Geografická báza údajov 1
Cvičenie 5

Vladimír Pelech

pelech2@uniba.sk, G-23

PRÍRODOVEDECKÁ FAKULTA
Univerzita Komenského
v Bratislave

Cvičenie 5

- Náplň:
  - Opakovanie z minulej hodiny
  - Obmedzenia a referencie
  - Kombinovanie dopytov pomocou UNION, INTERSECT, EXCEPT
  - Hlbší pohľad na DROP TABLE
  - Cvičné úlohy

Opakovanie z minulej hodiny

- Ako je možné prepájať tabuľky?
- Aké typy JOINov poznáme?
- Čo získame pri použití FULL OUTER JOIN?

Príprava na cvičenie

- Skontrolujte si, či v schéme *cviko3* máte tabuľky *firmy*, *oddelenia* a *zamestnanci.*
- Ak nie, tak si stiahnite údaje k cvičeniu 4 (cvicenie4Data.zip) a odzipujte ich.
- Vytvorte v schéme *cviko3* tabuľky *firmy*, *oddelenia* a *zamestnanci.*
- Názvy stĺpcov a ich dátové typy voľte podľa stĺpcov z odzipovaných *.csv* súborov.
- Importujte si csv súbory do príslušných tabuliek.

Účel obmedzení

- Zachovanie referenčnej integrity v tabuľke
- Znižovanie redundantnosti dát
- Kontrola vkladaných dát – znižuje sa chybovosť dát
- Vyžadovanie určitých atribútov

Obmedzenia (Constraints)

- Je nutné ich aplikovať na konkrétny stĺpec, alebo kombináciu stĺpcov, tabuľky. Na jeden stĺpec môže byť aj viacero obmedzení.
- V postgreSQL je sedem druhov obmedzení:
  - Nenulové hodnoty (NOT NULL)
  - Primárny kľúč (PRIMARY KEY)
  - Jedinečnosť (UNIQUE)
  - Cudzí kľúč (FOREIGN KEY)
  - Kontrola (CHECK)
  - Prednastavená hodnota (DEFAULT) – prísne vzaté to nie je obmedzenie
  - Vylúčenie (EXCLUDE)-iba na ukážku

Vytvorenie a odstránenie obmedzenia

- Vytvorenie je možné troma základnými spôsobmi:
  - Priamo počas vytvárania tabuľky v dopyte CREATE TABLE.
  - Pre už vytvorenú tabuľku je možné použiť:
    - dopyt ALTER TABLE,
    - alebo cez grafiku.
- Syntax SQL sa pre jednotlivé typy obmedzení mierne líši.
- Odstránenie obmedzenia pomocou ALTER TABLE alebo grafiku.

„Nenullové“ hodnoty (NOT NULL)

- Pri tomto obmedzení bude do príslušného stĺpca vždy vyžadovaná hodnota. Stĺpec nemôže obsahovať hodnotu NULL – bez hodnoty.
- Pri CREATE TABLE stačí za dátový typ stĺpca zadať NOT NULL.

```
4  CREATE TABLE cviko3.firmy1 (
5  id_firmy smallint NOT NULL,
6  firma varchar (30) NOT NULL);
7
8  INSERT INTO cviko3.firmy1 (id_firmy) VALUES (5);
```

Data Output | Explain | Messages | Notifications

```
ERROR:  null value in column "firma" violates not-null constraint
DETAIL:  Failing row contains (5, null).
SQL state: 23502
```

„Nenullové“ hodnoty (NOT NULL)

- Pri využití ALTER TABLE sa použije syntax:

  ALTER TABLE názov schémy.názov tabuľky ALTER [COLUMN] názov stĺpca SET NOT NULL;

```
 8  ALTER TABLE cviko3.firmy
 9  ALTER COLUMN firma  SET NOT NULL;
10
11  INSERT INTO cviko3.firmy (id_firmy) VALUES (5);
```

Data Output | Explain | Messages | Notifications

```
ERROR:  null value in column "firma" violates not-null constraint
DETAIL:  Failing row contains (5, null).
SQL state: 23502
```

- Na odstránenie je potrebné v predchádzajúcom príkaz nahradiť „*SET*“ slovom „*DROP*“.

