for, where és return záradékot. A „let" ideiglenes változók definícióit vezeti be, a „for" ciklusokat készít, a „where" teszteli a feltételeket, a „return" pedig definiálja a lekérdezés eredményét.
- *Összehasonlító operátorok XQueryben és XPath-ben:* A szokásos összehasonlító operátorok, amilyen a <, alkalmazható az adatok szekvenciájára, és „létezik olyan" jelentéssel bír. Ezek az összehasonlítások igazak, ha a megadott reláció fennáll a tételek bármely olyan párjára, amelyek az egyes listákból származnak. Annak biztosítására, hogy csak egy-egy tétel kerüljön összehasonlításra, a betűkkel jelölt operátorokat kell alkalmaznunk, amilyen az `lt` a „kisebb mint" jelentéssel.
- *Egyéb XQuery-kifejezések:* Az XQueryben számos olyan művelet van, amely hasonlít az SQL-ben lévőkre. Ezek az operátorok magukban foglalják az egzisztenciális és univerzális kvantorokat, az összesítéseket, a duplikátumok megszüntetését és az eredmény rendezését.
- *XSLT:* Ezt a nyelvet az XML-dokumentumok transzformációjára tervezték, ennek ellenére úgy is használható, mint egy lekérdező nyelv. Egy ezen a nyelven írt „program" alakja olyan, mint egy XML-dokumentumé speciális névtérrel, amely biztosítja számunkra a transzformációk leírására szolgáló jelölők használatát.
- *Sablonok:* Az XSLT szíve a sablon, amely a bemeneti dokumentum egyes elemeire illeszkedik. A sablon leírja a kimeneti szöveget, és képes arra, hogy értékeket kapjon a bementi dokumentumból a kimenetben való elhelyezés céljából. Egy sablon meghívhat újabb sablonokat az adott elem gyerekeire való rekurzív alkalmazással.
- *XSLT-programkonstrukciók:* Egy sablon tartalmazhat XSLT-szerkezeteket, amelyek úgy viselkednek, mint egy iteratív programozási nyelv. Ezek a szerkezetek a for ciklusok és az if-szerkezetek.