Relacyjne bazy danych w interpretacji MySQL - podstawy

Ćwiczenie 4 i 5

Połączenie z serwerem MySQL

1. Zaloguj się na swoje osobiste konto na serwerze info3.meil.pw.edu.pl (dostęp do bazy danych jest możliwy tylko z tego serwera).
2. Klient MySQL'a urchamia komenda: `mysql -u mysql` Program wita nas krótkim komunikatem oraz znakiem zachęty:

```
mysql >
```

3. Dostępne bazy danych możemy wyświetlić za pomocą polecenia:

```
mysql > SHOW DATABASES;
```

4. Jest ich dużo. My chcemy skorzystać z bazy danych o nazwie *test*:

```
mysql > USE test;
```

Baza danych *test*

1. Sprawdź jakie tabele zawarte są w bazie danych *test*:

```
mysql> SHOW TABLES;
```

   Okaże się, że są to: *City*, *Country* i *CountryLanguage*. Informacje jakie są przechowywane w poszczególnych tabelach (definicję tabeli) wyświetli poniższe polecenie, np. dla tabeli *City*:

```
mysql> DESCRIBE City;
```

   Sprawdź jakie wartości przechowują poszczególne tabele.

2. Podstawowe zapytania (zapytanie - *query*)

- Wyświetl wszystkie informacje zawarte w tabeli Country:

```
SELECT * FROM Country;
```

- Wyświetl wartości wszystkich rekordów dwóch pól (kolumn) (np. Name, Region):

```
mysql> SELECT Name, Region FROM Country;
```

- Wyświetl nazwy wszystkich państw leżących w Europie wraz z długością życia ich mieszkańców:

```
mysql> SELECT Name, LifeExpectancy FROM Country
-> WHERE Continent='Europe';
```

- Wyświetl nazwy wszystkich państw leżących w Europie i Azji wraz z długością życia ich mieszkańców:

```
mysql> SELECT Name, LifeExpectancy FROM Country
-> WHERE Continent IN ('Europe', 'Asia');
```

- Wyświetl informację z punktu poprzedniego, ale posortowaną względem długości życia (dodaj do poprzedniej komendy frazę - `ORDER BY LifeExpectancy`).
- Wyświetl liczbę ludności żyjącej w Europie:

```
mysql> SELECT sum(Population) FROM Country WHERE Continent='Europe';
```

- Policz średnie zaludnienie krajów w Europie:

```
mysql> SELECT avg(Population) FROM Country WHERE Continent='Europe';
```

- Znajdź nazwy i kody wszystkich państw, których nazwy zaczynaj się od „Ch":

```
mysql> SELECT Name, Code FROM Country WHERE Name LIKE 'Ch%';
```

- Wyświetl wszystkie informacje o miastach w Finlandii (CountryCode Finlandii to FIN):

```
mysql> SELECT * FROM City WHERE CountryCode = 'FIN';
```

- Wyświetl wszystkie informacje o miastach w Polsce (CountryCode Polski to POL) i posortuj je według województw:

```
mysql> SELECT * FROM City WHERE CountryCode = 'POL'
-> ORDER BY District;
```

- Wyświetl nazwy krajów, które uzyskały niepodległość po roku 1980. Wyświetl również rok uzyskania niepodległości.

```
mysql> SELECT Name, IndepYear FROM Country WHERE IndepYear > 1980;
```

- Wyświetl nazwy krajów Ameryki Północnej, które uzyskały niepodległość pomiędzy rokiem 1800 a rokiem 1900. Wyświetl również rok uzyskania niepodległości. Posortuj dane według tej daty:

```
mysql> SELECT Name, IndepYear from Country
-> WHERE Continent = 'North America' AND
-> IndepYear > 1800 AND
-> IndepYear < 1900
-> ORDER BY IndepYear;
```

3. Zapytania bardziej zaawansowane

- Wyświetl nazwy miast o ludności przekraczającej 3 000 000. Wyświetl również kody państw, w których te miasta leżą i liczbę ludności. Posortuj dane w kolejności malejącej według kodu kraju, a następnie populacji (w przypadku alfabetu, kolejnosc malejaca oznacza porzadek od Z do A):

```
mysql> SELECT Name, CountryCode, Population FROM City
-> WHERE Population > 3000000
-> ORDER BY CountryCode DESC, Population DESC;
```