Primárny kľúč (PRIMARY KEY)

- Obsahuje iba jedinečné hodnoty slúžiace na identifikáciu záznamu v tabuľke.
- Každá tabuľka môže obsahovať iba jeden primárny kľúč.
- Stĺpec s obmedzením primárneho kľúča nemôže obsahovať hodnoty NULL.
- Môže byť vytvorený aj ako kombinácia viacerých stĺpcov, pričom výsledná kombinácia musí byť jedinečná.
- O prítomnosti primárneho kľúča v tabuľke svedčí označenie [PK] v niektorom zo stĺpcov.

Primárny kľúč (PRIMARY KEY)

- Počas vytvárania tabuľky stačí zadať za dátový typ stĺpca PRIMARY KEY.

```
CREATE TABLE cviko3.firmy (
id_firmy smallint PRIMARY KEY,
firma varchar (30));
```

- Alebo ho definovať na konci po definovaní všetkých stĺpcov.

```
CREATE TABLE cviko3.firmy (
id_firmy smallint,
firma varchar (30),
PRIMARY KEY (id_firmy));
```

Primárny kľúč (PRIMARY KEY)

- Pri využití ALTER TABLE sa použije syntax:

ALTER TABLE názov schémy.názov tabuľky ADD PRIMARY KEY (názov stĺpca/ov);

```
ALTER TABLE cviko3.firmy
ADD PRIMARY KEY (id_firmy);
```

- Na odstránenie sa používa syntax:

ALTER TABLE názov schémy.názov tabuľky DROP CONSTRAINT názov tabuľky+_+pkey;

```
ALTER TABLE cviko3.firmy
DROP CONSTRAINT firmy_pkey;
```

Obmedzenie jedinečnosti (UNIQUE)

- Daný stĺpec môže obsahovať iba unikátne hodnoty.
- Je podobný ako obmedzenie primárneho kľúča, obmedzení typu UNIQUE však môže byť v jednej tabuľke viac.
- Počas vytvárania tabuľky stačí zadať za dátový typ stĺpca UNIQUE.

```
CREATE TABLE cviko3.firmy (
id_firmy smallint UNIQUE,
firma varchar (30));
```

Obmedzenie jedinečnosti (UNIQUE)

- Pri využití ALTER TABLE sa použije syntax:

  ALTER TABLE názov_schémy.názov_tabuľky ADD UNIQUE (názov_stĺpca);

```
ALTER TABLE cviko3.firmy
ADD UNIQUE(id_firmy);
```

- Na odstránenie sa používa syntax:

  ALTER TABLE názov schémy.názov tabuľky DROP CONSTRAINT názov tabuľky+_+názov stĺpca+_+key;

```
ALTER TABLE cviko3.firmy
DROP CONSTRAINT firmy_id_firmy_key;
```

Pomenovanie obmedzení

- Z predchádzajúceho príkladu odstránenia je zrejmé, že ide o pomerne ťažkopádnu syntax, kde je potrebné poznať názov tabuľky, stĺpca a aj obmedzenie.
- Preto je výhodnejšie aj pri tvorbe obmedzenia zadávať jeho názov.
- Zadať obmedzeniu názov je možné pri oboch spôsoboch vytvorenia, aj pri vytváraní tabuľky, aj pri zmene už vytvorenej tabuľky.
- Pre názvy platia rovnaké pravidlá ako pre názvy stĺpcov.
- Názvy obmedzení v rámci schémy musia byť jedinečné.

