Kapitola 5

Load, DDL - Create, Alter, Drop

Dôležité
Správne vytvorenie databázy a naloadovanie všetkých dát je nevyhnutným predpokladom pre prácu na ďalších cvičeniach.

!

Preto dôsledne dodržiavajte všetky pokyny!!!!

5.1 Zadanie cvičenia - Load

1. Stiahnite a rozbaľte si nasledovné dáta pre vytvorenie databázy letisko `soc_poist.zip`
   Tento `soc_poist.zip` archív obsahuje 3 typy súborov rozlíšené podľa prípony:
   (a) `soc_poist.sql` - DDL príkazy na vytvorenie schémy databázy
   (b) `*.unl` - dátové súbory - dáta, ktoré je potrebné vložiť do jednotlivých tabuliek
   (c) `*.ctl` - control súbory - obsahujú príkazy ako vložiť dáta z dátového súboru do tabuľky
2. Pomocou súboru `soc_poist.sql` vytvorte tabuľky pre informačný systém sociálnej poisťovne.
3. Vytvorte chýbajúce control súbory pre správne naloadovanie dát do tabuliek vytvorených v predchádzajúcom kroku.

Tabuľka	Dáta	Control súbory
p_krajina	krajina.unl	krajina.ctl
p_kraj	kraj.unl	kraj.ctl
p_okres	okres.unl	okres.ctl
p_mesto	mesto.unl	mesto.ctl
p_zamestnavatel	zamestnavatel.unl	zamestnavatel.ctl
p_platitel	platitel.unl	platitel.ctl
p_osoba	osoba.unl	...
p_ztp	ztp.unl	ztp.ctl
p_zamestnanec	zamestnanec.unl	zamestnanec.ctl
p_poistenie	poistenie.unl	poistenie.ctl
p_poberatel	poberatel.unl	...
p_typ_postihnutia	typ_postihnutia.unl	typ_postihnutia.ctl
p_odvod_platba	odvod.unl	odvod.ctl
p_typ_prispevku	typ_prispevku.unl	typ_prispevku.ctl
p_historia	historia.unl	historia.ctl
p_prispevky	prispevky.unl	prispevky.ctl

4. Chýbajúci control súbor najjednoduchšie vytvoríme prepísaním existujúceho control súboru. Kde prepíšeme názov dátového súboru, názov tabuľky, do ktorej ideme loadovať dáta a zoznam stĺpcov.

```
LOAD DATA
INFILE 'zamestnanec.unl'
INTO TABLE p_zamestnanec
FIELDS TERMINATED BY '|'
(
  ID_ZAMESTNAVATELA,
  ROD_CISLO,
  DAT_OD DATE 'MM/DD/YYYY',
  DAT_DO DATE 'MM/DD/YYYY',
  ID_POISTENCA
)
```

   Ak importujeme len dátum samotný, nie čas, časová zložka bude nahradená minimálnou hodnotou - 0. Podobne, hodnoty atribútu `obdobie` importujeme len v presnosti mesiac a rok. Deň bude teda nahradený číslom 1 (pozri `odvod.ctl` a `odvod.unl`).

5. Poradie stĺpcov v control súbore musia byť podľa toho ako sú dáta.
   V prípade našich chýbajúcich súborov sedí poradie dát s poradím stĺpcov ako boli pri vytvorení tabuliek (OKREM TABUĽKY OSOBA).

6. Pre správne loadovanie dátumov je potrebné nastaviť správny formát dátumu podľa dát na mesiac/den/rok

   - v prípade verzie Oracle 8:

     ```
     $export NLS_DATE_FORMAT="MM/DD/RRRR"
     ```

   - od verzie Oracle 10g):
     Nastavte formát dátumu priamo v `ctl` súbore priamo pri zadanom atribúte (pozri súbor `l_zamestnanec.ctl`):

     ```
     napr. DATUM_PRIJATIA DATE 'DD.MM.YYYY'
     ```

7. Naplňte tabuľky dátami z `unl` súborov pomocou control súborov. Nezabudnite, že loadovanie musí byť v správnom poradí tabuliek (rovnako ako insert, opačne ako drop).

