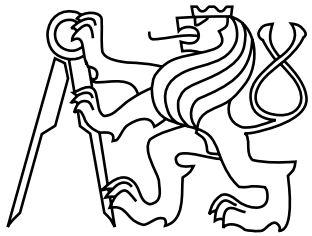
Jazyk SQL 3 - DML, DDL, TCL, DCL

Michal Valenta

Katedra softwarového inženýrství FIT
České vysoké učení technické v Praze

BI-DBS, ZS 2014/15

https://edux.fit.cvut.cz/courses/BI-DBS/



SQL DML

- příkazy INSERT, UPDATE, DELETE, MERGE
- transakční zpracování
  - záleží na nastavení klienta (autocommit on|off)
  - předpokládejme autocommit off, potom:
    - transakce je tvořena sekvencí DML a SELECT příkazů
    - potvrzena příkazem COMMIT
    - odvolána příkazem ROLLBACK
    - ostatní session vidí v průběhu transakce staré hodnoty, dokud tato není potvrzena
    - příkazy DDL nebo DCL provedou implicitní COMMIT
  - systematický výklad transakčního zpracování bude podán v samostatné přednášce
- při zpracování DML příkazů si DBMS hlídá platnost integritních omezení

Aktualizace v SQL – příklady

Smazání řádku / řádek

```
DELETE FROM Filmy
WHERE jmeno_f = 'Puška'; - - pozor na správně formulovanou podmínku
```

zmněna hodnoty řádku / řádek

```
UPDATE Zakaznici SET jmeno = 'Gotzová'
WHERE rod_c = '4655292130'; - - pozor na správně formulovanou podmínku

UPDATE Zákazníci SET jméno = 'Müller'
WHERE jméno = 'Muller';
```

UPDATE více řádek pomocí vnořeného dotazu

Doplň k tabulce Zakaznici redundantní atribut pocet_pujcek a jednorázově dopočti jeho hodnoty.

```
ALTER TABLE zakaznici
Add Pocet_pujcek Number; - - přidáme sloupec to tabulky zákazníci

UPDATE Zákazníci Z - - jednorázově do něj doplníme hodnoty
SET Pocet_pujcek = (SELECT count(*) from Vypucky V
                    WHERE V.rod_c = Z.rod_c);
```

Pokud se databáze mění, je třeba udržovat tuto informaci konzistentní!!!. Lze toho dosáhnout například pomocí triggerů.

INSERT

Vložení jednoho řádku do tabulky Zakaznici.

```
INSERT INTO Zakazníci (rod_c, jmeno)
                VALUES ('4804230160','Novák');
```

Vložení více řádek vnořeným SELECTem

```
CREATE TABLE Kolik_kopii - - nejprve vytvoříme tabulku
(rod_c CHAR(10), pocet SMALLINT);

INSERT INTO Kolik_kopii
SELECT rod_c, COUNT(c_kopie) FROM Vypujcky
GROUP BY rod_c; - - ... a pak ji naplníme daty
nebo celé v jednom příkazu:
CREATE TABLE Kolik_kopii
(rod_c CHAR(10), pocet SMALLINT)
AS SELECT rod_c, COUNT(c_kopie) FROM Vypujcky
GROUP BY rod_c;
```

Pohledy - syntaxe, příklad

Definice pohledu – syntaxe

```
CREATE VIEW jméno-pohledu [(v-jméno-atr[,v-jméno-atr]...)]
AS dotaz
WITH CHECK OPTION;
```

Pohled obsahující pouze Pražáky

```
CREATE VIEW Prazaci AS
SELECT c_ct, jmeno, adresa
FROM Zakaznici WHERE adresa LIKE '%PRAHA%';

DROP VIEW Prazaci;

CREATE VIEW Dluznici (rod_c, pocet_vypujcek) AS
SELECT rod_c, COUNT(c_kopie) FROM Vypucky
GROUP BY rod_c;

SELECT * from Dluznici
where pocet_vypujcek > 5;
```

