**Relációs adatmodell**

Az adatmodell a valóság objektumait (egyedeit), ezek tulajdonságait és a köztük lévő kapcsolatokat ábrázolja. Az adatmodellek több különböző fajtája ismert: hierarchikus, hálós, relációs, stb. A relációs adatmodell az egyik legelterjedtebb adatmodell. A relációs adatmodell lényege, hogy az egyedeket, tulajdonságokat és kapcsolatokat egyaránt táblázatok, úgynevezett adattáblák (relációk) segítségével kezeli.

Népszerőségét annak köszönheti, hogy nagyon egyszerő deklaratív nyelvvel rendelkezik az adatok kezelésére, illetve lekérdezésére. Értékorientált, ez ahhoz vezet, hogy a relációkon értelmezett műveletek eredményei szintén relációk.

Reláció: adott n darab halmaz, ezeket jelöljük D1,…Di ,…Dn-nel. Reláción e halmazok direkt szorzatának a részhalmazát értjük és R-rel jelöljük. Az R relációt tehát felírhatjuk a következő formában: R={(a1,…ai ,…an), ai∈Di , i=1,…,n}.

A tábla (reláció) tulajdonságai:

1. Minden táblának (relációnak) egyedi neve van. pl. DIAK, DOLGOZO
2. A tábla oszlopainak (attribútumainak) neve van, és ezek a nevek egy táblán belül egyediek.
3. Az oszlopok száma (a reláció foka) az adott táblában állandó.
4. Az oszlopok sorrendje tetszőleges, azonosításuk a nevük alapján és nem a helyük alapján történik.
5. Az egyes oszlopok csak meghatározott értékeket vehetnek fel egy adott értéktartományból (Di halmazból), ezt az értéktartományt domain-nek nevezzük. Minden sor minden oszlopában egy és csakis egy elemi érték szerepelhet, és egy értéknek szerepelnie is kell. (A gyakorlatban megengedett, hogy egy attribútum valamelyik előfordulásán ne vegyen fel értéket a domainből, ekkor ez az előfordulás definiálatlan lesz és NULL értéknek nevezzük.)
6. A sorok sorrendje tetszőleges, azonosításuk az értékük, tartalmuk alapján és nem a sorszámukkal történik. A sorok száma változhat, adott pillanatban a sorok száma megadja a reláció számosságát.
7. A táblában minden sor különböző. Nem létezhet két teljesen megegyező sor a táblában, mint ahogy nem létezik két azonos elem a relációban, mint halmazban sem.
8. A 7. tulajdonságból következik, hogy minden táblában (relációban) létezik az oszlopoknak egy olyan halmaza, mely a tábla (reláció) bármely sorát egyértelműen meghatározza. Ezt az oszlop kombinációt hívjuk a tábla (reláció) kulcsának.

Egy helyes relációs adatbázis sémát több módszerrel is kaphatunk:

1. az egyed/kapcsolat diagramot átírjuk relációsémákká,
2. Feladat specifikációból (feladat leírás) kiindulva normalizálással
3. UML, stb.

Relációk kapcsolata alkot egy adatbázist. Elsődleges Kulcs (PK). Kapcsoló kulcs (Külső kulcs -FK).

Kapcsolat Típusok 1:1, 1:N, N:N.

**Funkcionális függőségek**

**Funkcionális függőség**: Egy vagy több adat konkrét értékéből más adatok egyértelműen következnek: Személyi\_szám -> Név (fordítva nem),

Autó\_rendszám -> Tipus, Tulajdonos

Hely, Időpont -> Hőmérséklet

**Funkcionális függőség**: adott az R reláció, azt mondjuk, hogy Y értelmezési tartománya funkcionálisan függ az X értelmezési tartományától (X → Y) akkor és csak akkor, ha X minden értéke egyértelműen meghatározza Y-t. Ez a meghatározás nem csak az aktuális előfordulásokra, hanem mindig érvényes.

Az elsődleges kulcstól minden értelmezési tartomány funkcionálisan függ.

SzállításiInformációk(SzállID, SzállNév, SzállCím, ÁruID, ÁruNév, MértEgys, Ár)

SzállID  {SzállNév,SzállCím}

ÁruID  {ÁruNév, MértEgys}

*Parciális függőség*: Ha *C* egy kulcsa az *R* relációnak, az *Y* attribútumhalmaz valódi részhalmaza a *C*-nek () és *B* egy attribútum, mely nem része az *Y*-nak (), akkor az -t egy parciális függőség. (B függ a kulcs egy részétől.)

###### **példa:** {SzállID, ÁruID} a kulcs, tehát {SzállID, ÁruID}SzállNév,

mivel a kulcs funkcionálisan meghatároz minden más attribútumot, de a SzállNév függ a kulcs egy részétől is. SzállID  {SzállNév} Parciális függőség

*Tranzitív függőség*: Legyen  egy attribútumhalmaz és *B* egy attribútum, mely nem része *Y*-nak (). Egy  funkcionális függőség tranzitív függőség, ha *Y* nem szuperkulcs *R* relációban és nem is valódi részhalmaza *R* egy kulcsának.

Rendelések (RendelésSzám, Dátum, VevőID, VevőNév, Részletek)

RendelésSzám VevőID.

VevőIDVevőNév. Tehát a VevőNév tranzitív függőséggel függ a RendelésSzámtól.

