Jazyk SQL - TCL, DML, DDL, DCL

Michal Valenta

Katedra softwarového inženýrství
Fakulta informačních technologií
České vysoké učení technické v Praze

BI-DBS, LS 2021/2022

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

Příkazy COMMIT, ROLLBACK, SAVEPOINT

session

Jedno navázané spojení s DB serverem. V session probíhají transakce. Lze mít otevřeno více session i pod jedním DB uživatelem. Při DB vývoji se tak běžně pracuje.

- 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 vztáhnout ROLLBACK.

AUTOCOMMIT ON/OFF

- AUTOCOMMIT řídí chování na úrovni session
- ON - každý DML příkaz je automaticky potvrzen (COMMIT)
- OFF - musí přijít explicitní COMMIT nebo ROLLBACK

PostgreSQL

- psql: `\echo :AUTOCOMMIT`, `\set AUTOCOMMIT ON|OFF`
- lze použít explicitní příkaz `BEGIN TRANSACTION` (funguje i v případě AUTOCOMMIT ON)
- nastavení v nástroji DataGrip

Důrazné varování

Nespoléhejte na AUTOCOMMIT ON a v insert scriptech používejte explicitně příkaz `COMMIT`.

SQL DML

- příkazy INSERT, UPDATE, DELETE, MERGE
- transakční zpracování (viz předchozí slide)
- při zpracování DML příkazů si DBMS
  automaticky hlídá platnost (deklarativních) integritních omezení
- detailní výklad transakčního zpracování bude předmětem samostatné přednášky

Aktualizace v SQL – příklady

Smazání řádku / řádek

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

změna 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;
```

příkaz MERGE

- prakticky užitečná kombinace příkazů INSERT a UPDATE
- na základě **referenční tabulky** provádíme buď insert nebo update záznamů v **cílové tabulce**
  - je-li záznam v cílové i referenční tabulce, provede se v cílové jeho update
  - je-li záznam pouze v referenční, pak se do cílové vloží nový

Některé DB stroje (např. **PostgreSQL**) používají místo příkazu MERGE příkaz UPSERT (kombinace UPdate a inSERT).

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 systémovém katalogu 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[NAZEV_K]
R2:=FILM(HEREC='BRANDO')[JMENO_F]
R:=R1 x R2
S:= R/ PROGRAM[NAZEV_K, JMENO_F]
T:=S[NAZEV_K]
U:=PROGRAM[NAZEV_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);
```

příkaz WITH - rekurzivní dotazování

Příkaz WITH se od standardu SQL99 používá také k **rekurzivnímu dotazování**. Do předmětu BI-DBS jsme jej ale nezařadili.

Oracle měl již v dřívějších verzích implementováno vlastní rekurzivní dotazování. Příkaz SELECT je rozšířen o klauzule START WITH a CONNECT BY.

Info

Následujících 6 slajdů s výjimkou slajdu „Okamžik kontroly IO, dočasné vypnutí/zapnutí IO“ jsou již v první části prezentace o transformaci ER schématu na relační.

V rámci logiky výkladu to bylo nezbytné. Zde jsou uvedeny pro úplnost.

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]
```

```
DROP TABLE KINA CASCADE;
```

Integritní omezení v SQL

- Integritní omezení sloupce:
  - NOT NULL
  - DEFAULT
  - UNIQUE
  - PRIMARY KEY
  - REFERENCES
  - CHECK
- Integritní omezení 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

Integritní omezení v SQL

```
DROP TABLE KINA CASCADE CONSTRAINTS;
CREATE TABLE KINA ...
...
CREATE TABLE PREDSTAVENI
(NAZEV_K Char Varying(20) NOT NULL,
NAZEV_F Character Varying(20) NOT NULL,
DATUM  date  NOT NULL,
CONSTRAINT PREDSTAVENI_PK
PRIMARY KEY (NAZEV_K, NAZEV_F),
CONSTRAINT PREDSTVENI_KINA_FK
FOREIGN KEY (NAZEV_K) REFERENCES KINA,
CONSTRAINT PREDSTAVENI_FILMY_FK
FOREIGN KEY (NAZEV_F) REFERENCES FILMY);
```

Referenční integrita, kaskádní reakce

Referenční integrita (cizí klíč) – v SQL čtyři 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 - enable/disable constraint

- 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.

Datové typy v SQL

- numerické
- textové
- datum a čas
- ... a mnoho dalších závisí na konkrétním RDBMS

poznámka

NULL je prvkem každého datového typu
(pokud nespecifikujeme NOT NULL).
Tříhodnotová logika: TRUE, FALSE, UNKNOWN.
Konverze: implicitní, explicitní pomocí funkce CAST
(v PostgreSQL se používá též: "::").

Datové typy v PostgreSQL (přehledově)

DCL - příkazy GRANT a REVOKE

- schema, uživatel, role
  pozor, v Oracle schema = uživatel, v PostgreSQL nikoliv
- kdo vytvoří objekt, je jeho vlastníkem 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;
```

Systémový katalog

- Metadata (informace o obsahu databáze).
- V relační databázi má podobu (pevně daných) tabulek.
- Nad nimi jsou (obvykle) k dispozici pohledy, které může používat poučený uživatel i nástroje (např. DataGrip).
- Interní tabulky (systémového katalogu) jsou pro každou databázi specifické.
- SQL standard definuje jednotné informační schéma. Některé databáze (například Oracle) ho ale napoužívají.
- Se znalostí systémového katalogu lze velmi efektivně "skriptovat"v SQL. Využíváno (nejen) DB administrátory ke změnám ve více schématech nebo na více objektech. Například: "smaž všechny tabulky ve schématu", "všem uživatelům dej privilegium XY", ...

PostgreSQL - systémový katalog

Schémata

Databáze v PostgreSQL může obsahovat více schémat. Schéma lze chápat jako namespace, ve kterém jsou uložené DB objekty (tabulky a další). Schémata můžete přidávat/mazat a přepínat se mezi nimi.

- information_schema - blíží se standardu
  pohledy tables, views, columns, ...
- pg_catalog - postgresql specific
  krom informací o objektech též mnoho dalších informací například pro sledování okamžitého zatížení, transakcí, využití indexů, ...

Oracle - systémový katalog

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;
```

K zapamatování

- Důležitost transakčního zpracování zejména pro DML.
  Pozor na nastavení autocommit on/off.
- DML: INSERT, UPDATE, DELETE – též v semestrálce a zkouškovém testu.
- Pohledy (VIEW) – vyžadujeme i v semestrálce.
- Příkaz WITH – dočasné pohledy
- DDL: CREATE, ALTER, DROP (a deklarativní IO)
- DCL: GRANT, REVOKE (a ROLE), pojem SCHEMA
  V Oracle USER = SCHEMA, v PostgreSQL nikoliv.
- TCL: COMMIT, ROLLBACK (a SAVEPOINT)
- O konceptu **transakce** bude ještě samotatná přednáška, pro databázové systémy je to kriticky důležité.
- Systémový katalog a práce s ním bývá velmi „DB specific“.