

Relacijska algebra in SQL

Relacijski model

- ▶ star več kot 35 let,
- ▶ uporablja se v večini večjih poslovnih sistemov,
- ▶ enostaven za razumevanje, pregleden,
- ▶ omogoča zmogljive poizvedbe v standardiziranem jeziku SQL,
- ▶ podpira učinkovite implementacije.

Relacijska algebra in SQL

- ▶ *Relacijska algebra* je matematični opis operacij nad relacijami (tabelami).
- ▶ Jezik SQL (Standard Query Language) je implementacija relacijske algebre v obliki proizvedovalnega jezika.
- ▶ Operatorji so operacije, ki sprejmejo relacije (tabele) in vrnejo (nove) relacije (tabele).

Operatorji relacijske algebre

- ▶ *Schema relacije* = definicija tabele (imena + tipi).
- ▶ Operatorji so odvisni od shem relacij nad katerimi jih izvajamo.
- ▶ $\sigma_p(R)$ - izberi vrstice v relaciji R , ki ustrezajo pogoju p . Pogoju je lahko logični izraz. Schema vrnjene tabele je ista.
- ▶ $\pi_{a_1, a_2, \dots, a_n}(R)$ - izberi stolpce z imeni a_1, a_2, \dots, a_n relacije R in vrni novo tabelo s shemo, ki jo določajo definicije teh stolpcev. Število vrstic ostane enako.
- ▶ $\rho_{a/b}(R)$ - spremeni ime stolpcu a v b . Vrni enako tabelo (glede vrstic), le z drugo shemo.
- ▶ $R \cup S$ - vrni relacijo z unijo vrstic, če imata relaciji R in S enaki shemi.
- ▶ $R \setminus S$ - vrni relacijo z razliko vrstic, če imata relaciji R in S enaki shemi.
- ▶ $R \times S$ - vrni kartezični produkt relacij (vsaka vrstica R z vsako vrstico S). Schema rezultata sta združeni shemi.

JOIN

$$R \bowtie S = \pi_{schema(R) \cup schema(S)}(\sigma_{R.a_1=S.a_1 \wedge R.a_2=R.a_2 \wedge \dots}(R \times S))$$

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

Dept

DeptName	Manager
Finance	George
Sales	Harriet
Production	Charles

Employee \bowtie Dept

Name	EmpId	DeptName	Manager
Harry	3415	Finance	George
Sally	2241	Sales	Harriet
George	3401	Finance	George
Harriet	2202	Sales	Harriet

Vir: Wikipedia.

SQL

- ▶ Standard Query Language.
- ▶ Primeri iz tabele na SQLZOO
 - ▶ http://sqlzoo.net/wiki/SELECT_from_WORLD_Tutorial
 - ▶ http://sqlzoo.net/wiki/SELECT_from_BBC_Tutorial
 - ▶ http://sqlzoo.net/wiki/SELECT_from_Nobel_Tutorial
- ▶ Ogledali si bomo stavke SELECT, INSERT, UPDATE, DELETE.

SELECT

Stavek SELECT kot projekcija.

- ▶ Izberi stolpca population in name iz tabel world:

```
SELECT population, name FROM world
```

- ▶ Izberi stolpca name in population in jih preimenuj v drzava ter st_prebivalcev.

```
SELECT name AS drzava, population AS st_prebivalcev  
FROM world
```

- ▶ Izberi stolpec name in ga preimenuj v Ime države.

```
SELECT name AS "Ime države" FROM world
```

- ▶ To v splošnem ni dobra praksa (šumniki v imenih, zgodovinsko, ...)

SELECT

- ▶ Katere različne regije nastopaj v tabeli bbc?

```
SELECT DISTINCT region FROM bbc
```

- ▶ Kateri različni kontinenti nastopajo v tabeli world?

```
SELECT DISTINCT continent FROM world
```


SELECT ... WHERE

Stavek `SELECT ... WHERE` kot projekcija z izbiro vrstic v skladu s pogoji.

