SQL

SQL --- Structured Query Language

SQL je v súčasnosti najpoužívanejší dotazovací jazyk.

- štandard už z 80. tych rokov (1986); posledných 10 rokov sa štandard veľmi nemenil
- aj keď existuje štandard, „SQL“ je rôzne pre každý databázový systém
- dotazy napísané pre jeden databázový systém (PosgreSQL) nemusia fungovať v inom (MySQL), hlavne kvôli rôznym užitočným rozšíreniam SQL, ktoré poskytuje daný databázový systém
- súčasťou je Data Manipulation Language (DML) a Data Definition Language (DDL)

Základná syntax:

- **SELECT** <zoznam atribútov>
  FROM <zoznam relácií>
  WHERE <podmienka>
  [ORDER BY atribút1, atribút2...]
  [LIMIT 100] [OFFSET 0]

Príklad SQL dotazu

```
SELECT
   concat(e.firstname,’ ‘,e.lastname) AS ename,
   (CASE
     WHEN e.comm IS NULL THEN e.sal
     ELSE e.comm + e.sal
   ) AS ‘total_salary’
FROM emp
WHERE deptno>=20 AND lower(e.firstname)=‘john’
```

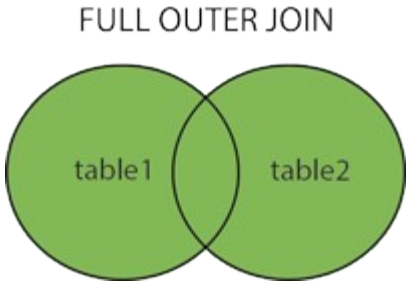
Multimnožiny

- Jazyk SQL uvažuje relácie ako multimnožiny, čiže môžu obsahovať duplikáty riadkov (na rozdiel od Datalogu).
- Ak duplikáty nechceme, musíme si to dodatočne vynútiť (obmedzením **UNIQUE** pri vytváraní tabuliek --- viac o tom neskôr pri DDL --- a pomocou **DISTINCT** v dotazoch).

JOINs

- *join je s*pojenie dvoch tabuliek; je to podmnožina kartoziánskeho súčinu tabuliek špecifikovaná dodatočnými podmienkami na prepájanie (karteziánsky súčin – každý riadok s každým)
- FULL JOIN
- INNER JOIN alebo iba JOIN
- LEFT JOIN
- RIGHT JOIN

Kartézsky súčin (FULL JOIN):



Name	Deptno
John	10
Thomas	20
Joe	40

X

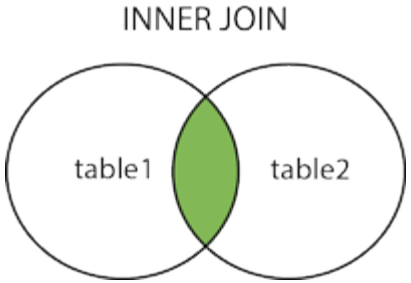
Deptno	Dept. name
10	Accounting
20	PR
30	Development

=

Name	Deptno	Deptno	Dept. name
John	10	10	Accounting
John	10	20	PR
John	10	30	Development
Thomas	20	10	Accounting
Thomas	20	20	PR
Thomas	20	30	Development
Joe	40	10	Accounting
Joe	40	20	PR
Joe	40	30	Development

SELECT * FROM emp, dept

INNER JOIN = JOIN:



Name	Deptno
John	10
Thomas	20
Joe	40

JOIN

Deptno	Dept. name
10	Accounting
20	PR
30	Development

=

Name	Deptno	Deptno	Dept. name
John	10	10	Accounting
~~John~~	~~10~~	~~20~~	~~PR~~
~~John~~	~~10~~	~~30~~	~~Development~~
~~Thomas~~	~~20~~	~~10~~	~~Accounting~~
Thomas	20	20	PR
~~Thomas~~	~~20~~	~~30~~	~~Development~~
~~Joe~~	~~40~~	~~10~~	~~Accounting~~
~~Joe~~	~~40~~	~~20~~	~~PR~~
~~Joe~~	~~40~~	~~30~~	~~Development~~

```
SELECT * FROM emp e, dept d
WHERE e.deptno = d.deptno
```

```
SELECT * FROM emp e
    JOIN dept d ON e.deptno = d.deptno
```

```
SELECT * FROM emp e natural join dept as d
```

INNER JOIN = JOIN:

Name	Deptno
John	10
Thomas	20
Joe	40

JOIN

Deptno	Dept. name
10	Accounting
20	PR
30	Development
10	Human res.

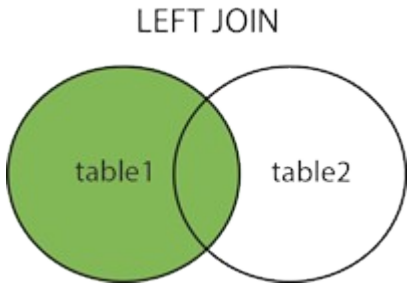
=

Name	Deptno	Deptno	Dept. name
John	10	10	Accounting
~~John~~	~~10~~	~~20~~	~~PR~~
~~John~~	~~10~~	~~30~~	~~Development~~
John	10	10	Human res.
~~Thomas~~	~~20~~	~~10~~	~~Accounting~~
Thomas	20	20	PR
~~Thomas~~	~~20~~	~~30~~	~~Development~~
~~Thomas~~	~~20~~	~~10~~	~~Human res.~~
~~Joe~~	~~40~~	~~10~~	~~Accounting~~
~~Joe~~	~~40~~	~~20~~	~~PR~~
~~Joe~~	~~40~~	~~30~~	~~Development~~
~~Joe~~	~~40~~	~~10~~	~~Human res.~~

```
SELECT *
FROM emp e
    JOIN dept d ON e.deptno = d.deptno
```

Ako by ste zapísali JOIN v Datalogu?