Pomenovanie obmedzení

- Na pomenovanie obmedzení je nutné použiť kľúčové slovo CONSTRAINT:
  - Pri vytváraní tabuľky

```
CREATE TABLE cviko3.firmy (
id_firmy smallint CONSTRAINT unikatne_obmedzenie UNIQUE,
firma varchar (30));
```

  - Pri ALTER TABLE

```
ALTER TABLE cviko3.firmy
ADD CONSTRAINT unikatne_obmedzenie UNIQUE(id_firmy);
```

Pomenovanie obmedzení

- Pomenovanie obmedzení veľmi zjednodušuje ich prípadné odstránenie podľa syntaxe:

ALTER TABLE názov schémy.názov tabuľky DROP CONSTRAINT názov obmedzenia;

```
ALTER TABLE cviko3.firmy
DROP CONSTRAINT unikatne_obmedzenie;
```

- Pomenovanie je možné použiť aj pri iných typoch obmedzenia.
- V prípade neuvedenia názvu pre obmedzenie mu databázový systém názov vytvorí automaticky.

Vytvorenie tzv. tabuľkového obmedzenia

- Tabuľkové obmedzenie funguje rovnako ako bežné obmedzenie, avšak je viazané na kombináciu stĺpcov.

```
CREATE TABLE cviko3.firmy (
id_firmy smallint,
firma varchar (30),
CONSTRAINT tab_obmedzenie UNIQUE(id_firmy,firma));
```

Cudzí kľúč (FOREIGN KEY)

- Stĺpec, ktorý je cudzím kľúčom v nejakej tabuľke, obsahuje hodnoty primárneho kľúča z inej tabuľky.
- Spolu s primárnym kľúčom slúži na vzájomné prepojenie hodnôt medzi tabuľkami.
- Na poslednej hodine sa prepájali tabuľky aj bez toho, aby obsahovali obmedzenia vo forme primárnych a cudzích kľúčov, na čo sú potom potrebné?

Cudzí kľúč (FOREIGN KEY)

- Pri cudzom kľúči potomkovskej tabuľky platí, že môže obsahovať iba také hodnoty, aké obsahuje aj primárny kľúč rodičovskej tabuľky.
- Uvedená vlastnosť slúži na zachovanie referenčnej integrity dát.
- Výhody spočívajú v tom, že dáta sú na navzájom seba naviazané a slúžia na prípadnú kontrolu.
- Primárny kľúč môže byť v tabuľke len jeden, aj keď môže byť tvorený kombináciou viacerých stĺpcov. Cudzích kľúčov môže byť viac a na rôzne tabuľky.

Cudzí kľúč (FOREIGN KEY)

- Počas vytvárania tabuľky sa cudzí kľúč, alebo kľúče, vytvoria definovaním po všetkých stĺpcoch tabuľky podľa syntaxe:

  CONSTRAINT názov cudzieho kľúča FOREIGN KEY (stĺpec) REFERENCES názov schémy.názov rodičovskej tabuľky(názov stĺpca jej primárneho kľúča)

- Pre názov obmedzenia typu cudzieho kľúča platí dohoda: FK_názov potomka_názov rodiča

```
CREATE TABLE cviko3.oddelenia (
id_odd smallint,
oddelenie varchar (30),
firma smallint,
CONSTRAINT FK_oddelenia_firmy FOREIGN KEY (firma) REFERENCES cviko3.firmy(id_firmy));
```

Cudzí kľúč (FOREIGN KEY)

- Pri využití ALTER TABLE sa použije syntax:

  ALTER TABLE názov schémy.názov tabuľky ADD CONSTRAINT názov cudzieho kľúča FOREIGN KEY (stĺpec) REFERENCES názov schémy.názov rodičovskej tabuľky (názov stĺpca jej primárneho kľúča);

```
ALTER TABLE cviko3.oddelenia
ADD CONSTRAINT FK_oddelenia_firmy FOREIGN KEY (firma) REFERENCES cviko3.firmy(id_firmy);
```

