# Úvod do SQL

Štandardizovaný jazyk pre manipuláciu s dátami (fakticky SQL zahrnuje ako DML tak DDL)

**S**tructured **Q**uery **L**anguage

Existuje viacero noriem : SQL92, SQL2, SQL3.
Databázové systémy podporujú rôzne podmnožiny a rozšírenia týchto štandardov, väčšinou všetko čo je tu uvedené.

**Základný tvar:** (všeliké detaily sú zanedbané)

Select zoznam atribútov
From zoznam relácii (ak ich je viac kartézsky súčin alebo join)
Where selekčná podmienka (zahrnuje aj spájacie podmienky);



# Selekcia

```
SELECT  *
FROM    Podniky
WHERE   okres = "Senec" AND cena > 50;
```

Dá sa použiť:
- mená atribútov za FROM.
- porovnávacie operátory: =, <>, <, >, <=, >=
- aritmetické operátory: +, -, *, /
- operácie s reťazcami *napr. zreťazenie ( //, & )*
- logické spojky: NOT, AND, OR
- porovnanie regulárnych výrazov: s LIKE p
- špeciálne funkcie pre dátum a čas



# Projekcia a premenovanie

Výber podmnožiny atribútov:

```
SELECT  meno, cena
FROM    Podniky
WHERE   okres= "Senec" AND cena  > 50;
```

Premenovanie atribútov vo výslednej tabuľke:

```
SELECT  meno AS názov, cena AS ponuka
FROM    Podniky
WHERE   okres = "Senec" AND cena > 50;
```



# Usporiadanie výsledkov

```
SELECT  meno, cena
FROM    Podnik
WHERE   okres = "Senec" AND cena > 50
ORDER BY  okres DESC, meno ASC;
```

Usporiadanie je rastúce, pokiaľ nepoužijeme kľúčové slovo DESC, určujúce klesajúce usporiadanie.

V zozname za ORDER BY sa môže vyskytnúť aj viac atribútov, pre každý nezávisle môže byť vzostupné alebo klesajúce utriednie.

## Spojenia (joins)

```
SELECT   meno, sklad
FROM     Osoba, Kontrakt
WHERE    meno = "Kováč" AND mesto = "Trnava"
         AND výrobok = "meč"
```

Produkt ( meno, cena, kategória, výrobca)
Kontrakt (predávajúci, kupujúci, sklad, výrobok)
Podnik (meno, adresa, okres, cena)
Osoba (meno, rodné_číslo, telefón, mesto)



## Jednoznačnosť atribútov

Nájdite mená osôb, ktoré si kúpili elektroniku:

```
SELECT   Osoba.meno
FROM     Osoba, Kontrakt, Produkt
WHERE    Osoba.meno = kupujúci
           AND  výrobok = Produkt.meno
           AND  Produkt.kategória = "elektronika";
```

Produkt ( meno, cena, kategória, výrobca)
Kontrakt (predávajúci, kupujúci, sklad, výrobok)
Osoba (meno, rodné_číslo, telefón, mesto)



## n-ticové premenné

Dotaz na dvojice podnikov vyrabajúcich tie isté kategórie tovarov.

```
SELECT   P1.výrobca, P2.výrobca
FROM     Produkt AS P1, Produkt AS P2
WHERE    P1.kategória = P2. kategória
         AND P1. výrobca <> P2. výrobca;
```

Produkt ( meno, cena, kategória, výrobca)



## Zjednotenie, prienik a rozdiel

```
(SELECT  meno
FROM     Osoba
WHERE    mesto = "Trnava")
UNION
(SELECT  meno
FROM     Osoba, Kontrakt
WHERE    kupujúci = name AND sklad = "Zohor");
```

Podobne sa používa prienik INTERSECT a rozdiel EXCEPT.
Tabuľky musia mať tie isté atribúty (inak ich treba premenovať).

# Poddotazy

```
SELECT  Kontrakt.výrobok
FROM    Kontrakt
WHERE   kupujúci =
        (SELECT  meno
         FROM    Osoba
         WHERE   rodné_číslo  = "450626/7887");
```

Poddotaz môže vrátiť najviac jednu hodnotu.
Inak nastane chyba: run-time error.



