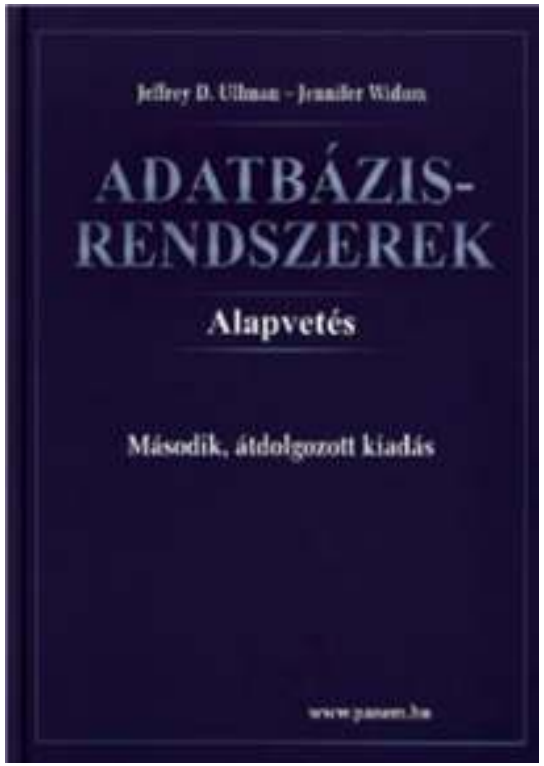
Bevezetés: Relációs adatmodell

Tankönyv: Ullman-Widom:
Adatbázisrendszerek Alapvetés
Második, átdolgozott kiadás,
Panem, 2009



2.1. Adatmodellek áttekintése

2.2. A relációs modell alapjai

2.3. SQL: Relációsémák definiálása

2.4. Relációs algebra

6.1. SQL: Egyszerű egytáblás lekérdezések az SQL-ben

-- A SQL gyakorlat felépítése miatt a Tk. 2.fejezettel kezdünk,
-- Később lesz az 1.fejezet Az adatbázisrendszerek világáról.

Bevezető példa (reláció = tábla)

- Naponta találkozunk adatbázisokkal
  - 1960-as évektől a korai DBMS: banki rendszerek, repülőgép-helyfoglalás, vállalati nyilvántartások
  - Napi szinten: Google, Yahoo, Amazon.com, egyetemi tanulmányi rendszerek (ETR, Neptun)
- 1.példa: A jelentkezési adatok egyeztetésére táblázat (lásd papíron a jelenléti ív /Ez az első előadás példája)
- Reláció = tábla (a jelenléti ív)
- Séma = a reláció szerkezetének leírása (tábla fejléce)
- Előfordulás v. példány = a tábla sorai, adott időpontban a tábla aktuális tartalma dinamikusan változik (példában szereplő tábla tartalma a tantárgyfelvétel időszakában)

Bevezetés: Mi az adatbázis?

- Olyan adatok együttese, amit egy az adatbázis-kezelő rendszer kezel. Mit várunk az ABKR-től? (DBMS-től?)
  - Támogassa nagy méretű (több terabyte mennyiségű) adat hosszú időn keresztül való tárolását, és tegye lehetővé a hatékony hozzáférést a lekérdezések és adatbázis-módosítások számára.
  - Biztosítsa a tartósságot, az adatb. helyreállíthatóságát, biztonságos (konzisztens állapot biztosítsa, védve legyen a hardware, software és felhasználói hibáktól).
  - Felügyelje a több felhasználó által egy időben történő adathozzáféréseket úgy, hogy ezek a műveletek ne legyenek hatással a többi felhasználóra számára (konkurencia-vezérlés)

Mit várunk egy ABKR-től?

- (folyt.) Mit várunk az ABKR-től? (DBMS-től?)
  - Tegye lehetővé a felhasználók számára, hogy új adatbázisokat hozhassanak létre, és azok sémáját, vagyis az adatok logikai struktúráját egy speciális nyelven adhassák meg: Adatdefiníciós nyelv (DDL)
  - Tegye lehetővé a felhasználóknak, hogy az adatokat egy megfelelő nyelv segítségével lekérdezhessék vagy módosíthassák: Adatkezelő nyelv (DML)
  - Kényelmes (fizikai adatfüggetlenség, magas szintű deklaratív nyelv, mint például az SQL szabvány)
  - Hatékony legyen a megvalósítás.
- Erre (Tk.1.fejezetére) később a lekérdezések (relációs algebra és SQL SELECT utasítás) után fogunk visszatérni.

Bevezetés: Ki ismeri az SQL-t?

- Ki ismeri az SQL-t? Van-e különbség?

  Tetszőleges táblát lekérdezve

  megegyezik-e az eredmény?

R

A	B
5	20
10	30
20	40
…	…

(1)
```
SELECT B FROM R
    WHERE A<10 OR A>=10;
```

(2)
```
SELECT B FROM R;
```

- Itt mi a helyzet ezzel?

(3)
```
SELECT A FROM R, S
    WHERE R.B = S.B;
```

(4)
```
SELECT A FROM R
    WHERE B IN (SELECT B FROM S);
```

Mi az adatmodell?

- Az adatmodell a valóság fogalmainak, kapcsolatainak, tevékenységeinek magasabb szintű ábrázolása
- Kettős feladat: az adatmodell megadja, hogy a számítógép számára és a felhasználó számára hogy néznek ki adatok.
- Az adatmodell: adatok leírására szolgáló jelölés.
  Ez a leírás általában az alábbi három részből áll:

1. Az adat struktúrája (struktúrák, tömbök, rekordok, fizikai és fogalmi adatstruktúrák, hamarosan példákat is mutatunk)
2. Az adaton végezhető műveletek (lekérdezések,módosítások, feldolgozások legyenek megfogalmazhatók és hatékonyan legyenek implementálva)
3. Az adatokra tett megszorítások (milyen adatokat engedélyezünk, milyen megszorításokat teszünk?)

A fontosabb adatmodellek

- Hálós, hierarchikus adatmodell (gráf-orientált, fizikai szintű, ill. apa-fiú kapcsolatok gráfja, hatékony keresés)
- Relációs adatmodell (táblák rendszere, könnyen megfogalmazható műveletek), magában foglalja az objektumrelációs kiterjesztést is (strukturált típusok, metódusok), SQL/Object, SQL/CLI, SQL/PSM (PL/SQL)
- Objektum-orientált adatmodell (az adatbázis-kezelés funkcionalitásainak biztosítása érdekében gyakran relációs adatmodellre épül), ODMG: ODL és OQL
- Logikai adatmodell (szakértői rendszerek, tények és következtetési szabályok rendszere)
- Dokumentum típusú adatok, félig-strukturált adatmodell (XML-dokumentum), további modellek: gráf adatbázisok