   (a) spusti príkaz pre load do tabuľky

   ```
   $ sqlldr login@orcl control='subor.ctl'
   ```

   (b) hneď po spustení loadu musíte skontrolovať, či load prešiel správne.

   - ak control súbor bol `load_person.ctl`, potom po spustení príkazu vznikne súbor s rovnakým menom, len s príponou `log`, t.j. `load_person.log`, v ktorom budú chybové hlášky a počet naloadovaných riadkov.

   ! ***Dôležité***
   Po každom spustení sqlldr je nevyhnutné kontrolovať log súbor!
   Na konzolu vypisuje len počet načítaných riadkov z dátového súboru, nie počet správne naloadovaných riadkov.

   - Ak loadovanie neprešlo správne v súbore k pôvodnému dátovému súboru bude vytvorený bad súbor s chybnými riadkami.

8. V správnom poradí naplňte tabuľky dátami.

9. Po každom kroku nezabudnite skontrolovať, či load prebehol úspešne.

10. Na záver vyexportujte všetky tabuľky sociálnej poisťovne spolu s dátami.
11. Po úspešnom vyexportovaní dropnite všetky tabuľky sociálnej poisťovne.
12. Naimportujte späť tabuľky sociálnej poisťovne spolu s dátami.

5.2 Zadanie cvičenia - Export/import

1. Spravte si zálohu vášho konta

   `$exp` *`login`*`@orcl tables='`*`zoznam tabuliek`*`' file='`*`export_subor.exp`*`'`

   > Skôr ako vykonáte ďalší krok skontrolujte, či export je kompletný
   > (všetky tabuľky, všetky dáta) !

2. Dropnite všetky tabuľky, aby ste vyskúšali import zo súboru.
3. Obnovte si databázu z exportu.

   `$imp` *`login`*`@orcl file='`*`export_subor.exp`*`'`

4. Vytvorte si export pomocou dump file.
5. Spravte si zálohu vášho konta

   > Skôr ako vykonáte ďalší krok skontrolujte, či export je kompletný
   > (všetky tabuľky, všetky dáta) !

6. Dropnite všetky tabuľky, aby ste vyskúšali import z dump file.
7. Obnovte si databázu z exportu (cez dump file).

5.3 Zadanie cvičenia - DDL

1. Vytvorte tabuľku `t1` s primárnym kľúčom `ID`, atribútom `KPK` a ešte aspoň jedným atribútom.
2. Vytvorte tabuľku `t2` s kompozitným primárnym kľúčom pozostávajúceho z atribútov `ID` a `Datum`. (Pozor – `ID` nech je rovnakého typu ako `ID` z tabuľky `T1`)
3. Vytvorte vzťah 1:M medzi tabuľkami `T1` a `T2` (spojenie je cez `ID`).
4. Do tabuľky `T2` pridajte atribút `poznamka` ľubovoľného typu.
5. Vytvorte (duplicitný) index na atribút `Datum` tabuľky `T2`.
6. Vytvorte unikátny index na `KPK` z tabuľky `T1`.
7. Zmažte tabuľku `T1` a `T2`.