- Wyświetl wszystkie miasta w Norwegii (załóż, że nie znasz wartości *CountryCode* tego państwa):

```
mysql> SELECT * FROM City WHERE CountryCode=
-> (SELECT Code FROM Country WHERE Name='Norway');
```

- Wyświetl nazwę najbardziej zaludnionego państwa w Ameryce południowej; obok nazwy wyświetl liczbę jego ludności:

```
mysql> SELECT Name, Population FROM Country
-> WHERE Population=
-> (SELECT MAX(Population) FROM Country
-> WHERE Continent='South America');
```

- Wyświetl liczbę państw leżących na każdym kontynencie:

```
mysql> SELECT Continent, Count(*) AS 'Total Population'
-> FROM Country GROUP BY Continent;
```

- Wyświetl nazwy wszystkich stolic Europejskich (wykorzystaj fakt, że kolumna *ID* w tabeli *City* odpowiada kolumnie *Capital* w tabeli *Country*):

```
mysql> SELECT Name FROM City WHERE ID IN
-> (SELECT Capital FROM Country WHERE Continent='Europe');
```

Tak skonstruowane zapytanie działa bardzo wolno ponieważ sprowadza się do wielokrotnego przeszukiwania tabeli *Country*. W takich przypadkach należy posłużyć się konstrukcją `tab1 INNER JOIN tab2 ON condition`

```
mysql> SELECT City.Name FROM City INNER JOIN Country
-> ON City.ID=Country.Capital
-> WHERE Continent='Europe';
```

- Wyświetl informacje o językach używanych w europejskich państwach:

```
mysql> SELECT Country.Name, CountryLanguage.Language FROM Country
-> INNER JOIN CountryLanguage
-> ON Country.Code=CountryLanguage.CountryCode
-> WHERE Country.Continent='Europe';
```

- Wyświetl nazwę i powierzchnię najmniejszego państwa na świecie:

```
mysql> SELECT Name, SurfaceArea FROM Country
-> WHERE SurfaceArea =
-> (SELECT MIN(SurfaceArea) FROM Country);
```

- Wyświetl nazwę i powierzchnię najmniejszego państwa w Afryce.
- Wyświetl nazwy państw i nazwy ich stolic.
- Wyświetl nazwy państw azjatyckich i ich stolic.
- Wyświetl nazwy państw afrykańskich i ich stolic posortowane według nazwy kraju (użyj aliasów tabel).
- Wyświetl informacje o językach oficjalnych używanych w europejskich państwach.
- Wyświetl wszystkie państwa, w których ludzie mówią po Polsku.
- Wyświetl jakimi językami posługują się mieszkańcy Hiszpanii.
- Wyświetl nazwy państw, które uzyskały niepodległość po roku 1900, w których to państwach językiem oficjalnym jest hiszpański.
- Powtórz powyższe zapytanie dla języków: francuskiego i angielskiego.

