Relációs adatmodell

Gugolya László

Bevezetés

- A relációs adatmodellt és az ezt használó adatbázis-koncepciót Edgar Frank Ted Codd dolgozta ki az IBM munkatársaként 1969-ben.
- 1970-ben megjelent cikkében ismertette szélesebb körben.
- A relációs adatbázisok megvalósítása erőforrásigényes, így jó ötletnek gondolták, de a megvalósíthatóságában kételkedtek
- Az eszközök nagymértékű fejlődése miatt elhárultak az akadályok a megvalósítás elöl
- Maga az IBM sem fűzött hozzá komolyabb reményeket, hat év alatt hozták létre az első modern adatbázis-kezelőt, a System-R (ma DB/2) kódnevű szoftvert.

Bevezetés

- CIA (Larry Ellison, Bob Miner) egy grandiózus projektbe fogott bele, amelynek célja egy olyan adattár létrehozása volt, amely a CIA minden felmerülő kérdését gyorsan, hatékonyan, és aránylag olcsón megválaszolja.
- A projekt az orákulum nevet kapta, angolul: Oracle.
- A projekt pénz hiányában 1976-ban nem folytatódott, de Ellison és Miner továbbvitte (RSI, 1982-től Oracle Corp.)→1977-ben létrejött az első Oracle.
- A lekérdező nyelv problémáját azonban nem ők, hanem a System-R fejlesztői oldották meg. Mivel az IBM akkor még nem különösebben érdeklődvén az adatbázisok iránt, nem tiltotta le az adatbázisokkal kapcsolatos információk kiadását, így az RSI nagyon könnyen hozzájutott az SEQUEL nyelv (ma SQL) leírásához.

Bevezetés

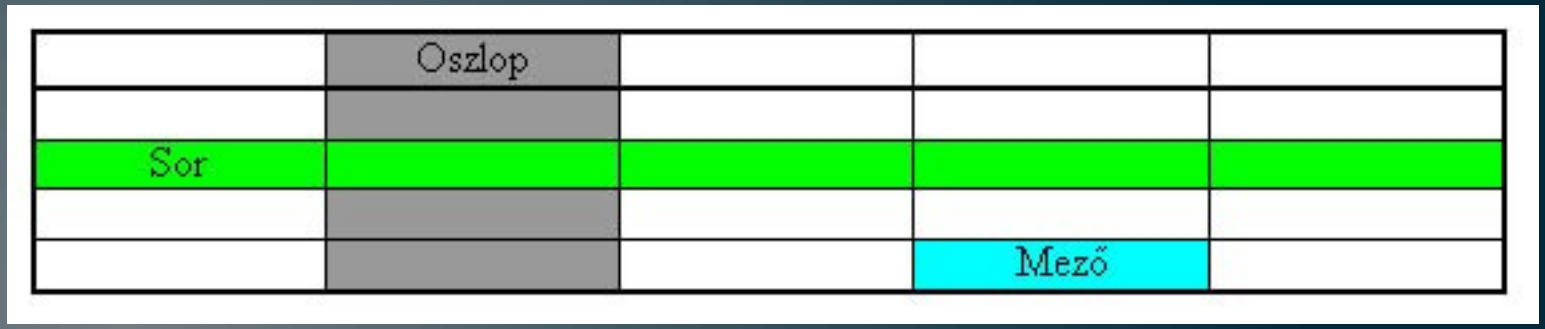
- Fontos: a relációs adatmodell a logikai adatbázis (vagy fogalmi adatbázis) kereteit határozza meg, nem foglalkozik azzal a problémával, hogy adatokat ténylegesen hogyan kell tárolni a háttértáron, hogyan kell módosítani, illetve a memóriába betölteni a felhasználó által igényelt adatokat.

