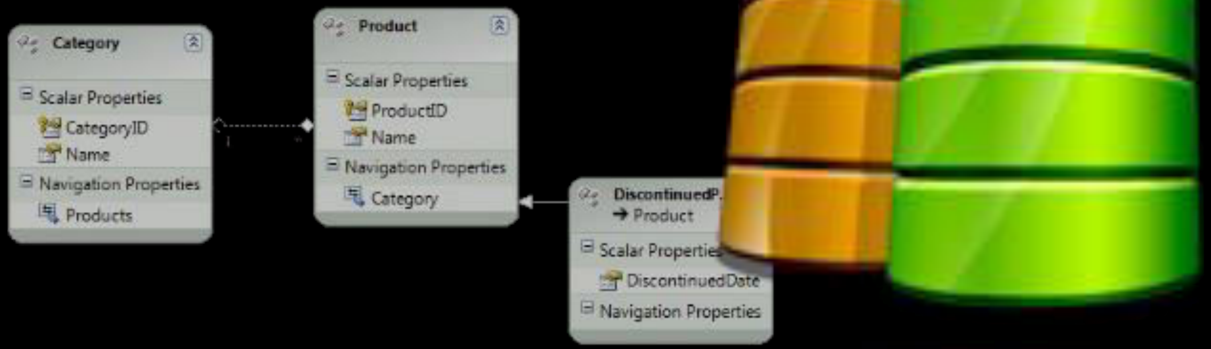
Table Relations

Database Design and Rules



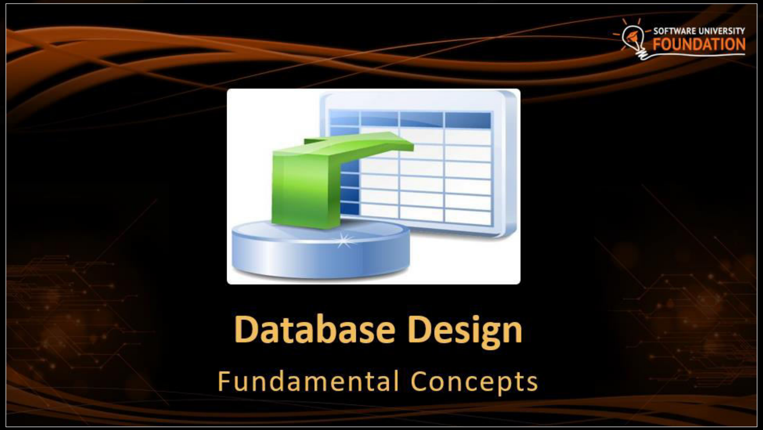
SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Content

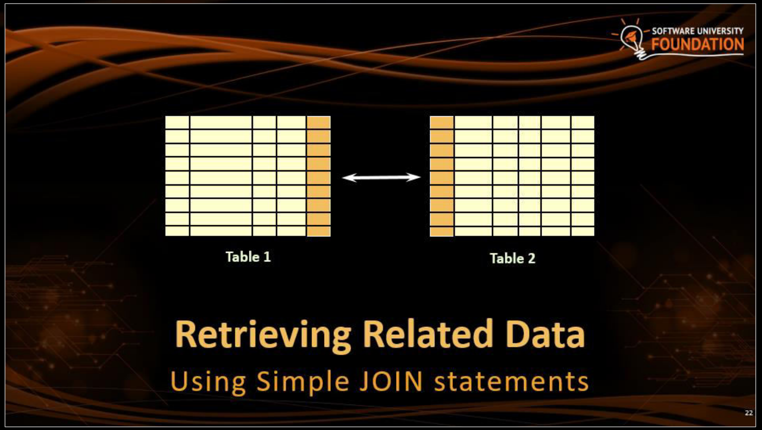


Database Design

Fundamental Concepts

Table Relations

Relational Database Model in Action

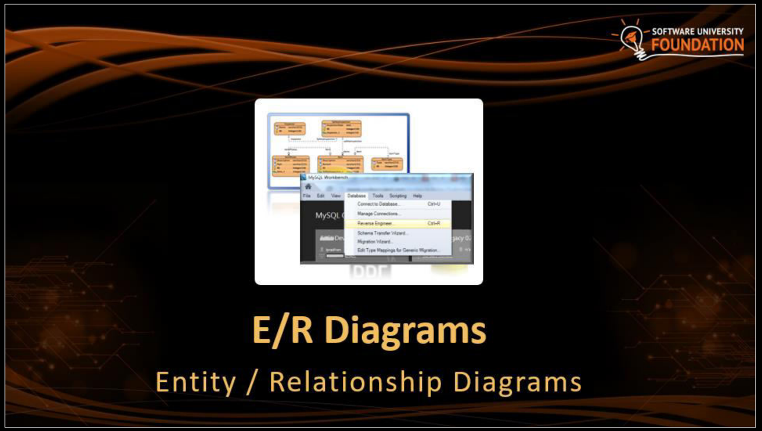


Retrieving Related Data

Using Simple JOIN statements

Cascade Operations

Cascade Delete/Update



E/R Diagrams

Entity / Relationship Diagrams

Questions

sli.do

#JavaDB

Database Design

Fundamental Concepts

Steps in Database Design

SOFTWARE UNIVERSITY FOUNDATION

1. Identification of the entities
2. Defining table columns
3. Defining primary keys
4. Modeling relationships
5. Defining constraints
6. Filling test data

Identification of Entities

- Entity tables represent objects from the real world
  - Most often they are nouns in the specification
  - For example:

```
We need to develop a system that stores
information about students, which are trained
in various courses. The courses are held in
different towns. When registering a new
student the following information is entered:
name, faculty number, photo and date.
```

  - Entities: **Student**, **Course**, **Town**

Identification of the Columns

- Columns are clarifications for the entities in the text of the specification, for example:

```
We need to develop a system that stores
information about students, which are trained
in various courses. The courses are held in
different towns. When registering a new
student the following information is entered:
name, faculty number, photo and date.
```

- Students have the following characteristics:
  - Name, faculty number, photo, date of enlistment and a list of courses they visit

How to Choose a Primary Key?