- ▶ Vse vrstice v tabeli `world`, ki pripadajo državam v Evropi

```
SELECT * FROM world WHERE continent = 'Europe'
```

- ▶ Možni relacijski operatorji so:
 - ▶ `=`, `<>`, `!=`, `<`, `<=`, `>`, `>=`
 - ▶ `IS NULL`, `IS NOT NULL`
 - ▶ `e BETWEEN a AND b`
 - ▶ `e NOT BETWEEN a AND b`
 - ▶ `e IN (v1, v2, ...)`
 - ▶ `e NOT IN (v1, v2, ...)`
- ▶ Pogoje lahko sestavljamo z logičnimi vezniki : `NOT`, `AND`, `OR`, `XOR`

Izrazi v SELECT

- ▶ Za vsako državo v tabeli world izračunaj razmerje med prebivalstvom in površino.

```
SELECT name, population / area
FROM world
WHERE continent = 'Europe'
```

- ▶ Še z ustreznim preimenovanjem.

```
SELECT name AS ime_drzave,
       population / area AS gostota_prebivalstva
FROM world
WHERE continent = 'Europe'
```

Izrazi v SELECT

- Dodajmo še pogoj da je število prebivalcev večje od 2mio.

```
SELECT name AS "ime države",  
       ROUND(population / area, 2)  
       AS "gostota prebivalstva"  
FROM world  
WHERE continent = 'Europe' AND population > 2000000
```

- Funkcije, ki jih lahko uporabljamo.

```
SELECT name, ROUND(population/1000000) AS prebiMilion  
FROM world  
WHERE continent IN ('Asia', 'Europe')  
      AND name LIKE 'C%'
```

ORDER BY - urejanje tabel

- Urejanje po stolpcu population v tabeli world

```
SELECT name, population  
FROM world  
ORDER BY population
```

- Urejanje po 2. stolpcu.

```
SELECT name, population  
FROM world  
ORDER BY 2
```

- V padajočem vrstnem redu.

```
SELECT name, population  
FROM world  
ORDER BY population DESC
```

ORDER BY - urejanje tabel

- ▶ Urejanje po izračunanem stolpcu.

```
SELECT name  
FROM world  
WHERE continent = 'Europe'  
ORDER BY population/area
```

- ▶ Urejanje po continent in potem še po name.

```
SELECT continent, name  
FROM world  
ORDER BY continent, name
```

Podpoizvedbe

- ▶ Uporabimo rezultat poizvedbe za izračun pogojev v drugi poizvedbi.
- ▶ Katere države na svetu imajo manj prebivalcev kot Slovenija? Podatki iz tabele world.

```
SELECT name FROM world WHERE  
population <= (SELECT population  
FROM world WHERE name='Slovenia')
```

- ▶ Katere države na svetu imajo več ali enako prebivalcev kot Kanada in manj ali enako kot Alžirijo?

```
SELECT name FROM world WHERE population BETWEEN  
(SELECT population FROM world WHERE name='Canada')  
AND  
(SELECT population FROM world WHERE name='Algeria')
```

Podpoizvedbe

- ▶ Podpoizvedbe morajo imeti ustrezno število stolpcev in vrstic glede na uporabljene operatorje.
- ▶ V katerih letih je bila podeljena Nobelova nagrada za fiziko in ni bila za kemijo? Podatki iz tabel nobel

```
SELECT DISTINCT yr
FROM nobel
WHERE subject = 'physics' AND
      yr NOT IN (SELECT yr FROM nobel
                  WHERE subject = 'chemistry')
```

Združevalne funkcije

- Povprečno število prebivalcev na državo v Evropi. Podatki iz tabele world.

```
SELECT AVG(population)
FROM world
WHERE continent='Europe'
```

- Maksimalno število prebivalcev v afriški državi.

```
SELECT MAX(population)
FROM world
WHERE continent='Africa'
```


Združevalne funkcije

- ▶ Najmanjša površina države na svetu.

```
SELECT MIN(area) FROM world
```

- ▶ Zakaj je enaka 0?

```
SELECT name, area FROM world  
WHERE area = 0
```

```
SELECT MIN(area) FROM world WHERE area > 0
```

```
SELECT SUM(gdp) FROM world  
WHERE continent='Europe'
```

- ▶ Nekatere vrstice imajo polje gdp enako NULL. Funkcije za združevanje ignorirajo vrednosti NULL.

