Technologia Informacyjna

LABORATORIUM

ĆWICZENIE 1

Zapoznanie ze środowiskiem MySQL Workbench

oraz projekt prostej bazy danych.

Opracował Dawid Budnarowski

CZĘŚĆ TEORETYCZNA

Definicja relacyjnej bazy danych.

System zarządzania relacyjną bazą danych (RDBMS) jest dominującym modelem na rynku. Wiele usług sieciowych i aplikacji biznesowych korzysta z tego typu baz w celu ustrukturyzowania danych i wydobycia z nich istotnych informacji. Niektóre duże firmy z branży chmurowej opracowały własne modele (np. Db2, który należy do IBM, Oracle Database i Microsoft SQL Server). Wiele systemów RDBMS to projekty open source'owe:

- **MySQL** to najpopularniejszy system na świecie. W 2009 r. został przejęty przez Oracle, pierwotnie był jednak projektem darmowym i otwartym. W reakcji na to przejęcie deweloperzy stworzyli jego następcę, również o otwartym kodzie źródłowym: MariaDB
- **PostgreSQL** to również rozwiązanie typu open source. Rozwijany jest przez zespół niezależnych deweloperów. Jego działanie różni się nieznacznie od działania MySQL, ponieważ jest to model obiektowo-relacyjny (RDBMS)
- **SQLite** to biblioteka publiczna, jej cechą szczególną jest to, że jest zainstalowana jest bezpośrednio w programie, a nie w trybie klient-serwer.

Skrót SQL pochodzi od nazwy: Structured Query Language. Jest to standardowy interfejs dla relacyjnej bazy danych. Jego instrukcje są wykorzystywane do strukturyzacji danych i organizacji informacji, które chcemy z nich wydobyć.

Architektura relacyjnej bazy danych

Organizacja baz danych opiera się na teorii Edgara F. Codda z 1971 r., który wprowadził główne założenia dotyczące modelu relacyjnego. Zgodnie z tą teorią każde „naturalne" zapytanie można przetłumaczyć na algebrę relacyjną, a tym samym na język zapytań czytelny dla komputera (za pomocą SQL). W tym modelu relacja (lub tabela) składa się z kilku atrybutów uporządkowanych w kilku wierszach i kolumnach, zwanych *krotkami*. Cała tabela jest postrzegana jako zbiór *krotek*.

Aby administrować bazą danych, użytkownicy komunikują się z interfejsem zarządzania za pomocą języka SQL, który jest oparty na algebrze relacyjnej. Wszystkie polecenia są wyrażone w języku SQL i szczegółowo opisują każdy etap tworzenia bazy danych. Język ten umożliwia wybór pożądanych informacji, wskazywanie ich lokalizacji w bazie danych, interakcje między nimi, etc.

W ramach laboratorium z przedmiotu 'Technologia Informacyjna' będziemy poznawać bazy MySQL. Są one dystrybuowane obecnie na dwóch licencjach:

- MySQL Enterprise (płatna z większą listą rozszerzeń)
- MySQL Community Edition (darmowa).

