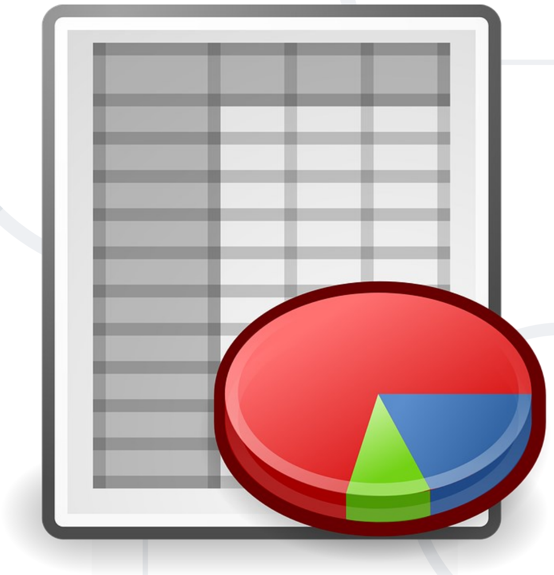
Table Relations

Database Design and Rules



SoftUni Team

Technical Trainers

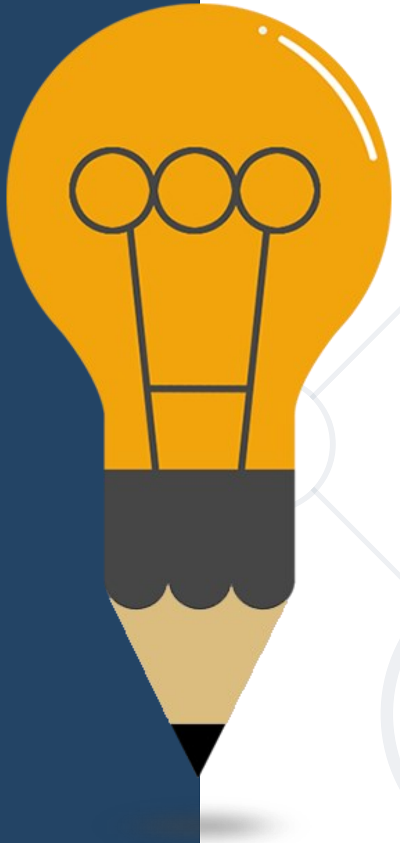
Software University

https://softuni.bg

Table of Contents

Fundamental Concepts

Steps in Database Design



- Steps in the database design process:
  - Identify entities
  - Identify table columns
  - Define a primary key for each table
  - Identify and model relationships
  - Define other constraints
  - Fill tables with test data

DB Design: Identify 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**

DB Design: Identify Table 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 **`IDENTITY`** 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)

DB Design: Identify Entity 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

Database Normalization

Database Normalization

- It is a technique of organizing the data in the database
- **Normalization** is a systematic approach of **decomposing** tables to eliminate data redundancy (repetition) and undesirable characteristics like insertion, update and deletion **anomalies**
- It is a multi-step process that puts data into **tabular form** removing duplicated data from the relation tables

Normal Forms

- **First Normal Form (1NF)**
  - Table should only have single(atomic) valued attributes/columns
  - Values stored in a column should be of the same domain (same type)
  - All the columns in a table should have unique names
  - The order in which data is stored should not matter
- **Second Normal Form (2NF)**
  - The table should be in the **First Normal form**
  - It shouldn't have **Partial Dependency** (dependency on part of the primary key)
- **Third Normal Form (3NF)**
  - The table is in the **Second Normal form**
  - It doesn't have **Transitive Dependency**