- Na odstránenie cudzieho kľúča stačí využiť už známu syntax s názvom:

```
ALTER TABLE cviko3.oddelenia
DROP CONSTRAINT FK_oddelenia_firmy;
```

Cudzí kľúč (FOREIGN KEY)

- Ak ste cudzí kľúč odstránili, tak ho pridajte ešte raz.

```
ALTER TABLE cviko3.oddelenia
ADD CONSTRAINT FK_oddelenia_firmy FOREIGN KEY (firma) REFERENCES cviko3.firmy(id_firmy);
```

- Pokúste sa vložiť do tabuľky *oddelenia* nasledovný záznam:

INSERT INTO cviko3.oddelenia VALUES (14,'Nové oddelenie',6);

```
ERROR:  insert or update on table "oddelenia" violates foreign key constraint "fk_oddelenia_firmy"
DETAIL:  Key (firma)=(6) is not present in table "firmy".
SQL state: 23503
```

Kontrola (CHECK)

- Počas vytvárania tabuľky stačí zadať za dátový typ stĺpca CHECK a definovať výraz, ktorý sa požaduje kontrolovať.

```
CREATE TABLE cviko3.firmy (
id_firmy smallint CONSTRAINT kontrola CHECK (id_firmy < 10),
firma varchar (30));
```

- Kontrolovať je možné aj požadované reťazce s využitím operátora IN:

  CHECK (názov stĺpca IN ('hodnota1','hodnota2',atď.))

- Kontrolovaný výraz môže obsahovať aj logické spojky:

  CHECK (názov stĺpca1 > 10 AND názov stĺpca1 > názov stĺpca2)

Prednastavená hodnota (DEFAULT)

- Nejde o obmedzenie v pravom zmysle slova.
- V prípade, že pri importe záznamov do tabuľky nebude zadaná presná hodnota alebo NULL, tak bude daný atribút vyplnený prednastavenou hodnotou.
- Počas vytvárania tabuľky stačí zadať za dátový typ stĺpca DEFAULT a prednastavenú hodnotu (číslo, dátum, reťazec,...) podľa typu stĺpca.

```
CREATE TABLE cviko3.oddelenia (
id_odd smallint,
oddelenie varchar (30) DEFAULT 'oddelenie',
firma smallint);
```

Prednastavená hodnota (DEFAULT)

- Pri využití ALTER TABLE sa použije syntax:

  ALTER TABLE názov schémy.názov tabuľky ALTER COLUMN názov stĺpca SET DEFAULT hodnota;

```
ALTER TABLE cviko3.oddelenia
ALTER COLUMN oddelenie
SET DEFAULT 'oddelenie';
```

- Pre odstránenie sa použije syntax:

  ALTER TABLE názov schémy.názov tabuľky ALTER COLUMN názov stĺpca DROP DEFAULT;

Vylúčenie (EXCLUDE)

- Na jeho vytvorenie je potrebné najprv do databázy pridať EXTENSION *btree_gist.*
- Zaistí, aby z akýchkoľvek dvoch riadkoch v kontrolovaných stĺpcoch alebo výrazoch, aspoň jeden z nich vrátil hodnotu NULL alebo FALSE.

Vylúčenie (EXCLUDE)

- Počas vytvárania tabuľky sa obmedzenie vylúčenia vytvorí podľa syntaxe:

  CONSTRAINT názov obmedzenia EXCLUDE USING gist (stĺpec1 WITH operátor[, stlpec2 WITH operátor,..])

```
CREATE TABLE cviko3.firmy (
id_firmy smallint,
firma varchar (30),
CONSTRAINT vylucenie EXCLUDE USING gist
(firma WITH <>));
```

- Pokúste sa do takto vytvorenej tabuľky vložiť dvojicu nových záznamov. Nemalo by sa vám to podariť, ak zadávate názov, ktorý sa v tabuľke nevyskytuje.