Ranking popularności baz danych:
https://db-engines.com/en/ranking

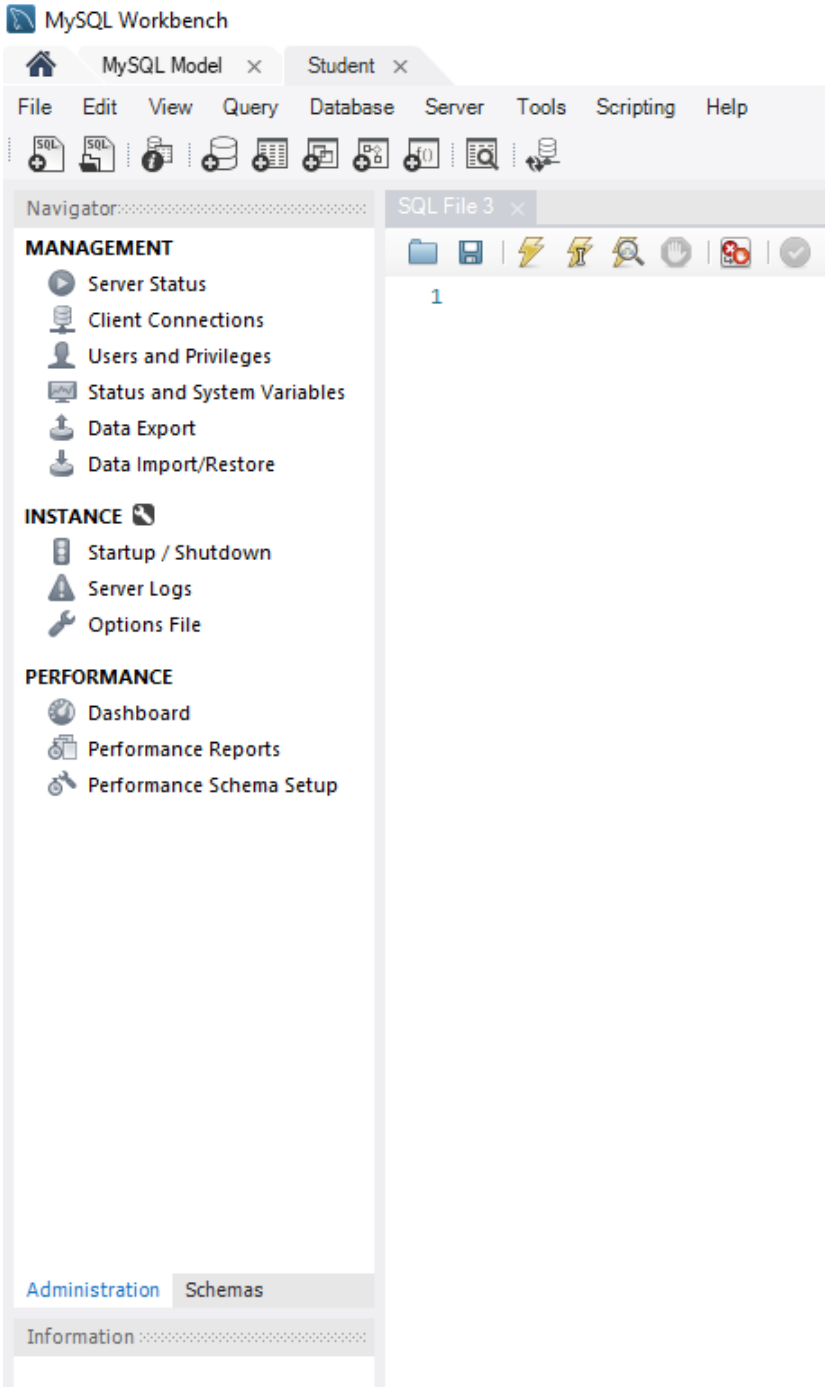
Oprogramowanie MySQL Workbench umożliwia zarządzanie i modelowanie bazą danych MySQL. Za jego pomocą można edytować konfigurację serwera i jego komponentów, a także zaprojektować i stworzyć schematy (wizualne reprezentacje tabel, widoków itp.) nowych baz danych, wykonać dokumentację istniejących oraz zapewnić wsparcie przy procesach migracji do MySQL.

