Lecture 1 – An introduction to database systems and data modeling

Gianluca Quercini

gianluca.quercini@centralesupelec.fr

Master DSBA 2020 – 2021
In this lecture you will learn:

- What a **database system** is.
- Why a database system is important.
- What a **data model** is.
- The main notions of the **relational data model**.
- How to **design a database** with the **entity-relationship model**.
What is a database system?

Definition (Database system)

A **database system** is a computerized system whose overall purpose is to **store data** and to allow users to **retrieve** and **update** that data on demand.

- **Data.** Anything that matters to the user.
  - **Example.** Personal information on the employees of a company.

- Database systems allow users a number of **operations** on the data.
  - **Read** operations: retrieve some data.
  - **Write** operations: store, update, delete some data.
Data: an example

We identify three main components in data:

- **Entities.** What the data describes.
  - Example. Employees, departments.

- **Attributes.** Properties of the entities.
  - Example. First name, last name, position.

- **Relationships.** How entities relate to each other.
  - Example. Joseph Bennet works in the Administration department.

<table>
<thead>
<tr>
<th>Employees</th>
<th>Departments</th>
</tr>
</thead>
<tbody>
<tr>
<td>First name</td>
<td>Last name</td>
</tr>
<tr>
<td>Joseph</td>
<td>Bennet</td>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
</tr>
<tr>
<td>Patricia</td>
<td>Fisher</td>
</tr>
<tr>
<td>Mary</td>
<td>Green</td>
</tr>
<tr>
<td>William</td>
<td>Russel</td>
</tr>
<tr>
<td>Elizabeth</td>
<td>Smith</td>
</tr>
<tr>
<td>Michael</td>
<td>Watson</td>
</tr>
<tr>
<td>Jennifer</td>
<td>Young</td>
</tr>
</tbody>
</table>
Database system components

- **Users.** Possibly many accessing the database system **concurrently**.
- **Applications.** Accessing the database system on behalf of users.
- **DataBase Management System (DBMS).** Manages all requests for access to the database.
- **Database.** Collection of data.
Why database systems?

Concurrent access

Data redundancy

<table>
<thead>
<tr>
<th>first</th>
<th>last</th>
<th>position</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joseph</td>
<td>Bennet</td>
<td>Office assistant</td>
<td>55,000</td>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
<td>Budget manager</td>
<td>60,000</td>
</tr>
<tr>
<td>Patricia</td>
<td>Fisher</td>
<td>Credit analyst</td>
<td>45,000</td>
</tr>
</tbody>
</table>

Data inconsistency

Concurrent access

Data redundancy

<table>
<thead>
<tr>
<th>first</th>
<th>last</th>
<th>position</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joseph</td>
<td>Bennet</td>
<td>Office assistant</td>
<td>Administration</td>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
<td>Budget manager</td>
<td>Finance</td>
</tr>
<tr>
<td>Patricia</td>
<td>Fisher</td>
<td>Secretary</td>
<td>Education</td>
</tr>
</tbody>
</table>
What is a database?

Definition (Database)

A **database** is a collection of data that is **persistent**, **integrated** and **shared**.

- **Persistent.** Data are only removed at the user’s request.
- **Integrated.** Unification of several otherwise distinct files.
  - Redundancy might partly or completely eliminated.
- **Shared.** Different users can have access to the same data, possibly at the same time.

Users access the data through the **Database Management System (DBMS)**.
DBMS functions

- **Query/update.** Allow users to retrieve and modify data.
- **Indexing.** Optimize the retrieval of data from the database.
- **Integrity.** Preserve the relationship among different related data.
- **Security.** Limit data access to authorized users or programs.
- **Concurrency.** Prevent two users from interfering with each other when they access the same data.
- **Backup/Recovery.** Ensure that the database can be restored to a valid state after a failure.

💡 The DBMS hides the data storage hardware details from the user by using a data model.
A data model is an abstract, self-contained, logical definition of the structure of the data and the operators that manipulate the data.

- A data model defines the representation of:
  - The **entities** (how is an *employee* represented?).
  - The **attributes** (how is *salary* represented?).
  - The **relationships** (how is the fact that an *employee* works in a certain *department* represented?).

- Users interact directly with the data model.
- The DBMS hides the way data are physically stored in the machine.
Major data models

- **Relational** data model (70s).
- **Object-oriented** data model (90s).
- **NoSQL** data models (2000).
  - **Key-value** data model.
  - **Document** data model.
  - **Column family** data model.
  - **Graph** data model.

In this course, we study the relational and the NoSQL data models, in particular document and graph models.

**Definition (Relational data model)**

The *relational data model*, (or, simply, *relational model*) is characterized by the following three aspects.

- **Structural aspect.** The data in the database is perceived by the user as **tables** (or, **relations**), and nothing but tables.