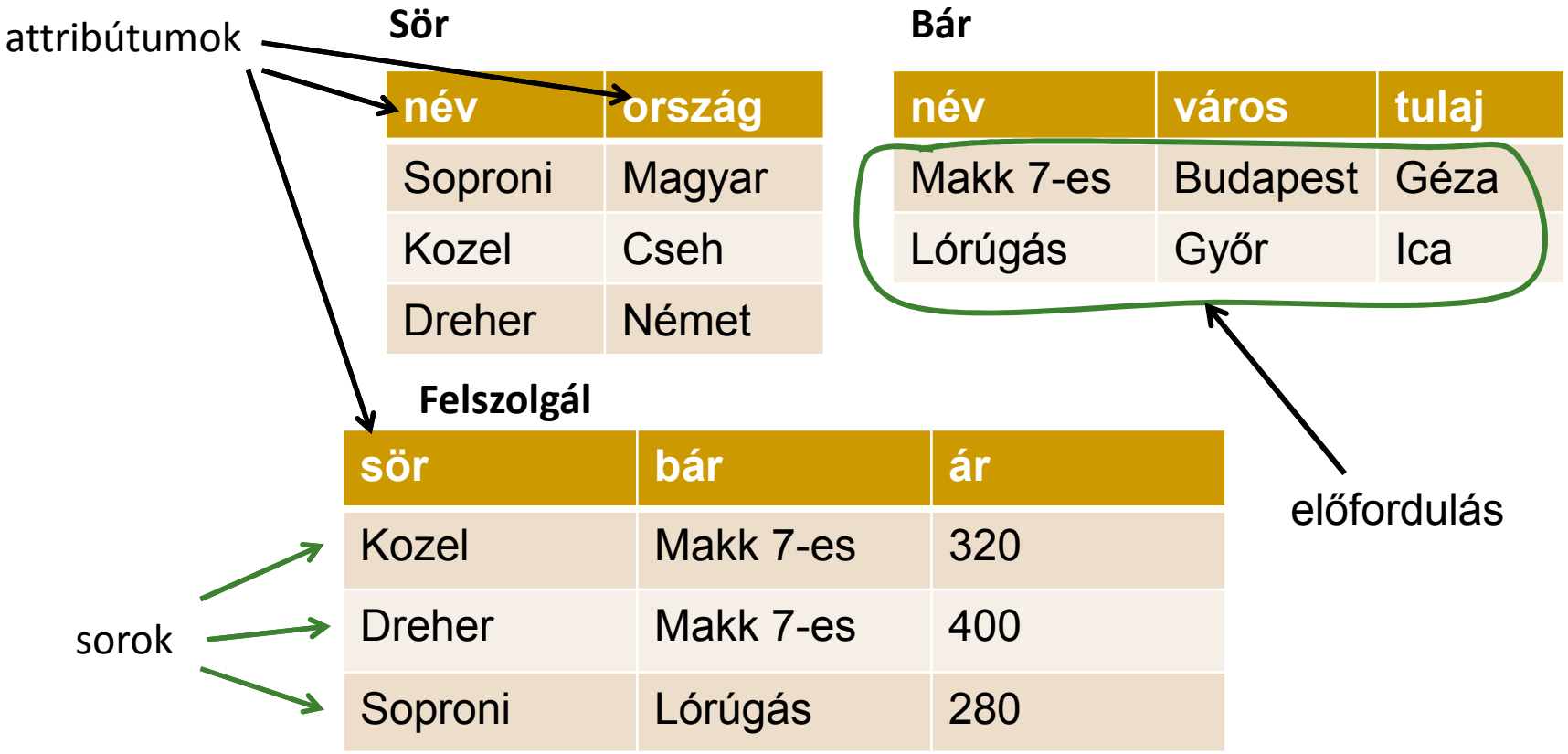
Példa féligstrukturált adatra (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<söröző típus="étterem">
    <név>Makk 7-es</név>
    <város>Budapest</város>
    <tulaj>Géza</tulaj>
    <telefon>+36-70-123-2345</telefon>
    <telefon>+36-70-123-2346</telefon>
</söröző>
<söröző típus="kocsma">
    <név>Lórúgás</név>
    <város>Eger</város>
    <telefon>+36-30-451-1894</telefon>
</söröző>
</xml>
```

Példa relációs adatmodellre

Egy reláció sémája: Sör (név, ország).

Az adatbázis sémája: Sör (név, ország), Söröző (név, város, tulaj),
Felszolgál (sör, söröző, ár).



Sör

név	ország
Soproni	Magyar
Kozel	Cseh
Dreher	Német

Bár

név	város	tulaj
Makk 7-es	Budapest	Géza
Lórúgás	Győr	Ica

Felszolgál

sör	bár	ár
Kozel	Makk 7-es	320
Dreher	Makk 7-es	400
Soproni	Lórúgás	280

Relációs adatmodell története

- E.F. Codd 1970-ben publikált egy cikket
  A Relational Model of Data for Large Shared Data Banks
  Link: http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

  amelyben azt javasolta, hogy az adatokat táblázatokban, **relációkban** tárolják. Az elméletére alapozva jött létre a relációs adatmodell, és erre épülve jöttek létre a relációs adatmodellen alapuló relációs adatbázis-kezelők.
- Relációs (objektum-relációs) **adatbázis-kezelők** például:

  ORACLE , INFORMIX , SYSBASE , INGRES, DB2, stb
- Adatbázisok-1 gyakorlaton ORACLE adatbázis-kezelő rendszert használunk.

Relációs adatmodell előnyei

Miért ez a legelterjedtebb és legkifinomultabb?

- Az adatmodell egy egyszerű és könnyen megérthető strukturális részt tartalmaz. A természetes táblázatos formát nem kell magyarázni, és jobban alkalmazható.
- A relációs modellben a fogalmi-logikai-fizikai szint teljesen szétválik, nagyfokú logikai és fizikai adatfüggetlenség. A felhasználó magas szinten, hozzá közel álló fogalmakkal dolgozik (implementáció rejtve).
- Elméleti megalapozottság, több absztrakt kezelő nyelv létezik, például relációs algebra (ezen alapul az SQL automatikus és hatékony lekérdezés optimalizálása).
- Műveleti része egyszerű kezelői felület, szabvány SQL.

Relációs lekérdező nyelvek

Három nyelv szerepel (Adatbázisok-1 gyakorlaton is lesz)

- Relációs algebra: algebrai megközelítés , megadjuk a kiértékelési eljárásokat, többféle lehetőség összevetése, hatékonysági vizsgálatok.
- SQL szabvány relációs lekérdező nyelv: SQL története, szabványok, felépítése. Lekérdezések SELECT-utasítás, SQL DDL, DML, DCL, és SQL/PSM ill. PL/SQL (Oracle)
- Datalog: logika alapú megközelítés (korábban elsőrendű logikának megfelelő kalkulus típusú lekérdezések voltak, erről az MSc-n lesz majd bővebben). Itt az Adatbázisok-1 keretében a Datalog az összetett lekérdéseknél segítség, például a rekurzív lekérdezéseknél.

Relációs adatmodell: relációs séma

- Adatok gyűjteményét kezeli (gyűjtemény azonosítása: név)
  A gyűjtemény - R reláció (tábla, táblázat) megadása
- A gyűjtemény milyen típusú adatokat gyűjt?
  adattípus: sor-típus. A sor-típus (egy n-es) megadása:
  <Attribútumnév$_1$: értéktípus$_1$, … , Attrnév$_n$: értéktípus$_n$>
  röviden <A_1, … , A_n>
- Relációséma: Relációnév (sortípus) (itt: kerek zárójelben!)
  vagyis R(Anév$_1$: értéktípus$_1$, …, Anév$_n$: értéktípus$_n$)
  röviden R(A_1, … , A_n) ill. U = {A_1, … , A_n} jelöléssel R(U)
- PÉLDA: jelenléti ív fejléce – relációséma: megadjuk a tábla szerkezetét, oszlopnevek és típusuk. Milyen megszorításokat (pl. kulcs) tudunk megadni a sémán?

Relációs adatmodell: előfordulás

- Mit jelent egy konkrét sor? sor $<A_1$: érték$_1$, …, A_n: érték$_n>$
- Reláció előfordulás (példány, instance)

  A sor-típusnak megfelelő véges sok sor (sorok halmaza).

  $\{t_1, \ldots, t_m\}$ ahol t_i (tuple, sor, rekord) i= 1, …, m (véges sok)

  $t_i = <v_{i1}, \ldots, v_{in}>$ (vagyis egy sor n db értékből áll)

  m - számosság (sorok száma)

  n - dimenzió (attribútumok száma)
- Értéktartományok: A reláció minden attribútumához tartozik egy értéktartomány (adott értéktípusú értékek halmaza) (1normálforma, 1NF feltétel: atomi típusú)
- PÉLDA: Jelenléti ív táblázat (a táblázatban az adatok)

Szemléltetés: táblázatos forma

- Szemléltetése: a táblázatos forma (reláció, tábla)

R	A_1	…	A_j	…	A_n
t_1	v_{11}		…		v_{1n}
…			…		
t_i	…		v_{ij}		…
…			…		
t_m	v_{m1}		…		v_{mn}

A_j - attribútumnév

t_i - itt csak szimbolikusan vezetem be, hogy tudjak a sorokra hivatkozni (Oracle-ben: rowid)

Szemléltetés: függvénnyel

- A táblázatos szemléltetésből áttérhetünk a sorok egy másik szemléltetésére:
- $t \in R$ sor felfogható függvényként is
- $t_i : U \rightarrow$ értékek, ahol $U = \{A_1, \ldots, A_n\}$

  Ezzel a jelöléssel $t_i (A_j) = v_{ij}$ ekvivalens jelöléssel $t_i [A_j]$ vagy $t_i .A_j$ (objektum-orientált/metódus típusú jelöléssel)