CZĘŚĆ PRAKTYCZNA

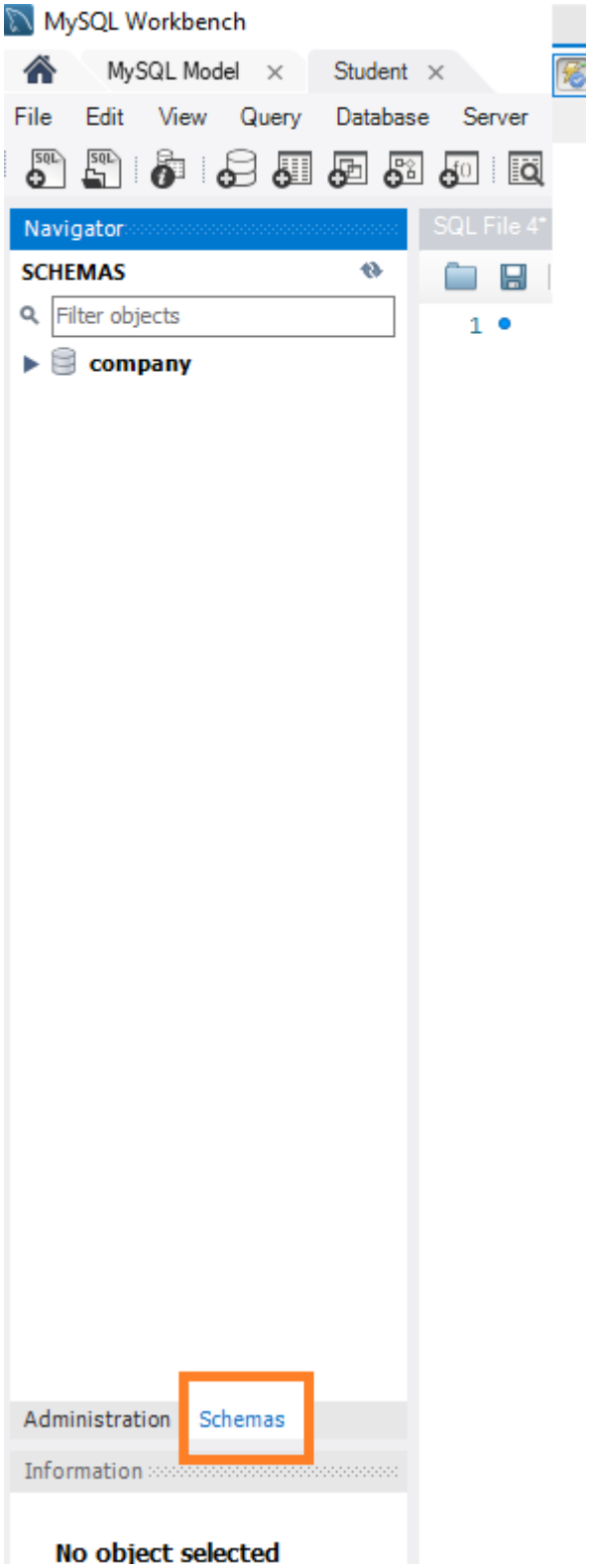
Cel ćwiczenia

- Wykonanie projektu prostej bazy danych przy pomocy dołączonego kodu SQL.
- Napisanie kilku zapytań SQL realizujących zadane w ostatnim punkcie zagadnienia.

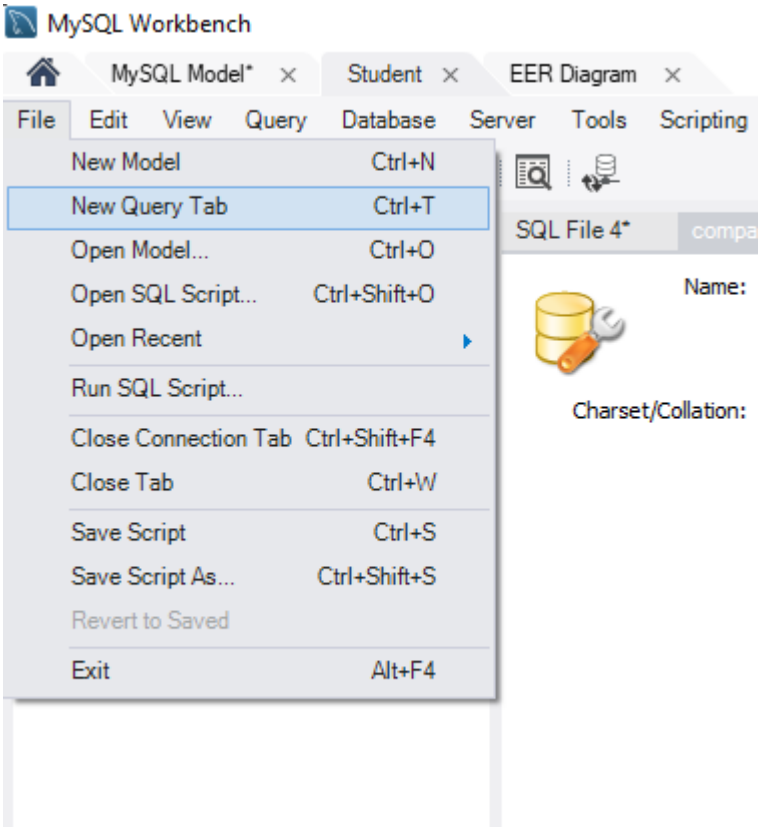
1. Zapoznaj się z zakładką ‘Administration (Managment, Instance, Performance)’.



