

2005-12-06

Uplatnění substitučních parametrů:

Výpis studentů na podkladě obsluhou zadaného jména a adresy:

```
SQL>INPUT
  SELECT * FROM student
  WHERE jmeno='&1' AND adresa='&2'

SQL>SAVE TEST                                (uloží příkazovou dávku pod jménem TEST)
SQL>SET VERIFY OFF                          (potlačí systémové hlášky)
SQL>START TEST Novák Praha 1                (vypsání Nováků z Prahy 1)
```

„Spolování“ výsledku do textového souboru:

```
SQL>Spool TEMP
SQL>Clear Columns
SQL>Clear Breaks
SQL>Clear Computes
SQL>Column CIND Heading 'CISLO INDEXU' format A14
SQL>Column JMENO Heading 'JMENO STUDENTA' format A14
SQL>Column ADRESA Heading 'BYDLISTE' format A20
SQL>Break On ADRESA Skip 1 On Report
SQL>Compute COUNT of CIND ON ADRESA
SQL>Compute COUNT of CIND On Report
SQL>Set PageSize 30
SQL>Set LineSize 50
SQL>Ttitle Left 'INFORMACE O STUDENTECH' right 'Strana: ' Format 99 SQL.PNO Skip 2
SQL>Btitle CENTER 'Urceno pouze pro studijni odeleni'
SQL>Select CIND,JMENO,ADRESA From STUDENT Order By ADRESA'
SQL>Spool Out                                (vytiskne vybrané věty na připojenou tiskárnu)
```

Program bude uložen pod názvem **TEST.SQL** a lze jej odstartovat příkazem **START TEST**. Požadovaný výpis studentů lze nalézt (tisknout a prohlížet libovolným editorem) v souboru TEMP.LST.

Tisk hodnoty sloupce v záhlaví VÝPISU:

```
SQL>TTITLE LEFT 'Bydliste: ' Bydl SKIP 2;
SQL>BREAK ON adresa SKIP PAGE;
SQL>SELECT adresa,jmeno,cind FROM student ORDER BY adresa;
SQL>COLUMN adresa NEW_VALUE Bydl;
```

Bydl představuje paměťovou proměnnou naplňovanou hodnotami z položky ADRESA příkazem **COLUMN ADRESA NEW_VALUE Bydl**;

Výpis bude stránkovat podle položky ADRESA, přičemž v záhlaví výpisu se objeví konkrétní hodnota této položky.

Výpočty dílčích a celkových agregací:

```
SQL>BREAK ON adresa;
SQL>COMPUTE COUNT OF cind ON adresa;
SQL>SELECT adresa,jmeno,cind FROM student ORDER BY adresa;
```

Za každou skupinu studentů se stejnou adresou se zobrazují počet studentů se shodným bydlištěm. Platnost příkazu **COMPUTE** zrušíte odesláním příkazu **CLEAR COMPUTES**;

Počet studentů za celý soubor:

```
SQL>BREAK ON REPORT;
SQL>COMPUTE COUNT OF cind ON REPORT;
SQL>SELECT adresa,jmeno,cind FROM student ORDER BY adresa;
```

Výhody používání SQL:

- ü **Snížení ceny na zaškolení pracovníků** – tvůrci a uživatelé aplikací mohou snadněji přecházet od jednoho SŘBD k jinému.
- ü **Přenositelnost vytvořené aplikace** – aplikace vytvořená v konkrétním SŘBD může být provozovaná v jiném prostředí bez změny – vhodné odladit danou aplikaci na personálním počítači a výsledek pak přenést na jinou kategorii počítačů.
- ü **Délka života aplikace se zvyšuje** – v případě nutnosti je možno přejít na jiný databázový prostředek a prodloužit tak životnost dané aplikace.
- ü **Společný přístup k datům v heterogenním prostředí** – distribuované databáze provozované pod různým SŘBD.

Nevýhody SQL – nedostatky:

- ü Normalizaci **ANSI** dodržují jen někteří producenti.
- ü Problematická konstrukce výrazu **SELECT FROM WHERE**:
projekce(selekce(kartézský_součin)relace1...relacen) je nevhodná, lépe: selekce(projekce(relace)).
- ü Nelze definovat **vlastní funkce**.
- ü Problematická existence **WHERE** a **HAVING**:

```
SQL>SELECT * FROM student WHERE stipendium>2000;
SQL>SELECT rocnik,COUNT(*) FROM student GROUP BY rocnik HAVING rocnik<'3'
```

- ü SQL slouží pro komunikaci se SŘBD, neobsahuje **prostředky běžného programování**.
- ü Nelogická **duplicita** konstruktů:

```
SQL>CREATE TABLE prednasejici
  (evcped CHAR(3) NOT NULL PRIMARY KEY,
   jmeno (CHAR(12),
   datnar DATE,
   plat NUMBER(5) CHECK (plat BETWEEN 7000 AND 50000));
```

```
SQL>CREATE TABLE predmety
  (evcped CHAR(5),
   nazev (CHAR(15),
   katedra CHAR(3),
   evcped CHAR(3),
   FOREIGN KEY (evcped) REFERENCES prednasejici(evcped));
```