# Poddotazy vracajúce tabuľku

Najdite spoločnosti ktorých produkty kupuje Jožko Bomba.

```
SELECT  Podnik.meno
FROM    Podnik, Produkt
WHERE   Podnik.meno = výrobca
        AND  Produkt.meno  IN
             (SELECT výrobok
              FROM   Objednávka
              WHERE  kupujúci = "Jožko Bomba");
```

Dá sa použiť aj: s > ALL R
s > ANY R
EXISTS R (true ak R je neprázdne, false ak R je prázdne)



# Podmienky na n-tice

```
SELECT  Podnik.meno
FROM    Podnik, Produkt
WHERE   Podnik.meno = výrobca
        AND  (Produkt.meno, cena)  IN
             (SELECT výrobok, cena
              FROM    Kontrakt
              WHERE   kupujúci = "Jožko Bomba");
```



# Opakovaný výskyt tabuľky v dotaze

Nájdite názvy filmov, ktoré sa vyskytli viac než raz.

```
SELECT  názov
FROM    Film AS F1
WHERE   rok < ANY
              (SELECT  rok
               FROM    Film
               WHERE   názov  =  F1.title);
```

Film (názov, rok, režisér, dĺžka)

Názvy filmov neidentifikujú film (názov sa môže použiť znovu v niektorom z neskorších rokov).

Všimnite si platnosť (scope) premenných !

# Odstránenie duplikátov

SELECT DISTINCT *P1.výrobca, P2.výrobca*
FROM *Produkt* AS *P1, Produkt* AS *P2*
WHERE *P1.kategória = P2. kategória*
AND *P1. výrobca <> P2. výrobca*;

Produkt ( meno, cena, kategória, výrobca)



# Zachovánie duplikátov

Operátory UNION, INTERSECT a EXCEPT pracujú s množinami a nie s multimnožinami.

(SELECT meno
FROM Osoba
WHERE Mesto = “Trnava”)

UNION ALL

(SELECT meno
FROM Osoba, Kontrakt
WHERE kupujúci = meno AND sklad = “Zohor”);



# Agregačné funkcie

SELECT Sum(cena)
FROM Produkt
WHERE výrobca = “Vinárske Závody”

SQL obsahuje viacero agregačných funkcií:

SUM, MIN, MAX, AVG, COUNT, ...

S výnimkou COUNT, všetky agregačné funkcie sa aplikujú na jeden atribút.

SELECT Count(*)
FROM Kontrakt



# Zoskupenie a agregácia

Väčšinou chceme aplikovať agregačné funkcie na časti tabuľky.

Zistite objem predaja jednotlivých výrobkov.

SELECT *Produkt.meno,* Sum(*cena*)
FROM *Produkt, Kontrakt*
WHERE *Produkt.meno = Kontrakt.výrobok*
GROUP BY *Produkt.meno*

1. Vypočíta sa relácia (t.j., SELECT ... FROM ... WHERE ...).
2. Výsledok sa zoskupí podľa atribútov za GROUP BY.
3. Aplikuje sa agregačná funkcia na každú skupinu (jedna n-tica a agregát).

SELECT môže obsahovať (1) zoskupené atribúty (2) agregáty, t.j. všetky atribúty v GROUP BY alebo v agregačných funkciach.

# Podmienky na agregáty

Podobná otázka ako predošlá, ale len na tie produkty, ktoré majú viac ako 100 kupujúcich.

```
SELECT      výrobok, Sum(cena)
FROM        Produkt,  Kontrakt
WHERE       Produkt.meno = výrobok
GROUP BY  výrobok
HAVING      Count(kupujúci) > 100
```

Za HAVING nasledujú podmienky na agregované skupiny.



# Aktualizácia databázy

Sú tri druhy aktualizácie: vkladanie, vynechanie, zmena.

Všeobecný tvar vloženia:

INSERT INTO R(A1,…., An) VALUES (v1,…., vn)

Insert a new purchase to the database:

```
INSERT INTO   Kontrakt(predávajúci, kupujúci, výrobok, sklad)
        VALUES  (Jožo, Fero, rádiobudík, "Tesko-2")
```

Môžeme vynechať mená atribútov, ak uvedieme všetky v správnom poradí.