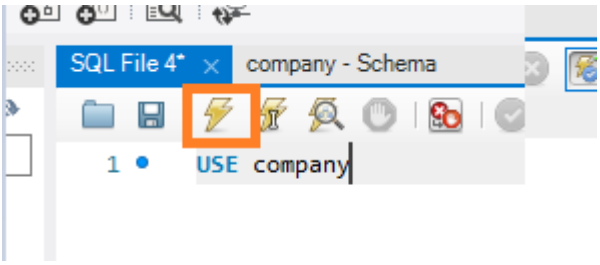
2. Utwórz schemat (‘Schemas’) o nazwie company (prawy przycisk myszki -> ‘Create Schema..’). W następnych krokach wygenerujesz z przygotowanego kodu SQL dwie tabele o nazwie ‘employees’ oraz ‘departments’.



3. Wybierz File -> New Query Tab



4. W nowo otwarty dokumencie wpisujemy ‘USE company’ i klikamy żółty piorun (skrót klawiszowy Ctrl + Shift + Enter)



W tym momencie zdefiniowaliśmy na którym schemacie będziemy pracować.

5. Tworzymy tabelę z oddziałami poprzez komendę:

```
CREATE TABLE departments
    ( department_id INTEGER PRIMARY KEY
    , department_name VARCHAR(30)
    , location_id INTEGER
    ) ;
```

Wykonujemy kod.

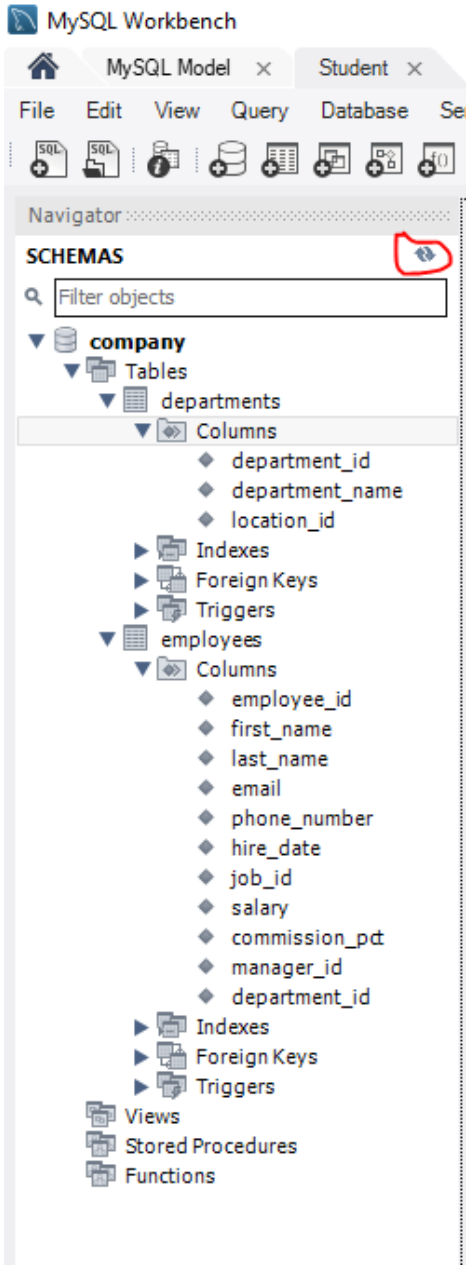
6. Tworzymy tabelę z pracownikami poprzez komendę:

```
CREATE TABLE employees
```

```
( employee_id INTEGER
, first_name VARCHAR(20)
, last_name VARCHAR(25)
, email VARCHAR(25)
, phone_number VARCHAR(20)
, hire_date DATE
, job_id VARCHAR(10)
, salary INTEGER
, commission_pct INTEGER
, manager_id INTEGER
, department_id INTEGER
, constraint pk_emp primary key (employee_id)
, constraint fk_deptno foreign key (department_id) references departments(department_id)
) ;
```