- Always define an additional column for the primary key
  - Don't use an existing column (for example SSN)
  - Must be an integer number
  - Must be declared as a **`PRIMARY KEY`**
  - Use **`auto_increment`** to implement auto-increment
  - Put the primary key as a first column
- Exceptions
  - Entities that have well known ID, e.g. countries (BG, DE, US) and currencies (USD, EUR, BGN)

Identification of Relationships

- Relationships are dependencies between the entities:

```
We need to develop a system that stores
information about students, which are trained in
various courses. The courses are held in
different towns. When registering a new student
the following information is entered: name,
faculty number, photo and date.
```

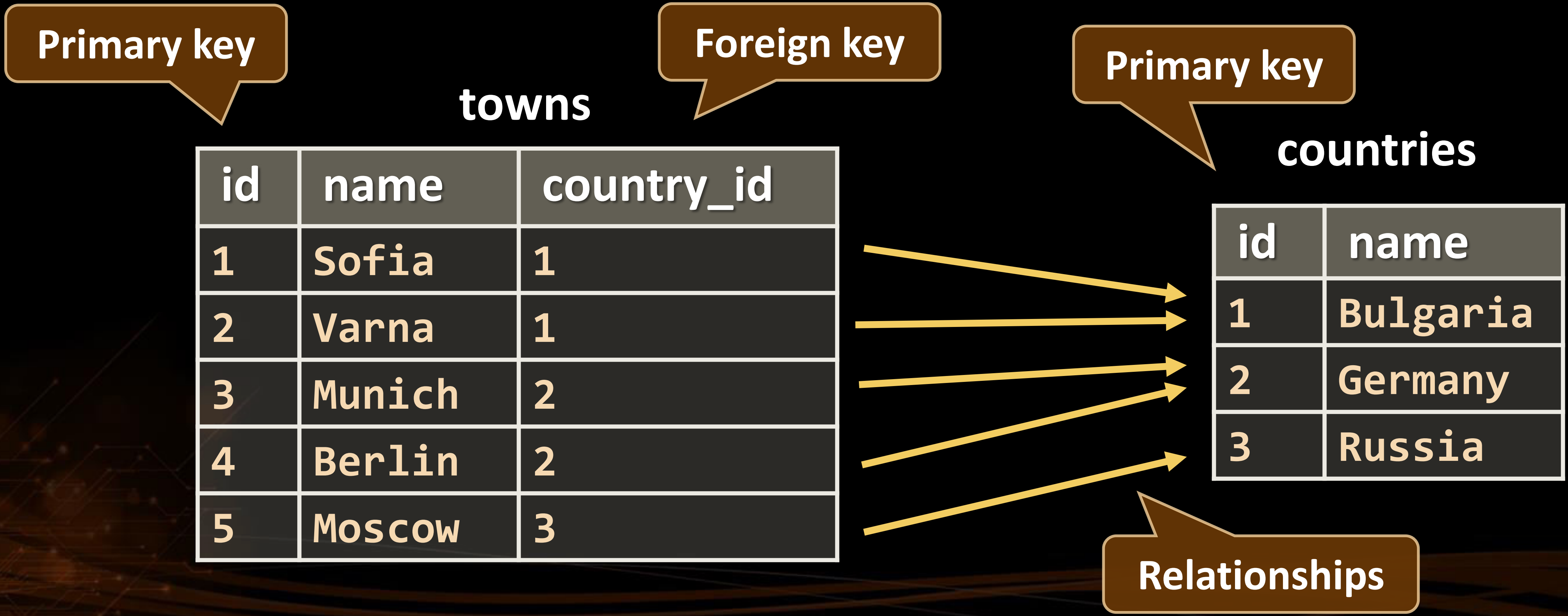
- "Students are trained in courses" – many-to-many relationship.
- "Courses are held in towns" – many-to-one (or many-to-many) relationship

Table Relations

Relational Database Model in Action

Relationships

- Relationships between tables are based on interconnections: **PRIMARY KEY / FOREIGN KEY**



towns

id	name	country_id
1	Sofia	1
2	Varna	1
3	Munich	2
4	Berlin	2
5	Moscow	3

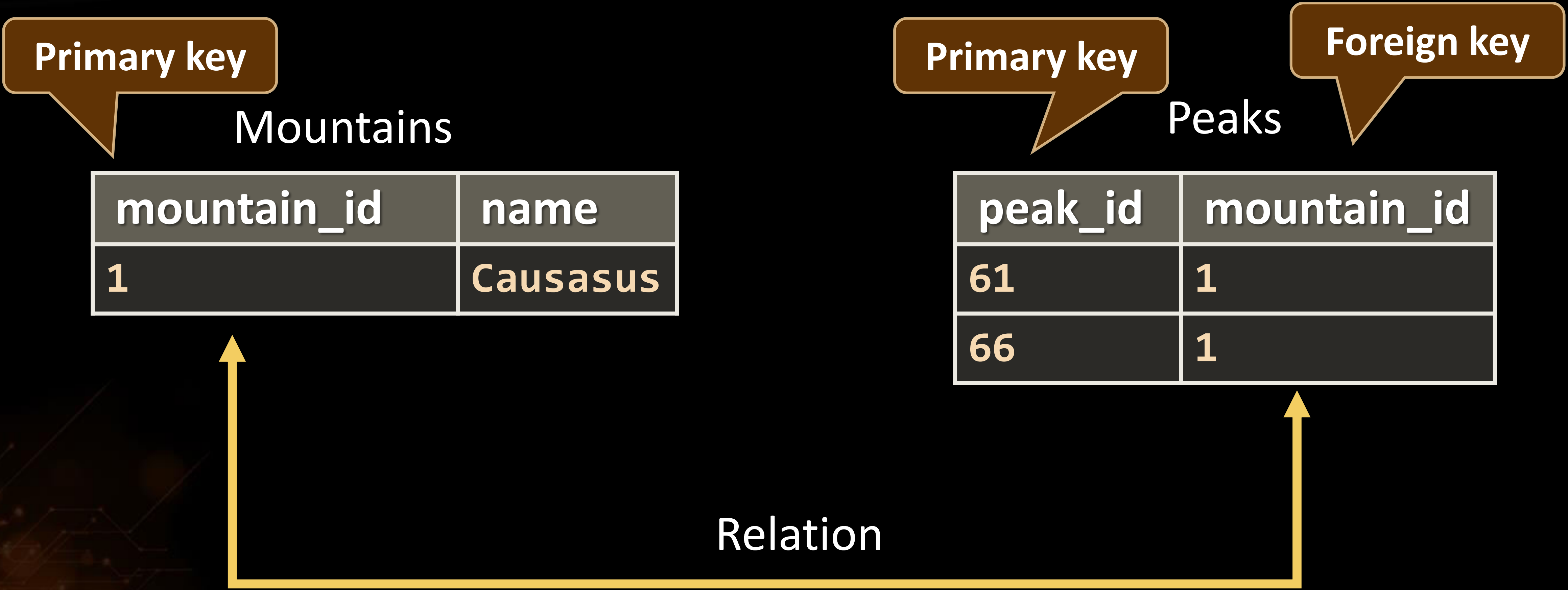
countries

id	name
1	Bulgaria
2	Germany
3	Russia

Relationships (2)