4. Pytania dodatkowe:

- Wyświetl nazwy Europejskich krajów, w których czas życia jest krótszy od 70 lat.
- Policz liczbę Europejskich krajów, w których czas życia jest dłuższy od 70 lat.
- Wyświetl średni czas życia na świecie.
- Wyświetl nazwy Europejskich krajów, w których czas życia jest krótszy od średniego czasu życia na świecie.
- Wyświetl nazwy Europejskich krajów, w których czas życia jest krótszy od średniego czasu życia w Europie.
- Wyświetl nazwę najbardziej zaludnionego państwa w Ameryce południowej; obok nazwy wyświetl liczbę jego ludności.
- Wyświetl nazwy krajów na świecie, w których czas życia jest krótszy niż połowa najdłuższego czasu życia w Europie (w kolejności malejącej).
- Wyświetl nazwy krajów na świecie, dla których nie ma danych na temat czasu życia.
- Wyświetl liczbę państw leżących na każdym kontynencie, których ludność liczy powyżej 50 000 000.
- Dla każdego państwa wyświetl sumę ludności mieszkającej w miastach (wykorzystaj kod tego państwa).
- Dla każdego państwa wyświetl sumę ludności mieszkającej w miastach (wykorzystaj kod tego państwa), ale tylko jeśli suma ta przekracza 10 000 000. Otrzymane wartości posortuj w kolejności malejącej.
- Jak w punkcie powyżej, tylko w miejsce kolejnych wywołań `SUM(Population)` użyj aliasu.
- Jak w punkcie powyżej, tylko na wszelki wypadek wyklucz wiersze, w których wystąpił brak danych (`NULL`).
- Jak w punkcie powyżej, tylko weź pod uwagę jedynie miasta mające powyżej 100 000 mieszkańców.
- Wykonaj poniższe zapytanie i zinterpretuj wynik:

```
mysql> SELECT Country.Name, City.Name FROM Country, City;
```

- Wykonaj poniższe zapytanie i zinterpretuj wynik:

```
mysql> SELECT Country.Name, City.Name FROM Country, City
->  WHERE Country.Name = 'Poland';
```

- Wyświetl wszystkie miasta w Europie i nazwę państwa w którym leżą.
- Wyświetl wszystkie miasta w Norwegii. Załóż, że nie znasz wartości CountryCode.
- Wyświetl wszystkie miasta w Polsce. Załóż, że nie znasz wartości CountryCode.
- Wykonaj polecenie z powyższego punktu przy pomocy złączenia tabel.

- Wyświetl wszystkie miasta leżące w kraju, w którym leży Warszawa. Użyj samozłączenia (czyli złączenia tabeli samej ze sobą).
- Znajdź liczbę wystąpień każdego miasta na świecie.
- Znajdź liczbę wystąpień każdego miasta na świecie. Wyświetl jedynie te miasta, które występują przynajmniej 3 razy. Wyniki posortuj.
- Jak w punkcie wyżej tylko przy każdym mieście wyświetlić państwo w którym leży.
- Zakładając, że nie znasz daty uzyskania niepodległości przez Watykan (kod: VAT), wyświetl te europejskie państwa, które uzyskały niepodległość przed uzyskaniem niepodległości przez Watykanem.

Tworzenie i używanie nowych baz danych

0. Utwórz własną bazę danych `studxy` (*xy* to numer przydzielony na początku semestru), która zawierać będzie informacje o osobach i należących do nich numerach telefonów:

```
CREATE DATABASE studxy;
```

Przejdź do nowo utworzonej bazy danych:

```
USE studxy;
```

1. Utwórz pierwszą tabelę, która zawierać będzie informacje o osobach:

```
mysql> CREATE TABLE Person (
-> ID int,
-> Surname varchar(30) NOT NULL,
-> FirstName varchar(30) DEFAULT 'brak',
-> PRIMARY KEY (ID) );
```

Pole ID będzie identyfikatorem osoby. Będzie ono typu całkowitego. Pole Surname będzie zawierało nazwisko osoby. Może ono przechowywać maksymalnie 30 znaków i musi być podane przy dodawaniu nowej osoby (NOT NULL), choć może być podane puste! Pole FirsName będzie przechowywać imię. Załóżmy, że można imienia nie podać i jeśli się tego nie zrobi to domyślnie zostanie wprowadzony tekst 'brak'. Na koniec musimy podać, które pole będzie używane jako klucz główny. W naszym przypadku będzie to pole ID – każdy użytkownik musi mieć numer unikatowy.