Przeanalizuj powyższy kod. Zwróć uwagę na różne typy danych przy polach w tabelach.

7. Odświeżamy widok ‘Schemas’

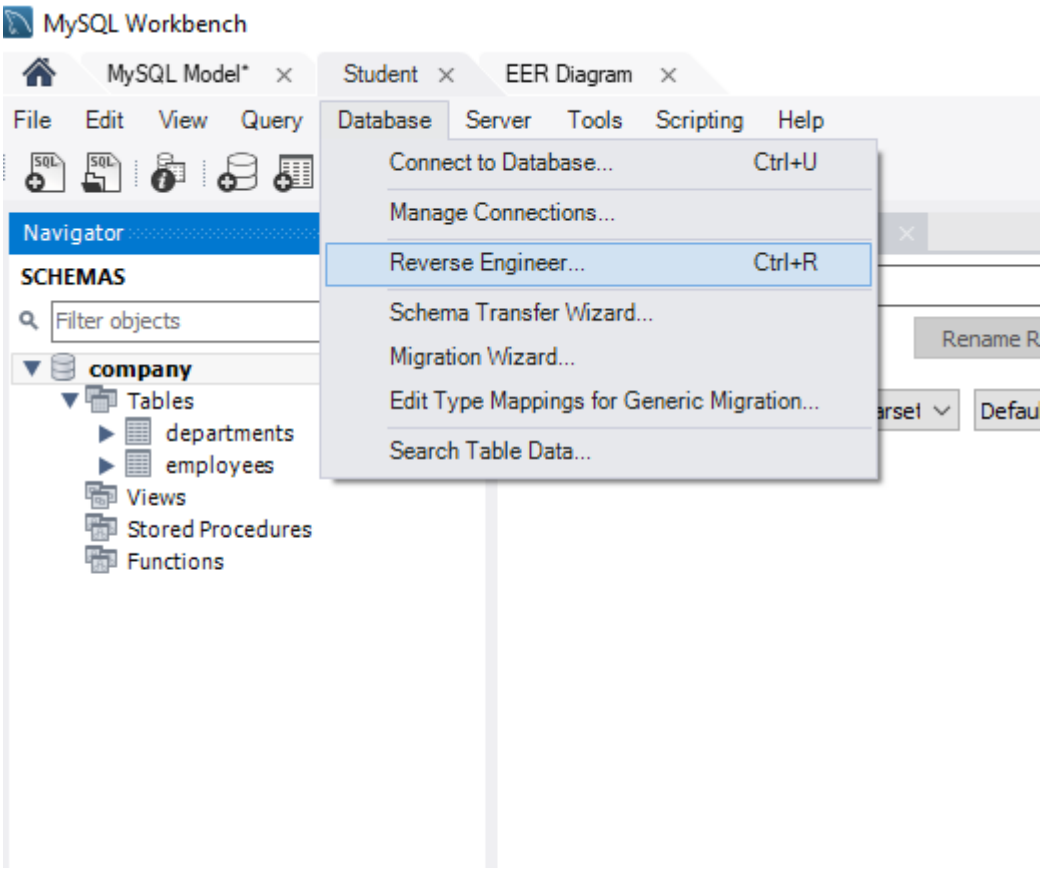


8. Utworzone tabele są puste. Dodaj rekordy do tabel poprzez komendę:

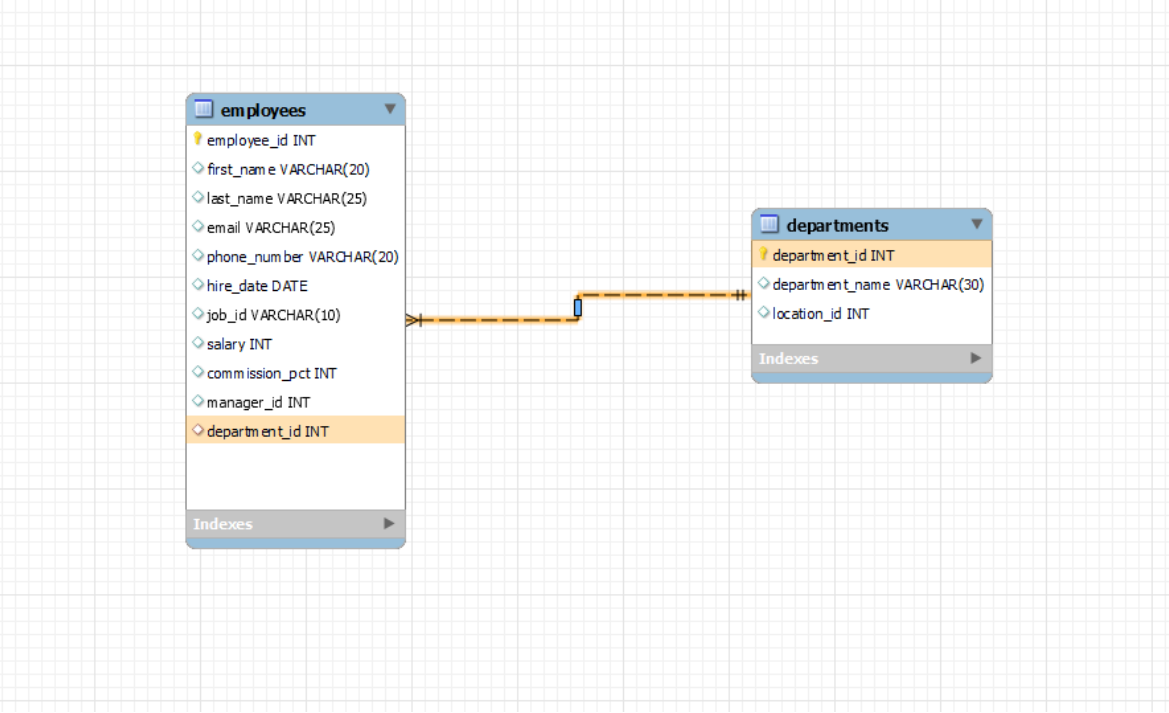
```
## Insert into Departments table
 INSERT INTO departments VALUES ( 20,'Marketing',  180);
 INSERT INTO departments VALUES ( 30,'Purchasing',  1700);
 INSERT INTO departments VALUES ( 40, 'Human Resources',  2400);
 INSERT INTO departments VALUES ( 50, 'Shipping',  1500);
 INSERT INTO departments VALUES ( 60 , 'IT',  1400);
 INSERT INTO departments VALUES ( 70, 'Public Relations',  2700);
 INSERT INTO departments VALUES ( 80 , 'Sales',  2500 );
 INSERT INTO departments VALUES ( 90 , 'Executive',  1700);
 INSERT INTO departments VALUES ( 100 , 'Finance',  1700);
 INSERT INTO departments VALUES ( 110 , 'Accounting',  1700);
 INSERT INTO departments VALUES ( 120 , 'Treasury' ,  1700);
 INSERT INTO departments VALUES ( 130 , 'Corporate Tax' ,  1700 );
 INSERT INTO departments VALUES ( 140, 'Control And Credit' ,  1700);
 INSERT INTO departments VALUES ( 150 , 'Shareholder Services', 1700);
 INSERT INTO departments VALUES ( 160 , 'Benefits', 1700);
 INSERT INTO departments VALUES ( 170 , 'Payroll' , 1700);


## Insert into Employees table
INSERT INTO employees VALUES (100, 'Steven', 'King', 'SKING', '515.123.4567', '1987-06-17' , 'AD_PRES', 24000 , NULL, NULL, 20);
INSERT INTO employees VALUES (101, 'Neena' , 'Kochhar' , 'NKOCHHAR' , '515.123.4568' , '1989-11-21' , 'AD_VP' , 17000 , NULL , 100 , 20);
INSERT INTO employees VALUES (102 , 'Lex' , 'De Haan' , 'LDEHAAN' , '515.123.4569' , '1993-09-12' , 'AD_VP' , 17000 , NULL , 100 , 30);
