29. Jazyk SQL – selekce, projekce, agregační funkce, množinové operace, typy spojení, vnořené dotazy, spouště a uložené procedury

- SQL – Structured Query Language
- SQL je standardizovaný jazyk pro přístup k relačním databázím složen z:
- jazyk pro definici dat **DDL** (Data Definition Language)
    - definování databázového schéma – datové typy, struktury
    - vytváření, odstraňování, modifikace definic relací a pohledů (virtuální relace)
    - definování integritních omezení
- jazyk pro manipulaci s daty **DML** (Data Manipulation Language)
    - dotazování SELECT
    - INSERT, UPDATE, DELETE

Charakteristiky SQL

- neprocedurální jazyk popisuje co od DB chceme a nikoliv jak je to třeba provést
- není citlivý na velká/malá písmen

CRUD operace

Create

`INSERT`

Read (Retrieve)

`SELECT`

Update

`UPDATE`

Delete

`DELETE`

Datové typy

- přesné numerické např.
    - INTEGER (4B), SMALLINT (2B),
    - DECIMAL (p, s), kde p – počet cifer, s – počet desetinných míst, NUMERIC (p, s)
- aproximativní numerické např.
    - FLOAT (volitelná přesnost), REAL (pevně daná přesnost), DOUBLE PRECISION
- znakové řetězce např.
    - CHARACTER(n) – fixní délka, doplnění prázdnými znaky (trim)
    - CHARACTER VARYING(n) – max. délka
- časové např.
    - DATE, TIME, TIMESTAMP (vs. INTEGER, CHAR(n))
- intervalové
    - INTERVAL
- bitové řetězce – sekvence 0 a 1, nepodporované od SQL:2003
    - BIT(n)
    - BIT VARYING(n)

- logické (tříhodnotové)
  - BOOLEAN (TRUE, FALSE, NULL (unknown))
- nestandartní např.
  - MONEY, GEOMETRIC SHAPE

- možnost konverzí, přetypování pomocí CAST

IO sloupce

- NOT NULL – žádná hodnota v daném sloupci nesmí být NULL
  - vždy musí být vyplněna datami
  - implicitně je nastavení NULL (není to však ani 0 ani mezera)
- UNIQUE – všechny hodnoty v sloupci musí být unikátní
  - je přípustná jedna hodnota NULL
  - vhodné pro kandidátní klíče
  - kombinace více atributů může být UNIQUE
- PRIMARY KEY – sloupec je primárním klíčem
- REFERENCES – sloupec je cizím klíčem, definuje referenční integritu vzhledem k jiné tabulce
- CHECK – IO zadané logickým výrazem

- //DEFAULT – slouží k určení implicitní hodnoty sloupce
  - NULL nebo hodnota
  - co se stane, když NOT NULL DEFAULT NULL?

Selekce

σ_podmínka (R)

- výsledek je relace – podmnožina n-tic z R, které splňují danou podmínku

- výběr řádků

- D2. Najdi všechny prihlášky na obor IT na TUL

Všechny informace o promítáních po 5.4.2002

```
SELECT *
FROM Program
WHERE datum_prom > '5.4.2002';
```

Projekce

Π_seznam atributů (R)

- výsledek je relace – vertikální podmnožina z R, která obsahuje pouze atributy ze seznamu a eliminuje duplicitní hodnoty

- výběr sloupců

- D3. Najdi názvy univerzit, na které se studenti hlásil

Kombinace selekce a projekce

D5. Najdi jména a ID studentů, kteří mají průměr míň jako 2.0

Agregační funkce

Agregační funkce slouží k výpočtu agregovaných hodnot z jednoho nebo více sloupců. Například **SUM**, **AVG, COUNT, MAX a MIN** jsou běžně používané agregační funkce.

Množinové operace

• **UNION, INTERSECT a EXCEPT**

Filmy, které natočil režisér Hřebejk nebo jsou promítány 2.4.2002

```
(SELECT k_filmu FROM Filmy
WHERE reziser = 'Hřebejk')
UNION
(SELECT k_filmu FROM Promítání
WHERE datum = '2.4.2002');
```

Typy spojení

• **INNER JOIN, LEFT JOIN, RIGHT JOIN a FULL JOIN**

• spojení přes největší množinu atributů, které se vyskytují v obou relacích

• společný atribut se ve výsledné relaci nachází jen jedno

Vnořené dotazy

Adresy kin, ve kterých dávají film Gladiátor natočený roku 2000

```
SELECT Adresa FROM Kina
WHERE k_kina IN
        (SELECT k_kina FROM Promítání P WHERE P.k_filmu =
                (SELECT F.k_filmu FROM Filmy F WHERE F.nazev_f = „Gladiátor“ AND rok_v = 2000));
```

Spouště

• Triggr je procedura, která je automaticky spuštěná DBMS jako reakce na specifickou akci v databáze

• Když nastane událost, zkontroluj podmínku, pokud podmínka platí, proveď akci

  • např. když se vloží přihláška studenta, který má průměr 1.0, automaticky ho akceptuj

Událost – změna v DB, která vyvolá spuštění triggru (nezáleží, kdo jí vyvolal)

- definovaná na relaci - ON jméno_tabulky
- INSERT | DELETE | UPDATE OF [seznam_sloupců]

Podmínka – dotaz nebo test, který je proveden pokud je triggr aktivovaný

- když výsledek dotazu je NOT NULL, podmínka platí

Akce – procedura, která je provedena při spuštění pokud podmínka platí

```
DECLARE
BEGIN
    SQL příkazy
END
```

BEFORE triggr

- granularita FOR EACH ROW
- pracuje se stavem databáze před provedením vlastního dotazu, který triggr spustil.
- vlastní dotaz bude proveden až po ukončení všech akcí triggru.