Relációk és a velük kapcsolatos alapfogalmak

Fogalmak

- A reláció nem más mint egy táblázat, a táblázat soraiban tárolt adatokkal együtt.
- A relációs adatbázis pedig relációk és **csak** relációk összessége.
- Az egyes relációkat egyedi névvel látjuk el.
- A relációk oszlopaiban azonos mennyiségre vonatkozó adatok jelennek meg. Az oszlopok névvel rendelkeznek, melyeknek a reláción belül egyedieknek kell lenniük, de más relációk tartalmazhatnak azonos nevű oszlopokat.
- A reláció soraiban tároljuk a logikailag összetartozó adatokat. A reláció sorainak sorrendje közömbös, de nem tartalmazhat két azonos adatokkal kitöltött sort.
- Egy sor és oszlop metszésében található táblázat elemet mezőnek nevezzük, a mezők tartalmazzák az adatokat. A mezőkben oszloponként különböző típusú (numerikus, szöveges stb.) mennyiségek tárolhatók.
- A reláció helyett sokszor a tábla vagy táblázat, a sor helyett a rekord, az oszlop helyett pedig az attribútum elnevezés is használatos.

Példa



Személy

Személyi szám	Név	Város	Foglalkozás
1 650410 1256	Kiss lászló	Győr	kőműves
2 781117 0131	Nagy Ágnes	Szeged	tanuló
1 610105 1167	Kiss László	Budapest	lakatos

Relációs adatmodell - táblázat

- A logikai szerkezet és a tárolás független.
- Az egyedet egy táblázattal adjuk meg, táblázat oszlopai a tulajdonságok, a sorai az egyed értékei.
- Az adatokat táblákban (table) tároljuk.
  - Minden táblának van egy egyértelmű, azt azonosító neve.
  - A tábláknak sorai (row) és oszlopai (column) vannak.
    Minden sor, oszlop kereszteződésben adat (érték) helyezkedik el.
  - Az adatmodell szintjén a táblákat egyedtípusnak
    (entity-type vagy entity) vagy relációnak (relation) nevezik.

Relációs adatmodell - oszlopok

- A tábla minden oszlopában azonos jellegű (homogén) adatokat tárolunk.
- Minden oszlopnak van egy azt azonosító (az adott táblán belül egyedi) neve, típusa és mérete.
- Az oszlop az adatmodell szintjén az adott egyed egy tulajdonsága (entity-attribute).
- Az oszlopok sorrendje nem befolyásolja az adatmodellt.

Relációs adatmodell - sorok

- A sorokat az adatmodell szintjén az adott egyedtípus egy-egy előfordulásának (entity-instance) is nevezzük.
- Minden sor különböző.
- A táblában a sorok sorrendje közömbös.
- A táblák közötti kapcsolatokat közös tulajdonságokkal, indexek használatával valósítjuk meg.

Elnevezések

Relációs modell	Logikai modell	Fizikai megjelenési forma
tábla (table)	egyedtípus (entity)	fájl(ok) (file)
oszlop (column)	tulajdonság (attribute)	mező (field)
sor (row)	előfordulás (instance)	rekord (record)

Példa adattáblára: könyvek nyilvántartása

Attribútum: egy tulajdonság, amelyhez név és értéktartomány tartozik.
Jelölés: a *Z* attribútum értéktartománya dom(*Z*).
Példa: könyvszám.

Értéktartomány megadása: típus, hossz, esetleg korlátozó feltételek.

Példa: dom(könyvszám) = 4-jegyű decimális számok halmaza, ahol az első jegy 1-től 6-ig változhat, és a könyvet tartalmazó polcot azonosítja.

A relációs modellben az értéktartomány csak atomi értékekből állhat!

Relációséma: egy attribútumhalmaz, amelyhez név tartozik.

Példa: Könyv (könyvszám, szerző, cím)

dom(könyvszám) = 4-jegyű decimális számok halmaza, ahol 1000 < könyvszám < 7000.

dom(szerző) = 20 karakteres stringek halmaza.

dom(cím) = max. 80 hosszú stringek halmaza.

Jelölések: $R(A_1,...,A_n)$.

$R(A)$, ahol $A = \{A_1,...,A_n\}$.

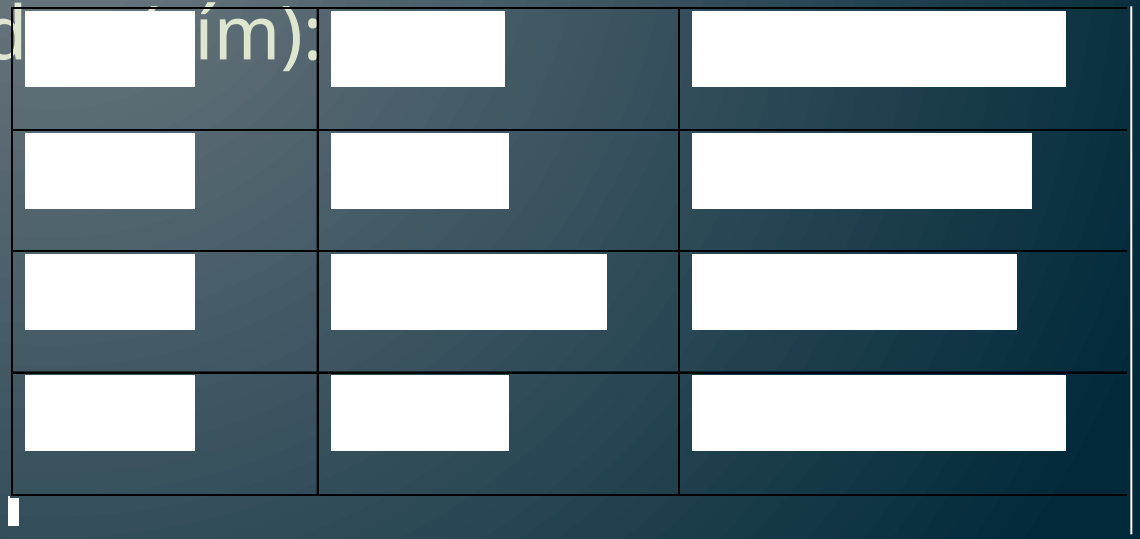
$R.A_1$ az R séma A_1 attribútuma.

Reláció, vagy adattábla az $R(A_1,...,A_n)$ relációs séma felett: $T \subseteq \text{dom}(A_1) \times ... \times \text{dom}(A_n)$.

T elemei $(a_1,...,a_n)$ alakúak, ahol $a_i \in \text{dom}(A_i)$ $(i=1,..., n)$.

Példa: a Könyv (könyvszám, szerző, cím) relációséma feletti adattábla:

$T \subseteq$ dom(könyvszám) x dom(szerző) x dom(cím):



Miért hívják relációnak az adattáblát?
Magyarázat: a matematikai relációfogalom

Z: természetes számok halmaza

Z* x *Z: az összes (*a, b*) párok halmaza

Relációjel: pl. $\subseteq$

A "kisebb" reláció definíciója:
$K \subseteq Z \times Z$, ahol K azon (*a, b*) párok halmaza, amelyekre $a < b$.

Példa: $(2, 3) \in K$, de $(5, 2) \notin K$

Általánosítás: $K \subseteq A \times B \times C$, ahol K azon (*a, b, c*) hármasok halmaza, amelyekre valamilyen feltétel teljesül.

Megjegyzések

- Az adattábla ***sorok halmaza***, ezért a relációs modellben a tábla ***minden sora különböző***, és a soroknak ***nincs kitüntetett sorrendje.***
- Elvileg a tábla ***oszlopainak sincs kitüntetett sorrendje***. Ehhez módosított definíció:
  $D = \mathrm{dom}(A_1) \cup \ldots \cup \mathrm{dom}(A_n)$,
  $T = \{t_1,\ldots,t_k\}$, ahol $t_i: A \rightarrow D$ úgy, hogy
  $t_i(A_j) \in \mathrm{dom}(A_j)$ minden j-re.
- Az RDBMS-ek általában megengednek azonos sorokat is, és a soroknak ill. oszlopoknak szükségképpen van egy tárolási sorrendje.

Elnevezések

- *Relációséma:* a tábla felépítési sémája.
- *Reláció* vagy *adattábla* vagy *tábla:* az adatokat tartalmazza.
- *Sor, oszlop.*
- *Rekord:* a tábla egy sora.
- *Mező:* a séma egy attribútuma, vagy egy bejegyzés a táblába.
- *NULL:* definiálatlan mezőérték.
- *Relációs adatbázis:* táblák együttese.

Könyvtári adatbázis

Relációsémák:

Könyv (könyvszám, szerző, cím, olvasószám, kivétel)

Olvasó (olvasószám, név, lakcím)

A sémák között a *közös attribútum* (olvasószám) biztosítja a kapcsolatot.

A KÖNYV adattábla:

k.szám	*szerző*	*cím*	*olvasószám*	*kivétel*
1121	Sályi	Adatbázisok		
3655	Radó	Világatlasz	122	2006.07.12
2276	Karinthy	Így írtok ti		
1782	Jókai	Aranyember	355	2006.09.23

Az OLVASÓ adattábla:

olvasószám	*név*	*lakcím*
122	Kiss István	Szeged, Virág u. 10.
612	Nagy Ágnes	Szentes, Petőfi út 38.
355	Tóth András	Budapest, Jég u. 3.

Kulcsok

Szuperkulcs (superkey): olyan attribútumhalmaz, amely egyértelműen azonosítja a tábla sorait.

Pontosabban:

Egy $R(A)$ relációsémában $K (\subseteq A)$ ***szuperkulcs***, ha bármely R feletti T tábla bármely két sora K-n különbözik.

Formálisan: bármely $t_i \in T$ és $t_j \in T$ esetén

$t_i \neq t_j \Rightarrow t_i(K) \neq t_j(K)$.

Kulcs (key):

K (⊆ *A*) *kulcs*, ha ***minimális*** szuperkulcs, vagyis egyetlen valódi részhalmaza sem szuperkulcs.

Példák: a **Könyv(könyvszám, szerző, cím)** sémában

- {könyvszám} szuperkulcs és kulcs is,
- {könyvszám, szerző} szuperkulcs de nem kulcs,
- {szerző} nem szuperkulcs (és nem kulcs),
- {szerző, cím} nem szuperkulcs (és nem kulcs).

Egyszerű kulcs: ha egyetlen attribútumból áll.
Összetett kulcs: ha több attribútumból áll.

Példák:

Könyvtárban:

Könyv (könyvszám, szerző, cím)
Egyszerű kulcs: {könyvszám}

Könyvesboltban:

Könyv (ISBN, szerző, cím, kiadásiév, kiadó, példány)
Egyszerű kulcs: {ISBN}
Összetett kulcs: {szerző, cím, kiadásiév}

Megjegyzések

- A teljes A attribútumhalmaz mindig szuperkulcs.
- A kulcs valójában egy ***feltétel előírása*** a relációsémára.
- A kulcs a séma tulajdonsága, nem a tábláé.
- Egy sémában több kulcs lehet.

Elsődleges kulcs (primary key):

Ha csak egy kulcs van, az lesz az elsődleges kulcs.

Ha több kulcs van, egyet önkényesen kiválasztunk.

Jele: aláhúzás.

Példák:

Könyv (ISBN, szerző, cím, kiadásiév, kiadó, példány)

Kulcsok: {ISBN}, {szerző, cím, kiadásiév}

Elsődleges kulcs: {ISBN}

Fuvar (gkvez, rendszám, indul, érkezik)

Kulcsok: {gkvez, indul}, {gkvez, érkezik},
{rendszám, indul}, {rendszám, érkezik}.

Elsődleges kulcs: {rendszám, indul}

Fontos különbség:

- ***Relációs modell:*** a tábla minden sora különböző, ezért *mindig van kulcs*.
- ***Konkrét RDBMS:*** ha azonos sorokat is megengedünk, akkor *nincs kulcs*!

Példa: Vásárlás (dátum, terméknév, mennyiség)

2011.09.04.	banán	4.0
2011.09.05.	alma	3.0
2011.09.05.	szilva	1.5
2011.09.05.	alma	3.0

Itt megengedhető, hogy ne legyen kulcs.

Külső kulcs (idegen kulcs, foreign key):

Egy relációséma attribútumainak valamely részhalmaza *külső kulcs*, ha egy másik séma elsődleges kulcsára hivatkozik.

Jelölés: dőlt betű, vagy a hivatkozott kulcsra mutató nyíl.

A külső kulcs értéke a hivatkozott táblában előforduló kulcsérték vagy NULL lehet. (Hivatkozási integritás)

A KÖNYV adattábla:

k.szám	*szerző*	*cím*	*olvasószám*	*kivétel*
1121	Sályi	Adatbázisok		
3655	Radó	Világatlasz	122	2010.07.12
2276	Karinthy	Így írtok ti		
1782	Jókai	Aranyember	355	2010.09.23

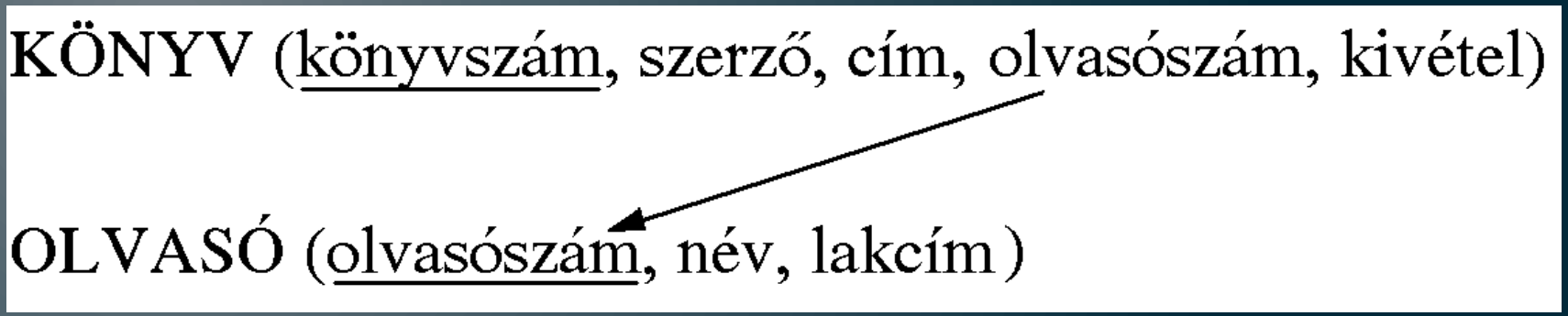
Az OLVASÓ adattábla:

olvasószám	*név*	*lakcím*
122	Kiss István	Szeged, Virág u. 10.
612	Nagy Ágnes	Szentes, Petőfi út 38.
355	Tóth András	Budapest, Jég u. 3.

Példa:

KÖNYV (könyvszám, szerző, cím, *olvasószám,* kivétel)

OLVASÓ (olvasószám, név, lakcím)



Megjegyzés: A külső kulcs valójában egy ***feltétel előírása*** a relációsémákra.

Relációs adatbázis séma

relációsémák + kulcs feltételek
(elsődleges kulcsok, külső kulcsok)

Példa: egészségügyi adatbázis

Beteg (betegId, betegnév, lakcím)

Orvos (orvosId, orvosnév, osztály, kórház)

Kezelés (kezelésId, megnevezés, kategória)

Ellátás (*betegId, orvosId, kezelésId*, dátum, költség)

Egy valós példa: helyiség nyilvántartás

Ingatlan (inszam, megnev)

Telek (*inszam*, sorszam, cim, jelleg, hrsz, tullap, tulajd, tulcim, tulhanyad, kezelo, kezcim, kezhanyad, berlo, bercim, berhanyad, megjegyzes)

Epulet (*inszam*, sorszam, epulet, epnev, vedett, rendelt, terulet, szintszam, nszter, bszter, nterfogat, terfogat, epitev, rekonev, felujev, felujkts, ertek, rajznev, fh, megjegyzes)

Szint (epulet, szint, sznev, rajzjel)

Helyiseg (*epulet, szint*, szam, hnev, *funkod*, belmag, term2, legm3, fh, *egyseg,* megjegyzes)

Funkcio (id, megnev, *foegys*, val, alnum, flg, nflg)

Egyseg (id, megnev, *foegys*, *tipus*, alnum, flg, nflg)

Tipus (tipus, megnev)

Megjegyzések a valós példához

- A feladatot a megrendelő specifikálja (például egy ingatlanhoz több telek tartozhat).
- Egyes tábláknál sok attribútum.
- Különböző táblák azonos nevű attribútumokat tartalmazhatnak (például Funkcio, Egyseg).
- Külső kulcs neve eltérhet a hivatkozott kulcstól (például Helyiseg.funkod → Funkcio.id).
- Ékezetes betűk kerülendők.

Indexelés

Index: kiegészítő adatstruktúra. Célja:

- rendezés,
- keresések gyorsítása (pl. külső kulcs).

Indexkulcs: valamely $L \subseteq A$ attribútumhalmaz. Megegyezhet a tényleges kulccsal, de bármi más is lehet.

„Indextábla”: Index (indexkulcs, rekordsorszám)

A KÖNYV adattábla

k.szám	*szerző*	*cím*	*olvasószám*	*kivétel*
1121	Sályi	Adatbázisok		
2276	Karinthy	Így írtok ti		
3655	Radó	Világatlasz	122	2006.07.12
1782	Jókai	Aranyember	355	2006.09.23

Szerző szerinti index

Szerző	*Sorszám*
Jókai	4
Karinthy	2
Radó	3
Sályi	1

Cím szerinti index

Cím	*Sorszám*
Adatbázisok	1
Aranyember	4
Így írtok ti	2
Világatlasz	3

Indexek használata - 1

Az indextáblákat általában ***B-fa struktúraként*** valósítják meg (B = balanced):

- Minden művelet után kiegyensúlyozott.
- Egy csomópont = egy lemezblokk.
- Egy csomópontnak akár 50-100 leszármazottja lehet.
- A keresés ritkán mélyebb 3 szintnél (gyakorlatilag konstans keresési idő).
- Gyökér a memóriában.

Indexek használata - 2

- ***Index létrehozása:*** minden rekordhoz (indexkulcs, rekordsorszám) létrehozása, fára fűzése.
- ***Adott indexkulcsú rekord keresése:*** logaritmikus időben. (A gyakorlatban max. 3 szint.)
- ***Rendezett listázás:*** B-fa bejárása, lineáris időben.
- ***Új rekord felvétele*** a tábla végére, index beszúrás a B-fába.
- ***Rekord törlés:*** táblából logikailag, B-fában

Megvalósítás SQL segítségével

SQL

- Az adatdefiniciós nyelv segítségével hozhatjuk létre illetve szüntethetjük meg a relációkat, az indexeket illetve a nézet táblázatokat.
- A nézet táblázat az adatbázisban fizikailag nem létező relációs műveletek (szelekció, projekció, összekapcsolás, halmazműveletek) segítségével létrehozott táblázat, mely a relációkhoz hasonlóan kezelhető.

SQL

- A relációk létrehozására a CREATE TABLE SQL utasítás szolgál, melynek általános alakja a következő
- CREATE TABLE reláció_név
  (attribútum_név adattípus [(szélesség)] [CONSTRAINT megszorítás_név] [oszlop_megszorítás],
  attribútum_név adattípus [(szélesség)] [CONSTRAINT megszorítás_név] [oszlop_megszorítás],
  …) [CONSTRAINT megszorítás_név] [tábla_megszorítás];

SQL

- A szélesség megadása el is maradhat. A reláció név és általában a nevek megadására a következő szabályok érvényesek:
- a névben csak az angol ABC betűi, a számjegyek és az _, #, $ karakterek szerepelhetnek
- a névnek betűvel kell kezdődnie
- a neveknek hatáskörükön belül egyedinek kell lennie (például nem lehet egy adatbázisban két azonos nevű reláció, egy relációban két azonos nevű attribútum, stb.)
- A nevek hossza korlátozott
- Az SQL az azonosítókban és a parancs szavakban általában nem tesz különbséget a kis és nagybetűk között.

- Az attribútumokra megadható adattípusok köre adatbáziskezelőnként változhat
- A legtöbb relációs adatbáziskezelőben használhatók:
- CHAR [(hossz)]
  megadott maximális hosszúságú karakterlánc, csak a karakterláncnak megfelelő hosszúságú területet foglalja el, szinonimája a VARCHAR. Maximum 255 karakterig.
- NUMBER [(szélesség, tizedes)]
  valós szám megadása, a szélesség mellett a tizedespont utáni jegyek száma is megadható, hasonlóan használható a FLOAT
- INTEGER
  egész típusú érték megadása, hozzá hasonló, de számábrázolási tartományában eltérő típus még a SMALLINT, szinonimája a DECIMAL
- DATE
  dátum megadása
- RAW
  a karakterhez hasonló típus, de az adatok értelmezésére nincs semmilyen feltételezés, így segítségükkel tetszőleges bináris adatokat tárolhatunk, például ábrákat is. Maximális hossza 255.
- LONG
  Maximum 64 KByte hosszú szöveg. Relációnként csak egy ilyen lehet, és csak korlátozott helyeken használható
- LONGRAW
  Mint a RAW, de 64 KByte hosszú adat.

Megszorítások

- Az attribútum típusát megadva, a reláció azon oszlopába más típusú adat nem vihető be, erről az adatbáziskezelő gondoskodik illetve szükség esetén figyelmeztet.
- Az opcionális oszlop megszorítás a következőket tartalmazhatja.
  - NULL az attribútum definíciójában arra utal, hogy az adat megadása nem kötelző, ez az alapértelmezés ezért a legritkább esetben írják ki.
  - NOT NULL az attribútum definíciójában arra utal, hogy az adat megadása kötelező, azaz nem vihető be olyan sor a relációban, ahol az így definiált adat nincs kitöltve.
  - PRIMARY KEY ez az oszlop a tábla elsődleges kulcsa.
  - UNIQUE ez az oszlop a tábla kulcsa.
  - CHECK(feltétel) csak feltételt kielégítő értékek kerülhetnek be az oszlopba.
  - [FOREIGN KEY] REFERENCES tábla [(oszlop)], ez az oszlop külső kulcs.

Megszorítások

- A tábla megszorításban több oszlopra vonatkozó korlátozásokat adhatunk meg.
- PRIMARY KEY(oszlop1[, oszlop2, ...]) ezek az oszlopok együtt alkotják az elsődleges kulcsot.
- UNIQUE(oszlop1[, oszlop2, ...]) ezek az oszlopok együtt kulcsot alkotnak.
- CHECK(feltétel) csak feltételt kielégítő sorok kerülhetnek be a táblába.
- FOREIGN KEY (oszlop1[, oszlop2, ...]) REFERENCES tábla(oszlop1[, oszlop2, ...]), az oszlopok külső kulcsot alkotnak a megadott tábla oszlopaihoz.
- A megszorításokhoz nevet is rendelhetünk, mely hasznos lehet a megszorítás módosítása, törlése esetén (ALTER TABLE)

Példák

- CREATE TABLE Diakok
(Diak_azonosito NUMERIC (4) PRIMARY KEY,
Nev CHAR (30) NOT NULL,
Cim CHAR (40) NOT NULL,
Telefon CHAR (15)
Osztaly CHAR (3) NOT NULL);

- CREATE TABLE Tanarok
(Tanar_azonosito NUMERIC (4) PRIMARY KEY,
Nev CHAR (30) NOT NULL,
Cim CHAR (40) NOT NULL,
Telefon CHAR (15));

- CREATE TABLE Orarend
(Tanar_azonosito NUMERIC (4) NOT NULL
REFERENCES Tanarok(Tanar_azonosito),
Tantargy CHAR (20) NOT NULL,
Idopont NUMERIC (2), NOT NULL,
Osztaly CHAR (3) NOT NULL),
Terem NUMERIC (3) NOT NULL)
PRIMARY KEY(Tanar_azonosito, Idopont)
UNIQUE(Osztaly, Idopont);

- CREATE TABLE Osztalyzatok
(Diak_azonosito NUMERIC (4)
NOT NULL REFERENCES
Diakok(Diak_azonosito),
Tantargy CHAR (20) NOT NULL,
Datum DATE NOT NULL,
Osztalyzat NUMERIC (1)
VALID(BETWEEN 1 AND 5));