INSERT INTO employees VALUES (103 , 'Alexander' , 'Hunold' , 'AHUNOLD' , '590.423.4567' , '1990-09-30', 'IT_PROG' , 9000 , NULL , 102 , 60);
INSERT INTO employees VALUES (104 , 'Bruce' , 'Ernst' , 'BERNST' , '590.423.4568' , '1991-05-21',  'IT_PROG' , 6000 , NULL , 103 , 60);
INSERT INTO employees VALUES (105 , 'David' , 'Austin' , 'DAUSTIN' , '590.423.4569' , '1997-06-25',  'IT_PROG' , 4800 , NULL , 103 , 60);
INSERT INTO employees VALUES (106 , 'Valli' , 'Pataballa' , 'VPATABAL' , '590.423.4560' , '1998-02-05',  'IT_PROG' , 4800 , NULL , 103 , 40);
INSERT INTO employees VALUES (107 , 'Diana' , 'Lorentz' , 'DLORENTZ' , '590.423.5567' , '1999-02-09',  'IT_PROG' , 4200 , NULL , 103 , 40);
INSERT INTO employees VALUES (108 , 'Nancy' , 'Greenberg' , 'NGREENBE' , '515.124.4569' , '1994-08-17',  'FI_MGR' , 12000 , NULL , 101 , 100);
INSERT INTO employees VALUES (109 , 'Daniel' , 'Faviet' , 'DFAVIET' , '515.124.4169' , '1994-08-12',  'FI_ACCOUNT' , 9000 , NULL , 108 , 170);
INSERT INTO employees VALUES (110 , 'John' , 'Chen' , 'JCHEN' , '515.124.4269' , '1997-04-09',  'FI_ACCOUNT' , 8200 , NULL , 108 , 170);
INSERT INTO employees VALUES (111 , 'Ismael' , 'Sciarra' , 'ISCIARRA' , '515.124.4369' , '1997-02-01',  'FI_ACCOUNT' , 7700 , NULL , 108 , 160);
```

```
INSERT INTO employees VALUES (112 , 'Jose Manuel' , 'Urman' , 'JMURMAN' , '515.124.4469' , '1998-06-03',
'FI_ACCOUNT' , 7800 , NULL , 108 , 150);
INSERT INTO employees VALUES (113 , 'Luis' , 'Popp' , 'LPOPP' , '515.124.4567' , '1999-12-07',  'FI_ACCOUNT' , 6900 ,
NULL , 108 , 140);