- The foreign key is an identifier of a record located in another table (usually its primary key)
- By using relationships we avoid repeating data in the database
- Relationships have multiplicity:
  - **One-to-many** – e.g. country / towns
  - **Many-to-many** – e.g. student / course
  - **One-to-one** – e.g. example driver / car

One-to-Many/Many-to-One



Mountains

mountain_id	name
1	Causasus

Peaks

peak_id	mountain_id
61	1
66	1

Setup

```
CREATE TABLE mountains(
  mountain_id INT PRIMARY KEY,
  mountain_name VARCHAR(50)
);
CREATE TABLE peaks(
  peak_id INT PRIMARY KEY,
  mountain_id INT,
  CONSTRAINT fk_peaks_mountains
  FOREIGN KEY (mountain_id)
  REFERENCES mountains(mountain_id)
);
```

Primary key

Table Peaks

Foreign Key

Foreign Key

Constraint Name

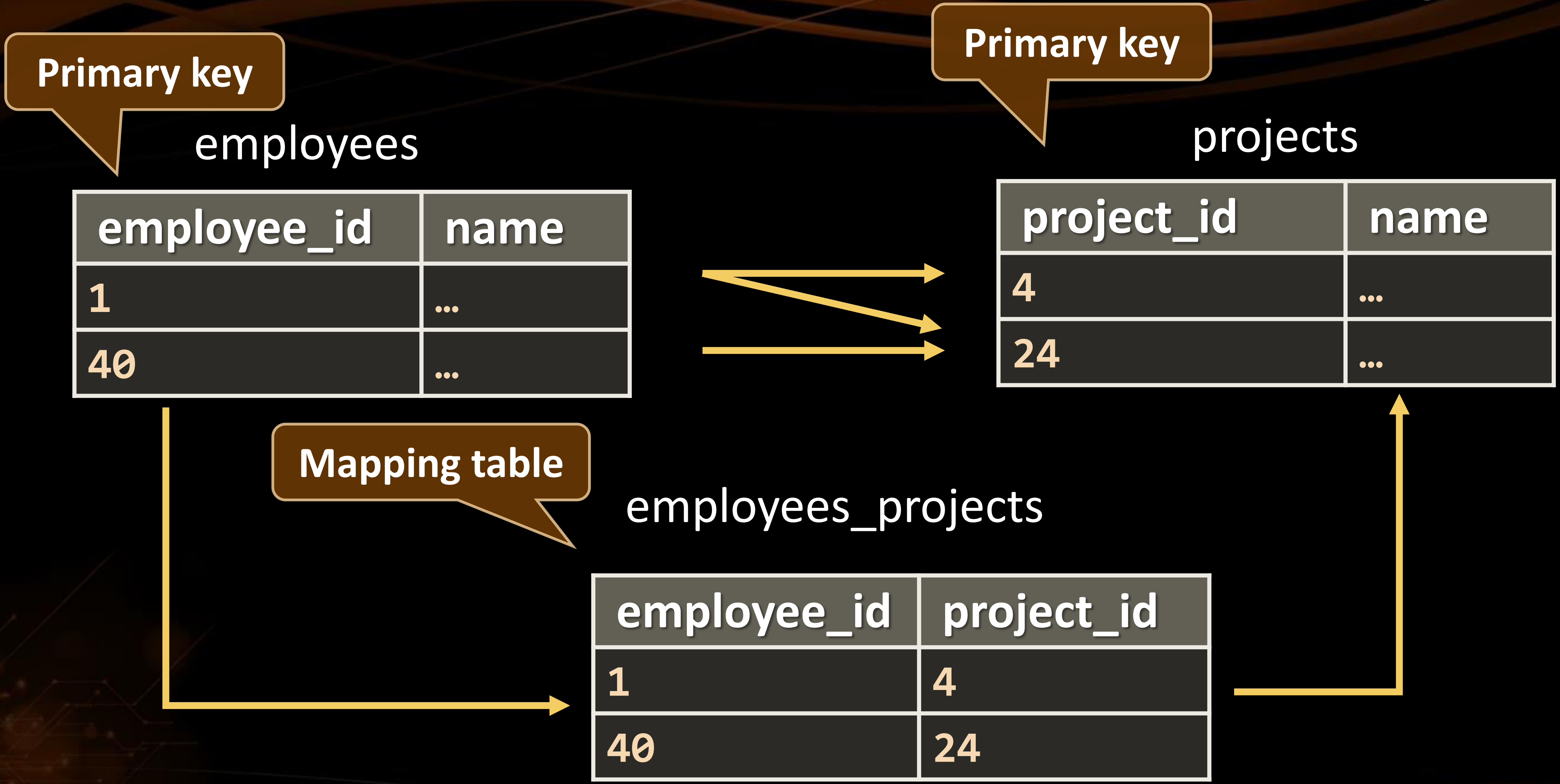
```
CONSTRAINT fk_peaks_mountains
FOREIGN KEY (mountain_id)
REFERENCES mountains(mountain_id);
```

Foreign Key

Referent Table

Primary Key

Many-to-Many



employees

employee_id	name
1	…
40	…

projects

project_id	name
4	…
24	…

employees_projects

employee_id	project_id
1	4
40	24

Setup(1)

```
CREATE TABLE employees(
  employee_id INT PRIMARY KEY,
  employee_name VARCHAR(50)
);
```

Table Employees

```
CREATE TABLE projects(
  project_id INT PRIMARY KEY,
  project_name VARCHAR(50)
);
```

Table Projects

Setup(2)