LEFT [OUTER] JOIN:



Name	Deptno
John	10
Thomas	20
Joe	40

LEFT JOIN

Deptno	Dept. name
10	Accounting
20	PR
30	Development
10	Human res.

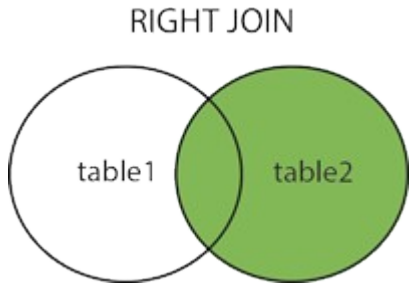
=

Name	Deptno	Deptno	Dept. name
John	10	10	Accounting
John	10	10	Human res.
Thomas	20	20	PR
Joe	40	null	null

```
SELECT *
FROM emp as e
    LEFT JOIN dept as d
        ON e.deptno = d.deptno
```

Ako by ste zapísali LEFT JOIN v Datalogu?

RIGHT [OUTER] JOIN:



Deptno	Dept. name
10	Accounting
20	PR
30	Development
10	Human res.

RIGHT JOIN

Name	Deptno
John	10
Thomas	20
Joe	40

=

Name	Deptno	Deptno	Dept. name
John	10	10	Accounting
John	10	10	Human res.
Thomas	20	20	PR
Joe	40	null	null

To isté ako LEFT JOIN, akurát v obrátenom poradí

```
SELECT *
FROM dept AS d
    RIGHT JOIN emp AS e ON e.deptno = d.deptno
```

Operátory, výrazy a funkcie

- v podmienke za WHERE môžete používať operátory:
  - =, <>, >, <, >=, <=, BETWEEN, LIKE, IN, IS NULL, IS NOT NULL
  - AND, OR, ! (NOT)
- taktiež môžete používať aritmetické výrazy a hromadu ďalších funkcií
  - napr. *concat(e.firstname,' ', e.lastname)*
  - funkcie na prácu s dátumami, číslami a pod.
  - zoznam podporovaných funkcií a ich syntax závisí na konkrétnom databázovom systéme
    - https://www.postgresql.org/docs/current/static/functions.html

Vnorený SELECT (subselect)

- SELECT name FROM emp e WHERE e.ID IN (SELECT ID FROM managers)
- SELECT name FROM emp e WHERE EXISTS (SELECT * FROM managers WHERE id=e.id)

- v prípade vnorených selectov si treba dávať pozor na efektívnosť
  - **JOIN** operácie vie databázový systém vcelku dobre optimalizovať
    (ak máte správne navrhnutú DB – o tom neskôr)
  - aj **EXISTS** a **NOT EXISTS** sú priamočiare
  - optimalizovať **IN** a **NOT IN** môže byť problém, ľahšie napíšete „neefektívny“ dotaz
- premyslite si, že **EXISTS** možno vždy prepísať ako **JOIN**; **NOT EXISTS** ako rozdiel (**EXCEPT**)

Množinové operácie: UNION, EXCEPT

- SELECT name
  FROM emp_dallas WHERE sal>=1000

  UNION [ALL]

  SELECT name
  FROM emp_huston WHERE sal>=500

- počet a typy atribútov v jednotlivých častiach musia byť rovnaké
- UNION je zjednotenie, EXCEPT je rozdiel (multi)množín

Pomocné tabuľky a CTE

```
WITH emp_houston AS (
    SELECT * FROM emp as e, dept as d
    WHERE e.deptno=d.deptno and d.dname='houston'
)
SELECT * FROM emp_houston WHERE sal>=1000
```

- CREATE TEMPORARY TABLE emp_houston (
  SELECT * FROM emp as e, dept as d
  WHERE e.deptno=d.deptno and d.dname='houston'
  );
- SELECT * FROM emp_houston WHERE sal>=1000

PostgreSQL

- na cvičeniach budeme pracovať s databázovým systémom PostgreSQL
- väčšina databázových systémov má architektúru klient-server
  - **server**
    - obsahuje dáta
    - rozumie SQL dotazom
    - klienti sa na neho pripájajú väčšinou cez socket (TCP/IP), príp. named pipes či iné kanály podporované OS
  - **klient**
    - aplikácia, ktorá potrebuje pracovať s dátami
    - serveru posiela dotazy v SQL jazyku
    - zobrazí / spracuje odpoveď od servera

Niektoré funkcie možno implementovať na ľubovoľnej strane (napr. stránkovanie).

PostgreSQL

- máme otvorené dve terminálové okná na cvika.dcs.fmph.uniba.sk
  - v jednom okne editujeme súbor so zadaním, napr. **vim queries.sql**
  - v druhom okne editovaný súbor spustíme (všetky dotazy v ňom) príkazom

    psql -f queries.sql

  - neuškodí tretie okno, kde spustíme **psql** a používame ho na prezeranie obsahu databázy a ladenie dotazov

- každý pracuje nad svojou databázou, ktorá je automaticky vybratá po spustení psql

Práca s konzolou PostgreSQL

- konzolu (interaktívny terminál) PostgreSQL sputíte príkazom **psql**
- v konzole následne môžete písať dotazy, napr. **SELECT * FROM emp;**
- zaujímavé špeciálne príkazy:
  - \d emp resp. \d+ emp - zobrazí štruktúru tabuľky
  - \d - zobrazí zoznam tabuliek v aktuálnej databáze
  - \db - zobrazí zoznam databáz
  - \c emp - pripojí sa k databáze emp
  - \q - ukončenie konzoly
- dokumentácia konzolových príkazov: http://www.postgresql.org/docs/current/static/app-psql.html