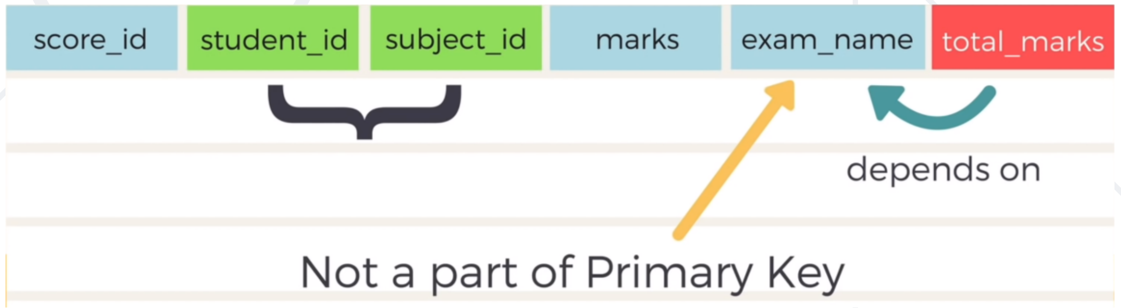
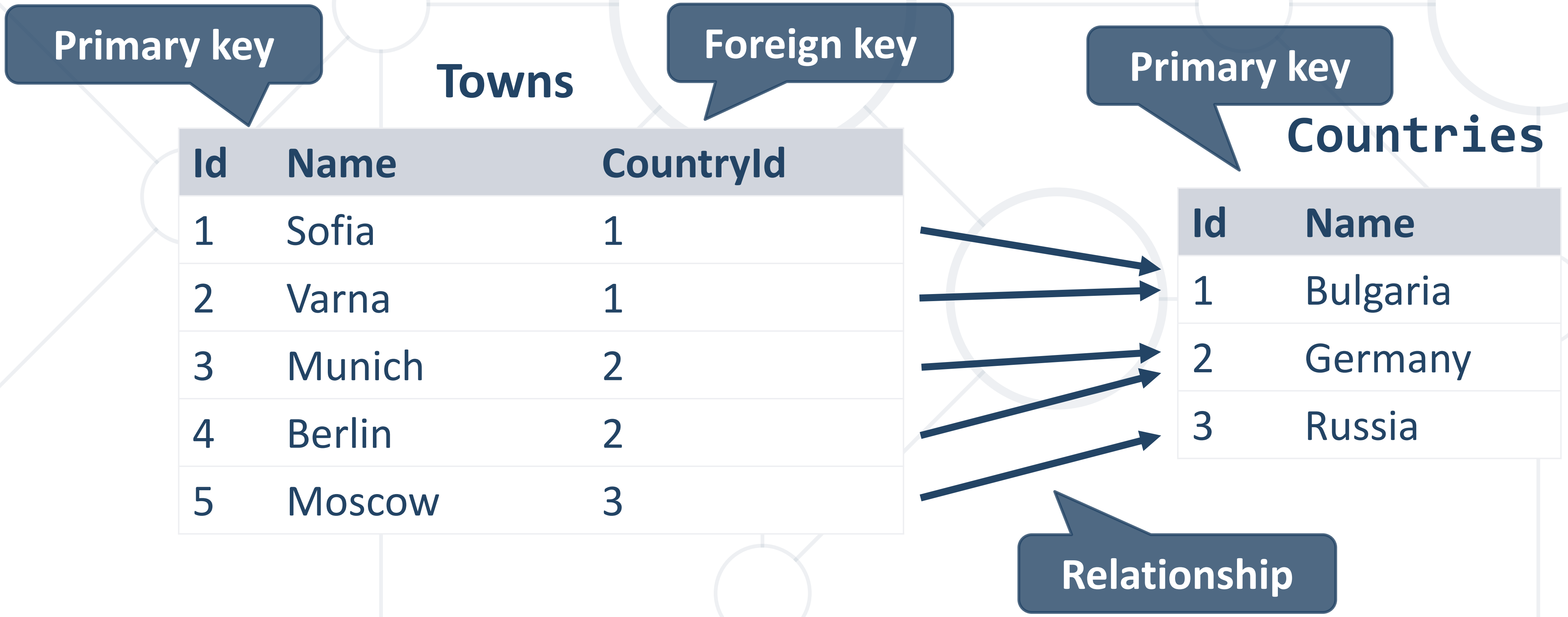


Table Relations

Relational Database Model in Action

Table Relations

- **Relationships** between tables are based on interconnections: **primary key → foreign key**



Towns

Id	Name	CountryId
1	Sofia	1
2	Varna	1
3	Munich	2
4	Berlin	2
5	Moscow	3

Countries

Id	Name
1	Bulgaria
2	Germany
3	Russia

Custom Column Properties

- Primary Key

```
Id INT NOT NULL PRIMARY KEY
```

- Identity (auto-increment)

```
Id INT PRIMARY KEY IDENTITY
```

- Unique constraint – no repeating values in entire table

```
Email VARCHAR(50) UNIQUE
```

Table Relations: Foreign Key

- The **foreign key** is an **identifier** of a record located in **another table** (usually a primary key)
- Using relationships, we **refer** to data instead of **repeating** data
  - Country name is **not repeated**, it is **referred** to by its **primary key**