Mapping Table

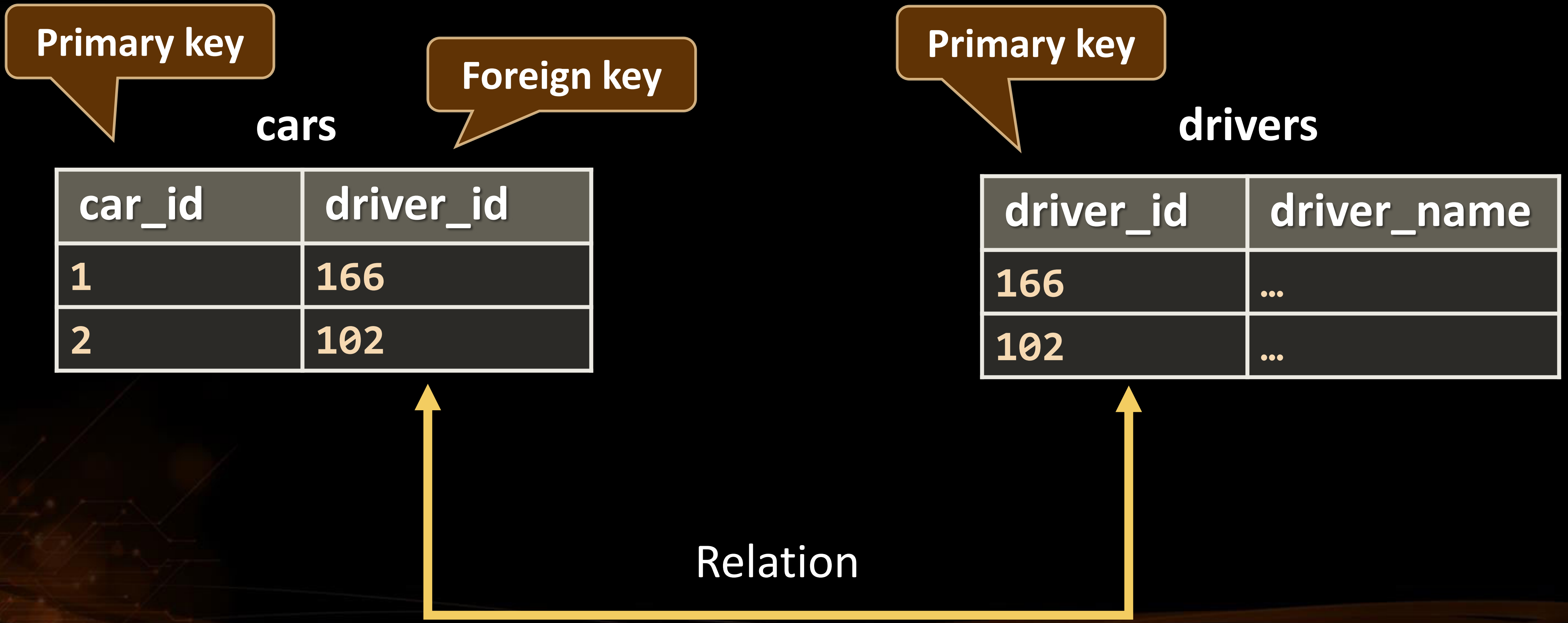
```
CREATE TABLE employees_projects(
  employee_id INT,
  project_id INT,
  CONSTRAINT pk_employees_projects
  PRIMARY KEY(employee_id, project_id),
  CONSTRAINT fk_employees_projects_employees
  FOREIGN KEY(employee_id)
  REFERENCES employees(employee_id),
  CONSTRAINT fk_employees_projects_projects
  FOREIGN KEY(project_id)
  REFERENCES projects(project_id)
);
```

Primary Key

Foreign Key

Foreign Key

One-to-One



cars

car_id	driver_id
1	166
2	102

drivers

driver_id	driver_name
166	…
102	…

Setup

```
CREATE TABLE drivers(
  driver_id INT PRIMARY KEY,
  driver_name VARCHAR(50)
);

CREATE TABLE cars(
  car_id INT PRIMARY KEY,
  driver_id INT UNIQUE,
  CONSTRAINT fk_cars_drivers FOREIGN KEY
  (driver_id) REFERENCES drivers(driver_id)
);
```

Primary key

One driver per car

Foreign Key

Foreign Key

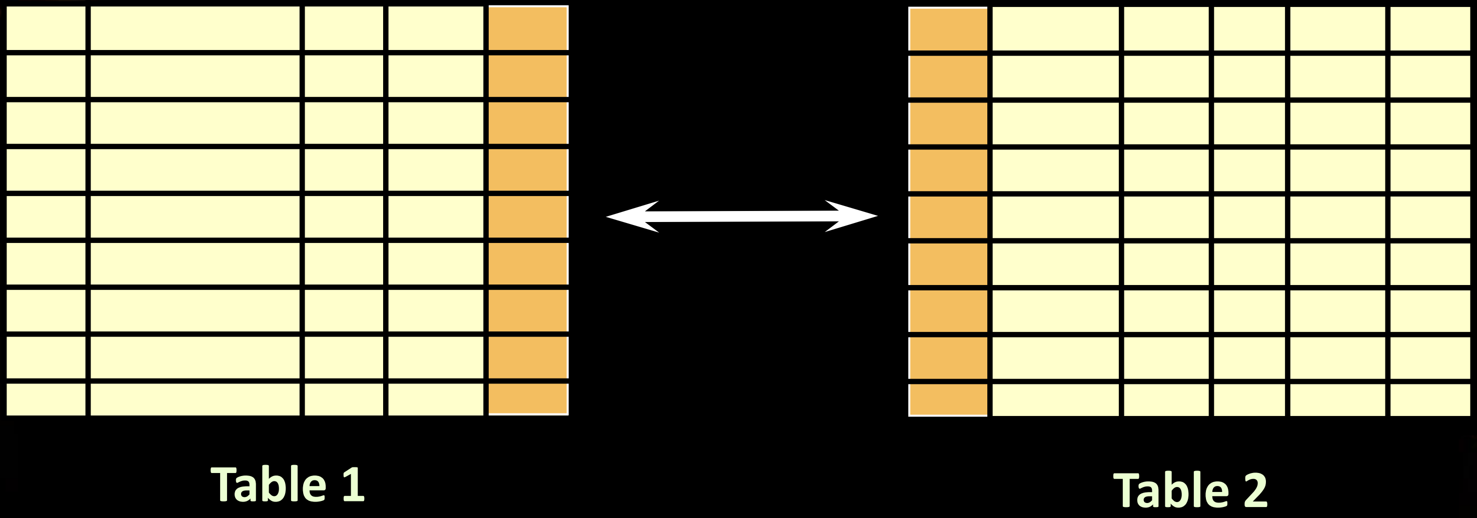
Constraint Name

```
CONSTRAINT fk_cars_drivers
FOREIGN KEY (driver_id)
REFERENCES drivers(driver_id)
```