- **Manipulative aspect.** The **operators** available to the user for manipulating the tables and derive tables from tables.

- **Integrity aspect.** The tables satisfy certain **integrity constraints**.
In the **relational model**, a database is a collection of **tables**, or **relations**.

- A **row** in a table (or, a **tuple** in a **relation**) describes an **entity**.
- A **column** in a table (or, an **element** in a **tuple**) represents an **attribute** of an entity.
- A **relationship** between two entities is expressed as common values in one or more columns of their respective tables.
- The relational model provides an *open-ended* collection of **scalar types** (e.g., `boolean`, `integer` . . .).
  - Open-ended: users are allowed to define custom types.

- The values in a given column must have the **same type**.
## Example of relational tables

<table>
<thead>
<tr>
<th>codeE</th>
<th>first</th>
<th>last</th>
<th>position</th>
<th>salary</th>
<th>codeD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joseph</td>
<td>Bennet</td>
<td>Office assistant</td>
<td>55,000</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>Doe</td>
<td>Budget manager</td>
<td>60,000</td>
<td>62</td>
</tr>
<tr>
<td>3</td>
<td>Patricia</td>
<td>Fisher</td>
<td>Secretary</td>
<td>45,000</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
<td>Green</td>
<td>Credit analyst</td>
<td>65,000</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>William</td>
<td>Russel</td>
<td>Guidance counselour</td>
<td>35,000</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>Elizabeth</td>
<td>Smith</td>
<td>Accountant</td>
<td>45,000</td>
<td>62</td>
</tr>
<tr>
<td>7</td>
<td>Michael</td>
<td>Watson</td>
<td>Team leader</td>
<td>80,000</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>Jennifer</td>
<td>Young</td>
<td>Assistant director</td>
<td>120,000</td>
<td>14</td>
</tr>
</tbody>
</table>

### Department

<table>
<thead>
<tr>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
</tr>
<tr>
<td>25</td>
<td>Education</td>
<td>150,000</td>
</tr>
<tr>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
</tr>
<tr>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
</tr>
</tbody>
</table>
The relational model provides some **operators** to manipulate tables.

These operators are collectively known as the **relational algebra**.

**Definition (Relational algebra)**

The **relational algebra** is a collection of **operators** that take tables as their operands and return a table as their result (**relational closure property**).

The relational algebra consists of 8 operators organized into two groups:

- **Set operators.** *Union*, *intersection*, *difference* and *Cartesian product*.
- **Relational operators.** *Select* (a.k.a. *restrict*), *project*, *join* and *divide*. 
Original operators of the relational algebra

- **Union**
- **Intersection**
- **Difference**
- **Product**
- **Project**
- **(Natural) join**

**Select**

**Union**

**Intersection**

**Difference**

**Product**

**Project**

**(Natural) join**

**Division**
An **integrity constraint** is a boolean expression that is associated with a database and is required to evaluate at all times to TRUE.

Examples of integrity constraints:

- **Type constraints.** A value in a relational table column must have a specific type.
- **Key constraints.** No two distinct employees have the same employee number.
- **Foreign key constraints.** Every employee involves an existing department.
- The salary of an employee must be between 30K and 150K.
- The salary of a credit analyst must not exceed the salary of an assistant director.
Correctness and consistency

The database system cannot check if the data is correct.

Closed world assumption

Everything that is in the database is assumed to be true; everything that is not in the database is assumed to be false.

The database system can check if the data is consistent.

- Correct implies consistent (but not the other way round).
- Inconsistent implies incorrect (but not the other way round).
Key constraints

Definition (Key)

A set $X$ of columns of a table $R$ is **key** (a.k.a, **superkey**) of $R$ if and only if there are not two or more distinct rows of $R$ that have the same value for all the columns in $X$.

<table>
<thead>
<tr>
<th>Department</th>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Education</td>
<td>150,000</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
<td></td>
</tr>
</tbody>
</table>

Which of the following sets is a **key**?

- $\{\text{codeD, nameD, budget}\}$
- $\{\text{budget}\}$
- $\{\text{codeD, nameD}\}$
- $\{\text{codeD}\}$
- $\{\text{nameD}\}$

- **Simple key.** Key composed of one column.
- **Composite key.** Key composed of more than one column.
Key constraints

Definition (Key)

A set $X$ of columns of a table $R$ is **key** (a.k.a, **superkey**) of $R$ if and only if there are not two or more distinct rows of $R$ that have the same value for all the columns in $X$.

<table>
<thead>
<tr>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
</tr>
<tr>
<td>25</td>
<td>Education</td>
<td>150,000</td>
</tr>
<tr>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
</tr>
<tr>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
</tr>
</tbody>
</table>

Which of the following sets is a **key**?