Vylúčenie (EXCLUDE)

- Pri využití ALTER TABLE sa použije syntax:

  ALTER TABLE názov_schémy.názov_tabuľky ADD CONSTRAINT názov obmedzenia EXCLUDE USING gist (stĺpec1 WITH operátor[, stlpec2 WITH operátor,..])

```
ALTER TABLE cviko3.firmy
ADD CONSTRAINT vylucenie EXCLUDE USING gist
(firma WITH <>);
```

- Na odstránenie obmedzenia stačí využiť už známu syntax s názvom.

Viacnásobné odstránenie a/alebo pridanie obmedzení

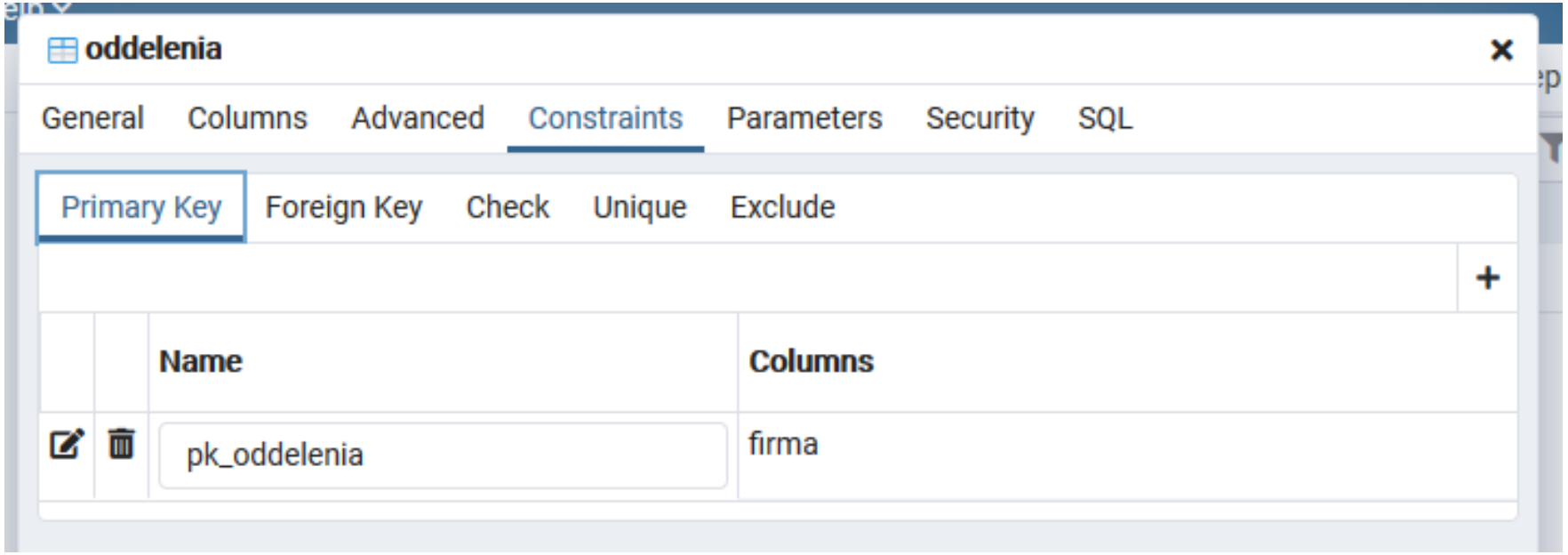
- Pri vytváraní tabuľky uviesť jednotlivé obmedzenia na príslušnom mieste.
- Pri ALTER TABLE len pridať čiarku medzi jednotlivé vytvárané alebo odstraňované obmedzenia.

```
ALTER TABLE cviko3.firmy
DROP CONSTRAINT vylucenie,
ADD CONSTRAINT vylucenie EXCLUDE USING gist
(firma WITH <>),
ADD CONSTRAINT primarny_kluc PRIMARY KEY(id_firmy);
```

- Je vhodné ukladať dopyty, ktoré slúžili na vytvorenie tabuliek a obmedzení.