Funkcionális függőségeket akkor határozunk meg amikor Normalizálás segítségével akarunk megtervezni egy adatbázist. Egy relációban felmerülő problémák kiküszöbölésére:

*Problémák*:

Azokat a problémákat, amelyek akkor jelennek meg, amikor túl sok információt probálunk egyetlen relációba belegyömöszölni, *anomáliá*nak nevezzük. Az anomáliáknak alapvető fajtái a következők:

* *Redundancia*: Az információk feleslegesen ismétlődnek több sorban, mint például a SzállításiInformációk reláció esetében a szállító címe ismétlődik.
* *Módosítási problémák*: Megváltoztatjuk az egyik sorban tárolt információt, miközben ugyanaz az információ változatlan marad egy másik sorban. Például, ha a szállító címe változik, de csak egy sorban változtatjuk meg, nem tudjuk, melyik a jó cím. Jó tervezéssel elkerülhetjük azt, hogy ilyen hibák felmerüljenek.
* *Törlési problémák*: Ha az értékek halmaza üres halmazzá válik, akkor ennek mellékhatásaként más információt is elveszthetünk. Ha például töröljük a Rolicom által szállított összes árut, az utolsó sor törlésével elveszítjük a cég címét is.
* *Illesztési problémák*: Ha hozzáilleszteni akarunk egy szállítót, amely nem szállít egy árut sem, a szállító címét kitöltjük úgy, hogy az áruhoz „null” értékeket viszünk be, melyet majd utólag ki kell törölni, ha el nem felejtjük.

**Normalizálás. Normál formák(1NF, 2NF, 3NF)**

Az (az előbbi oldalon felsorolt) anomáliák megszüntetésének elfogadott útja a relációk *felbontása* (*dekompozíció-*ja). *R* felbontása egyrészt azt jelenti, hogy *R* attribútumait szétosztjuk úgy, hogy ezáltal két új reláció sémáját alakítjuk ki belőlük.

**Normalizálás:** Logikai adatbázis megtervezésére szolgáló módszer. Táblázat szétbontó relációs műveletek sorozata, eredményeképpen egymással kapcsolatban álló, az eredetinél kisebb tárolási igényű relációkat kapunk.

Célja: anomáliák, redundanciák kiküszöbölése.

Normálformák: 1NF, 2NF, 3NF, BCNF, 4NF, 5NF

1. normálforma (1NF): az R reláció első normálformában van, ha a relációban szereplő minden érték elemi, azaz minden attribútum minden sorban csak egy értéket vesz fel az értelmezési tartományából.

2. normálforma (2NF): az R reláció második normálformában van akkor és csak akkor, ha 1NFben van és minden olyan attribútuma, mely nem része az elsődleges kulcsnak, funkcionálisan teljesen függ az elsődleges kulcstól.

3. normálforma (3NF): az R reláció harmadik normálformában van akkor és csak akkor, ha 2NFben van és minden olyan attribútuma, mely nem része az elsődleges kulcsnak, funkcionálisan teljesen függ az elsődleges kulcstól és **csak attól**.

Másik definició: Az R reláció harmadik normálformában (3NF) van akkor és csak akkor, ha 2NF-ben van és nem tartalmaz tranzitív függőségeket.

**1NF-ra alakítás**

Az R relációt felbontjuk két relációra, ahol: C Kulcs, I Ismétlődő, J a többi:

S(C, I), T(C, J),

**2NF-ra alakítás**

Az R relációt felbontjuk két relációra, ahol D B, A minden

T(D, B) és S(A-B)

**3NF-ra alakítás**

Az R relációt felbontjuk két relációra, ahol D B, A minden

T(D, B) és S(A-B)

**Az SQL lekérdezőnyelv -** Structured Query Language

Halmazorientált. Nincsenek benne ciklusok, feltételes elágazások, stb. ez már PL/SQL.

**DDL** adatdefiníciós nyelv (Data DefinitionLanguage) adatbázis és adatbázis objektumok létrehozása: CREATE, ALTER, DROP, RENAME

**DML** adatmanipulációs nyelv (Data Manipulation Language) adatok karbantartása (bevitele, módosítása, törlése), lekérdezése: INSERT, UPDATE, DELETE, SELECT

**DCL** adatvezérlő nyelv (Data Control Language) tranzakció kezelése: COMMIT, ROLLBACK, SAVEPOINT. Adatvédelem, felhasználói hozzáférés szabályozása: GRANT, REVOKE

SELECT ... oszlopok kiválasztása (projekció) (6)

FROM ... táblanév(-ek) (Descartes szorzat) (1)

[WHERE ... ] sorok kiválasztása (szelekció) (2)

[CONNECT BY ... [START WITH ...]] hierarchia kezelés (3)

[GROUP BY ... ] csoportosítás (4)

[HAVING ... ] csoportok közötti válogatás (5)

[{UNION [ALL] | INTERSECT | MINUS} alselect] halmazműveletek (7)

[ORDER BY ... ] eredménysorok rendezése (8)

[FOR UPDATE OF …] kiválasztott sorok zárolása update idejére (9)

NULL érték, =, !=, <>, , <=, >=, IN, BETWEEN, LIKE, IS, ANY, ALL, UNION, INTERSET, UNION, -, EXISTS, AVG, SUM, MIN, MAX, STDDEV, VARIANCE, COUNT, SYSDATE,

**GROUP BY**: az oszlopkifejezés alapján csoportosítja a leválogatott sorokat és egyetlen, összesített információt tartalmazó sort állít elő minden csoporthoz

**HAVING**: megadható a csoportok közötti válogatás feltétele.

Táblák összekapcsolása: Equi Join, Natural join, Cross Join, Self Join, Outer Join

Allekérdezések, Korrelált Allekérdezések

CONSTRAINT, DATA DICTIONARY, stb….