Dotazovací jazyky SQL – systémové informace:

- ü Informace o struktuře databáze (definované tabulky, pohledy, indexy atd.) jsou uloženy stejným způsobem jako vlastní data, tj. ve formě tabulky. Takovéto uložení je výhodné, protože informace o databázi je možno běžně vybírat pomocí příkazu SELECT.
- ü Výpis jména objektu, vlastníka a typu objektu:

```
SQL>SELECT tname,creator,tablety FROM katalog;
TNAME          CREATOR        TABLETY
-----
seznam_studentu  NĚMEC          VIEW
student          NĚMEC          INDEX
student          NĚMEC          TABLE
```

Oprávnění přístupu k datům:

- ü Nejlépe řešit na základě identifikace uživatele, kterého postupně zvyšujeme v jeho pravomocích.
- ü Autor tabulky má automaticky všechna oprávnění pro všechny operace s ní.
- ü **GRANT {ALL | <seznam-oprávnění>} ON <jméno-tabulky> TO {PUBLIC | <seznam-uživatelů>}**
- ü **Dílicí oprávnění:**
 - SELECT – povoluje pouze čtení dat z tabulky
 - INSERT – povoluje vkládání dat do tabulky
 - UPDATE – povoluje opravu dat v tabulce
 - DELETE – povoluje rušení dat z tabulky
 - ALL – povoluje veškerá oprávnění

- ü **Příklad:**

```
SQL>GRANT SELECT,UPDATE ON student TO referent1;
```

Zpětné odnětí přidělených práv – příkaz REVOKE:

Ů REVOKE {ALL | <seznam-oprávnění>} ON <jméno-tabulky> TO {PUBLIC | <seznam-uživatelů>}

Ů **Příklad:**

```
SQL>REVOKE UPDATE ON student TO referent1;
```

Přidělení oprávnění uživatelským skupinám – tzv. role:

Ů Nutno vytvořit katalog rolí a jim přidělit potřebná oprávnění.

Ů Příklad – založení role PRODEKAN se všemi právy pro tabulku STUDENT a přidělit ji Prof. Prokešovi:

```
SQL>CREATE ROLE prodekan;
SQL>GRANT ALL ON student TO prodekan;
SQL>GRANT prodekan TO Prokes;
```

Transakční zpracování:

Ů Transakční zpracování zajišťuje konzistence dat v existujících tabulkách.