5.4 Príklady na precvičenie - Select, Insert, Update, Delete pre model sociálnej poisťovne:

1. Vypíšte menný zoznam žien.
2. Vypíšte menný zoznam osôb, ktoré mali narodeniny minulý mesiac (podľa aktuálneho dátumu).
3. Vypíšte menný zoznam osôb, ktoré aktuálne nemajú platný ZŤP preukaz (platnosť všetkých preukazov osoby už vypršala).
4. Vypíšte zoznam osôb, ktoré nikdy nemali žiaden ZŤP preukaz.
5. Vypíšte zoznam osôb, ktoré oslavujú tento rok okrúhle narodeniny.
6. Vypíšte aktuálny počet zamestnancov, za ktorých platí poistenie zamestnávateľ s názvom Tesco.
7. Vypíšte zoznam osôb, ktoré nikdy nepoberali príspevok v nezamestnanosti.
8. Vypíšte zoznam osôb, ktorí poberajú nejaký príspevok, ktorý nie je príspevkom v nezamestnanosti. (príspevok v nezamestnanosti môže a nemusí poberať).
9. Vypíšte počet osôb, ktoré pracujú v Tescu na dobu neurčitú. Názov atribútu vytvoreného v selecte nech je `celkovy_pocet`.
10. Vypíšte celkovú sumu zaplatenú (t.j. `dat_platby` je vyplnený) spoločnosťou Tesco za minulý rok zamestnancom so zmluvou na dobu neučitú.
11. BONUS: Vypíšte zoznam menovcov z tabuľky osoba (majú rovnaké meno i priezvisko) - napr. Tomas Kovac je tam 3x.
12. Zamestnajte osobu Karol Matiasko na ZU od budúceho mesiaca na dobu neurčitú. Potrebné údaje si vymyslite.
13. Do tabuľky `p_prispevky` vložte údaje - výplatnú listinu pre aktuálnych poberateľov príspevkov, pričom obdobie bude budúci mesiac, kedy nastavte na 15-teho daného mesiaca a sumu vypočítajte podľa aktuálnej sadzby a percetualneho vyjadrenia priznaného príspevku (`p_poberatel.perc_vyj`).
14. Vložte poistencovi s id=6268 platby za všetky mesiace roku 1965. Sumu si vymyslite. Platbu vždy realizujte posledný deň daného mesiaca.
15. Všetkým, ktorých percentuálna miera je menšia ako 10, pozastavte vyplácanie príspevkov ku koncu aktuálneho roka.
16. Zvýšte aktuálnu sumu príspevkov v nezamestnanosti o 10% (Nezabudnite aktualizovať tabuľku `p_historia`).
17. Pozastavte poistenie (nastavte hodnotu atribútu `dat_do` na aktuálny dátum) všetkým, ktorí doteraz nezaplatili na odvodoch nič a nie sú oslobodení.
18. Pozastavte poberanie príspevku v nezamestnanosti všetkým osobám, ktoré zároveň pracujú.
19. Vymažte informácie o poistení, ktoré skončilo pred rokom 2000. Pozor na referenčnú integritu.
20. Vymažte informácie o poberateľoch príspevkov, ktorým sme ukončili vyplácanie pred viac ako 10 rokmi.

5.5 Doplnok

5.5.1 Load

- Majme súbor s dátami, napr. os_ud00103.unl v tvare:

```
780416/1212|Karolina|Velka|Zilina|
```

- Napíšte v editore súbor load_ou.ctl s nasledovným obsahom:

```
LOAD DATA
INFILE 'os_ud00103.unl'

[ { REPLACE | APPEND } ]
```

//ak nič, tak musí byť tabuľka os_udaje prázdna

```
 INTO TABLE os_udaje
 FIELDS TERMINATED BY '|' [ OPTIONALY ENCLOSED BY '|' ]
```

// klauzula OPTIONALY ENCLOSED je pre ukončenie riadku, je potrebná len ak máme iný oddeľovač stĺpcov a riadkov

```
(
  rod_cislo, meno, priezvisko, obec
)
```

Poznámky k syntaxi príkazov
Tieto sa v príkaze nepíšu, len vysvetľujú syntax :

- [ ] . . . nepovinné
- { } ... výber z možností
- | . . . oddeľovač jednotlivých možností v { }

Obyčajné zátvorky () sú súčasťou príkazu a je nutné ich písať.

!

Poznámka
Pre súbory *.csv platí:

```
fields terminated by ','optionally enclosed by '‘‘ '''
```

// (apostrof úvodzovka uvodzovka apostrof)

!