Foreign Key

Referent Table

Primary Key



Retrieving Related Data

Using Simple JOIN statements

Joins

- Table relations are useful when combined with JOINS
- With JOINS we can get data from two tables simultaneously
  - JOINS require at least two tables and a "join condition"
  - Example:

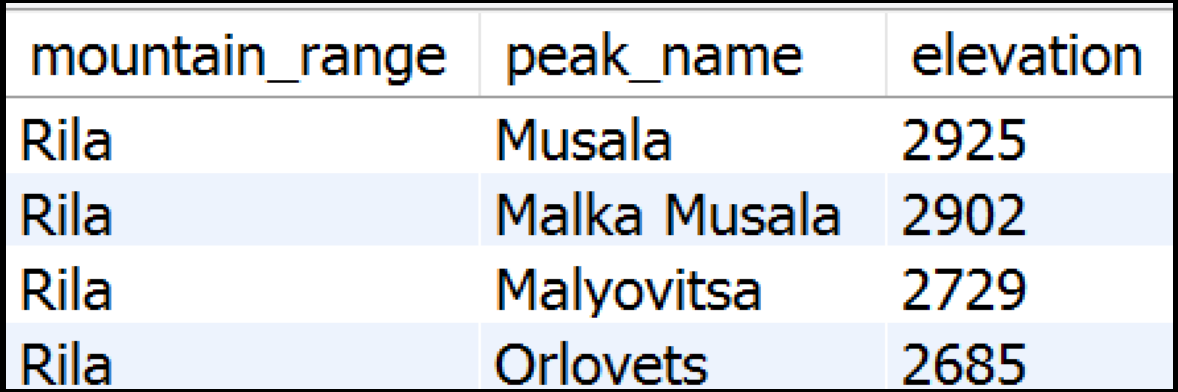
Select from Tables

```
SELECT * FROM table_a
  JOIN table_b ON
    table_b.common_column = table_a.common_column
```

Join Condition

Problem: Peaks in Rila

- Report all peaks for "Rila" mountain.
  - Report includes mountain's name, peak's name and also peak's elevation
  - Peaks should be sorted by elevation descending
  - Use database "Geography".



mountain_range	peak_name	elevation
Rila	Musala	2925
Rila	Malka Musala	2902
Rila	Malyovitsa	2729
Rila	Orlovets	2685

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/605#6

Solution: Peaks in Rila

Cross Table Selection

```
SELECT m.mountain_range, p.peak_name, p.elevation
  FROM peaks AS p
  JOIN mountains AS m ON m.id = p.mountain_id
  WHERE m.mountain_range = 'Rila'
  ORDER BY p.elevation DESC;
```

Join Condition

Sort

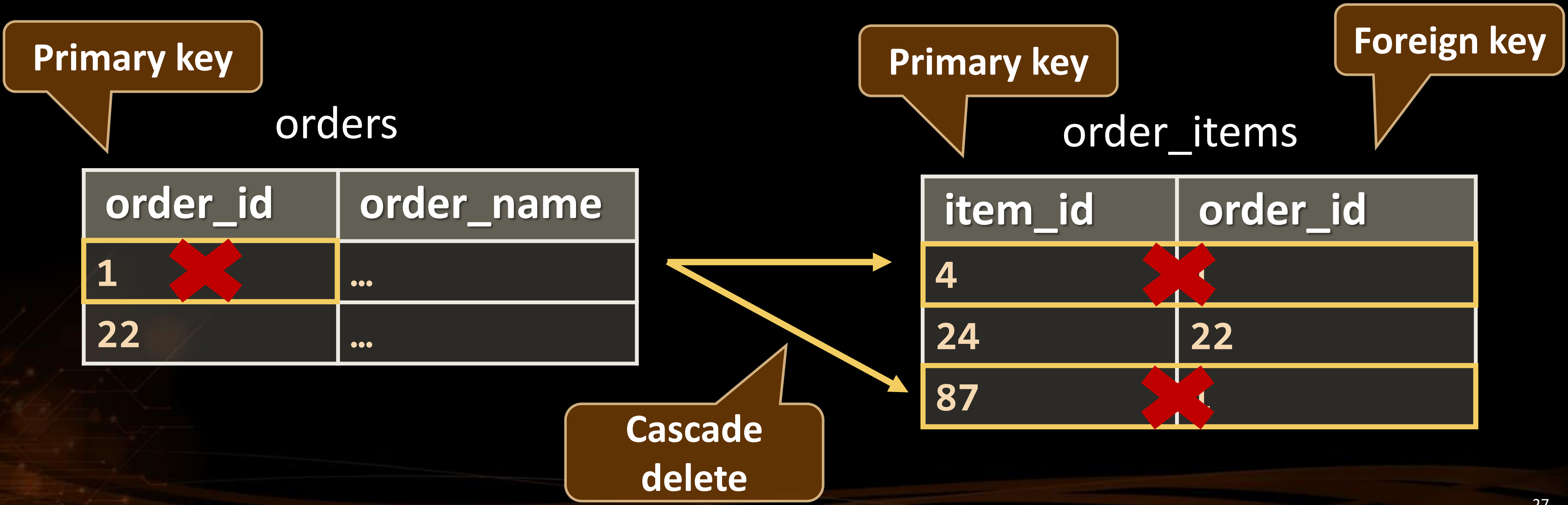
Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/605#6

Cascade Operations

Cascade Delete/Update

Definition

- Cascading allows when a change is made to certain entity, this change to apply to all related entities



orders

order_id	order_name
1	…
22	…

order_items

item_id	order_id
4	1
24	22
87	1

CASCADE DELETE

- **CASCADE** can be either **DELETE** or **UPDATE**.
- Use **CASCADE DELETE** when:
  - The related entities are meaningless without the "main" one
- Do not use **CASCADE DELETE** when:
  - You make "logical delete"
  - You preserve history
  - Keep in mind that in more complicated relations it won't work with circular references

CASCADE UPDATE

- Use **CASCADE UPDATE** when:
  - The primary key is **NOT** identity (not **`auto-increment`**) and therefore it can be changed
  - Best used with **`UNIQUE`** constraint
- Do not use **CASCADE UPDATE** when:
  - The primary is identity (**`auto-increment`**)
- Cascading can be avoided using triggers or procedures