- Előfordulás: sor-függvények véges halmaza
- Korlátozom a függvényt: $X \subseteq U = \{A_1, \ldots, A_n\}$

  Ha $X = \{A_{j_1}, \ldots, A_{j_k}\}$ attr.halmaz, akkor t[X] típusa:

  $t[X] = \langle A_{j_1}: t(A_{j_1}), \ldots, A_{j_k}: t(A_{j_k}) \rangle$
- Függvény szemléltetéssel könnyen tudom képezni a sorok egy részét és így állítom elő a megfelelő sort.

Relációs adatbázis felépítése

- Az adatbázis tulajdonképpen relációk halmaza.
- a megfelelő relációsémák halmaza adja az adatbázissémát (jelölése dupla szárú $\mathbb{R}$)

  $$\mathbb{R} = \{R_1, \ldots, R_k\}$$
- a hozzá tartozó előfordulások az adatbázis-előfordulás
- Előfordulás tartalma: egyes relációk előfordulásai

- Ez a koncepcionális szint, vagyis a fogalmi modell.
- Fizikai modell: a táblát valamilyen állományszerkezetben jeleníti meg (például szeriális állományban). A relációs adatbázis-kezelők indexelnek, indexelési mód: pl. B+ fa.

  (Ez az Adatbázis-2 kurzuson lesz a fizikai megvalósítás)

Logikai szinten: táblázatos szemléltetés

- A relációk táblákban jelennek meg. A tábláknak egyedi neve van.
  A relációk oszlopait az attribútumok címzik. A tábla sorait tetszőlegesen megcserélhetjük, sorok sorrendje lényegtelen (a halmazszemlélet miatt)

Mivel attribútumok halmazáról van szó, a Példa 1 és Példa 2 relációk nevüktől eltekintve azonosak.

Példa 1

A	B	C
a	b	c
d	a	a
c	b	d

Példa 2

B	C	A
b	c	a
a	a	d
b	d	c

Mivel sorok halmazáról van szó, a Példa 1 és Példa 3 relációk nevüktől eltekintve azonosak.

Példa 3

A	B	C
c	b	d
d	a	a
a	b	c

Példa 4

A	B	C
c	b	d
c	b	d
a	b	c

Ebben a modellben Példa 4 nem reláció, de a valóságban megengedünk mutihalmazokat lásd később SQL

Tankönyv példa: Filmek séma

Filmek(
cím:string,
év:integer,
hossz:integer,
műfaj:string,
stúdióNév:string,
producerAzon:integer)

FilmSzínész(
név:string,
cím:string,
nem:char,
születésiDátum:date)

Stúdió(
név:string,
cím:string,
elnökAzon:integer)

Mit jelentenek az aláhúzások?

Tankönyv példája, hibás fordítás:
title=(film)cím és address=(lak)cím
Tervezéssel később foglalkozunk, ez a példa hibás, az elnevezések, de így jó lesz, hogy a lekérdezéseknél megnézzük hogyan kezeljük.

SzerepelBenne(
filmCím:string,
filmÉv:integer,
színészNév:string)

GyártásIrányító(
név:string,
cím:string,
azonosító:integer,
nettóBevétel:integer)

Példa megszorításokra: Kulcs

- Előző példában: attribútumok aláhúzása mit jelent?
- Filmek: elvárjuk, hogy ne legyen a megengedett előfordulásokban két különböző sor, amelyek megegyeznek cím, év attribútumokon.
- Egyszerű kulcs egy attribútumból áll, de egy kulcs nem feltétlenül áll egy attribútumból, ez az összetett kulcs. Például a **Filmek** táblában a cím és év együtt alkotják a kulcsot, nem elég a cím, ugyanis van például (King Kong, 1933), (King Kong, 1976) és (King Kong, 2005).
- A kulcsot aláhúzás jelöli: Filmek (cím, év, hossz, …)

Kulcsra vonatkozó megszorítások

- Az attribútumok egy halmaza egy kulcsot alkot egy relációra nézve, ha a reláció bármely előfordulásában nincs két olyan sor, amelyek a kulcs összes attribútumának értékein megegyeznének.
- Formális megadása:

  $R(U)$, $X \subseteq U$, $U = \{A_1, \ldots, A_n\}$, $X = \{A_{j1}, \ldots, A_{jk}\}$

  $t \in R$, $t[X] < A_{j1} : t(A_{j1}), \ldots, A_{jk} : t(A_{jk}) >$

  ezzel a jelöléssel mit jelent, hogy X kulcs elvárás?

  ha $t_1 \in R$, $t_2 \in R$ és $t_1[X] = t_2[X]$ akkor $t_1 = t_2$

Az előadás példa: sörivók adatbázis

Sörök(<u>név</u>, gyártó)
Sörözők(<u>név</u>, város, tulaj, engedély)
Sörivók(<u>név</u>, város, tel)

Szeret(<u>név</u>, <u>sör</u>)
Felszolgál(<u>söröző</u>, <u>sör</u>, ár)
Látogat(<u>név</u>, <u>söröző</u>)

- Az aláhúzás jelöli a **kulcsot** (a sorok a kulcs összes attribútumán nem vehetik fel ugyanazt az értékeket).

SQL története, szabványok

- Szabvány adatbázis-kezelő nyelv: SQL
- SQL (angol kiejtésben SEQUEL) uis az SQL előfutára
  IBM fejlesztette ki a 70-es években: SEQUEL → SQL
  más is volt pl. Ingres : QUEL (ez kalkulus alapú lekérdezés)
- Szabványok (ANSI, ISO)
  SQL86, SQL89, SQL92 (SQL2), SQL:1999 (SQL3),
  SQL: 2003, SQL:2006, SQL:2008
- Nyelvjárások (Oracle, Sybase, DB2, Progress, MSSQL, mySQL, SQL Server, PostgreSQL, Access,…)
- Az SQL megvalósítások között vannak különbségek, gyakorlatokon az Oracle SQL-t nézzük meg részletesen.

SQL fő komponensei

- Sémaleíró nyelv, DDL (Data Definition Language)

  CREATE, ALTER, DROP
- Adatkezelő nyelv, DML (Data Manipulation Language)

  INSERT, UPDATE, DELETE, SELECT

  -- 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
  - Átmeneti munkatáblák (WITH utasítás)
- Alaptáblák megadása: CREATE TABLE

Tábla/reláció sémák SQL-ben

- A legegyszerűbb formája:

```
CREATE TABLE relációnév (
        Attribútum deklarációk listája,
        További kiegészítések
);
```

- Az attribútum deklaráció legalapvetőbb elemei:

  Attribútumnév típus [kiegészítő lehetőségek]

-- itt: a **típus** olyan, amit az SQL konkrét megvalósítása támogat (gyakorlaton Oracle környezetben nézzük meg), Típusok, pl: INTEGER, REAL, CHAR, VARCHAR, DATE

-- A kiegészítő lehetőségek például [PRIMARY KEY] vagy [DEFAULT érték] (köv.lapon példa)

Egyszerű példák táblák létrehozására

```
CREATE TABLE Sörözők (
    név CHAR(20),
    város VARCHAR2(40),
    tulaj CHAR(30),
    engedély DATE  DEFAULT SYSDATE
    );

CREATE TABLE Felszolgál (
          söröző CHAR(20),
          sör   VARCHAR2(20),
          ár   NUMBER(10,2) DEFAULT 100
    );
```

Az SQL értékekről (bővebben gyakorlaton)

- 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.
- 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)
- Bármely érték lehet NULL hiányzó érték:

Hiányzó értékek: NULL

- Az SQL lehetővé teszi a táblákban a hiányzó értékeket, vagyis a relációk soraiban az attribútum értéke ne legyen megadva, hanem egy speciális NULL nullérték legyen.
- **A nullérték értelmezésére** több lehetőségünk is van:
  - Nem-ismert érték: például tudom, „Joe's Bár"-jának 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 nullérték.
  - stb (van olyan cikk, amely több százféle okot felsorol)

Kulcs megadása

- PRIMARY KEY vagy UNIQUE
- Nincs a relációnak két olyan sora, amely a lista minden attribútumán megegyezne.
- Kulcs esetén nincs értelme a DEFAULT értéknek.
- Kulcsok megadásának két változata van:
  - Egyszerű kulcs (egy attribútum) vagy
  - Összetett kulcs (attribútumok listája)

Egyszerű kulcs megadása

- Ha a kulcs egyetlen attribútum, akkor ez az attribútum deklarációban megadható

  <attribútumnév> <típus> PRIMARY KEY

  vagy <attribútumnév> <típus> UNIQUE

- Példa:

```
CREATE TABLE Sörök (
        név         CHAR(20) UNIQUE,
        gyártó      CHAR(20)
);
```

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 (attrnév$_1$, … attrnév$_k$)
- Példa:

```
CREATE TABLE Felszolgál (
        söröző      CHAR(20),
        sör         VARCHAR2(20),
        ár          NUMBER(10,2),
        PRIMARY KEY (söröző, sör)
);
```

PRIMARY KEY vs. UNIQUE

- Csak egyetlen PRIMARY KEY lehet a relációban, viszont UNIQUE több is lehet.
- PRIMARY KEY egyik attribútuma sem lehet NULL érték egyik sorban sem. Viszont UNIQUE-nak deklarált attribútum lehet NULL értékű, vagyis a táblának lehet olyan sora, ahol a UNIQUE attribútum értéke NULL vagyis hiányzó érték.
- az SQL lekérdezésnél adjuk meg hogyan kell ezzel a speciális értékkel gazdálkodni, hogyan lehet NULL-t kifejezésekben és hogyan lehet feltételekben használni
- Következő héten visszatérünk a megszorítások és a hivatkozási épség megadására.

Táblák létrehozása után a táblák feltöltése adatokkal

Eddig láttuk, hogy a CREATE TABLE utasítással hogyan tudunk létrehozni táblákat és megadni a kulcsokat:

- SQL DDL: sémaleíró nyelv (Data Definition Language)
  CREATE TABLE, ALTER TABLE, DROP TABLE

Most nézzük meg a táblák tartalmának módosítását, hogyan tudjuk INSERT utasítással a táblát feltölteni adatsorokkal:

- SQL DML: adatkezelő nyelv (Data Manipulation Language)
  INSERT, UPDATE, DELETE, SELECT

Adatbázis tartalmának módosítása

- Lekérdező utasítás SELECT - lekérdezés
- A módosító utasítások nem adnak vissza eredményt, mint a lekérdezések, hanem az adatbázis tartalmát változtatják meg.
- 3-féle módosító utasítás létezik:

  INSERT - sorok beszúrása

  DELETE – sorok törlése

  UPDATE – sorok komponensei értékeinek módosítása

Beszúrás (insert into)

- Két alakja van: 1.) ha egyetlen sort szúrunk be:

  INSERT INTO <reláció>

  VALUES (<konkrét értékek listája>);

- 2.) ezt majd a lekérdezések után nézzük meg, hogyan tudunk több sort beolvasni a táblába, egy lekérdezés eredményét alkérdés segítségével:

  INSERT INTO <reláció>

  (<alkérdés>);

- INSERT INTO 1.) alakjára példa: a Szeret(név, sör) táblában rögzítjük, hogy Zsu szereti a Bud sört.

  INSERT INTO Szeret

  VALUES('Zsu', 'Bud');

Attribútumok megadása

- A reláció neve után megadhatjuk az attribútumait.
- Ennek alapvetően két oka lehet:
  1. Nem emlékszünk, hogy a reláció definíciójában, milyen sorrendben szerepeltek az attribútumok.
  2. Nincs minden attribútumnak értéke, és azt szeretnénk, ha a hiányzó értékeket NULL vagy default értékkel helyettesítenék.

Példa:

```
INSERT INTO Szeret(sör, név)
VALUES('Bud', 'Zsu');
```

Default értékek megadása

- A CREATE TABLE utasításban az oszlopnevet DEFAULT kulcsszó követheti és egy érték.
- Ha egy beszúrt sorban hiányzik az adott attribútum értéke, akkor a default értéket kapja.