Vytváranie obmedzení cez GRAFIKU

- Klik pravým tlačidlom myši na zvolenú tabuľku a výber možnosti *Properties* (Vlastnosti).
- V záložke Constraints sú uvedené jednotlivé obmedzenia podľa typu.



Kombinovanie dopytov

- Syntax SQL umožňuje vykonať sled dopytov, ako sled krokov, ak sú navzájom oddelené znakom bodkočiarky.
- Pri viacnásobnom použití dopytu SELECT, však každý nasledujúci premaže použitie predchádzajúceho.
- V prípade potreby zobrazenia výsledkov z viacerých dopytov súčasne je možné ich spájať pomocou kľúčových slov:
  - UNION
  - INTERSECT
  - EXCEPT
- Syntax platná pre všetky: dopyt1 UNION [ALL] dopyt2

Kombinovanie dopytov - UNION

- K výsledku prvého dopytu pridá výsledok druhého dopytu
- V prípade opakovane sa vyskytujúceho záznamu vráti iba jedinečný záznam, podobne ako pri DISTINCT, pokiaľ sa nepoužije UNION ALL.
- Pri všetkých typoch spojenia platí, že spájané dopyty musia vrátiť rovnaký počet stĺpcov, a zodpovedajúce stĺpce musia mať navzájom rovnaký dátový typ.

```
SELECT * FROM cviko3.cviko3 WHERE obec LIKE 'Lúčka%' --samostatne 4 záznamy
UNION
SELECT * FROM cviko3.cviko3 WHERE obec LIKE 'Lúč%'; --samostatne 10 záznamy
```

- Uvedený dopyt vráti 10 záznamov, s použitím UNION ALL 14 záznamov.

Kombinovanie dopytov - INTERSECT

- INTERSECT vráti spoločné záznamy pre oba dopyty, pričom bez použitia ALL ich vráti iba raz.

```
SELECT okres FROM cviko3.cviko3 WHERE okres LIKE 'Okres Michalovce' --samostatne 78 záznamov
INTERSECT
SELECT okres FROM cviko3.cviko3 WHERE okres LIKE 'Okres M%'; --samostatne 187 záznamov
```

- Uvedený dopyt vráti 1 záznam (pozor je žiadaný iba stĺpec okres), s použitím ALL 78 záznamov.

Kombinovanie dopytov - EXCEPT

- EXCEPT vráti záznamy z prvého dopytu, ktoré sa nevyskytujú v druhom dopyte, pričom bez použitia ALL ich vráti iba raz.

```
SELECT okres FROM cviko3.cviko3 WHERE okres LIKE 'Okres M%' --samostatne 187 záznamov
EXCEPT
SELECT okres FROM cviko3.cviko3 WHERE okres LIKE 'Okres Michalovce'; --samostatne 78 záznamov
```

- Uvedený dopyt vráti 4 záznamy (pozor je žiadaný iba stĺpec okres), s použitím ALL 109 záznamov.

Mazanie tabuliek s nastavenou referenciou

- Pokúste sa vymazať rodičovskú tabuľku, na ktorú je nastavená referencia v podobe cudzieho kľúča. Nemalo by sa to podariť.

```
CREATE TABLE cviko3.firmy (
id_firmy smallint,
firma varchar (30),
CONSTRAINT PK_firmy PRIMARY KEY(id_firmy));

CREATE TABLE cviko3.oddelenia (
id_odd smallint,
oddelenie varchar (30),
firma smallint,
CONSTRAINT PK_oddelenia PRIMARY KEY (firma),
CONSTRAINT FK_oddelenia_firmy FOREIGN KEY (firma) REFERENCES cviko3.firmy(id_firmy))
;

DROP TABLE cviko3.firmy;
```