Foreign Key Delete Cascade

```
CREATE TABLE drivers(
  driver_id INT PRIMARY KEY,
  driver_name VARCHAR(50)
);

CREATE TABLE cars(
  car_id INT PRIMARY KEY,
  driver_id INT,
  CONSTRAINT fk_car_driver FOREIGN KEY(driver_id)
  REFERENCES drivers(driver_id) ON DELETE CASCADE
);
```

Table Drivers

Table Cars

Foreign Key

Foreign Key Update Cascade

SOFTWARE UNIVERSITY FOUNDATION

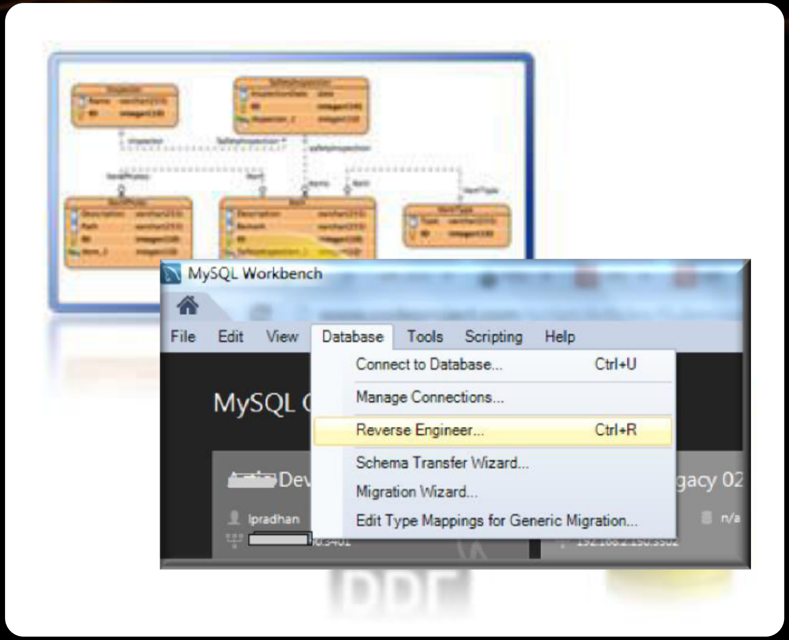
```
CREATE TABLE drivers(
   driver_id INT PRIMARY KEY,
   driver_name VARCHAR(50)
);

CREATE TABLE cars(
   car_id INT PRIMARY KEY,
   driver_id INT,
   CONSTRAINT fk_car_driver FOREIGN KEY(driver_id)
   REFERENCES drivers(driver_id) ON UPDATE CASCADE
);
```

Table Drivers

Table Cars

Foreign Key


MySQL Workbench
File
Edit
View
Database
Tools
Scripting
Help
Connect to Database... Ctrl+U
Manage Connections...
Reverse Engineer... Ctrl+R
Schema Transfer Wizard...
Migration Wizard...
Edit Type Mappings for Generic Migration...

E/R Diagrams

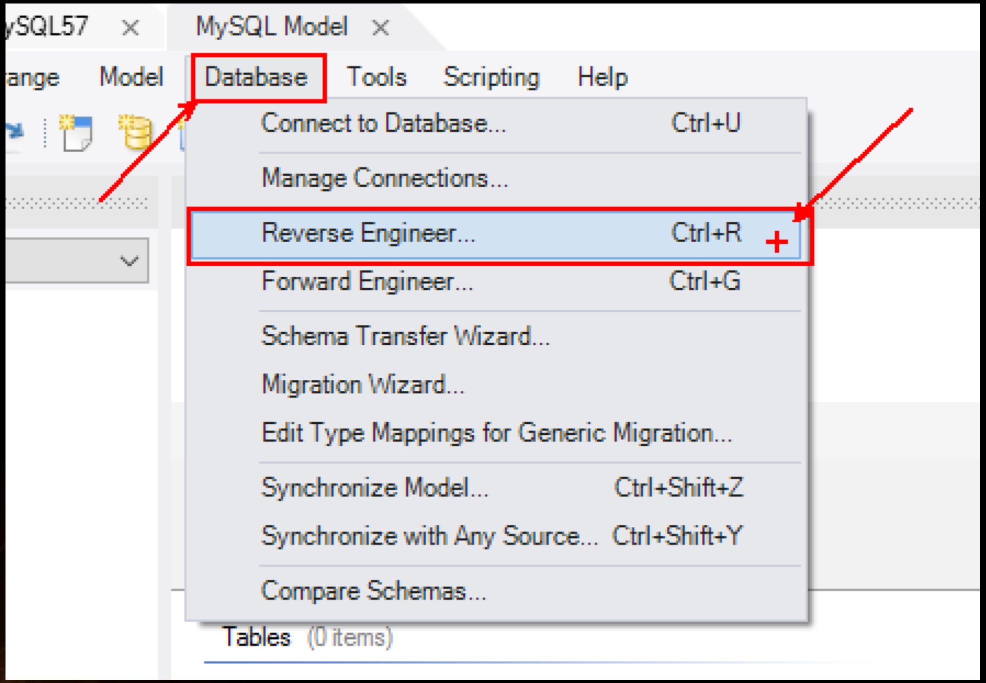
Entity / Relationship Diagrams

Relational Schema

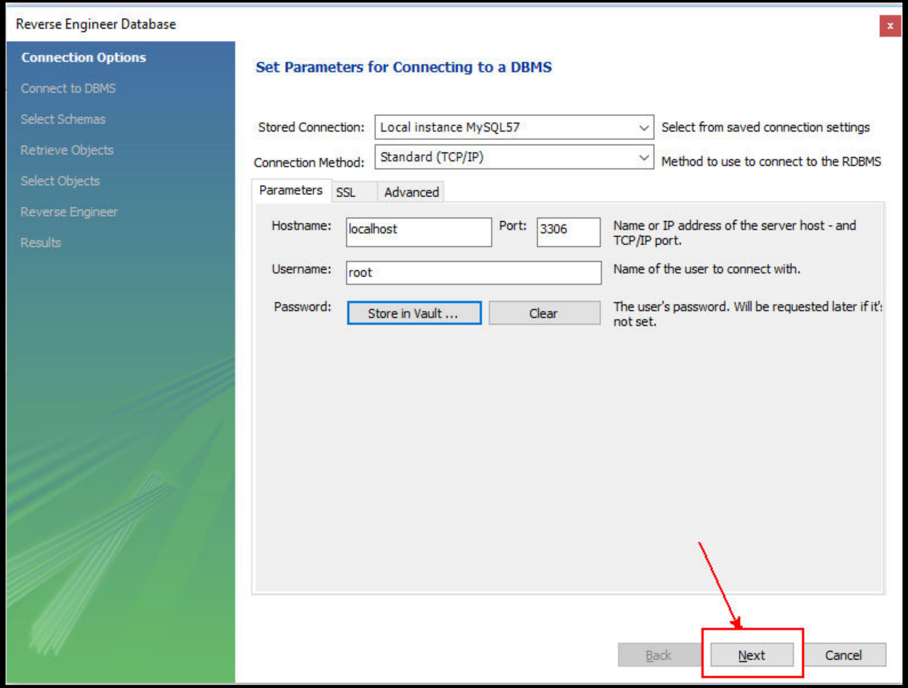
- Relational schema of a DB is the collection of:
  - The schemas of all tables
  - Relationships between the tables
  - Any other database objects (e.g. constraints)
- The relational schema describes the structure of the database
  - Doesn't contain data, but metadata
- Relational schemas are graphically displayed in Entity / Relationship diagrams (E/R Diagrams)