```
CREATE TABLE Sörivók(
    név CHAR(30) PRIMARY KEY,
    cím CHAR(50) DEFAULT 'Sesame St'
    telefon CHAR(16)  );
INSERT INTO Sörivók(név)
    VALUES('Zsu');
```

Az eredmény sor:

név	cím	telefon
Zsu	Sesame St	NULL

Gyakorlat: Oracle adatbázisban

- Az 1.gyakorlaton lesz: **Táblák létrehozására példák.** Az attribútumok típusának megadásakor az Oracle implementációban **milyen standard típusok** közül választhatunk, mi a különbség pl. CHAR és VARCHAR?
- Előkészítés: Töltsük fel a táblákat adatokkal, INSERT
- A lekérdezésekkel folytatjuk, relációs algebrai lekérdező nyelvvel, és ezen alapuló SQL SELECT utasítás is lesz.

Mit nevezünk algebrának?

- Nyelv: a kérdés szintaktikai alakja és a kérdés kiértékelése (algoritmus) kiértékelési szemantika
- Algebra műveleteket és atomi operandusokat tartalmaz.
- Relációs algebra: az atomi operandusokon és az algebrai kifejezéseken végzett műveletek alkalmazásával kapott relációkon műveleteket adunk meg, kifejezéseket építünk (a kifejezés felel meg a kérdés szintaktikai alakjának).
- Fontos tehát, hogy minden művelet végeredménye reláció, amelyen további műveletek adhatók meg.
- A relációs algebra atomi operandusai a következők:
  - a relációkhoz tartozó változók,
  - konstansok, amelyek véges relációt fejeznek ki.

Relációs algebrai lekérdező nyelv ---1

Relációs algebrai kifejezés, mint lekérdező nyelv

Lekérdező nyelv: L -nyelv

Adott az adatbázis sémája: $\mathbb{R} = \{R_1, \ldots, R_k\}$

$q \in L$ $\quad q: R_1, \ldots, R_k \rightarrow V$ (eredmény-reláció)

E - relációs algebrai kifejezés: $E(R_1, \ldots, R_k) = V$ (output)

Relációs algebrai kifejezések formális felépítése

- Elemi kifejezések (alapkifejezések)

  (i) $R_i \in \mathbb{R}$ (az adatbázis-sémában levő relációnevek)

  R_i kiértékelése: az aktuális előfordulása

  (ii) konstans reláció (véges sok, konstansból álló sor)

- Összetett kifejezések (folyt. köv.oldalon)

Relációs algebrai lekérdező nyelv ---2

(folyt.) Relációs algebrai kifejezések felépítése

- Összetett kifejezések
- Ha E_1, E_2 kifejezések, akkor a következő E is kifejezés
  - $E:=E_1 \cup E_2$ unió, ha azonos típusúak (és ez a típusa)
  - $E:= E_1 - E_2$ különbség, ha E_1, E_2 azonos típusúak (típus)
  - $E:= \Pi_{lista} (E_1)$ vetítés (típus a lista szerint)
  - $E:= \sigma_{Feltétel} (E_1)$ kiválasztás (típus nem változik)
  - $E:= E_1 \bowtie E_2$ term. összekapcsolás (típus attr-ok uniója)
  - $E:= \rho_{S(B1, \ldots, Bk)} (E_1 (A_1, \ldots A_k))$ átnevezés (típ.új attr.nevek)
  - $E:=(E_1)$ kifejezést zárójelezve is kifejezést kapunk
- Ezek és csak ezek a kifejezések, amit így meg tudunk adni

Halmazműveletek (jelölése a szokásos)

- Reláció előfordulás véges sok sorból álló halmaz. Így értelmezhetők a szokásos halmazműveletek: az unió (az eredmény halmaz, csak egyszer szerepel egy sor) értelmezhető a metszet és a különbség. Milyen művelet van még halmazokon? Értelmezhető-e relációkon?
- R, S és azonos típusú, $R \cup S$ és $R - S$ típusa ugyanez
  $R \cup S := \{t \mid t \in R \vee t \in S\}$, $R - S := \{ t \mid t \in R \wedge t \notin S\}$
- Az alapműveletekhez az unió és különbség tartozik, metszet műveletet származtatjuk $R \cap S = R - (R - S)$
-

A	B	C
a	b	c
c	d	e
g	a	d

A	B	C
a	b	c
c	d	e
g	d	f

Példa: különbségre

R – S

A	B	C
g	a	d

Vetítés (project, jelölése pí: Π)

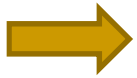
- Vetítés (projekció). Adott relációt vetít le az alsó indexben szereplő attribútumokra (attribútumok számát csökkentik)
- $\Pi_{lista}(R)$ ahol lista: $\{A_{i_1}, \ldots, A_{i_k}\}$ R-sémájában levő attribútumok egy részhalmazának felsorolása

  eredmény típusa $<A_{i_1}$: értéktípus$_{i_1}$, … , A_{i_k}:értéktípus$_{i_k}>$

  $\Pi_{lista}(R) := \{ t.A_{i_1}, t.A_{i_2}, \ldots, t.A_{i_k} \mid t \in R\} = \{ t[lista] \mid t \in R\}$
- Reláció soraiból kiválasztja az attribútumoknak megfelelő $A_{i_1}, \ldots, A_{i_k}$ -n előforduló értékeket, ha többször előfordul akkor a duplikátumokat kiszűrjük (hogy halmazt kapjunk)
- **Példa:**

A	B	C
a	b	c
c	d	e
c	d	d

$\Pi_{A, B}(R)$



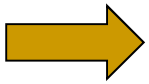
A	B
a	b
c	d

Kiválasztás (select, jelölése szigma: σ)

- Kiválasztás (szűrés). Kiválasztja az argumentumban szereplő reláció azon sorait, amelyek eleget tesznek az alsó indexben szereplő feltételnek.
- $\sigma_{\text{Feltétel}}(R)$ és R sémája megegyezik
- $\sigma_{\text{Feltétel}}(R) := \{ t \mid t \in R$ és t kielégíti az F feltételt$\}$
- $R(A_1, \ldots, A_n)$ séma feletti reláció esetén a σ_F kiválasztás F feltétele a következőképpen épül fel:
  - elemi feltétel: $A_i\ \theta\ A_j$, $A_i\ \theta$ c, ahol c konstans, θ pedig =, ≠, <, >, ≤, ≥
  - összetett feltétel: ha B_1, B_2 feltételek, akkor $\neg B_1$, $B_1 \wedge B_2$, $B_1 \vee B_2$ és zárójelezésekkel is feltételek
- **Példa:**

A	B	C
a	b	c
c	d	e
g	a	d

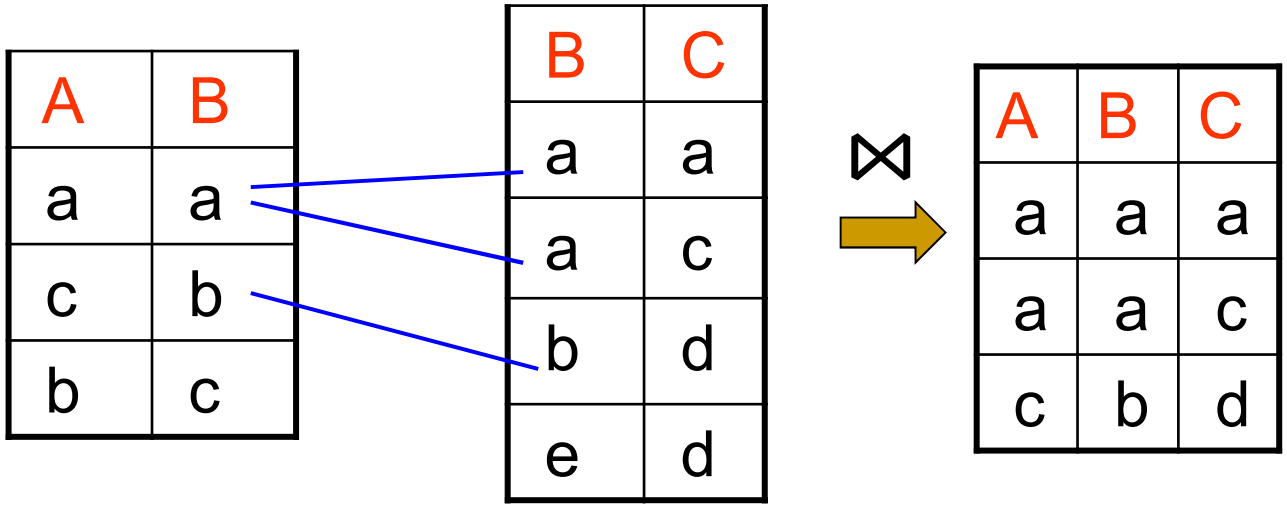
$\sigma_{A=a \vee C=d}(R)$



A	B	C
a	b	c
g	a	d

Természetes összekapcsolás ---1

- Szorzás jellegű műveletek (attribútumok számát növeli) többféle lehetőség, amelyekből csak egyik alapművelet:
- Angolul: Natural Join (jelölése: „csokornyakkendő”)
- Természetes összekapcsolás: közös attribútum-nevekre épül. R ⋈ S azon sorpárokat tartalmazza R-ből illetve S-ből, amelyek R és S azonos attribútumain megegyeznek.



Természetes összekapcsolás ---2

- Természetes összekapcsolás:
- Legyen $R(A_1,\ldots,A_k,B_1,\ldots,B_n)$, illetve $S(B_1,\ldots,B_n,C_1,\ldots,C_m)$
- $R \bowtie S$ típusa $(A_1,\ldots,A_k,B_1,\ldots,B_n,C_1,\ldots,C_m)$ vagyis a két attribútum-halmaz uniója
- $R \bowtie S = \{ <A_1: t(A_1),\ldots, A_k: t(A_k), B_1: t(B_1),\ldots, B_n: t(B_n),$
  $C_1: s(C_1),\ldots, C_m: s(C_m)> \mid t \in R, s \in S,$