- Z príkazového riadku OS spustite:

```
$sqlldr login@orcl control='load_ou.ctl'
```

5.5.2 UnLoad

stusti SQL*Plus (sqlplus)

```
SQL> SET ECHO OFF NEWPAGE 0 SPACE 0 PAGESIZE 0 FEED OFF
SQL> SPOOL os_udaje.unl
SQL> SELECT meno|| '|' || okres || '|'|| ...
  2        || '|' || priezvisko || '|'
  3  FROM os_udaje;
SQL> SPOOL OFF
```

5.5.3 Export

Z príkazového riadku OS spustite:

```
$exp login@orcl tables='l_person l_reader ....' file='db.exp'
```

5.5.4 Import

Z príkazového riadku OS spustite:

```
$imp login@orcl file='db.exp'
```

5.6 Jazyk DDL

5.6.1 Tabuľka

5.6.1.1 Základná syntax - tabuľka

1. Vytvorenie tabuľky

```
CREATE TABLE [schema.]nazov_tabulky
  ( { nazov_stlpca datatype [DEFAULT expr] { [column_constraint] }[...]
      |
      table_constraint
    }[...]
  )
```

2. Úprava štruktúry tabuľky

```
ALTER TABLE [schema.]nazov_tabulky
{ ADD (   {
           nazov_stlpca datatype [DEFAULT expr] { [column_constraint] }[...]
           |
           table_constraint
           } [, ...]
       )
  |
  MODIFY
  |
  DROP { COLUMN nazov_stlpca
         |
         CONSTRAINT nazov_constraintu
       }
}
```

3. Zrušenie tabuľky (schéma tabuľky spolu s dátami)

```
DROP TABLE [schema.]nazov_tabulky;
```

5.6.1.2 Základná syntax - Obmedzenia v tabuľke

1. Stĺpcové obmedzenie

```
column_constraint ::=
  [CONSTRAINT nazov_constraintu]
  {
     [NOT] NULL
   |
    {UNIQUE | PRIMARY KEY}
   |
    REFERENCES [nazov_schemy.] nazov_tabulky [ (  nazov_stlpca  ) ]
     [ON DELETE CASCADE ]
   |
    CHECK (condition)
  }
```

2. Tabuľkové obmedzenie

```
table_constraint ::=
  [CONSTRAINT nazov_constraintu]
  {
    { UNIQUE | PRIMARY KEY} ( { nazov_stlpca } [,...] )
     |
     FOREIGN KEY ( { nazov_stlpca } [,...] ) REFERENCES
       [nazov_schemy.]nazov_tabulky
          [ ( { nazov_stlpca } [,...] ) ] [ON DELETE CASCADE ]
     |
     CHECK (condition)
  }
```

5.6.1.3 Príklady

1. Vytvorenie tabuľky bez primárneho kľúča

```
CREATE TABLE os_udaje
 (
  rod_cislo CHAR(10) NOT NULL,
  meno VARCHAR2(15) NOT NULL,
  priezvisko VARCHAR2(15) NOT NULL,
  ulica VARCHAR2(20),
  obec VARCHAR2(20),
  psc CHAR(5) NOT NULL,
  okres VARCHAR2(20),
  st_prisl CHAR(2) NOT NULL
);
```

2. Vytvorenie tabuľky **s jednoduchým primárnym kľúčom**

   1. možnosť

```
CREATE TABLE os_udaje
(
  rod_cislo CHAR(10) NOT NULL PRIMARY KEY,
  ...
);
```

   2. možnosť

```
CREATE TABLE os_udaje
(
  rod_cislo CHAR(10) NOT NULL,
  ...,
  PRIMARY KEY (rod_cislo)
);
```

   3. možnosť

```
CREATE TABLE os_udaje
(
  rod_cislo CHAR(10) NOT NULL,
  ...
);

ALTER TABLE os_udaje
  ADD ( PRIMARY KEY (rod_cislo) );
```

3. Vytvorenie tabuľky **s kompozitným primárnym kľúčom**

   1. možnosť

```
CREATE TABLE st_odbory
(
  c_st_odboru  SMALLINT NOT NULL,
  c_specializacie SMALLINT NOT NULL,
  ...,
  PRIMARY KEY (c_st_odboru, c_specializacie)
);
```