Pohledy - charakteristika

- pohled je virtuální relace
- v datové slovníku je uložen ve formě SELECT příkazu, kterým je definován
- z hlediska dotazování je pohled zaměnitelný s tabulkou
- DML operace nad pohledy v omezené míře
  - musí se jednat o tzv “simple view” (neobsahuje operace join, agregace, výrazy, ...)
  - DML nesmí být zakázany v jeho definici – klauzule READ ONLY
  - je doporučeníhodné používat klauzuli WITH CHECK OPTION
- DML nad tzv. “complex views” lze realizovat pomocí instead-of triggerů
- k čemu pohledy slouží:
  - odstínění informací, které uživatel/role nemá vidět
  - zpřehlednění složitých dotazů
  - snazší vývoj aplikací
- pohledy nepřinášejí výkonové zrychlení – k tomu lze použít tzv. materializované pohledy (MATERIALIZED VIEWS)

Příkaz WITH - příklad

Najdi seznam kin, ve kterých hrají všechny filmy s M. Brandem

```
R1:=PROGRAM[NÁZEV_K]
R2:=FILM(HEREC="BRANDO")[JMÉNO_F]
R:=R1 x R2
S:= R/ PROGRAM[NÁZEV_K, JMÉNP_F]
T:=S[NÁZEV_K]
U:=PROGRAM[NÁZEV_K] -T
```

Příkaz WITH - příklad

Najdi seznam kin, ve kterých hrají všechny filmy s M. Brandem

```
With
  R1 As Select Nazev_k From PROGRAM,
  R2 As Select Jmeno_F From Film
        Where Herec='Brando',
  R As Select * From R1 Cross Join R2,
  S As (Select * From R)
     Except
       (Select Název_k, Jméno_f  From Program),
  T As Select Distinct Nazev_k From S

(Select Distinct Nazev_k From Program)
   Except
(Select * From T);
```

CREATE TABLE

- CREATE TABLE

```
CREATE TABLE tabulka (
  sloupec datovy_typ [io_sloupce [, io_sloupce...]],
    ...
  [io_tabulky [, io_tabulky ...]] );
```

```
CREATE TABLE VYPUJCKY (
  c_kopie CHAR (3) NOT NULL,
  c_zak   CHARACTER (6) NOT NULL,
  cena    DECIMAL(5,2),
  rod_c   CHARACTER (10) NOT NULL,
  datum_v DATE);
```

ALTER TABLE, DROP TABLE

- ALTER TABLE

```
ADD sloupec, DROP sloupec, ALTER sloupec,
ADD CONSTRAINT io, DROP CONSTRAINT io
```

```
ALTER TABLE KINA ADD pocet_mist INTEGER;
```

- DROP TABLE

```
DROP TABLE tabulka [CASCADE CONSTRAINTS]
```

```
DROP TABLE KINA CASCADE CONSTRAINTS;
```

Integritní omezení v SQL

- IO sloupce:
  - NOT NULL
  - DEFAULT
  - UNIQUE
  - PRIMARY KEY
  - REFERENCES
  - CHECK
- IO tabulky - stejné jako IO sloupce
  (NOT NULL je speciálním případem CHECK)
  složené IO vždy na úrovni tabulky
- pojmenování IO
  není syntakticky nutné, ale vřele doporučované

Příklad DDL s IO

```
DROP TABLE KINA CASCADE CONSTRAINTS;
CREATE TABLE KINA ...
...
CREATE TABLE PŘESTAVENÍ
   (NAZEV_K Char_Varying(20) NOT NULL,
   JMENO_F Char_Varying(20) NOT NULL,
   DATUM  date  NOT NULL,
   CONSTRAINT PREDSTAVENI_PK
     PRIMARY KEY (NAZEV_K, JMENO_F)
   CONSTRAINT PREDSTVENI_KINA_FK
     FOREING KEY (NAZEV_K) REFERENCES KINA,
   CONSTRAINT PREDSTAVENI_FILMY_FK
     FOREING KEY (JMENO_F) REFERENCES FILMY);
```

Referenční integrita a kaskádní DELETE / UPDATE

Referenční integrita (cizí klíč) – v SQL 4 možné způsoby reakce:

```
[ CONSTRAINT constraint_name ]
   FOREIGN KEY ( column_name [, ... ] )
   REFERENCES reftable [ ( refcolumn [, ... ] ) ]
   [ ON DELETE action ] [ ON UPDATE action ]
action ::= [NO ACTION | RESTRICT |
            CASCADE | SET NULL | SET DEFAULT]
```

```
CREATE TABLE order_items (
    product_no integer REFERENCES products
                       ON DELETE RESTRICT,
    order_id integer REFERENCES orders
                       ON DELETE CASCADE,
    quantity integer,
    PRIMARY KEY (product_no, order_id));
```

Poznámka: implementace tohoto rysu nebývá kompletní.