- Sprawdź czy utworzona tabela ma poprawną postać:

```
mysql> DESCRIBE Person;
```

2. Utwórz drugą tabelę, która będzie zawierać informacje o numerach telefonów.

- Najpierw utwórz tabelę jedynie z jedną kolumną ID. Kolumna ta powinna być kluczem głównym tabeli `Phone` a wartość pola powinna być automatycznie zwiększana o 1 (przy dodaniu nowego rekordu):

```
mysql> CREATE TABLE Phone (ID int auto_increment PRIMARY KEY);
```

- Dodaj drugą kolumnę zawierającą numery telefonów:

```
mysql> ALTER TABLE Phone ADD Number varchar(16) DEFAULT '';
```

- W przypadku pomyłki możesz usunąć kolumnę z tabeli:

```
mysql> ALTER TABLE Phone DROP COLUMN Number;
```

- Dodaj kolejną kolumnę z zawierającą ID osoby, do której należy dany numer telefonu:

```
mysql> ALTER TABLE Phone ADD PersonID int NOT NULL;
```

- Załóżmy, że numery telefonów nie mogą się powtarzać, czyli że właścicielem danego telefonu może być tylko jedna osoba:

```
mysql> ALTER TABLE Phone ADD UNIQUE (Number);
```

- Podobnie jak poprzednio, sprawdź postać utworzonej tabeli Phone:

```
mysql> DESCRIBE Phone;
```

3. Wprowadź nieco danych do Twojej bazy

- Wprowadź pierwszą osobę. Podaj wszystkie wartości pól (w odpowiedniej kolejności):

```
mysql> INSERT INTO Person VALUES (1, 'Kowalski', 'Jan');
```

- Wprowadź nowe numery telefonów Jana Kowalskiego. Ponieważ pole `ID` tabeli `Phone` jest wypełniane automatycznie nie musimy go podać (choć możemy). Powinniśmy więc poinformować *MySQL'a*, które pola wypełniamy. Służy do tego lista pól podawana w nawiasach po nazwie tabeli:

```
mysql> INSERT INTO Phone (Number, PersonID)
-> VALUES ('022 358 85 58', 1);
```

```
mysql> INSERT INTO Phone (Number, PersonID)
-> VALUES ('0 600 560 780', 1);
```

- Wprowadź nową osobę. Tym razem nie podawaj imienia użytkownika, a nazwisko ustaw na NULL:

```
mysql> INSERT INTO Person (ID, Surname) VALUES (2, NULL);
```

(Która własność tabeli Person spowodowała że wystąpił błąd?)

- Wprowadź nową osobę. Tym razem nie podawaj imienia użytkownika:

```
mysql> INSERT INTO Person (ID, Surname) VALUES (2, 'Dzik');
```

(Jakie imię zostało wpisane do bazy danych?)

- Spróbuj wprowadzić kolejną osobę podając jej imię a nie podając nazwiska:

```
mysql> INSERT INTO Person (ID, FirstName) VALUES (3, 'Adam');
```

(Jakie nazwisko zostało wpisane do bazy danych?)

- Spróbuj wprowadzić kolejną osobę podając jej nazwisko i numer `ID` identyczny z już istniejącym:

```
mysql> INSERT INTO Person (ID, Surname) VALUES (2, 'Lis');
```

(Która własność tabeli Person spowodowała że wystąpił błąd?)

- Jeśli chcesz zmodyfikować pewne dane możesz użyć komendy UPDATE, np.:

```
mysql> UPDATE Person SET FirstName='Adam' WHERE ID=2;
```

- Dodaj dwa telefony dla użytkownika od ID = 2.

4. Pobieranie informacji.

- Wyświetl wszystkie numery telefonów Kowalskiego.
- Wyświetl wszystkie numery telefonów zaczynające się od numeru 022.
- Jak w powyższym podpunkcie, ale wyświetl także właścicieli tych numerów.

5. Skrypty. Wpisywanie powtarzających się komend jest zazwyczaj męczące i zniechęcające. Można sobie ułatwić życie wpisując komendy *MySQL'a* do pliku. Utwórz plik o przykładowej nazwie *query.sql*. Umieść w tym pliku dwa zapytania:

```
SELECT * FROM Person;
```

```
SELECT * FROM Phone;
```

Plik możesz utworzyć za pomocą edytora *nano* na serwerze, lub przy pomocy dowolnego edytora na komputerze lokalnym. Ostatecznie plik powinien zostać umieszczony w katalogu, z którego logowałeś się do bazy danych. Teraz z poziomu *MySQL'a* wykonaj komendę:

```
mysql> SOURCE query.sql
```

6. Umieść w skrypcie query.sql instrukcję tworzącą tabelę Person (`CREATE ...`) i instrukcje wprowadzające do niej dane (`INSERT ...`). Ponieważ tabela Person już istnieje, przed wywołaniem instrukcji `CREATE` należy tą tabelę usunąć. Na początku skryptu wprowadź zatem następujący warunek:

```
DROP TABLE IF EXISTS Person;
```

Powyższa instrukcja jest charakterystyczna dla *MySQL'a* i może nie zadziałać w innych wersjach *SQL'a*.