COUNT

- ▶ Koliko je vrstic v tabeli world?

```
SELECT COUNT(*) FROM world
```

- ▶ Koliko je vrstic v tabeli world, ki imajo gdp različen od NULL?

```
SELECT COUNT(gdp) FROM world
```

- ▶ Koliko je različnih kontinentov?

```
SELECT COUNT(DISTINCT continent) FROM world
```

- ▶ Koliko krat se pojavi beseda Asia v kontinentu?

```
SELECT COUNT(*) FROM world  
WHERE continent LIKE '%Asia%'
```

Primeri

- ▶ Kako pa je z uporabo združevalnih funkcij v pogoju? Podatki iz tabele world.

```
SELECT name, population
FROM world
WHERE continent='Africa' AND
      population = MAX(population)
```

- ▶ Funkcije za združevanje lahko uporabljamo le v prvem delu stavka SELECT.

```
SELECT name, population
FROM world
WHERE continent = 'Africa' AND
      population = (SELECT MAX(population)
                    FROM world
                    WHERE continent = 'Africa'
                    )
```

Primeri

- Poišči imena tistih držav, ki imajo bruto družbeni proizvod večji od vseh evropskih držav. Podatki iz tabele world.

```
SELECT name FROM world
WHERE gdp > (SELECT MAX(gdp)
             FROM world
             WHERE continent = 'Europe')
```

```
SELECT name FROM world
WHERE gdp > ALL (SELECT gdp FROM world
                 WHERE continent = 'Europe'
                 AND gdp IS NOT NULL)
```

- Za primerjavo z NULL moramo vedno uporabiti IS NULL ali IS NOT NULL. Nikoli = ali <>

Primeri

- V tabeli bbc Poiščimo države z maksimalnim številom prebivalstva v svoji regiji.

```
SELECT region, name, population FROM bbc t1
WHERE population >= ALL
  (SELECT population FROM bbc t2
   WHERE t1.region = t2.region
    AND population > 0)
```

- Podpoizvedbo si predstavljamo, kot da je pri filtriranju posamezne vrstice parametrizirana z t1.region.

GROUP BY

- ▶ Maksimalno število prebivalcev države na vsaki celini. Podatki iz tabele world.

```
SELECT continent, population FROM world x
WHERE population >= ALL
  (SELECT population FROM world y
   WHERE y.continent = x.continent
    AND population > 0)
```

- ▶ GROUP BY - razdeli tabelo na skupine definirane z istimi vrednostmi stolpcev, ki so navedeni za GROUP BY.
- ▶ Vsaka skupina vrne kot rezultat eno vrstico. Zato morajo biti morebitni ostali stolpci navedeni v SELECT agregirani s kako od združevalnih funkcij.

```
SELECT continent, MAX(population)
FROM world
GROUP BY continent
```

HAVING

- Po posameznih regijah preštej tiste države, kjer je število prebivalcev več kot 200mio. Podatki iz tabele world.

```
SELECT continent, COUNT(*) AS kolikoDrzav  
FROM world  
WHERE population > 200000000  
GROUP BY continent
```

- Najprej smo izbrali ustrezne vrstice in jih nato pošteli.
- Kateri kontinenti imajo več kot 500 milijonov prebivalcev?
- Pozor: WHERE nam tu ne more pomagati!

```
SELECT continent, SUM(population)  
FROM world GROUP BY continent  
HAVING SUM(population) >= 500000000
```

HAVING

- ▶ HAVING je dejansko WHERE nad vrsticami, ki predstavljajo skupine dobljene z GROUP BY.
- ▶ Akumulirani stolpci uporabljeni v pogoju niso nujno v rezultatu.

```
SELECT continent  
FROM world  
GROUP BY continent  
HAVING SUM(population) >= 500000000
```


Primeri

- ▶ Podatki iz tabele nobel.
- ▶ Izpiši tista leta po letu 1970, ko je Nobelovo nagrado iz fizike (Physics) dobil le en posameznik.

```
SELECT yr FROM nobel
WHERE subject = 'Physics' AND yr > 1970
GROUP BY yr
HAVING COUNT(yr) = 1
```

- ▶ Kateri posamezniki so dobili Nobelovo nagrado na dveh ali več področjih?

```
SELECT winner FROM nobel
GROUP BY winner
HAVING COUNT(DISTINCT subject) > 1
```

Primeri

- ▶ Podatki iz tabele nobel.
- ▶ Prikaži tista leta in področja, kjer so bile v istem letu podeljene 3 nagrade ali več. Upoštevaj le leta po letu 2000.

```
SELECT yr, subject
FROM nobel
WHERE yr > 2000
GROUP BY yr, subject
HAVING COUNT(*) >= 3
ORDER BY yr
```