  $t(B_i) = s(B_i)\ i=1, \ldots, n \}$
- $R \bowtie S$ elemei $v \in R \bowtie S$

  $R \bowtie S = \{ v \mid \exists\, t \in R, \exists\, s \in S: t[B_1,\ldots,B_n] = s[B_1,\ldots,B_n] \wedge$
  $\wedge\ v[A_1,\ldots,A_k] = t[A_1,\ldots,A_k] \wedge v[B_1,\ldots,B_n] = t[B_1,\ldots,B_n] \wedge$
  $\wedge\ v[C_1,\ldots,C_m] = s[C_1,\ldots,C_m] \}$

Természetes összekapcsolás ---3

- Példákban: két azonos nevű attribútumot úgy tekintünk, hogy ugyanazt jelenti és a közös érték alapján fűzzük össze a sorokat.
- Milyen problémák lehetnek?
- Filmek adatbázisban ugyanarra a tulajdonságra más névvel hivatkozunk: Filmek.év és SzerepelBenne.filmÉv, illetve FilmSzínész.név és SzerepelBenne.színészNév
- Termékek adatbázisban pedig ugyanaz az azonosító mást jelent: Termék.típus más, mint Nyomtató.típus
- Emiatt a Filmek és a Termékek adatbázisokban ahhoz, hogy jól működjön az összekapcsolás szükségünk van egy technikai műveletre, és ez: az átnevezés (rename)

Átnevezés (rename, jelölése ró: ϱ)

- Miért van erre szükség? Nem tudjuk a reláció saját magával való szorzatát kifejezni, R ⋈ R = R lesz.
- Láttuk, hogy egyes esetekben szükség lehet relációnak vagy a reláció attribútumainak átnevezésére:

  $\rho_{T(B_1, \ldots, B_k)}(R(A_1, \ldots A_k))$

- Ha az attribútumokat nem szeretnénk átnevezni, csak a relációt, ezt $\rho_T(R)$-rel jelöljük. Ha ugyanazt a táblát használjuk többször, akkor a táblának adunk másik hivatkozási (alias) nevet.
- Az attribútumok átnevezése helyett alternatíva: R.A (vagyis relációnév.attribútumnév hivatkozás) amivel meg tudjuk különböztetni a különböző táblákból származó azonos nevű attribútumokat.

Szorzás jellegű műveletek ---1

- Szorzás jellegű műveletek többféle lehetősége közül csak az egyiket vesszük alapműveletnek: join vagy természetes összekapcsolást, amely közös attribútumnevekre épül. R ⋈ S azon sorpárokat tartalmazza R-ből illetve S-ből, amelyek R és S azonos attribútumain megegyeznek.
- Egy másik lehetőség: direkt-szorzat (Descartes-szorzat) Ez is tekinthető alapműveletnek (és bizonyos esetekben egyszerűbb ezt venni alapműveletnek) az ennél sokkal gyakrabban használt természetes összekapcsolás helyett.
- R × S: az R és S minden sora párban összefűződik, az első tábla minden sorához hozzáfűzzük a második tábla minden sorát

$$R \times S := \{ t \mid t[R] \in R \text{ és } t[S] \in S \}$$

Szorzás jellegű műveletek ---2

- A direkt-szorzat (vagy szorzat, Descartes-szorzat) esetén természetesen nem fontos az attribútumok egyenlősége. A két vagy több reláció azonos nevű attribútumait azonban meg kell különböztetni egymástól.
  Hivatkozás séma: oszlopok átnevezése illetve azonos nevű oszlop esetén: $R.A_1, \ldots, R.A_k, S.A_1, \ldots, S.A_k$
- **Példa:**

A	B	C
a	b	c
c	d	e
g	a	d

B	D
b	r
q	s

$R \times S$

A	R.B	C	S.B	D
a	b	c	b	r
a	b	c	q	s
c	d	e	b	r
c	d	e	q	s
g	a	d	b	r
g	a	d	q	s

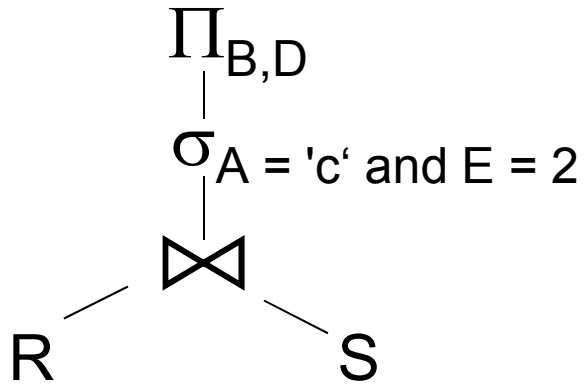
Szorzás jellegű műveletek ---3

- Ha R, S sémái megegyeznek, akkor $R \bowtie S = R \cap S$.
- Ha R, S sémáiban nincs közös attribútum, akkor $R \bowtie S = R \times S$.
- Később nézünk még további szorzás jellegű műveletet:

  Théta összekapcsolás $\bowtie_\theta$, félig összekapcsolás $\ltimes$, és a rel.algebra kiterjesztésénél külső összekapcsolásokat.
- Hogyan fejezhető ki az R x S direkt szorzat relációs algebrában? (ha a természetes összekapcsolást tekintjük alapműveletnek, ebből és az átnevezés segítségével felírható a direkt szorzat).
- Hogyan fejezhető ki a természetes összekapcsolás, ha a direkt szorzatot soroljuk az alapműveletek közé?

Lekérdezések kifejezése algebrában ---1

- Kifejezés kiértékelése: összetett kifejezést kívülről befelé haladva átírjuk kiértékelő fává, levelek: elemi kifejezések.
- A relációs algebra procedurális nyelv, vagyis nemcsak azt adjuk meg, hogy mit csináljunk, hanem azt is hogyan.
- Legyen R, S az R(A, B, C), S(C, D, E) séma feletti reláció $\Pi_{B,D}\, \sigma_{A = 'c' \text{ and } E = 2}\, (R \bowtie S)$
- Ehhez a kiértékelő fa: (kiértékelése alulról felfelé történik)



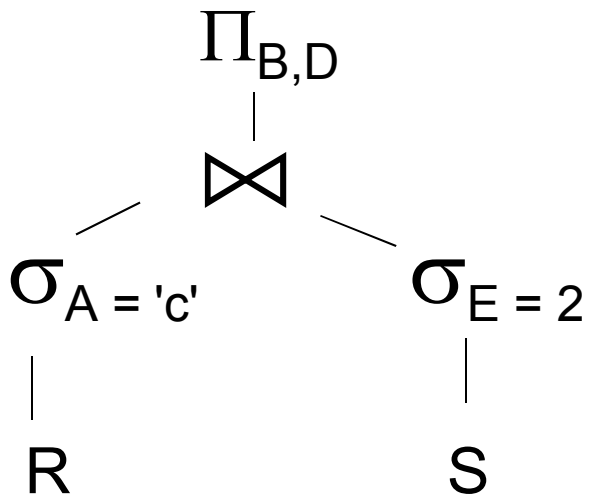
- Tudunk-e ennél jobb, hatékonyabb megoldást találni?

Lekérdezések kifejezése algebrában ---2

- Ekvivalens átalakítási lehetőségekkel, relációs algebrai azonosságokkal át tudjuk alakítani a fentivel ekvivalens másik relációs algebrai kifejezésre. Hatékonyabb-e?

$$\Pi_{B,D}\,(\sigma_{A = 'c'}(R) \bowtie \sigma_{E = 2}(S))$$

- Ehhez is felrajzolva a kiértékelő fát:



Lekérdezések kifejezése algebrában ---3

- Ekvivalens átalakítás: oly módon alakítjuk át a kifejezést, hogy az adatbázis minden lehetséges előfordulására (vagyis bármilyen is a táblák tartalma) minden esetben ugyanazt az eredményt (vagyis ugyanazt az output táblát) adja az eredeti és az átalakított kiértékelő fa.
- Adatbázisok-2 tárgyból lesznek az **ekvivalens átalakítási szabályok**, a szabály alapú optimalizálás első szabálya például, hogy a kiválasztási műveletet minél előbb kell végrehajtani (közbülső táblák lehetőleg kicsik legyenek)
- Ha egy-egy részkifejezést, ha gyakran használjuk, akkor új változóval láthatjuk el, segédváltozót vezethetünk be:

  $T(C_1, \ldots C_n) := E(A_1, \ldots A_n)$, de a legvégén a bevezetett változók helyére be kell másolni a részkifejezést.

Példa – Sörivók adatbázisséma

- Az előadások SQL lekérdezései az alábbi Sörivók adatbázissémán alapulnak

  (aláhúzás jelöli a kulcs attribútumokat)

Sörök(<u>név</u>, gyártó)
Sörözők(<u>név</u>, város, tulaj, engedély)
Sörivók(<u>név</u>, város, tel)

Szeret(<u>név</u>, <u>sör</u>)
Felszolgál(<u>söröző</u>, <u>sör</u>, ár)
Látogat(<u>név</u>, <u>söröző</u>)

Egyszerű példa Select-From-Where-re

- Használjuk Sörök(név, gyártó) relációsémát, mely söröket gyártja a Dreher?