7. Wykonaj polecenia z poprzedniego punktu w odniesieniu do tabeli *Phone*. (Dodaj nowe instrukcje do pliku *query.sql*).

8. Export. Możemy się niekiedy spotkać z potrzebą zapisania danych z tabel w formacie dogodnym dla innych aplikacji niż MySQL (np. format *.csv dla *Excela*). Napisz instrukcję eksportującą wszystkie dane z tabeli Person do pliku *res.txt*:

```
mysql> SELECT * INTO OUTFILE 'res.txt' FROM Person;
```

Jak widać po liście pól (tutaj *) należy użyć instrukcji `INTO OUTFILE` podając nazwę pliku docelowego. Plik zostanie utworzony, w katalogu z którego nastąpiło logowanie do bazy danych.

9. Export danych do formatu *.csv wymaga by pola w wierszu oddzielone były przecinkami:

```
mysql> SELECT * INTO OUTFILE 'res.txt' FIELDS TERMINATED BY ','
-> FROM Person;
```

10. Eksportuj dane zawierające następujący zestaw danych: imię, nazwisko i numer telefonu.

11. Usuwanie danych. Skoro posiadasz już wygodne narzędzie do odtwarzania tabel (skrypt *query.sql*) można przystąpić do testowania usuwania danych. Na początek usuń dane Kowalskiego z tabeli *Person*:

```
mysql> DELETE FROM Person WHERE Surname = 'Kowalski';
```

Odtwórz tabelę *Person* i usuń z tabeli *Phone* wszystkie telefony Kowalskiego (musisz połączyć instrukcję `DELETE` z instrukcją `SELECT` w celu pozyskania identyfikatora Kowalskiego).