E/R Diagram

- Click on "Database" then select "Reverse Engineer"



ySQL57
MySQL Model
ange
Model
Database
Tools
Scripting
Help
Connect to Database... Ctrl+U
Manage Connections...
Reverse Engineer... Ctrl+R
Forward Engineer... Ctrl+G
Schema Transfer Wizard...
Migration Wizard...
Edit Type Mappings for Generic Migration...
Synchronize Model... Ctrl+Shift+Z
Synchronize with Any Source... Ctrl+Shift+Y
Compare Schemas...
Tables (0 items)



Set Parameters for Connecting to a DBMS

Stored Connection: Local instance MySQL57 — Select from saved connection settings

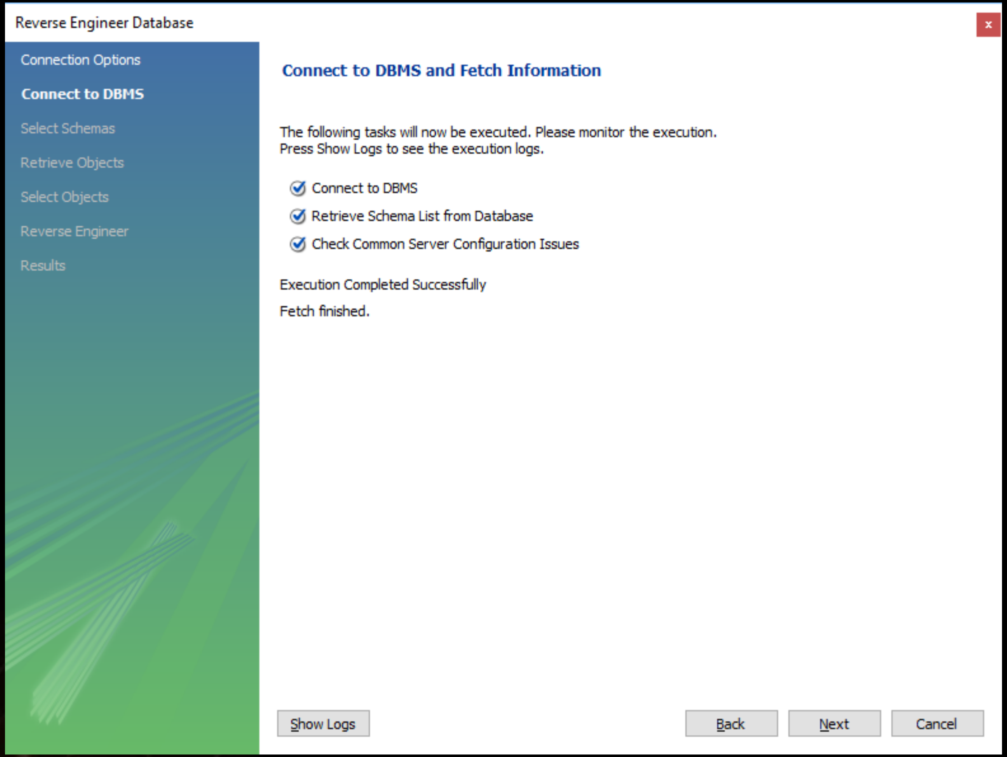
Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: localhost Port: 3306 — Name or IP address of the server host - and TCP/IP port.

Username: root — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

E/R Diagram



Reverse Engineer Database

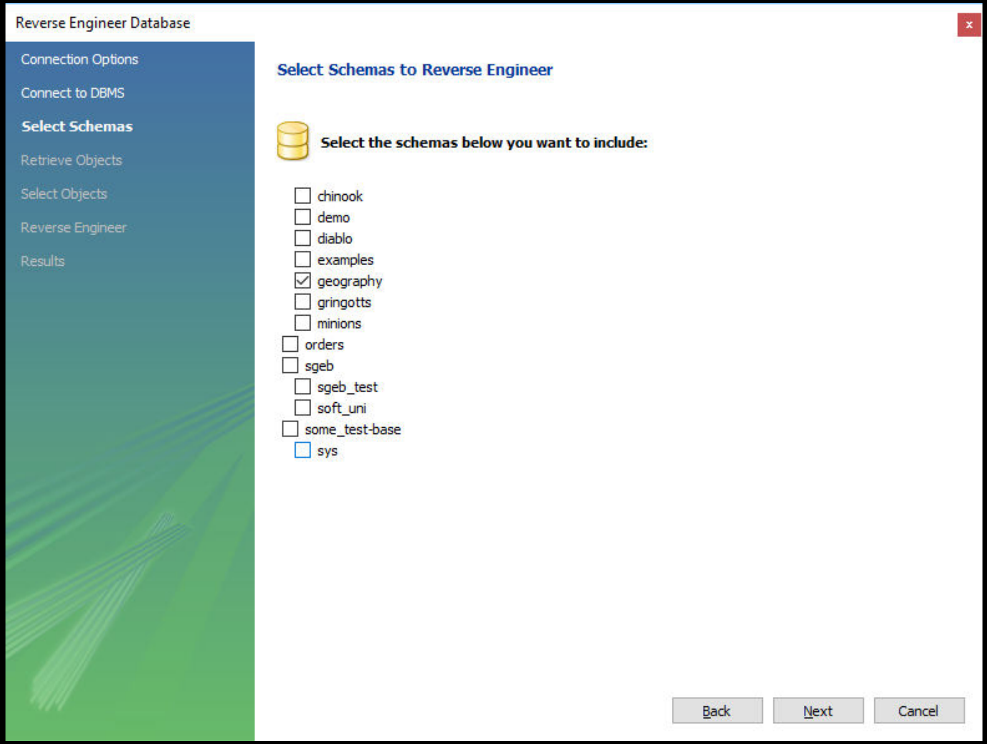
Connect to DBMS and Fetch Information

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

- ✓ Connect to DBMS
- ✓ Retrieve Schema List from Database
- ✓ Check Common Server Configuration Issues

Execution Completed Successfully

Fetch finished.