INSERT INTO employees VALUES (114 , 'Den' , 'Raphaely' , 'DRAPHEAL' , '515.127.4561' , '1994-11-08',  'PU_MAN' ,
11000 , NULL , 100 , 30);
INSERT INTO employees VALUES (115 , 'Alexander' , 'Khoo' , 'AKHOO' , '515.127.4562' , '1995-05-12',  'PU_CLERK' ,
3100 , NULL , 114 , 80);
INSERT INTO employees VALUES (116 , 'Shelli' , 'Baida' , 'SBAIDA' , '515.127.4563' ,'1997-12-13', 'PU_CLERK' , 2900 ,
NULL , 114 , 70);
INSERT INTO employees VALUES (117 , 'Sigal' , 'Tobias' , 'STOBIAS' , '515.127.4564' , '1997-09-10', 'PU_CLERK' , 2800 ,
NULL , 114 , 30);
INSERT INTO employees VALUES (118 , 'Guy' , 'Himuro' , 'GHIMURO' , '515.127.4565' , '1998-01-02',  'PU_CLERK' , 2600
, NULL , 114 , 60);
INSERT INTO employees VALUES (119 , 'Karen' , 'Colmenares' , 'KCOLMENA' , '515.127.4566' , '1999-04-08',
'PU_CLERK' , 2500 , NULL , 114 , 130);
INSERT INTO employees VALUES (120 , 'Matthew' , 'Weiss' , 'MWEISS' , '650.123.1234' ,'1996-07-18',  'ST_MAN' , 8000
, NULL , 100 , 50);
INSERT INTO employees VALUES (121 , 'Adam' , 'Fripp' , 'AFRIPP' , '650.123.2234' , '1997-08-09',  'ST_MAN' , 8200 ,
NULL , 100 , 50);
INSERT INTO employees VALUES (122 , 'Payam' , 'Kaufling' , 'PKAUFLIN' , '650.123.3234' ,'1995-05-01',  'ST_MAN' ,
7900 , NULL , 100 , 40);
INSERT INTO employees VALUES (123 , 'Shanta' , 'Vollman' , 'SVOLLMAN' , '650.123.4234' , '1997-10-12',  'ST_MAN' ,
6500 , NULL , 100 , 50);
INSERT INTO employees VALUES (124, 'Kevin' , 'Mourgos' , 'KMOURGOS' , '650.123.5234' , '1999-11-12',  'ST_MAN' ,
5800 , NULL , 100 , 80);
INSERT INTO employees VALUES (125, 'Julia' , 'Nayer' , 'JNAYER' , '650.124.1214' , '1997-07-02',  'ST_CLERK' , 3200 ,
NULL , 120 , 50);