Towns

Id	Name	CountryId
1	Sofia	1
2	Varna	1
3	Munich	2
4	Berlin	2

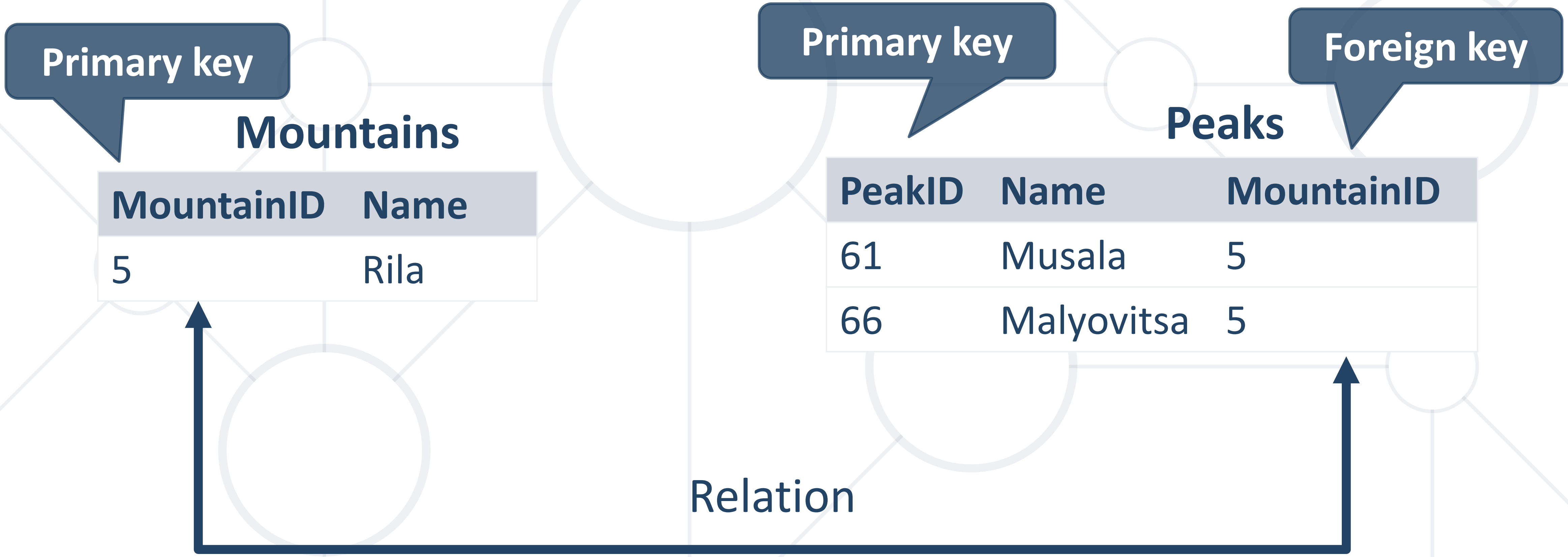
Countries

Id	Name
1	Bulgaria
2	Germany

Table Relations: Multiplicity

- **One-to-many** – e.g. country / towns
  - One country has many towns
- **Many-to-many** – e.g. student / course
  - One student has many courses
  - One course has many students
- **One-to-one** – e.g. example driver / car
  - One driver has only one car
  - Rarely used

One-to-Many/Many-to-One



Mountains

MountainID	Name
5	Rila

Peaks

PeakID	Name	MountainID
61	Musala	5
66	Malyovitsa	5

One-to-Many: Tables

```
CREATE TABLE Mountains(
  MountainID INT PRIMARY KEY,
  MountainName VARCHAR(50)
)
CREATE TABLE Peaks(
  PeakId INT PRIMARY KEY,
  MountainID INT,
  CONSTRAINT FK_Peaks_Mountains
  FOREIGN KEY (MountainID)
  REFERENCES Mountains(MountainID)
)
```

Primary key

Foreign Key

One-to-Many: Foreign Key

- The table holding the **foreign key** is the **child table**
- The table holding the **referenced primary key** is the **parent/referenced table**

Constraint Name

```
CONSTRAINT FK_Peaks_Mountains
FOREIGN KEY (MountainID)
REFERENCES Mountains(MountainID)
```