1. možnosť

```
CREATE TABLE st_odbory
(
  c_st_odboru      SMALLINT   NOT NULL,
  c_specializacie SMALLINT   NOT NULL,
  ...
);

ALTER TABLE st_odbory
  ADD ( PRIMARY KEY (c_st_odboru, c_specializacie) );
```

4. **Nesprávne pokusy** o vytvorenie kompozitného primárneho kľúča

   (a) NIE JE MOŽNÉ – Syntax ERROR – lebo PK môže byť len jeden

```
CREATE TABLE TAB1
( pk1  INTEGER  NOT NULL  PRIMARY KEY ,
  pk2  INTEGER  NOT NULL  PRIMARY KEY );
```

   (b) JE MOŽNÉ

```
CREATE TABLE TAB1
( pk1  INTEGER  NOT NULL  UNIQUE KEY ,
  pk2  INTEGER  NOT NULL  UNIQUE KEY );
```

   ale výsledok znamená, že relácia `TAB1` má dvoch kandidátov primárneho kľúča:

   - **KPK1:** `pk1`
   - **KPK2:** `pk2`

   ale **NIE** kompozitný PK zložený z atribútov `pk1, pk2`

5. Vytvorenie **cudzieho kľúča**

   1. možnosť – pozor na poradie tabuľka `os_udaje` aj spolu s primárnym kľúčom už musí existovať

```
CREATE TABLE student
(
  os_cislo  INTEGER NOT NULL PRIMARY KEY,
  rod_cislo  CHAR(10) NOT NULL,
  ...,
  FOREIGN KEY (rod_cislo) REFERENCES os_udaje (rod_cislo)
);
```

   2. možnosť – využíva sa aby nebolo potrebné dbať na poradie tabuliek, stačí len príkazy spúšťať v poradí – CREATE TABLE, ALTER TABLE ADD PRIMARY KEY, a potom ALTER TABLE ADD FOREIGN KEY:

```
ALTER TABLE student
  ADD ( FOREIGN KEY (rod_cislo) REFERENCES os_udaje (rod_cislo) );
```

5.6.1.4 Premenovanie

- **Premenovanie stĺpca**

```
ALTER TABLE table_name
  RENAME COLUMN old_name TO new_name;
```

- **Premenovanie tabuľky**

```
ALTER TABLE table_name
  RENAME TO new_table_name;
```

5.6.1.5 Zmena dátového typu stĺpca

Je možné uskutočniť len rozširujúce zmeny (napr. z pôvodného CHAR(2) na CHAR(6))

```
ALTER TABLE table_name
  MODIFY column_name new_data_type;
```

5.6.1.6 Zmena NULL/NOT NULL

Tieto úpravy samozrejme musia dovoliť dáta v tabuľke.

a)
```
ALTER TABLE table_name
   MODIFY column_name NOT NULL;
```

b)
```
ALTER TABLE table_name
   MODIFY column_name  NULL;
```

5.6.1.7 CHECK podmienka

1. Definícia pri stĺpci:

```
CREATE TABLE tab1
(
   id             integer   primary key,
   kladne_cislo   integer   CHECK ( kladne_cislo > 0 ),
   stav           char(1)   NOT NULL CHECK ( stav IN ('a','n') )
);
```

2. Definícia ako tabuľkové obmedzenie:

```
CREATE TABLE tab1
(
   id             integer   primary key,
   kladne_cislo   integer   CHECK ( kladne_cislo > 0 ),
   stav           char(1)   NOT NULL CHECK ( stav IN ('a','n') )
);
```

3. Doplenie pomocou ALTER TABLE:

```
ALTER TABLE tab1
    ADD CHECK ( kladne_cislo > 0 )
;
```

5.6.1.8 DEFAULT hodnota

1.
```
CREATE TABLE tab1
(
   id             integer   primary key,
   stav           char(1)   DEFAULT 'S'  CHECK( stlpec in ('S','T') ) NOT NULL
);
```

```
ALTER TABLE tab1
  MODIFY stav DEFAULT 'S';
```