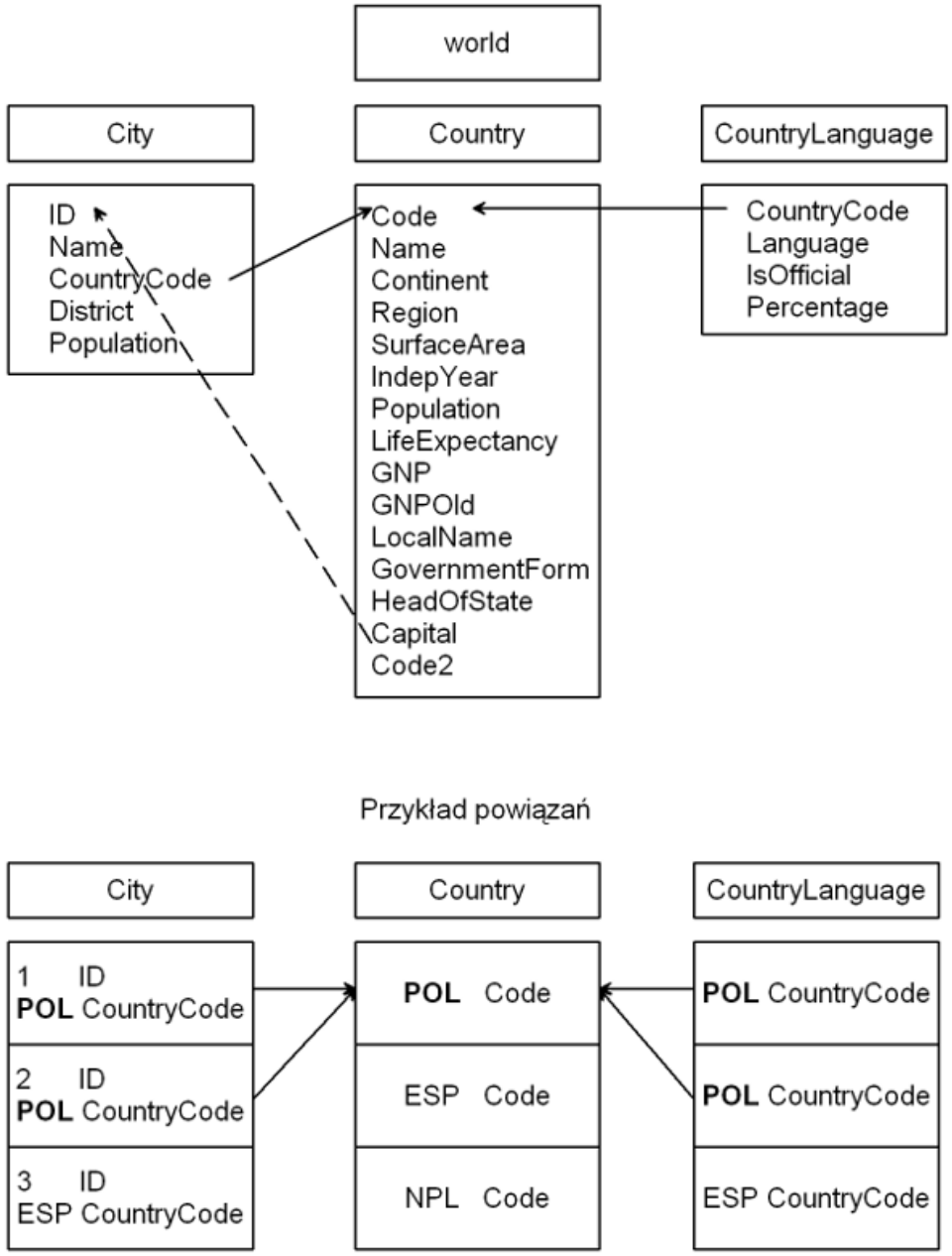

Figure 1:

Schemat tabel zawartych w bazie danych *test*

Baza danych test zawiera trzy tabele: `City`, `Country` i `CountryLanguage`. Powyższy schemat przedstawia powiązania jakie występują pomiędzy tymi tabelami. Pola: `ID` w tabeli `City` i `Code` w tabeli `Country` są unikatowe. To znaczy, że każdy rekord, np. w tabeli `City`, musi mieć inną wartość pola `ID`. Pole `CountryCode` w tabeli `City` przechowuje wartość pola `Code` z tabeli `Country`. W ten sposób można zidentyfikować, w którym państwie leży dane miasto. Podobna sytuacja występuje w powiązaniu tabeli `CountryLanguage` i `Country`. Tabela `CountryLanguage` zwiera dane o językach używanych we wszystkich państwach. Każdy rekord tej tabeli określa np. procentowy udział języka w danym państwie. Zatem, powiedzmy, język polski wystąpi w kilku rekordach tej tabeli, bo jest używany w kilku państwach.

Z powyższego wynika, że w przypadku obydwu powiązań, mamy do czynienia z relacją jeden-do-wielu. W przypadku tabel `City` i `Country`: każde miasto może wystąpić tylko w jednym państwie, ale każde państwo może posiadać wiele miast. W przypadku tabel `Country` i `CountryLanguage` jest to może mniej oczywiste: każdy rekord z tabeli `CountryLanguage` określający język w danym państwie może przynależeć tylko do jednego państwa. (Gdyby rekord ten określał język "w ogóle", to oczywiście mógłby być powiązany z wieloma rekordami z tabeli `Country`. Jednak wtedy nie można by w nim przechowywać danych charakterystycznych dla danego państwa, jak: czy jest to język oficjalny i jaki procent ludności nim włada.) Patrząc w drugą stronę: w każdym państwie może mieszkać wiele narodowości.