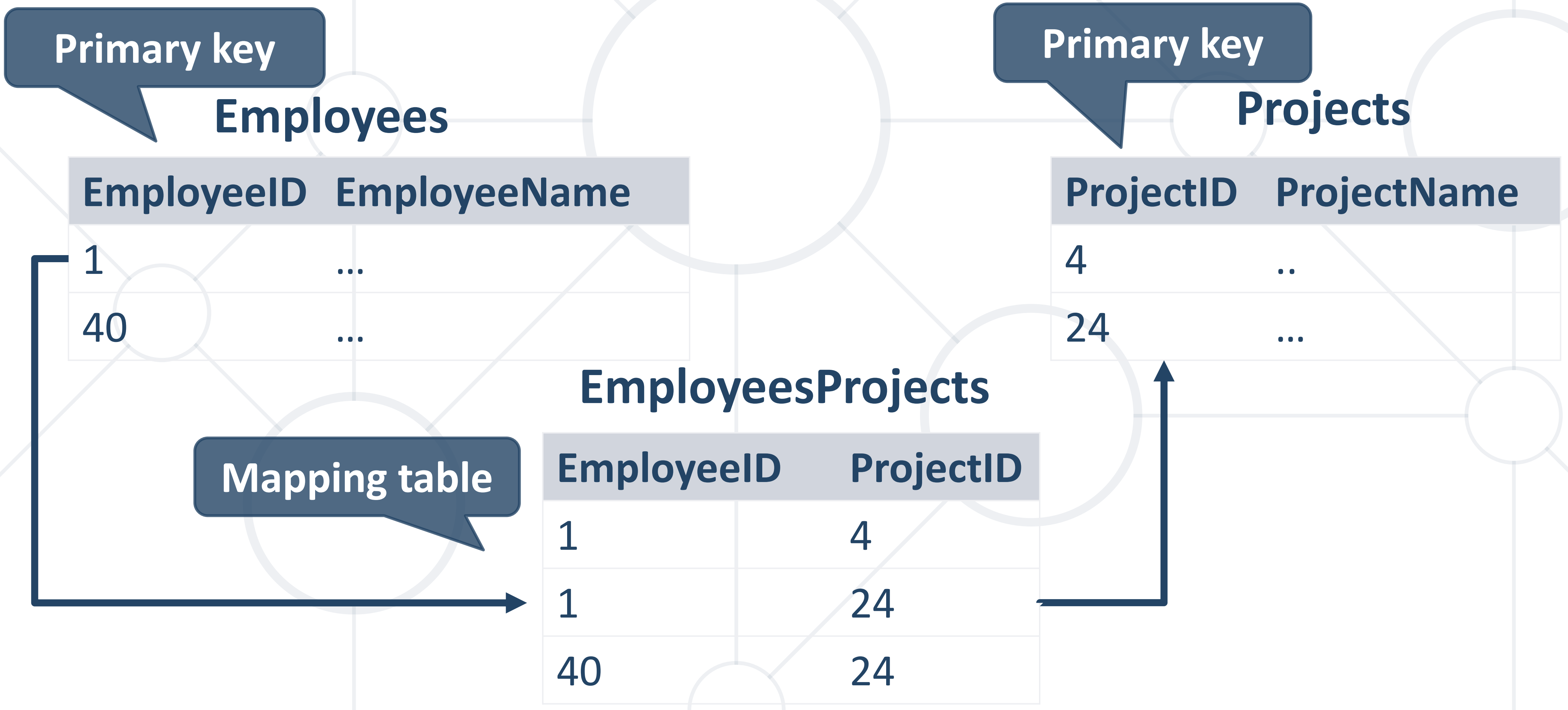
Foreign Key

Parent Table

Primary Key

Many-to-Many

- **Many-to-many relations** use a **mapping/join table**



Employees

EmployeeID	EmployeeName
1	…
40	…

Projects

ProjectID	ProjectName
4	..
24	…

EmployeesProjects

EmployeeID	ProjectID
1	4
1	24
40	24

Many-to-Many: Tables

```
CREATE TABLE Employees(
  EmployeeID INT PRIMARY KEY,
  EmployeeName VARCHAR(50)
)

CREATE TABLE Projects(
  ProjectID INT PRIMARY KEY,
  ProjectName VARCHAR(50)
)
```