```
SELECT név
FROM Sörök
WHERE gyártó = 'Dreher';
```

A lekérdezés eredménye

név
Arany Ászok
Dreher Classic
. . .

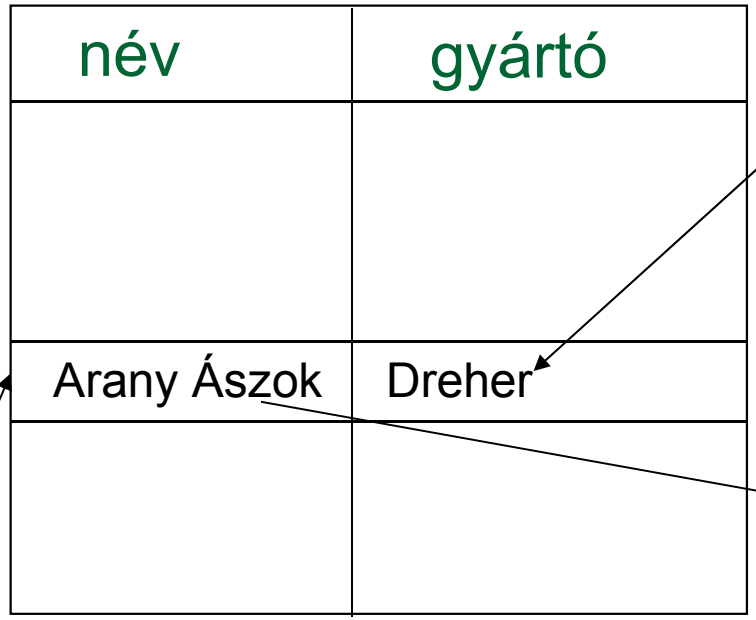
A lekérdezés eredménye egy reláció, amelynek egy attribútuma van (név) és a sorai az összes olyan sör neve, amelyet a Dreher gyárt.

Eltérés a relációs algebrától: Az SQL alapértelmezésben nem szűri ki a duplikátumokat, az eredmény multihalmaz.

Az egytáblás lekérdezés formális kiértékelése

- Kiindulunk a **FROM záradékból**, mely táblára vonatkozik a lekérdezés?
- Elvégezzük a **WHERE záradékban** szereplő feltételnek eleget tevő sorok kiválasztását
- Alkalmazzuk a **SELECT záradékban** jelölt kiterjesztett projekciót. Lényeges különbség a relációs algebra és SQL között, hogy **az SQL-ben az eredmény** alap-értelmezés szerint **nem halmaz**, hanem **multihalmaz,** egy sor az eredményben többször is előfordulhat, ennek az oka, hogy az olcsóbb és hatékonyabb kiértékelést tekintjük az SQL-ben alapértelmezésnek.
- Ahhoz, hogy halmazt kapjunk, azt külön kérni kell SELECT **DISTINCT** Lista FROM Táblanév

A műveletek szemantikája



1) Ellenőrizzük a feltételt, hogy a gyártó Dreher-e

2) Ha a feltétel teljesült, akkor képezünk egy t eredménysort

3) Ebből a t sorból a SELECT listának megfelelő típusú sort képezzük, példa: t.név

Az egytáblás SFW alapértelmezése

SELECT [DISTINCT] kif_1 [[AS] $onév_1$], … , kif_n [$onév_n$]
FROM táblanév [sorváltozó]
[WHERE feltétel]

Alapértelmezés (a műveletek szemantikája -- általában)

- A FROM záradékban levő relációhoz tekintünk egy **sorváltozót**, amely a reláció minden sorát bejárja
- Minden egyes „aktuális” sorhoz **kiértékeljük** a WHERE záradékot
- Ha helyes (vagyis **igaz**) választ kaptunk, akkor képezünk egy sort a SELECT záradékban szereplő kifejezéseknek megfelelően.

SELECT záradékban * jelentése

- Amikor csak egy reláció van a FROM záradékban, akkor a SELECT záradékban levő * jelentése: „a reláció minden attribútuma”
- Példa: Keressük a Sörök(név, gyártó) tábla alapján a Dreher-sörök adatait.
- A lekérdezés eredménye