Powiązania te ilustruje przykład przedstawiony na rysunku. Z pierwszym rekordem z tabeli `Country` (`POL Code`) powiązane są dwa miasta z tabeli `City` (`1 ID`, `2 ID`) i dwa języki z tabeli `CountryLanguage`.

Pola: `ID` (`City`), `Code` (`Country`), `CountryCode` (`City`, `CountryLanguage`) muszą zawsze być wypełnione, to znaczy, że możemy mieć pewność, że odpowiednie powiązania będą istnieć. Pomiędzy tabelami `Country` i `City` istnieje jeszcze jedno powiązanie oznaczone linią przerywaną: `Capital - ID`. Każde państwo może posiadać stolicę. Kod miasta będącego stolicą przechowywany jest w polu `Capital`. Pole to może mieć wartość `NULL`, ponieważ są pewne obszary globu (zazwyczaj jednak zależne od pewnych państw) nie posiadające wyraźnych struktur państwowych. Wystarczy sprawdzić jakie to terytoria:

```
mysql> SELECT Name FROM Country WHERE Capital IS NULL;
```

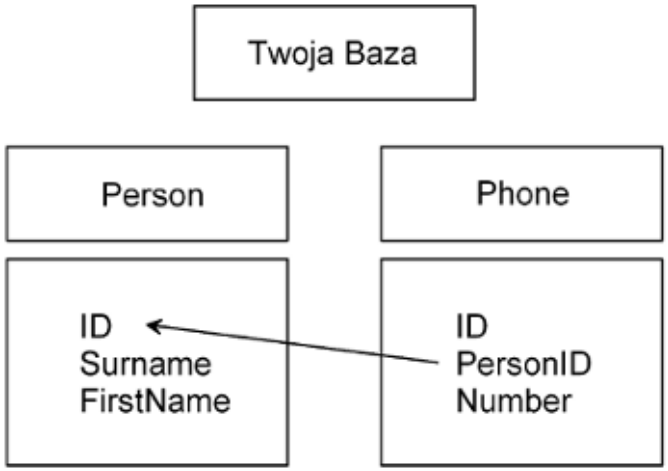

Figure 2:

Schemat tabel `Person` i `Phone`

Ściągawka

Komenda	Rezultat
USE *baza_danych*	wybór bazy danych
SHOW DATABASES	wyświetla wszystkie bazy danych
SHOW TABLES FROM *baza_danych*	wyświetla tabele danej bazy danych
SHOW TABLES	wyświetla tabele bieżącej bazy danych
DESCRIBE *tabela*	wyświetla strukturę tabeli
SELECT * FROM *tabela*	wyświetla wszystkie kolumny tabeli
SELECT *kolumna1*, *kolumna2* **FROM** *tabela*	wyświetla podane kolumny tabeli
SELECT *kolumna* **AS** *naglowek* **FROM** *tabela*	wyświetla kolumnę tabeli, przy czym jej standardowy nagłówek zostanie zastąpiony słowem (aliasem): naglowek; jeśli przypisywany alias jest wieloczłonowy należy wziąć go w podwójny cudzysłów: "nowy naglowek"
SELECT *kolumna* **FROM** *tabela* **WHERE** *warunek*	wyświetla te wiersze danej kolumny tabeli, które spełniają określony warunek

Komenda	Rezultat
SELECT *kolumna1* **FROM** *tabela* **WHERE** *kolumna2* **IS NOT NULL**	wyświetla te wiersze danej kolumny1, w których wartości kolumny2 są niepuste; puste wyszukuje instrukcja **IS NULL**
SELECT *kolumna1* **FROM** *tabela* **WHERE** *warunek1* **AND** *warunek2* **OR** *warunek3*	wyświetla te wiersze danej kolumny tabe które spełniają określony złożony warun klauzula **WHERE** może zawierać opera logiczne: **AND**, **OR**, **NOT**
SELECT *kolumna1*, *kolumna2* **FROM** *tabela* **ORDER BY** *kolumna2*	wyświetla podane kolumny tabeli w kolejności elementów kolumny2
SELECT *kolumna1*, *kolumna2* **FROM** *tabela* **ORDER BY** *kolumna2* **DESC**	wyświetla podane kolumny tabeli w kolejności odwrotnej elementów kolumny
LOWER(*tekst*)	funkcja zamienia tekst na małe litery
UPPER(*tekst*)	funkcja zamienia tekst na wielkie litery
TRIM(*tekst*)	funkcja obcina spacje początkowe i końc tekstu
SUM(*kolumna*)	funkcja wylicza sumę wartości z grupy wartości
AVG(*kolumna*)	funkcja wylicza średnią wartość z grupy wartości
MAX(*kolumna*)	funkcja znajduje maksymalną wartość z grupy wartości
MIN(*kolumna*)	funkcja znajduje minimalną wartość z gr wartości
SELECT DISTINCT *kolumna* **FROM** *tabela*	wyświetla wiersze danej kolumny którycl wartości nie powtarzają się
SELECT COUNT(*) **FROM** *tabela*	zlicza wiersze w tabeli, oprócz wierszy pustych; zwraca pojedynczy wynik
SELECT COUNT (*kolumna*) **FROM** *tabela*	zlicza wiersze podanej kolumny tabeli, oprócz wierszy pustych; zwraca pojedync wynik
SELECT COUNT (**DISTINCT** *kolumna*) **FROM** *tabela*	zlicza nie powtarzające się wiersze podar kolumny tabeli, oprócz wierszy pustych; zwraca pojedynczy wynik
SELECT *kolumna* **FROM** *tabela* **GROUP BY** *kolumna*	wyświetla pogrupowane wiersze kolumny tabeli; działanie podobne do instrukcji **DISTINCT** – otrzymamy tyle samo wierszy co w tej instrukcji; kolumna użyt klauzuli **GROUP BY** musi wystąpić w kolumn klauzuli **SELECT**

Komenda	Rezultat
SELECT *kolumna1*, **SUM** (*kolumna2*) **FROM** *tabela* **GROUP BY** *kolumna1*	wyświetla pogrupowane wiersze z *kolumna1* i sumę wartości wierszy z *kolumna2* liczoną oddzielnie dla każdej grupy *kolumna1*
SELECT *kolumna1*, **SUM**(*kolumna2*) **FROM** *tabela* **WHERE** *warunek1* **GROUP BY** *kolumna1* **HAVING** *warunek2* **ORDER BY** *kolumna1*	wyświetla pogrupowane wiersze *kolumna1* i sumę wartości wierszy *kolumna2* liczoną oddzielnie dla każdej grupy *kolumna1*, przy czym suma liczona jest tylko po wierszach spełniających dany *warunek1*; instrukcja **HAVING** określa *warunek2* wyświetlenia całej grupy; grupy są posortowane według *kolumna1*; kolumny użyte w instrukcji **HAVING** muszą wystąpić w instrukcji **SELECT**
DROP TABLE *tabela*	usuwa tabelę
CREATE TABLE *tabela* (definicje kolumn)	tworzy tabelę
ALTER TABLE *tabela* **ADD** definicja kolumny	dodaje do istniejącej tabeli kolumnę
ALTER TABLE *tabela* **DROP COLUMN** *kolumna*	usuwa z istniejącej tabeli kolumnę
INSERT INTO *tabela* (*kolumna1*, *kolumna2*) **VALUES** (*wartość1*, *'wartość2'*)	dodaje do tabeli rekord wstawiając odpowiednie wartości do odpowiednich kolumn. Wartości tekstowe powinny być ujęte w apostrofy
INSERT INTO *tabela* **VALUES** (wartości *kolumn*)	dodaje do tabeli rekord, w liście wartości należy wymienić wartości dla wszystkich kolumn
UPDATE *tabela* **SET** *kolumna* = *wartość*	zmienia wartość danej kolumny we wszystkich rekordach tabeli
UPDATE *tabela* **SET** *kolumna* = *wartość* **WHERE** *warunek*	zmienia wartość danej kolumny w rekordach spełniających dany warunek
DELETE FROM *tabela*	usuwa wszystkie rekordy danej tabeli
DELETE FROM *tabela* **WHERE** *warunek*	usuwa rekordy spełniające dany warunek