Okamžik kontroly IO, dočasné vypnutí/zapnutí IO

- SQL zavádí možnosti stanovit při deklaraci integritního omezení čas, kdy se má kontrolovat.
- Kontrolu IO lze definovat jako odložitelnou (DEFERRED) až na konec transakce.
- V rámci session pak lze stanovit zda se takové IO kontroluje IMMEDIATE (v rámci provedení DML) nebo až na konci transakce.
- Oracle dovoluje v příkazu ALTER TABLE také IO dočasně vypnout/zneplatnit DISABLE/ENABLE CONSTRAINT.
- Zpětné zapnutí IO pak může/nemusí vyžadovat kontrolu platnosti dat již vložených v databázi.

Typy dat v SQL

- numerické
- textové
- rozsáhlé znakové řetězce (CLOB)
- rozsáhlé bitové řetězce (BLOB)
- datum a čas
- interval

poznámka

NULL je prvkem každého datového typu
tříhodnotová logika: TRUE, FALSE, UNKNOWN
Konverze : implicitní, explicitní (pomocí funkce CAST)

Typy dat v SQL

přesné numerické typy

INTEGER, SMALLINT, NUMERIC, DECIMAL, NUMER
- DECIMAL(p,q)
p ... přesnost
q ... měřítko

aproximativní numerické typy

FLOAT
REAL
DOUBLE PRECISION

Typy dat v SQL

znakové řetězce

CHARACTER(n) (délka n,zprava mezery)
CHARACTER VARYING(n)

aproximativní numerické typy

DATE
TIMESTAMP
INTERVAL

DCL - příkazy GRANT a REVOKE

- schema (uživatel), role
- kdo vytvoří objekt, je jeho vlastníkem (je v jeho schématu) a může s ním manipulovat
- vlastník objektu může umožnit nakládání s objektem jinému uživateli nebo jiné roli
- grantovat lze (dle typu objektu):
  SELECT, INSERT, UPDATE, DELETE, ALTER, EXECUTE, INDEX, REFERENCE
- GRANT - přidá právo
- REVOKE - odebere právo

```
GRANT SELECT ON V_Filmy TO XNOVAKJ3;
GRANT ALL PRIVILEGES ON V_filmy TO PUBLIC;
REVOKE INSERT ON Filmy FROM XNOVAKJ3;
```

poznámky: systémová / objektová privilegia, role PUBLIC (oracle), schéma nemusí byt totožné s uživatelem (jak je tomu v Oracle)

Příkazy COMMIT, ROLLBACK, SAVEPOINT

- Potvrzení/odvolání změn provedených v příslušné transakci nezáleží na počtu DML a SELECT příkazů ani na počtu změněných řádků.
- COMMIT - potvrzení změn - všechny provedené změny jsou perzistentně uloženy v databázi, ostatní session od této chvíle provedené změny vidí.
- ROLLBACK - odvolání změn - všechny změny provedené v příslušné transakci jsou odvolány.
- SAVEPOINT - možnost definování “značky” uvnitř transakce, ke které lze vztáhnou ROLLBACK.

Systémový katalog

- systémový katalog, datový slovník (data dictionary), metadata, ...
  -> přístup k metadatům příslušné databáze
- standardizace existuje, ale není dodržována výrobci
- k datům v systémovém katalogu lze přistupovat pomocí příkazů SELECT
- dotazy nad systémovým katalogem umožňují “skriptování” databáze
- příklad Oracle:
  pohledy s prefixy USER_ , ALL_, DBA_

Příklady:

```
SELECT OBJECT_NAME, OBJECT_TYPE, OWNER
  FROM ALL_OBJECTS;
SELECT * FROM USER_CONSTRAINTS;
SELECT
 'DROP TABLE '||table_name||' CASCADE CONSTRAINTS;'
 FROM USER_TABLES;
```