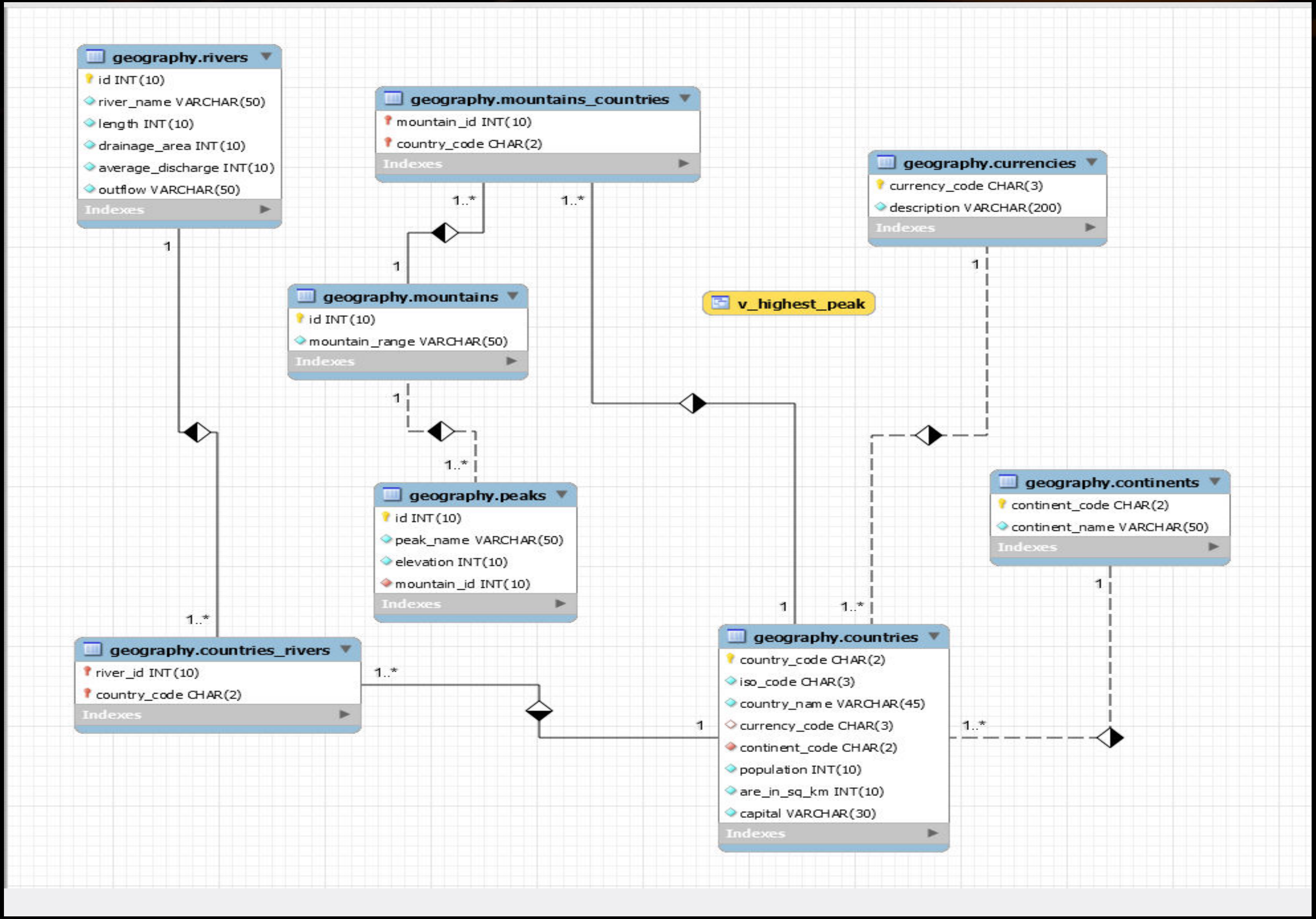
Reverse Engineer Database

Select Schemas to Reverse Engineer

Select the schemas below you want to include:

- ☐ chinook
- ☐ demo
- ☐ diablo
- ☐ examples
- ☑ geography
- ☐ gringotts
- ☐ minions
- ☐ orders
- ☐ sgeb
- ☐ sgeb_test
- ☐ soft_uni
- ☐ some_test-base
- ☐ sys

E/R Diagram


geography.rivers
id INT(10)
river_name VARCHAR(50)
length INT(10)
drainage_area INT(10)
average_discharge INT(10)
outflow VARCHAR(50)
Indexes
geography.mountains_countries
mountain_id INT(10)
country_code CHAR(2)
Indexes
geography.currencies
currency_code CHAR(3)
description VARCHAR(200)
Indexes
geography.mountains
id INT(10)
mountain_range VARCHAR(50)
Indexes
v_highest_peak
geography.peaks
id INT(10)
peak_name VARCHAR(50)
elevation INT(10)
mountain_id INT(10)
Indexes
geography.continents
continent_code CHAR(2)
continent_name VARCHAR(50)
Indexes
geography.countries_rivers
river_id INT(10)
country_code CHAR(2)
Indexes
geography.countries
country_code CHAR(2)
iso_code CHAR(3)
country_name VARCHAR(45)
currency_code CHAR(3)
continent_code CHAR(2)
population INT(10)
are_in_sq_km INT(10)
capital VARCHAR(30)
Indexes
1
1..*
1
1..*
1
1..*
1
1..*
1
1..*
1
1..*
1
1..*
1
1..*

Summary

- We design databases by specification entities and their characteristics
- Two types of relations:
  - One-to-many
  - Many-to-many
- We visualize relations via E/R diagrams

Table Relations

Questions?

https://softuni.bg/courses/databases-basics-mysql

License

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg