# 3. előadás – SQL

## Berezvai Dániel jegyzete <http://elte.3ice.hu/>

Adatbázis nyelv

* DML – Data Manipulation Language
* DDL – Data Definition Language

## SQL fő komponensei

Adatleíró, séma definiáló nyelv, DDL (Data Definition):
CREATE, ALTER, DROP

Adatkezelő, manipulációs nyelv, DML (Data Manipulation):
INSERT, UPDATE, DELETE, SELECT

## Tábla/Reláció sémák SQL-ben

CREATE utasítás.

Az attribútum deklaráció legalapvetőbb elemei:

Típus lehet egész szám, dátum, idő, logika boolean érték, string, stb.

Pl.: INTEGER, REAL, CHAR, VARCHAR, DATE

# Lekérdezések SQL-ben

## Egyetlen reláció lekérdezése – unér műveletek

Relációs algebrában ez a kiválasztás és a vetítés. (És az átnevezés)

## Három alapvető záradék

SELECT attribútum lista ()
FROM relációnév ()
WHERE feltétel ()

## Minden megjelenítése

Lehet kifejezéseket használni:

* Konstansok
* R attribútumai
* Műveletek (típusra használhatók) pl. R(A,B,C) esetén A+5B alakok használhatók.

## WHERE alakja

kifejezés kifejezés

 lehet

Ezekből összetett feltétel is építhető:

Logikai műveletek és zárójel segítségével

## SELECT után is lehet írni

attribútum név AS új attribútum név

Ha az eredményben más attribútumnevet szeretnénk használni, akkor “AS új\_oszlopnév” segítségével tudunk más oszlopnevet kiírni (Oracle-ben nem kell az "AS")

## Hiányzó értékek (NULL, konstans)

Miért lehet?

* Nem ismerjük
* Nincs értelme / nem létezik (Valakinek nincs telefonszáma)
* Nem nyilvános, titkolt

Az SQL lehetővé teszi, hogy a relációk soraiban az attribútum értéke egy speciális NULL érték legyen.

A null érték értelmezésére több lehetőségünk is van:

* Ismeretlen érték: például tudom, "Joe’s Bar"-nak van valamilyen címe, de nem tudom, hogy mi az.
* Nem-definiált érték: például a házastárs attribútumnak egyedülálló embereknél nincs olyan értéke, aminek itt értelme lenne, nincs házastársa, ezért null érték.

Where záradékban a null érték vizsgálata:

* kifejezés IS NULL
* kif IS NOT NULL
* Olyan nincs, hogy A=NULL

Kifejezés értéke mindig NULL marad, ha tartalmaz valahol NULL-t:

## Az ismeretlen (unknown) igazságérték

Összehasonlításban a null ismeretlen logikai értéket ad: kifejezésNULL=UNKNOWN

Háromértékű logika: TRUE, FALSE, UNKNOWN

(Inkább meghagyjuk az angol T, F, U rövidítéseket)

## A 3-értékő logika

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | y | x∧ y | x∨y | ¬x |
| T | T | T | T | F |
| T | U | U | T | F |
| T | F | F | T | F |
| U | T | U | T | U |
| U | U | U | U | U |
| U | F | F | U | U |
| F | T | F | T | T |
| F | U | F | U | T |
| F | F | F | F | T |

A szabályt könnyű megjegyezni, ha úgy tekintjük, hogy

Ekkor

## Több reláció lekérdezése

Gyakran előforduló relációs algebrai kifejezés típus:

SELECT s-lista

FROM f-lista -- relációk (táblák) összekapcsolása, illetve szorzata

[WHERE Feltétel]

FROM Lista elemei (ezek ismétlődhetnek):
táblanév [[AS] sorváltozó, …]

A sorváltozó neve lehet másodnév, ugyanis… (nem fontos)

### Elemi kifejezések alakja

konstans

[sorváltozó.] attribútumnév

Szükséges a sorváltozó, ha a FROM listában azonos reláció többször előfordul.

SELECT S-lista ()
FROM F-lista ()
[WHERE feltétel ()]

Miben több, mint az algebra:

* Feltételben kifejezés használni
* S-lista nem csak egy attribútumra vonatkozik, hanem kifejezéseket képezhetünk. kifejezés [AS a-név]

## SFW szabvány kiértékelése / szemantikája

SFW – SELECT FROM WHERE

* A FROM lista minden eleméhez egy beágyazott ciklus,
végigfut az adott tábla sorain
a ciklus minden lépésénél az darab sorváltozónak lesz egy-egy értéke a megfelelő relációból
* Ehhez kiértékeljük a WHERE feltételt, vagyis elvégezzük a WHERE záradékban szereplő feltételnek eleget tevő sorok kiválasztását
* Csak a helyesek, ahol TRUE választ kapunk, azok a sorok kerülnek az eredménybe.

Az SQL-ben az eredmény alapértelmezés szerint nem halmaz, hanem multihalmaz. Ugyan az a sor többször előfordulhat.

Ahhoz, hogy halmazt kapjunk, azt külön kérni kell:
SELECT DISTINCT Lista

A duplikált értékek kiszűrése költséges feladat.

## Természetes összekapcsolás SQL-ben

Legyen :

## Halmazműveletek SQL-ben

A relációs algebrai halmazműveletek: unió, különbség mellett az SQL-ben ide soroljuk a metszetet is (ugyanis SQL-ben megvan a metszetet implementációja is).

Az SQL-ben a halmazműveleteket úgy vezették be, hogy azt mindig két lekérdezés között lehet értelmezni, vagyis nem relációk között, mint R U S, hanem lekérdezem az egyiket is és a másikat is, majd a lekérdezések unióját veszem.

A két lekérdezés azonos típusú legyen.

A műveleteket halmazként értelmezi.

Eltérés: EXCEPT ALL (?)

## Relációs algebra átírása SQL-be

Kiértékelő fa (…):

Szorzat részkifejezés – SELECT FROM WHERE

Unió, metszet, különbség – UNION, INTERSECT, EXCEPT/MINUS

Alkérdések vagy Nézettáblák

# Alkérdések

A FROM listán és a WHERE záradékban (valamint később lesz a HAVING záradékban) zárójelezett SELECT-FROM-WHERE utasításokat (alkérdéseket) is használhatunk.

## Alkérdések használata

Hol használható? Ott, ahol relációnevet használunk:

(1) FROM listában: új listaelem (relációnév változó SQL-ben)

Ez felel meg annak, ahogyan a relációs algebrában tetszőleges helyen használhattuk a lekérdezés eredményét.

(2) WHERE záradékban: kifejezésekben, feltételekben

## Három típusú eredménye lehet

1. Az alkérdés eredménye egyetlen skalárérték:
Vagyis az alkérdés olyan, mint a konstans, ami egy új elemi kifejezésként tetszőleges kifejezésben használható.
2. Skalár értékekből álló multihalmaz logikai kifejezésekben használható:
 művelet a logikai összehasonlítás (=, ≠, stb.)
3. Teljes, többdimenziós tábla a visszatérő érték:
Azt ellenőrzi, hogy egy sor eleme-e egy táblának.

## Alkérdések kiértékelése, szemantikája, függő alkérdés

A kiértékelés az alkérdések kiértékelésével kezdődik.

Milyen változók szerepelhetnek egy alkérdésben?

* Lokális saját változói a saját FROM listáról
* Külső kérdés változói: ekkor az alkérdés korrelált.

Ha az alkérdés nem korrelált, önállóan kiértékelhető és ez az eredmény a külső kérdés közben nem változik, a külső kérdés szempontjából ez egy konstanstábla, akkor a kiértékelés mindig a legbelsőből halad kifelé.

Korrelált alkérdés, amely többször kerül kiértékelésre, minden egyes kiértékelés megfelel egy olyan értékadásnak, amely az alkérdésen kívüli sorváltozóból származik. Függő alkérdésre példa:

Előadás vége.

# Keddi előadás

Kezdés: 8:30

Ismétlés lesz.

## A hat alapművelet

 unió

 különbség

 vetítés

 kiválasztás

 természetes összekapcsolás

 átnevezés

Minimális készlet, vagyis bármelyiket elhagyva az a többivel nem fejezhető ki.

## Vizsgán nagy valószínűséggel lesz maximumkeresés

### !! i) Melyik gyártó gyártja a leggyorsabb számítógépet (PC-t vagy laptopot)?

Kiválasztjuk azokat a PC-ket, amelyiknél van gyorsabb, ha ezt kivonjuk a PC-ékből megkapjuk a leggyorsabbat:



Levelekben elemi kifejezések vannak.

Kiértékelés alulról fölfelé történik.

### !! j) Melyik gyártó gyárt legalább három, különböző sebességű PC-t?

Mint a legalább kettő, csak ott 2x, itt 3x kell a táblát önmagával szorozni.

Legyenek

## A leggyakrabban előforduló típus, amiből építkezünk

Ezt a komponenst támogatja legerősebben majd az SQL:

SELECT s-lista FROM f-lista WHERE feltétel;

## Szorgalmi feladatok lekérdezésekre

Szorgalmi feladatokat szorgalmi pontért be lehet adni

→AB1\_02A\_RelAlg2kif.pdf 21. oldaltól

 jelölje (név jelölje a sörivót)

|  |  |
| --- | --- |
| Dani | Víz |
| Bogi | Bor |
| Viktor | Sör |

 (most "ár" nélkül!)

|  |  |
| --- | --- |
| Konyha | Víz |
| Étterem | Bor |
| Kocsma | Sör |

|  |  |
| --- | --- |
| Dani | Konyha |
| Bogi | Étterem |
| Viktor | Kocsma |

Boldog ivó = látogat olyan bárt, ahol felszolgálnak olyan sört, amit kedvel. Fejezzük ki relációs algebrában!

 hármasok: ahol szereti -t és jár -be:

 (Megj. kommutatív)

|  |  |  |
| --- | --- | --- |
| D | V | K |
| B | B | É |
| V | S | K |

 hármasok: ahol szereti -t, és jár -be és felszolgálnak -t a -ben:

 (Megj. asszociatív)

|  |  |  |
| --- | --- | --- |
| D | V | K |
| B | B | É |
| V | S | K |

Megoldás:

Kik a Boldogtalan ivók?

### Ki jár olyan bárba, ahol van legalább két kedvenc söre?

### Ki jár CSAK olyan bárba, ahol legalább egy kedvenc söre kapható? (MÉG BOLDOGABB)

### Ki jár olyan bárba, ahol az összes kedvenc söre kapható? (NAGYON BOLDOG)

### Ki jár CSAK olyan bárba, ahol az összes kedvenc söre kapható? (SZUPER BOLDOG)

### Ki jár olyan bárba, ahol mindent szeret?

### Ki jár CSAK olyan bárba, ahol mindent szeret?

### "Tanácsadó szolgálat”: Hova menjen el két ivó sörözni, olyan bárt keresünk, ahova mind a ketten járnak és ahol mind a ketten találnak olyan sört, amit kedvelnek, stb.

# Bevezetés az SQL-be

→ Hétfői előadás jegyzet

## SQL fő komponensei kiegészítés

Az SQL elsődlegesen lekérdező nyelv (Query Language)

SELECT utasítás (az adatbázisból információhoz jussunk)

Adatvezérlő nyelv, DCL (Data Control Language)

GRANT, REVOKE

Tranzakció-kezelés

COMMIT, ROLLBACK, SAVEPOINT

Procedurális kiterjesztések

Oracle PL/SQL (Ada alapján), SQL/PSM (PL/SQL alapján)

## Adatbázis relációsémák definiálása

Az SQL tartalmaz adatleíró részt (DDL), az adatbázis

objektumainak a leírására és megváltoztatására.

Objektumok leíró parancsa a CREATE utasítás.

A relációt az SQL-ben táblának (TABLE) nevezik, az SQL alapvetően háromféle táblát kezel:

Alaptáblák (permanens) CREATE TABLE

Nézettáblák CREATE VIEW (ezt később nézzük)

Átmeneti munkatáblák (WITH utasítás is később)

Alaptáblák megadása: CREATE TABLE

→ fent. (Tábla/Reláció sémák SQL-ben)

## Példa

→ AB1\_02B\_BevSQL.pdf 6. oldal

## Az SQL értékekről

INTEGER, REAL, stb, a szokásos értékek, számok.

STRING szintén, de itt egyes-aposztróf közé kell tenni a ‘szöveget’ (vagyis nem „macskaköröm” közé).

Két egyes-aposztróf = egynek felel meg, például ’Joe’’s Bar’ megfelel a Joe’s Bar szövegnek.

Bármely érték lehet NULL (erről később a 6.1.fejezetben)

DATE és TIME típusok is vannak az SQL-ben.

A dátum formátumát meg kell adni DATE ’yyyy-mm-dd’

Például: DATE ’2007-09-30’ (2007. szept. 30)

Az idő formátumát is meg kell adni TIME ’hh:mm:ss’

Például: TIME ’15:30:02.5’ (délután fél 4 múlt két és fél másodperccel)

## Kulcs megadása

PRIMARY KEY vagy UNIQUE

Nincs a relációnak két olyan sora, amely a lista minden attribútumán megegyezne.

Kulcs esetén nincs értelme a DEAFULT értéknek.

Kulcsok megadásának két változata van:

* Egyszerű kulcs (egy attribútum) vagy
* Összetett kulcs (attribútumok listája)

## Összetett kulcs megadása

Ha a kulcs több attribútumból áll, akkor a CREATE TABLE utasításban az attribútum deklaráció után a kiegészítő részben meg lehet adni további tábla elemeket: PRIMARY KEY (attrnév1, … attrnévk)

Példa:

CREATE TABLE Felszolgál (

bár CHAR(20),

sör VARCHAR2(20),

ár NUMBER(10,2),

PRIMARY KEY (bár, sör)

);

## PRIMARY KEY vs. UNIQUE

Átugortuk. Később lesz.

## Idegen kulcsok megadása

Még egy kiegészítő lehetőség Mi köthet össze két táblát? Idegen kulcs (foreign key) megadása

Az egyik tábla egyik oszlopában szereplő értékeknek szerepelnie kell egy másik tábla bizonyos attribútumának az értékei között.

A hivatkozott attribútumoknak a másik táblában kulcsnak kell lennie! (PRIMARY KEY vagy UNIQUE)

Példa: Felszolgál(bár, sör, ár) táblára megszorítás, hogy a sör oszlopában szereplő értékek szerepeljenek a Sörök(sör, gyártó) táblában a sör oszlop értékei között.

## Idegen kulcsok megadása: attribútumként

REFERENCES kulcsszó használatának két lehetősége:

attribútumként vagy sémaelemként lehet megadni.

### 1.) Attribútumként (egy attribútumból álló kulcsra)

PÉLDA:

CREATE TABLE Sörök (

sör CHAR(20) PRIMARY KEY,

gyártó CHAR(20) );

CREATE TABLE Felszolgál (

bár CHAR(20),

sör CHAR(20) REFERENCES Sörök,

ár REAL );

### 2. Sémaelemként (egy vagy több attr.-ból álló kulcsra)

FOREIGN KEY (attribútum lista)

REFERENCES relációnév (attribútum lista)

PÉLDA: CREATE TABLE Sörök (

név CHAR(20) PRIMARY KEY,

gyártó CHAR(20) );

CREATE TABLE Felszolgál (

bár CHAR(20),

sör CHAR(20),

ár REAL,

FOREIGN KEY(sör) REFERENCES Sörök(név));

-- itt most másképp neveztük a hivatkozott attribútumot

## Relációs algebrai kifejezések felírása SELECT-tel

A lekérdezések megadására formálisan relációs algebrai

kifejezéseket adtunk meg (lásd előző 02A\_RelAlg2kif-ben).

Ezeket hogyan tudjuk átírni az SQL SELECT utasítására?

 Példa: Termék (gyártó, modell, típus)

PC (modell, sebesség, memória, merevlemez, cd, ár)

Laptop (modell, sebesség, memória, merevlemez, képernyő, ár)

Nyomtató (modell, színes, típus, ár) --- típus itt mást jelent!

Mi volt a leggyakrabban előforduló típus, amiből építkezek?

Ezt a komponenst támogatja legerősebben majd az SQL:

SELECT s-lista FROM f-lista WHERE feltétel

Halmazműveletek SQL-ben SELECT… lekérdezések között

UNION | EXCEPT/MINUS | INTERSECT

# Lekérdezések az SQL-ben

→ Fent:

## A műveletek szemantikája

→ Fent

## Egy táblát használó SFW alapértelmezése

→ Fent

## SELECT záradékban \* jelentése

→ Fent

## Attribútumok átnevezése

→ Fent

## SELECT záradékban levő kifejezések

→ Fent

## WHERE záradék (összetett feltételek)

→ Fent

## WHERE záradék (további lehetőségek)

SQL specialitások, amelyek könnyen átírhatóak relációs algebrai kifejezésre (összetett kiválasztási feltételre)

* BETWEEN .. AND .. intervallumba tartozás
* IN (értékhalmaz) egyszerű értékek halmaza

SQL specialitások, nem írhatók át relációs algebrába:

* Karakterláncok LIKE összehasonlítása mintákkal
* IS NULL összehasonlítás

## LIKE

Karakterláncok összehasonlítása mintákkal:

* <attribútum> LIKE <minta> vagy
* <attribútum> NOT LIKE <minta>

Minta egy olyan karakterlánc, amelyben használhatjuk a speciális % és \_ karaktereket. A mintában % megfelel bármilyen karakterláncnak és \_ bármilyen karakternek.

Példa: Azokat a bárokat keressük, amelynek a nevében van ‘s (mint például Joe’s Bar)

SELECT név

FROM Bár

WHERE név LIKE '%''s%‘;

## NULL értékek

→Fent

## UNKNOWN

→Fent

## 3-értékő logika

→Fent

## Az eredmény rendezése

SQL SELECT utasításban a záradékok

Az SQL lehetővé teszi, hogy a lekérdezés eredménye bizonyos sorrendben legyen rendezve. Az első attribútum egyenlősége esetén a 2.attribútum szerint rendezve, stb, minden attribútumra lehet növekvő vagy csökkenő sorrend.

Select-From-Where utasításhoz a következő záradékot adjuk, a WHERE záradék és minden más záradék (mint például GROUP BY és HAVING) után következik:

SELECT … FROM … [WHERE …] […]

ORDER BY {attribútum [DESC], …}

Példa: SELECT \* FROM Felszolgál ORDER BY ár DESC, sör

## Attribútumok megkülönböztetése

Ha egy attribútumnév több sémában is előfordul, akkor nem elég az attribútumnév használata, mert ekkor nem tudjuk, hogy melyik sémához tartozik.

Ezt a problémát az SQL úgy oldja meg, hogy megengedi egy relációnévnek és egy pontnak a használatát egy attribútum előtt: R.A (az R reláció A attribútumát jelenti).

Semmi nem tiltja, hogy ugyanaz a reláció többször is szerepeljen, szükség lehet arra, hogy ugyanaz a relációnév többször is előforduljon a FROM listában.

Ekkor a FROM listában másodnevet kell megadni, erre sorváltozóként is szoktak hivatkozni, megadjuk azt is, hogy melyik sorváltozó melyik relációt képviseli:

FROM

Ekkor a SELECT és WHERE záradékok kifejezésekben a hivatkozás: (vagyis )

## Tábla önmagával való szorzata

Bizonyos lekérdezéseknél arra van szükségünk, hogy ugyanannak a relációnak több példányát vegyük.

Ahhoz, hogy meg tudjuk különböztetni a példányokat, a relációkat átnevezzük, másodnevet adunk, vagyis sorváltozókat írunk mellé a FROM záradékban.

A relációkat mindig átnevezhetjük ily módon, akkor is, ha egyébként nincs rá szükség (csak kényelmesebb).

Példa: R(Szülő, Gyerek) séma feletti relációban adott szülő-gyerek adatpárokból állítsuk elő a megállapítható Nagyszülő-unoka párokat!

SELECT t1.Szülő Nagyszülő, t2.Gyerek Unoka

FROM R t1, R t2

WHERE t1.Gyerek = t2.Szülő;

Példa: Sörök(név, gyártó) tábla felhasználásával keressük meg az összes olyan sörpárt, amelyeknek ugyanaz a gyártója.

Ne állítsunk elő (Bud, Bud) sörpárokat.

A sörpárokat ábécé sorrendben képezzük, például ha (Bud, Miller) szerepel az eredményben, akkor (Miller, Bud) ne szerepeljen.

SELECT s1.név, s2.név

FROM Sörök s1, Sörök s2

WHERE s1.gyártó = s2.gyártó

AND s1.név < s2.név;

## Halmazműveletek

→Fent

Példát néztünk, de én kihagytam.

## Halmaz-multihalmaz szemantika

Jövő héten lesz, már most bemásolom diákról.

A SELECT-FROM-WHERE állítások multihalmaz szemantikát használnak, a halmazműveleteknél mégis a halmaz szemantika az érvényes. Azaz sorok nem ismétlődnek az eredményben.

Ha projektálunk, akkor egyszerűbb, ha nem töröljük az ismétlődéseket. Csak szépen végigmegyünk a sorokon.

A metszet, különbség számításakor általában az első lépésben lerendezik a táblákat. Ez után az ismétlődések kiküszöbölése már nem jelent extra számításigényt.

Motiváció: hatékonyság, minimális költségek

## Példa: ALL (multihalmaz szemantika)

Tehát: Ha mégis multihalmaz szemantikát akarunk, akkor oda kell írni az ALL szócskát.

Látogat(név, bár) és Kedvel(név, sör) táblák felhasználásával kilistázzuk azokat a sörivókat, akik több bárt látogatnak, mint amennyi sört szeretnek, és annyival többet, mint ahányszor megjelennek majd az eredményben

(SELECT név FROM Látogat)

EXCEPT ALL

(SELECT név FROM Kedvel);

## Alkérdések

→Fent

## Alkérdés FROM listán

Ilyet ne nagyon használjunk. Nem hatékony kiértékelés.

SELECT sör

FROM Kedvel, (SELECT név

FROM Látogat

WHERE bár = ’Joe’’s bar’) JD

WHERE Kedvel.név = JD.név;

## Alkérdések használata WHERE záradékban

→ Fent (Három típusú eredménye lehet)

Előadás vége.

AB1\_03A\_SQL1lekerd.pdf 18. oldal 5. sor "," helyett ";"

AB1\_03B\_SQL2sorvalt.pdf 5. oldal 8. sor, csukó zárójel hiányzik