Many-to-Many: Mapping Table

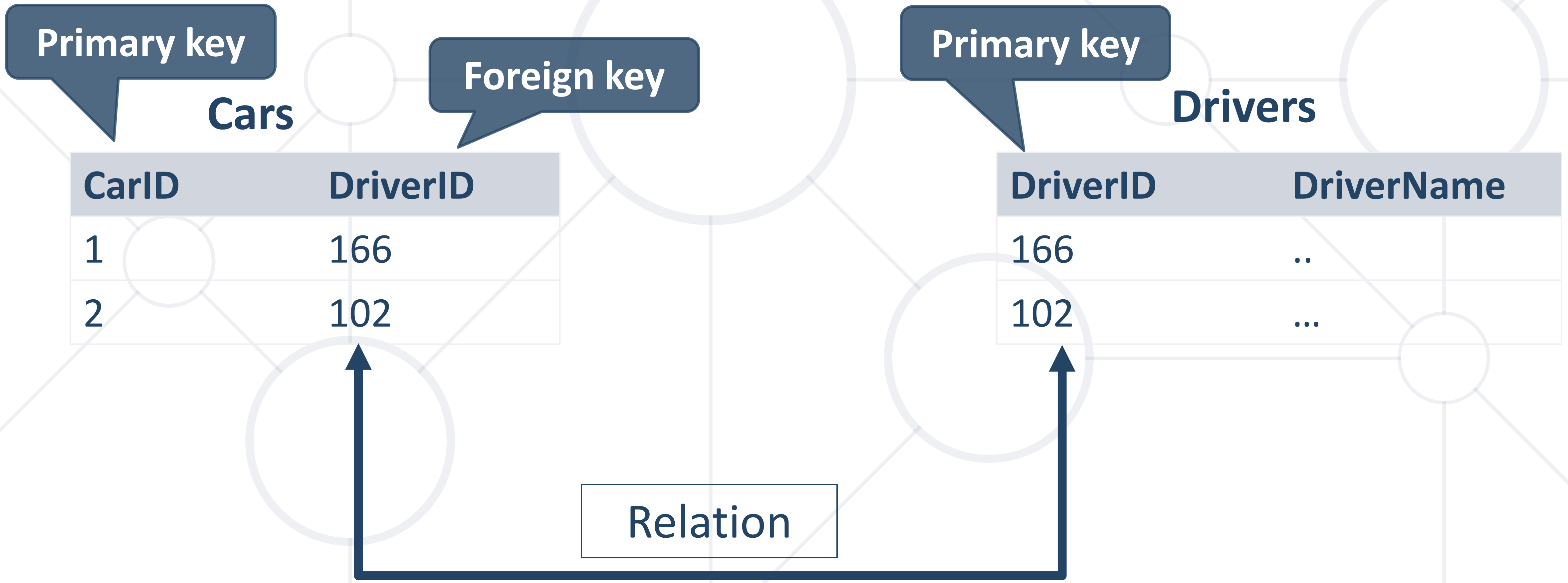
```
CREATE TABLE EmployeesProjects(
  EmployeeID INT,
  ProjectID INT,
  CONSTRAINT PK_EmployeesProjects
  PRIMARY KEY(EmployeeID, ProjectID),
  CONSTRAINT FK_EmployeesProjects_Employees
  FOREIGN KEY(EmployeeID)
  REFERENCES Employees(EmployeeID),
  CONSTRAINT FK_EmployeesProjects_Projects
  FOREIGN KEY(ProjectID)
  REFERENCES Projects(ProjectID)
)
```

Composite Primary Key

Foreign Key to Employees

Foreign Key to Projects

One-to-One



Cars

CarID	DriverID
1	166
2	102

Drivers

DriverID	DriverName
166	..
102	…

One-to-One

```
CREATE TABLE Drivers(
  DriverID INT PRIMARY KEY,
  DriverName VARCHAR(50)
)

CREATE TABLE Cars(
  CarID INT PRIMARY KEY,
  DriverID INT UNIQUE,
  CONSTRAINT FK_Cars_Drivers FOREIGN KEY
  (DriverID) REFERENCES Drivers(DriverID)
)
```

Primary key

One driver per car

Foreign Key

One-to-One: Foreign Key

Constraint Name

```
CONSTRAINT FK_Cars_Drivers
FOREIGN KEY (DriverID)
REFERENCES Drivers(DriverID)
```

Foreign Key

Referenced Table

Primary Key

Summary

1. Design a database using multiple tables with related data
2. Database Normalization
   - First Normal Form
   - Second Normal Form
   - Third Normal Form
3. Table Relation Types

Software University

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Trainings @ Software University (SoftUni)

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

License