Mazanie tabuliek s nastavenou referenciou

- Problémom je práve vytvorená referencia, alebo vytvorený pohľad (view) na danú tabuľku.
- V takom prípade je potrebné príkaz DROP TABLE doplniť.
- Celá syntax príkazu DROP TABLE:

  DROP TABLE [IF EXISTS] názov schémy.názov tabuľky [, ...] [CASCADE | RESTRICT];

  IF EXISTS – voliteľné, ak daná tabuľka neexistuje, tak nevráti Error

  CASCADE – voliteľné, odstráni objekty nadväzujúce na príslušnú tabuľku ako napr. pohľady, v prípade existencie referencie odstráni iba dané obmedzenie a nie potomkovskú tabuľku

  RESTRICT – voliteľné, neodstráni tabuľku v prípade existencie pohľadov alebo referencie, platí v prípade neuvedenia žiadnej voľby
- Potomkovskú tabuľku je možné mazať, ak na ňu neexistuje iná referencia.

Opakovanie z tohto cvičenia

- Vytváranie rôznych typov obmedzení,
- rôzne spôsoby vytvorenia obmedzení,
- pomenovanie obmedzení,
- referencie pomocou cudzieho kľúča,
- odstránenie obmedzení,
- hlbší pohľad na DROP TABLE.

Vaše otázky?

Otázky

- Načo sú dobré obmedzenia?
- Ako je možné vytvoriť obmedzenie existujúcej tabuľke?
- Je možné odstrániť obmedzenie? Ak áno, tak ako?
- Aký je rozdiel medzi obmedzeniami UNIQUE a PRIMARY KEY?
- Aký je vzťah medzi primárnym kľúčom a cudzím kľúčom?

- Majme tabuľku obce (id, nazov, okres, pocet_obyv) v schéme public, ktorá vznikla nasledujúcim príkazom:

```
CREATE TABLE obce (
id int PRIMARY KEY,
nazov varchar (30) UNIQUE,
okres varchar (30),
pocet_obyv int)
```

- Čo vykonajú nasledujúce dopyty?

```
INSERT INTO obce VALUES
(1,'Horný Koniec‘, 'Konce', 500),
(2,'Dolný Koniec','Konce', 1000),
(3,'Vyšný Okraj','Kraje',1500),
(4,'Nižný Okraj','Kraje',1500),
(5,'Horný Koniec','Kraje',1500);
```

```
INSERT INTO obce VALUES
(1,'Horný Koniec','Konce', 500),
(2,'Dolný Koniec','Konce', 1000),
(3,'Vyšný Okraj','Kraje',1500),
(3,'Nižný Okraj','Kraje',1500);
```

```
ALTER TABLE obce
DROP CONSTRAINT
obce_nazov_key;

INSERT INTO obce VALUES
(1,'Horný Koniec','Konce', 500),
(2,'Dolný Koniec','Konce', 1000),
(3,'Vyšný Okraj','Kraje',1500),
(4,'Nižný Okraj','Kraje',1500),
(5,'Horný Koniec','Kraje',1500);
```

Praktické úlohy

- V schéme *ulohy* si vytvorte tabuľku *Lucka,* kde budú všetky záznamy z tabuľky *cviko3.obyvatelia,* kde je v názve obce „Lúčka“.
- Vytvorenej tabuľke:
  - pridajte stĺpec ID s dátovým typom int a naplňte ho unikátnymi hodnotami,
  - pridajte primárny kľúč na stĺpec ID s názvom *primarny,*
  - pridajte obmedzenie neopakovania názvu okresu s názvom „unikatny“,
  - pridajte obmedzenie na kontrolu pridania údajov do stĺpca 2009 viac ako 0.
- V schéme *ulohy* si vytvorte tabuľku *Casti (id int, nazov varchar(20), id_obce int),* tak aby stĺpec *id* bol primárnym kľúčom a stĺpec *id_obce* bol cudzím kľúčom odkazujúci sa na primárny kľúč tabuľky *Lucka.*