5.6.2 DDL a vzťahy

5.6.2.1 DDL a kardinalita vzťahu

- **1:1**

```
ALTER TABLE FK_tabulka
  ADD FOREIGN KEY (FK_stlpce) REFERENCES PK_tabulka;

ALTER TABLE FK_tabulka
  ADD UNIQUE (FK_stlpce);
```

- **1:N**

```
ALTER TABLE FK_tabulka
  ADD FOREIGN KEY (FK_stlpce) REFERENCES PK_tabulka;
```

5.6.2.2 DDL a povinnosť členstva vo vzťahu

- povinné členstvo

```
CREATE TABLE FK_tabulka
  (
     ...
     FK_stlpec1       dat_typ    NOT NULL,
     FK_stlpec2       dat_typ    NOT NULL,
     ....);

ALTER TABLE FK_tabulka
  ADD FOREIGN KEY (FK_stlpce) REFERENCES PK_tabulka;
```

5.6.2.3 DDL a typ vzťahu

1. **Identifikačný vzťah** - FK sa stáva súčasťou PK v tabuľke `fk_tabulka`

```
CREATE TABLE fk_tabulka
(
    pk_stlpec1   dat_typ   NOT NULL,
    pk_stlpec2  dat_typ   NOT NULL,
   ....,
    PRIMARY KEY ( pk_stlpec1, pk_stlpec2 )
);


CREATE TABLE fk_tabulka
(
    pk_stlpec    dat_typ   NOT NULL,
    PFK_stlpec1  dat_typ   NOT NULL,
    PFK_stlpec2  dat_typ   NOT NULL,
    ...

    PRIMARY KEY ( pk_stlpec,  PFK_stlpec1, PFK_stlpec2 ),
    FOREIGN KEY ( PFK_stlpec1, PFK_stlpec2 ) REFERENCES pk_tabulka
);
```

2. **Neidentifikačný vzťah** - klasický FK, ktorý sa nestáva súčasťou PK

```
CREATE TABLE fk_tabulka
(
    pk_stlpec    dat_typ   NOT NULL,
    FK_stlpec1   dat_typ   ,
    FK_stlpec2   dat_typ   ,
    ...

    PRIMARY KEY ( pk_stlpec ),
    FOREIGN KEY ( PFK_stlpec1, PFK_stlpec2 ) REFERENCES pk_tabulka
);
```

5.6.3 Index

1. Vytvorenie indexu

```
CREATE [UNIQUE] [CLUSTER] INDEX [schema.]nazov_indexu
  ON [schema.]nazov_tabulky (nazov_stlpca [ASC | DESC],...)
```

2. Znovu vytvorenie indexu.

```
ALTER INDEX [schema.]nazov_indexu REBUILD;
```

3. Zrušenie indexu.

```
DROP INDEX [schema.]nazov_indexu;
```

5.6.3.1 Príklady

- Vytvorenie **duplikátneho** indexu

```
CREATE INDEX ind_zp_oc ON  zap_predmety (os_cislo);
```

- Vytvorenie **unikátneho** indexu

```
CREATE UNIQUE INDEX ind_ou ON  os_udaje (rod_cislo);
```

```
CREATE UNIQUE INDEX ind_zp ON
zap_predmety(os_cislo, skrok, cis_predmet);
```

- Index s určením **smeru triedenia**:

```
CREATE INDEX ind_vysl ON vysledok
  ( pocet_bodov DESC, id_cloveka ASC );

alebo

CREATE INDEX ind_vysl ON vysledok
  ( pocet_bodov DESC, id_cloveka );
```

- **Zrušenie** indexu

```
DROP INDEX ind_zp;
```

5.6.4 Funkcionálne indexy

```
CREATE [UNIQUE] [CLUSTER] INDEX meno_indexu
ON meno_tabulky (funkcia(parametre) [ASC | DESC],...)
```

Tento druh indexu namiesto indexovania priamo stĺpcov, umožní indexovať výsledok funkcie (štandardnej, alebo aj užívateľom definovanej), pričom parametrami môžu byť stĺpce, alebo konštanty. Dôvodom je optimalizácia SQL dotazov.

```
CREATE INDEX idx_meno
    ON os_udaje ( upper(priezvisko));
```