Ak neuvedieme všetky atribúty, budú namiesto neuvedených atribútov hodnoty NULL.



# Vloženie výsledku dotazu

```
INSERT   INTO   PRODUKT(name)
    SELECT  DISTINCT  výrobok
    FROM     Kontrakt
    WHERE  výrobok  NOT IN
                (SELECT  meno
                  FROM     Produkt)
```

Dotaz nahradzuje rezervované slovo VALUES.
*Dotaz nasleduje za príkazom vloženia .*



# Vynechanie

```
DELETE    FROM    Kontrakt
WHERE    predávajúci = "Jožo"   AND
              výrobok  = "holiaci strojček"
```

*Syntax podmienky je ako vo formule selekt.*

V SQL neexistuje žiadny spôsob odstrániť len jeden výskyt n-tice, ktorá sa v tabuľke vyskytuje viackrát.

# Modifikácia (zmena)

```
UPDATE Produkt
SET    cena = cena*(1 – 0.1)
WHERE  Produkt.meno  IN
            (SELECT výrobok
             FROM    Akcie
             WHERE  dátum = today);
```

Akcie(výrobok, dátum)



# Definícia dát v SQL

Definícia dát: definuje schému (množinu tabuliek).

- Vytvorenie tabuliek
- Odstránenie tabuliek
- Modifikácia schémy tabuliek

Najprv ale musíme:
Definovať typy dát.

Nakoniec: definujeme indexey a trigre.



# Typy dát v SQL

| | |
|---|---|
| • Reťazce (pevnej alebo premennej dĺžky) | CHAR, VARCHAR |
| • Bitové reťazce | Integer, SHORTINT |
| • Pohyblivá čiarka | Real, Double |
| • Dátum a čas | Date/Time |

Domény sa používajú pri definícii tabuliek.

Definícia domény:

```
CREATE DOMAIN  adresa  AS  VARCHAR(55);
```



# Definíciatabu ľky

```
CREATE  TABLE  Osoba(
        meno                VARCHAR(30),
        rodné_číslo         INTEGER,
        vek                 SHORTINT,
        mesto               VARCHAR(30),
        pohlavie            BIT(1),
        Dátum_narodenia     DATE
);
```

# Odstránenie a modifikácia tabuľky

Odstránenie: DROP Osoba;

Modifikácia:

```
ALTER TABLE   Osoba
        ADD   telefón  CHAR(16);

ALTER  TABLE   Osoba
        DROP  vek;
```



# Počiatočné (preddefinované) hodnoty

Preddefinovaná počiatočná hodnota je NULL.

Určenie počiatočných hodnôt:

```
CREATE TABLE  Osoba (
        meno            VARCHAR(30),
        rodné_číslo     INTEGER,
        vek             SHORTINT  DEFAULT 100,
        mesto             VARCHAR(30)  DEFAULT   “Trnava”,
        pohlavie                CHAR(1)   DEFAULT  “?”,
        Dátum_narodenia     DATE
);
```



# Indexy

Indexy sú veľmi dôležité pre urýchlenie dotazov. Na druhej strane spomalujú aktualizácie a zväčšujú rozsah databázy.

Majme tabuľku:

Osoba (meno, rodné_číslo, vek, mesto)

Index na “rodné_číslo” umožňuje prístup k riadku z daným rodným číslom rýchlejšie ako pri sekvenčnom prehľadávaní tabuľky).

Problém výberu indexov súvisí s návrhom a prevádzkovaním bázy dát. (Jeho exaktné riešenie je veľmi ťažké.)



# Vytvorenie indexov

CREATE INDEX  irc ON Osoba(rodné_číslo)

Môžeme vytvárať indexy aj pre viacero atribútov:

```
CREATE INDEX dvojitý ON
                Osoba (meno, rodné_číslo)
```

*Prečo sa automaticky neindexuje všetko ?*

# Pohľady (views)

Pohľady sú dotazy zapamätané v databáze. Používajú sa ako virtuálne tabuľky (za FROM). Možno na ne špecifikovať prístupové práva, aj indexy.
Sú užitočné pri štruktúrovaní zložitých dotazov. Umožňujú, aby ich prostredníctvom rôzní užívatelia videli DB rôzne.