```
SELECT *
FROM Sörök
WHERE gyártó = 'Dreher';
```

- A lekérdezés eredménye a Sörök tábla összes attribútumát tartalmazza. Első lépésben (kezdő gyakorlásnál kicsi táblákra) mindig lekérdezzük előbb a tábla tartalmát: SELECT * FROM Táblanév;

Attribútumok átnevezése

- Ha az eredményben (a fejlécben) más attribútumnevet szeretnénk használni, akkor “[AS] új_oszlopnév” segítségével tudunk más oszlopnevet kiírni.
  (Oracle: másodnévben nem kell ‘AS’, csak szóköz)
- Listán azt értjük, hogy vesszővel vannak elválasztva az elemek (attribútumnevek), ha a másodnévben szóköz szerepel, akkor azt macskaköröm közé kell tenni: ”… ”
- Példa: Sörök(név, gyártó)

```
SELECT név sör, gyártó ”Dreher gyártó”
FROM Sörök
WHERE gyártó = ’Dreher’;
```

- A lekérdezés eredményében az új oszlopnevek lesznek.

SELECT záradékban levő kifejezések

- Az attribútumnevek helyett tetszőleges kifejezések állhatnak (amelyek megfelelnek az adott típusra) a SELECT záradék elemeként.
- Lásd bővebben majd a gyakorlatok példáit, feladatait, felhasználjuk az Oracle DB SQL Language Reference megfelelő fejezeteit: Operators, Functions, Expressions.
- Példa: Felszolgál(söröző, sör, ár)

```
SELECT söröző, sör, ár*114 árYenben
FROM Felszolgál;
```

- Konstansok a kifejezésekben Szeret(név, sör):

```
SELECT név DABkedvelő
FROM Szeret
WHERE upper(sör) = 'DAB';
```

WHERE záradék (összetett feltételek)

- Hasonlóan, mint a relációs algebra kiválasztás (σ) feltételében elemi feltételekből építkezünk, ahol elemi feltételen két kifejezés =, <>, <, >, <=, >= aritmetikai összehasonlítását, a theta műveletet értjük.
- Logikai műveletek AND, OR, NOT és zárójel ( ) segítségével kapjuk az összetett feltételeket.
- Példa: Felszolgál (söröző, sör, ár) relációséma esetén keressük a „Joe's Bar"-ban árult „DAB" sörök árát:

```
SELECT ár
FROM Felszolgál
WHERE söröző = 'Joe''s Bar' AND
      sör = 'DAB';
```

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:

(--- ezek jönnek a köv.lapon…)

- 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 sörözőket keressük, amelyik nevének a második betűje „a” vagy a nevében van „‘s”, mint ahogyan például a „*Joe’s Bar*” névben is szerepel:

```
SELECT név FROM Sörözők
WHERE név LIKE '_a%' OR
      név LIKE '%''s%';
```

NULL (hiányzó) értékek

- Az SQL lehetővé teszi, hogy a relációk soraiban az attribútum értéke egy speciális NULL nullérték legyen.
- **A nullérték értelmezésére** több lehetőségünk is van:
  - Hiányzó érték: például tudom, „Joe's Bár"-jának 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 nullérték.
- **Where záradékban a nullérték** vizsgálata:
  - IS NULL
  - IS NOT NULL

NULL értékek használata

- **Where záradékban a nullérték** használata:
  - Amikor egy aritmetikai műveletben az egyik tag NULL, akkor az eredmény is NULL.
  - Amikor egy NULL értéket hasonlítunk össze bármely más értékkel (beleértve a NULL-t is) az összehasonlítási operátorok (=, <>, <, <=, >, >=) segítségével, akkor az eredmény UNKNOWN (ismeretlen).

Az ismeretlen (unknown) igazságérték

- Az SQL-ben szereplő logikai feltételek valójában háromértékű logika: TRUE, FALSE, UNKNOWN (magyarban igaz, hamis, ismeretlen rövidítése miatt inkább meghagyjuk az angol T, F, U rövidítéseket).
- A WHERE záradékban szereplő logikai feltételt a rendszer minden egyes sorra ellenőrzi és a logikai érték TRUE, FALSE vagy UNKNOWN valamelyike lehet, de az eredménybe csak azok a sorok kerülnek, amelyeknek a feltétel kiértékelése TRUE értéket adott.

A háromértékű logika

- Hogyan működnek az AND, OR, és NOT logikai műveletek a 3-értékű logikában?
- A szabályt könnyű megjegyezni, ha úgy tekintjük, hogy TRUE = 1, FALSE = 0, és UNKNOWN = ½.
- Ekkor AND = MIN, OR = MAX, NOT(*x*) = 1-*x*.
- Példa:

  TRUE AND (FALSE OR NOT(UNKNOWN)) =

  MIN(1, MAX(0, (1 - ½))) =

  MIN(1, MAX(0, ½)) =

  MIN(1, ½) = ½ = UNKNOWN
- A 3-értékű logika AND, OR és NOT igazságtáblázatát lásd a Tk. 6.2.ábráját (vagy kitöltése a fenti szabállyal)

A háromértékű logika (Tk.6.2. ábra)

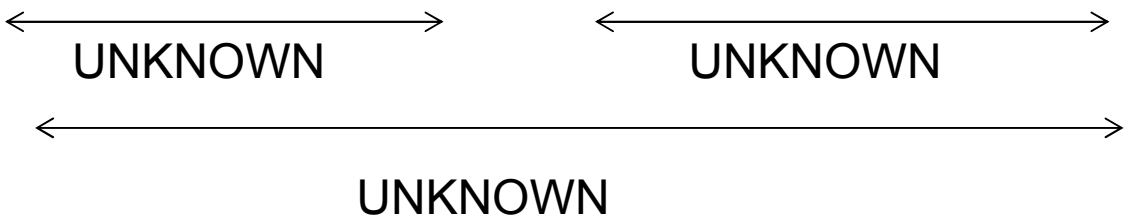
x	*y*	*x* AND *y*	*x* OR *y*	NOT *x*
T	T	T	T	F
T	U	U	T	F
T	F	F	T	F
U	T	U	T	U
U	U	U	U	U
U	F	F	U	U
F	T	F	T	T
F	U	F	U	T
F	F	F	F	T

Egy meglepő példa

- Példa: Felszolgál reláció legyen az alábbi:

söröző	sör	ár
Joe's Bar	Bud	NULL

```
SELECT söröző
FROM Felszolgál
WHERE ár < 2.00   OR   ár >= 2.00;
```



Oka: a 2-értékű != 3-értékű szabályok

- Bizonyos általános szabályok, mint például, hogy az AND kommutatív érvényes a 3-értékű logikában is.
- Ellenben nem igaz, például a kizáró szabály, vagyis

  p OR NOT *p* = TRUE nem teljesül, ha *p* = UNKNOWN, mert ekkor a baloldal: MAX(½, (1 – ½)) = ½ != 1 vagyis a 3-értékű logikában baloldal értéke nem TRUE.
- Ezért az előző példában nem az eredeti egy soros táblát, hanem az üres táblát (amelynek egy sora sincs) kaptuk meg az eredménytáblaként.

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

Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?
- Az első gyakorlaton: Példák táblák létrehozására, az attribútumok megadásakor milyen standard típusok közül választhatunk, és itt milyen különbségek vannak.
- Előkészítés: táblák létrehozása és feltöltése adatsorokkal
- Egy táblára vonatkozó lekérdezésekre példák, vetítés és kiválasztás művelete, függvények használata, a kiválasztott sorok rendezése.
- Gyakorlás: Oracle Példatár 1.fejezet feladatai:

  http://people.inf.elte.hu/sila/eduAB/Feladatok.pdf