Ů Možnosti – **příkazy:**

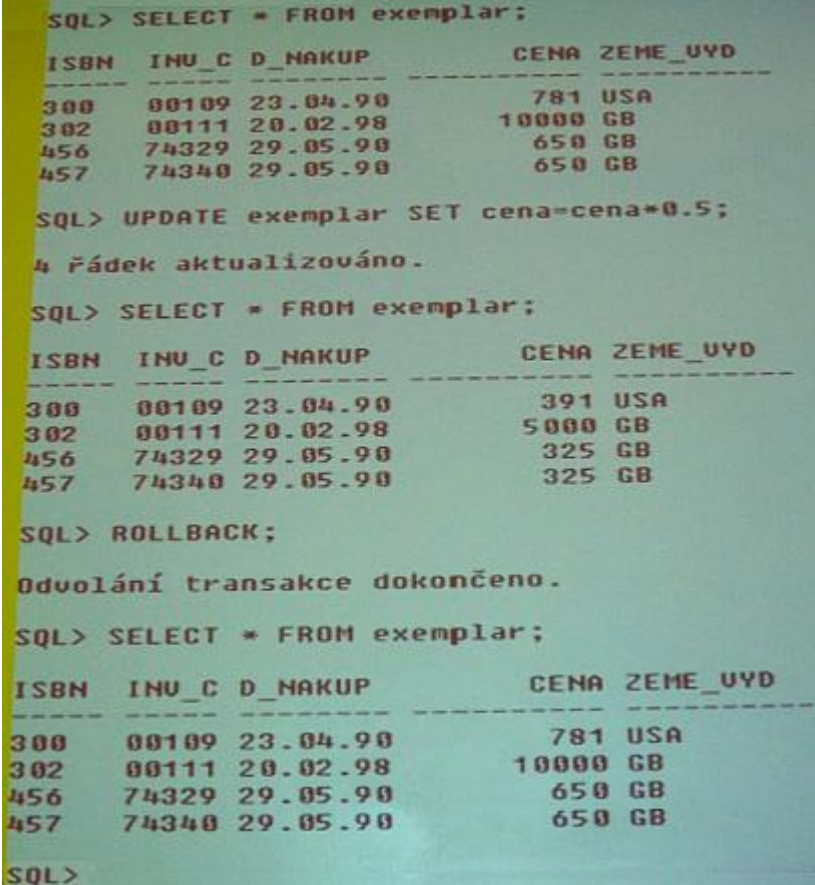
- **COMMIT:**

- Potvrzení platnosti transakce

```
SQL>COMMIT
Potvrzení dokončeno.
```

- **ROLLBACK:**

- Příkaz odvolávající všechny nepotvrzené změny, přičemž odvolat transakci může pouze ten uživatel, který operaci provedl.



```
SQL> SELECT * FROM exemplar;
-----
ISBN    INU_C  D_NAKUP      CENA  ZEME_UYD
-----
300      00109  23.04.98      781  USA
302      00111  20.02.98     10000  GB
456      74329  29.05.98      650  GB
457      74340  29.05.98      650  GB

SQL> UPDATE exemplar SET cena=cena*0.5;
4 řádek aktualizováno.

SQL> SELECT * FROM exemplar;
-----
ISBN    INU_C  D_NAKUP      CENA  ZEME_UYD
-----
300      00109  23.04.98      391  USA
302      00111  20.02.98     5000  GB
456      74329  29.05.98      325  GB
457      74340  29.05.98      325  GB

SQL> ROLLBACK;
Odvolání transakce dokončeno.

SQL> SELECT * FROM exemplar;
-----
ISBN    INU_C  D_NAKUP      CENA  ZEME_UYD
-----
300      00109  23.04.98      781  USA
302      00111  20.02.98     10000  GB
456      74329  29.05.98      650  GB
457      74340  29.05.98      650  GB

SQL>
```

- Je možno ho použít i pro případ omylem vymazaného záznamu.
- Příklad – vymažte v tabulce REZERV záznam s číslem čtenáře J60 a následně toto zrušení záznamu odvolejte:

```
SQL> SELECT * FROM rezerv;
```

ISBN	C_CT	D_REZ
300	H23	20.06.04
456	J60	19.05.04

```
SQL> DELETE FROM rezerv WHERE c_ct='J60';
```

1 řádka vymazána.

```
SQL> SELECT * FROM rezerv;
```

ISBN	C_CT	D_REZ
300	H23	20.06.04

```
SQL> ROLLBACK;
```

Odvolání transakce dokončeno.

```
SQL> SELECT * FROM rezerv;
```

ISBN	C_CT	D_REZ
300	H23	20.06.04
456	J60	19.05.04

```
SQL>
```

- **Vytvoření návratových bodů:** Předchozí varianty popisují způsob odvolání všech nepotvrzených změn od posledního příkazu COMMIT nebo od posledního regulérního ukončení konzolové aplikace. V případě neregulérního ukončení se provede automatický ROLLBACK. Tento příkaz však odvolává všechny doposud nepotvrzené změny vykonané uživatelem.
- Alternativa – pomocí příkazu **SAVEPOINT** vytvořit body návratu pro případné odvolání transakce k určitému bodu.

```
SQL> SAVEPOINT vygrazeni;
```

Bod návratu vytvořen.

```
SQL> DELETE FROM ctenar WHERE c_ct='H32';
```

1 řádka vymazána.

```
SQL> UPDATE ctenar SET dat_nar='07.07.77' WHERE c_ct='H31';
```

1 řádka aktualizována.

```
SQL> SELECT * FROM ctenar;
```

C_CT	JMENO	ADRESA	DAT_NAR
H23	Karel Novák	Praha 9	01.02.75
J45	Jiří Klouček	Praha 1	12.05.83
J60	Karel Kalaš	Praha 9	12.05.40
H24	Josef Novák	Praha 6	28.07.56
H25	Martin Dvořák	Praha 1	13.03.30
H26	Karel Říman	Praha 1	12.12.61
H27	Uáclav Nový	Praha 9	10.11.51
H28	Michal Nový	Praha 6	09.09.81
H29	Lubomír Valenta	Praha 3	07.08.85
H30	Rudolf Vaněk	Praha 1	31.03.77
H31	Martin Zounar	Praha 9	07.07.77

11 řádek vybráno.

SQL> ROLLBACK TO vyrazení;

Odvolání transakce dokončeno.

SQL> SELECT * FROM ctenar;

C_CT	JMENO	ADRESA	DAT_NAR
H23	Karel Novák	Praha 9	01.02.75
J45	Jiří Klouček	Praha 1	12.05.83
J60	Karel Kalaš	Praha 9	12.05.40
H24	Josef Novák	Praha 6	28.07.56
H25	Martin Dvořák	Praha 1	13.03.30
H26	Karel Říman	Praha 1	12.12.61
H27	Václav Nový	Praha 9	10.11.51
H28	Michal Nový	Praha 6	09.09.81
H29	Lubomír Valenta	Praha 3	07.08.85
H30	Rudolf Vaněk	Praha 1	31.03.77
H31	Martin Zounar	Praha 9	07.11.83
H32	Jiří Kadlec	Praha 3	29.11.90

12 řádek vybráno.

SQL> |