Pohľad: kontrakty na elektronické výrobky:

```
CREATE VIEW  nákupy_elektroniky AS
  SELECT výrobok, kupujúci, predávajúci, sklad
  FROM  Kontrakt, Produkt
  WHERE  Kontrakt.výrobok = Produkt.meno
         AND  Produkt.kategória = “elektronika”
```



# Použitie pohľadu

```
CREATE VIEW     Trnavskí_zákazníci  AS
        SELECT   kupujúci, predávajúci, výrobok, sklad
        FROM     Osoba, Kontrakt
        WHERE  Osoba.mesto = “Trnava”    AND
               Osoba.meno = Kontrakt.kupujúci
```

Neskoršie použitie pohľadu:

```
SELECT   meno, sklad
FROM     Trnavskí_zákazníci , Produkt
WHERE   Trnavskí_zákazníci .product = Product.name
        AND  Produkt.kategória = “textíl”
```

Čo sa deje pri spracovaní dotazu na pohľad ?



# Aktualizácia cez pohľady

Možno aktualizovať cez neexistujúce tabuľky?

```
CREATE VIEW  nákupy_tesko  AS
   SELECT   sklad, predávajúci, kupujúci
   FROM     Kontrakt
   WHERE    sklad = “Tesko”
```

Čo spôsobí nasledujúce vloženie:

```
INSERT INTO nákupy_tesko
      VALUES  (“Tesko”, Jožo, “Fero”);
```

Vloží riadok
(“Tesko”, Jožo, NULL, “Fero”)
do tabuľky kontrakt.



# Neaktualizovateľné pohľady

```
CREATE VIEW  Iná_Trnava  AS
      SELECT  predávajúci, výrobok, sklad
      FROM    Osoba, Kontrakt
      WHERE   Osoba.mesto = “Trnava”  AND
              Osoba.meno = Kontrakt.kupujúci
```

Ako vložíme nasledujúci riadok ?

(Jožo, “sandále”, “Kmart”)

# Nulové hodnoty (Null)

SQL rozširuje všetky domény o hodnotu Null.
Null predstavuje neznámu, neexistujúcu aj neurčenú hodnotu.

Pre všetky operácie a funkcie platí: Ak jeden z operandov alebo argumentov je Null, potom výsledok Null.

Logika s hodnotou Null:

| ¬ | |
|---|---|
| T | F |
| F | T |
| N | N |

| ∧ | T | F | N |
|---|---|---|---|
| T | T | F | N |
| F | F | F | N |
| N | N | N | N |

| ∨ | T | F | N |
|---|---|---|---|
| T | T | T | N |
| F | T | F | N |
| N | N | N | N |

Dôsledok: Neplatí princíp vylúčenia tretieho.



# Varianty syntaxe

Syntax spojení:

SELECT **x, y, z**
FROM R (inner) JOIN S ON R.**y** = S.**y**;

SELECT **x, y, z**
FROM R, S
WHERE R.**y** = S.**y**;



# Vonkajšie spojenia

SELECT **x, y, z**
FROM R LEFT (outer) JOIN S ON R.**y** = S.**y**;

SELECT **x, y, z**
FROM R, S
WHERE R.**y** = S.**y**
UNION
SELECT **x, y, NULL**
FROM R
WHERE **y** NOT IN (SELECT **y** FROM S);

Úloha: Definujte pravé vonkajšie spojenie (RIGHT outer JOIN).



# Temné SQL dotazy

```
SELECT   R.A
FROM     R, S, T
WHERE    R.A = S.A   OR   R.A = T.A;
```

*Vyberá hodnoty z R, ktoré sú v S alebo T.*

Ale čo ak S alebo T je prázdne? A čo dotaz ?

```
SELECT   R.A
FROM     R, S, T
WHERE    R.A = S.A   AND   R.A = T.A;
```

Vyskúšajte na počítači !