Použití:

- validace vstupních dat
- automatické generování a doplňování dat pro nové řádky
- nepoužívají se pro další modifikace, neobsahují INSERT, UPDATE, DELETE

AFTER triggr

- granularita FOR EACH ROW a taky FOR EACH STATEMENT
- pracuje se stavem databáze po provedení vlastního dotazu, který triggr spustil. Až po kontrole IO spojených s vlastním dotazem.
- nejdřív se provede vlastní dotaz, pak akce triggru

- Použití:
  - aplikační logika

BEFORE triggry → vlastní dotaz (IO) → AFTER triggry

Příklad triggeru - automaticky odstranit záznam pracovníka, jehož záznam v tabulce LIDÉ mažeme

```
CREATE TRIGGER aktualizuj_platy
ON lidé
FOR DELETE
AS
        DELETE FROM platy
        WHERE platy.osoba_id = lidé.id
```

Uložené procedury

Uložená procedura je databázový objekt, který neobsahuje data, ale část programu, který se nad daty v databázi má vykonávat. Lze se k ní chovat stejně jako ke každému jinému objektu databáze (založit, upravovat a smazat) pomocí příkazů dotazovacího jazyka databáze. Uložené procedury jsou vykonávány přímo databázovým serverem. Je to vlastně jakýsi skript - přesněji řečeno dávka - která je uložena přímo v databázi. Procedury mohou obsahovat vstupní parametry, výstupní parametry a návratové hodnoty.

- Uložená procedura (stored procedure) je podprogram uložený a spuštěný v rámci DB serveru
- Podle návratové hodnoty
  - když vrací hodnotu – funkce
  - když nic nevrací – procedura
- Použití
  - validace dat (datové API)
  - přepoužití často používaných kombinací SQL příkazů (volání stejné procedury z více triggrů)
  - posun programování logiky do DB
- syntax silně závislá na DBMS

```
definice procedury:
CREATE [OR REPLACE] PROCEDURE jméno_procedury
[(parameter1 [typ] datový_typ,
  parameter2 [typ] datový_typ], ...)]
[IS|AS deklarace proměnných]
BEGIN
akce
[EXCEPTION výjimky]
END;

volání procedury:
EXECUTE jméno_procedury(param1, param2, ...)

vymazání procedury:
DROP PROCEDURE jméno_procedury;
```

- Parametre slouží na výměnu dat mezi procedurou a programem, který ji volal.
  - **IN** - pochází z programu, procedura s ním pracuje, ale nemůže jej měnit. (defaultní hodnota)
  - **OUT** - procedura s ním nepracuje, ale může jej měnit. Slouží jako návratová hodnota, která se předá programu.
  - **IN OUT** - procedura s ním pracuje a může jej i měnit.

UP2: O zadaném studentovi vrátí jméno a průměr.

```
CREATE OR REPLACE query_studenti
(p_sId IN Studenti.sId%TYPE,
 p_name OUT Studenti.sJmeno%TYPE,
 p_prumer OUT Studenti.prumer%TYPE)
IS
BEGIN
    SELECT sJmeno, prumer INTO p_name, p_prumer
    FROM Studenti
    WHERE p_sID = Studenti.sId;
END;
```

Příklad uložené procedury - vytvoření procedury vracející string a její následné volání

```
CREATE PROCEDURE my_hello_world
AS
        SELECT 'Hello, World!';
GO

my_hello_world;
```

Příklady příkazů

Kolik filmů natočili jednotliví režiséři?

```
SELECT reziser, COUNT(k_filmu)
FROM Filmy
GROUP BY reziser;
```

Kteří režiséři natočili alespoň tři filmy?

```
SELECT reziser
FROM Filmy
GROUP BY reziser
HAVING COUNT(k_filmu) > 2;
```

Jaká je průměrná cena vstupenek u jednotlivých filmů?

```
SELECT X.k_filmu, nazev_f, AVG(Cena)
FROM Filmy X, Promítání Y
WHERE X.k_filmu = Y.k_filmu
GROUP BY k_filmu;
```

Hodnoty sloupce nazev_f libovolné délky začínající na PS

```
nazev_f  LIKE 'PS%'
```

Jména filmů, kteří se v některém kině promítají.

```
SELECT nazev_f FROM Filmy F
WHERE EXISTS (SELECT * FROM Promítání
        WHERE k_filmu = F.k_filmu);
```