JOIN

- ▶ Podatki iz tabel movie, actor in casting.
- ▶ V katerih filmih je igral John Wayne?

```
SELECT title FROM movie
  JOIN casting ON movie.id=movieid
  JOIN actor   ON actorid=actor.id
 WHERE actor.name='John Wayne'
```

- ▶ Z enim ali več JOIN združimo potrebne tabel in na ta način posredno izvajamo sklicevanje med tabelami.
- ▶ Kdo je poleg Jamesa Deana še igral v filmu Giant?

```
SELECT name FROM movie
  INNER JOIN casting ON movie.id = movieid
  INNER JOIN actor   ON actor.id = actorid
 WHERE title = 'Giant' AND name <> 'James Dean'
```

Primer

- ▶ Izpiši tiste igralce, ki so bili glavni igralci (`ord = 1`) v vsaj 10 filmih?

```
SELECT actor.name FROM actor
  INNER JOIN casting ON actorid = id
WHERE ord = 1
GROUP BY actorid
HAVING COUNT(id) >= 10
```

- ▶ Pozor: ali lahko v `SELECT` izberemo stolpec, ki ne nastopa v `GROUP BY`?
- ▶ Glede na to da, vemo da se za nek `actorid` pojavi lahko samo eno ime `actor.name` lahko naredimo takole:

```
SELECT actor.name FROM actor
  INNER JOIN casting ON actorid = id
WHERE ord = 1
GROUP BY actorid, actor.name
HAVING COUNT(id) >= 10
```

Primer

- Zanima nas še v koliko filmih so bili glavni ti igralci glavni igralci?

```
SELECT actor.name, count(actor.name) AS 'FILMI' FROM actor
  JOIN casting ON actorid = id
 WHERE ord = 1
  GROUP BY actorid, actor.name
  HAVING COUNT(id) >= 10
  ORDER BY FILMI DESC
```

Primer

- Kateri igralci so igrali v več kot enem filmu, ki ima v naslovu 'love'?

```
SELECT name, COUNT(*) from movie
  JOIN casting ON movie.id = movieid
  JOIN actor  ON actor.id = actorid
WHERE title LIKE '%love%'
GROUP BY name
HAVING COUNT(*) > 1
```

Primer

- ▶ Zanimajo nas naslovi in glavni igralec vseh tistih filmov, kjer je igral Al Pacino in ni bil v glavni vlogi.

```
SELECT movie.title, actor.name FROM movie
  JOIN casting ON movie.id = movieid
  JOIN actor ON actor.id = actorid
WHERE casting.ord = 1 AND movie.id IN
  (SELECT movieid FROM casting
    JOIN actor ON actor.id = actorid
    WHERE actor.name = 'Al Pacino'
  ) AND actor.name <> 'Al Pacino'
```

INSERT {.build} - vstavljanje vrstic

- ▶ INSERT - stavek za vstavljanje vrstic.

```
INSERT INTO ime_tabele VALUES  
    (vrednost_prvega_stolpca, ... , vred_zad_st)
```

- ▶ Naštejemo vse vrednosti za vse stolpce, tudi če so NULL ali avtomatično generirani.
- ▶ Poznati moramo vrstni red stolpcev v shemi.

```
INSERT INTO ime_tabele (stolpec_1, stolpec_2, ..., stolpc_n)  
    VALUES (vred_stolpca_1, ... , vred_stolpca_n)
```

- ▶ Naštejemo le vrednosti za izbrane stolpce
- ▶ Ostali se nastavijo na NULL ali privzeto vrednost.

INSERT

- ▶ za VALUES lahko naštejemo več vektorjev vrednosti (vrstic) in jih ločimo z vejico.
- ▶ Vstavljamo lahko tudi rezultat SELECT stavka, če se ujema s shemo tabele.

```
INSERT into ime_tabele  
      SELECT ...
```

UPDATE {.build} - popravljanje vrstic

- popravljanje vrednosti v tabeli v vrsticah, ki zadoščajo pogoju ter stolpcih v teh vrsticah, ki jih želimo spremeniti.

```
UPDATE ime_tabele SET st1 = v1, st2 = v2, ...  
WHERE pogoj
```

DELETE {.build} - brisanje vrstic

- brisanje vrstic, ki zadoščajo pogoju.

```
DELETE FROM ime_tabele  
WHERE pogoj
```