INSERT INTO employees VALUES (126, 'Irene' , 'Mikkilineni' , 'IMIKKILI' , '650.124.1224' , '1998-11-12', 'ST_CLERK' ,
2700 , NULL , 120 , 50);
INSERT INTO employees VALUES (127, 'James' , 'Landry' , 'JLANDRY' , '650.124.1334' , '1999-01-02' , 'ST_CLERK' , 2400 ,
NULL , 120 , 90);
INSERT INTO employees VALUES (128, 'Steven' , 'Markle' , 'SMARKLE' , '650.124.1434' , '2000-03-04' , 'ST_CLERK' , 2200
, NULL , 120 , 50);
INSERT INTO employees VALUES (129, 'Laura' , 'Bissot' , 'LBISSOT' , '650.124.5234' ,'1997-09-10' , 'ST_CLERK' , 3300 ,
NULL , 121 , 50);
INSERT INTO employees VALUES (130, 'Mozhe' , 'Atkinson' , 'MATKINSO' , '650.124.6234' , '1997-10-12' , 'ST_CLERK' ,
2800 , NULL , 121 , 110);
```

9. Utworzyliśmy dwie tabele wypełnione danymi. Sprawdźmy jak wygląda diagram EER naszej bazy. W tym celu wybierz ‘Database’ -> Reverse Engineer i przejdź przez kolejne ekrany.



10. Zastanów się z czego wynika połączenie między tabelami:



11. Wróć do zakładki z konsolą SQL. Wpisz kolejne komendy i przeanalizuj otrzymane odpowiedzi:

```
SELECT first_name,
    last_name,
    job_id,
    salary FROM employees


SELECT * FROM departments


SELECT employee_id,
    first_name,
    last_name,
    job_id,
    salary
 FROM employees
 WHERE salary > 5000;
```

Na podstawie powyższych przykładów spróbuj napisać samodzielnie komendy SQL które zwrócą:

1. Imię oraz nazwisko wszystkich pracowników zatrudnionych na stanowisku IT_PROG.
2. Wyświetl wszystkie nazwy departamentów.
3. Wyświetl imiona oraz nazwiska pracowników których numer telefonu zaczyna się od liczby 5. (podpowiedź -> LIKE '5%')
4. Wyświetl listę imion i nazwisk pracowników zatrudnionych w 1994. (podpowiedź -> YEAR(hire_date))