- CREATE TABLE Hianyzasok
(Diak_azonosito NUMERIC (4)
NOT NULL REFERENCES
Diakok(Diak_azonosito),
Datumtol DATE NOT NULL,
Datumig DATE,
Igazolt CHAR (1))
PRIMARY KEY (Diak_azonosito,
Datumtol);

Szerkezet módosítása

- A CREATE TABLE utasítással létrehozott táblázatok definícióját csak korlátozottan módosíthatjuk
- A reláció újabb attribútummal való bővítésére az alábbi parancs szolgál:

  ALTER TABLE reláció_név
  ADD attribútum_név adattipus [(szélesség)]
  Pl.:ALTER TABLE Diakok ADD szul_ev INT (4);
- Egy reláció attribútumának szélességét meg lehet növelni az ALTER TABLE paranccsal.

  ALTER TABLE reláció_név
  MODIFY attribútum_név adattipus (új_szélesség) [NOT NULL];
  Pl.: ALTER TABLE Diakok MODIFY nev CHAR (40) NOT NULL;

Reláció törlése

- Teljes relációk törlésére is lehetőséget biztosít az SQL nyelv a

  DROP TABLE reláció_név;

  utasítással. Ezután a relációban tárolt valamennyi adat és a reláció definíciója is törlődik.

- A diákok személyi adatait tartalmazó reláció törlése a következő paranccsal lehetséges:
  DROP TABLE Diakok;

Nézet tábla

- A nézettáblázat az adatbázisban létező reláción vagy relációkon végrehajtott művelet eredményét tartalmazó olyan új táblázat, amely mögött a valóságban nem áll megfelelő táblázat. Nézettáblát a
  CREATE VIEW nézettábla_név [alias_név, alias_név ...
  AS lekérdezés;
  paranccsal hozhatunk létre.
- A lekérdező utasítás formáit a lekérdező nyelv tárgyalása során részletezzük. A 3/b osztály névsorát tartalmazó nézettáblázatot hoz létre a következő parancs.
  CREATE VIEW 3b AS SELECT * FROM Diakok WHERE osztaly = '3/b';
- Akár több táblázatból is vehetünk oszlopokat a nézettáblába. A nézettábla segítségével a három relációra felbontott órarend relációt összevonhatjuk egy táblázatba a kényelmesebb kezelés érdekében

Nézet tábla

- Példa több táblára

  CREATE VIEW orarend FROM tanr-to_id, tantargy-osztaly, ora
  AS SELECT tanar_azonosito, tantargy, osztaly, idopont, terem
  FROM Tanar-to_id A, Tantargy-osztaly B, Ora C
  WHERE C.to_id = B.to_id AND C.to_id = A.to_id;
- A nézettáblák megszüntetése a relációkhoz hasonlóan a

  DROP VIEW nézettábla_név;

  paranccsal lehetséges.

Index

- A relációkhoz indexeket is rendelhetünk, melyek helyes megválasztása esetén a lekérdezések felgyorsíthatók. Az indexek létrehozására a következő utasítás szolgál:

  CREATE [UNIQUE] INDEX index_név ON reláció (attribútum, attribútum, ...);

- Az index létrehozásánál a UNIQUE módosító megadásával a reláció valamennyi sorában különbözőnek kell lennie az index kulcsnak. Általában a reláció kulcsok esetén használható csak (index kulcs = reláció kulcs).
- Indexet a Diákok reláció Diák_azonosító attribútuma alapján:

  CREATE UNIQUE INDEX Diak ON Diakok (Diak_azonosito);

Index

- Az indexek megszüntetése a

  DROP INDEX index_név ON [reláció]

  parancs segítségével történhet.
- A relációkhoz kapcsolódó indexek használatáról az adatbáziskezelő optimalizáló algoritmusok alapján dönt.
- Az indexeket létrehozásuktól a megszüntetésükig az adatbáziskezelő automatikusan frissíti, a módosításoknak megfelelően.
- Figyelem, az indexek számának növelésével a relációkon végrehajtott módosítások, törlések végrehajtási ideje növekszik.