- $\{\text{codeD, nameD, budget}\}$  
- $\{\text{budget}\}$
- $\{\text{codeD, nameD}\}$
- $\{\text{codeD}\}$
- $\{\text{nameD}\}$

- **Simple key.** Key composed of one column.
- **Composite key.** Key composed of more than one column.
Key constraints

Definition (Key)
A set \( X \) of columns of a table \( R \) is **key** (a.k.a, superkey) of \( R \) if and only if there are not two or more distinct rows of \( R \) that have the same value for all the columns in \( X \).

<table>
<thead>
<tr>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
</tr>
<tr>
<td>25</td>
<td>Education</td>
<td>150,000</td>
</tr>
<tr>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
</tr>
<tr>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
</tr>
</tbody>
</table>

Which of the following sets is a **key**?
- \{codeD, nameD, budget\}  
- \{budget\}
- \{codeD, nameD\}
- \{codeD\}
- \{nameD\}

- **Simple key.** Key composed of one column.
- **Composite key.** Key composed of more than one column.
**Key constraints**

**Definition (Key)**

A set $X$ of columns of a table $R$ is **key** (a.k.a, **superkey**) of $R$ if and only if there are not two or more distinct rows of $R$ that have the same value for all the columns in $X$.

<table>
<thead>
<tr>
<th>code</th>
<th>name</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
</tr>
<tr>
<td>25</td>
<td>Education</td>
<td>150,000</td>
</tr>
<tr>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
</tr>
<tr>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
</tr>
</tbody>
</table>

Which of the following sets is a **key**?

- $\{\text{code}, \text{name}, \text{budget}\}$ ✓
- $\{\text{budget}\}$ ❌
- $\{\text{code}, \text{name}\}$ ✓
- $\{\text{code}\}$
- $\{\text{name}\}$

- **Simple key.** Key composed of one column.

- **Composite key.** Key composed of more than one column.
Key constraints

Definition (Key)
A set $X$ of columns of a table $R$ is **key** (a.k.a, **superkey**) of $R$ if and only if there are not two or more distinct rows of $R$ that have the same value for all the columns in $X$.

<table>
<thead>
<tr>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
</tr>
<tr>
<td>25</td>
<td>Education</td>
<td>150,000</td>
</tr>
<tr>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
</tr>
<tr>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
</tr>
</tbody>
</table>

Which of the following sets is a **key**?
- $\{\text{codeD, nameD, budget}\}$ ✓
- $\{\text{budget}\}$ ✗
- $\{\text{codeD, nameD}\}$ ✓
- $\{\text{codeD}\}$ ✓
- $\{\text{nameD}\}$

- **Simple key.** Key composed of one column.
- **Composite key.** Key composed of more than one column.
Key constraints

Definition (Key)

A set $X$ of columns of a table $R$ is **key** (a.k.a, **superkey**) of $R$ if and only if there are not two or more distinct rows of $R$ that have the same value for all the columns in $X$.

<table>
<thead>
<tr>
<th>Department</th>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration</td>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
</tr>
<tr>
<td>Education</td>
<td>25</td>
<td>Education</td>
<td>150,000</td>
</tr>
<tr>
<td>Finance</td>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
</tr>
<tr>
<td>Human Resources</td>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
</tr>
</tbody>
</table>

Which of the following sets is a **key**?

- $\{\text{codeD, nameD, budget}\}$  ✓
- $\{\text{budget}\}$  ✗
- $\{\text{codeD, nameD}\}$  ✓
- $\{\text{codeD}\}$  ✓
- $\{\text{nameD}\}$  ✓

- **Simple key.** Key composed of one column.
- **Composite key.** Key composed of more than one column.
Candidate and primary key

Definition (Candidate Key)

A **candidate key** of a table $R$ is a key $X$ such that no proper subset of $X$ is a key.

<table>
<thead>
<tr>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
</tr>
<tr>
<td>25</td>
<td>Education</td>
<td>150,000</td>
</tr>
<tr>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
</tr>
<tr>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
</tr>
</tbody>
</table>

Which of the following sets is a candidate key?
- $\{\text{codeD, nameD, budget}\}$
- $\{\text{codeD, nameD}\}$
- $\{\text{codeD}\}$
- $\{\text{nameD}\}$

- A table may have several candidate keys.
- One candidate key is chosen as the **primary key**.
A **candidate key** of a table $R$ is a key $X$ such that no proper subset of $X$ is a key.

A table may have several candidate keys.

One candidate key is chosen as the **primary key**.
Candidate and primary key

Definition (Candidate Key)

A candidate key of a table $R$ is a key $X$ such that no proper subset of $X$ is a key.

<table>
<thead>
<tr>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
</tr>
<tr>
<td>25</td>
<td>Education</td>
<td>150,000</td>
</tr>
<tr>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
</tr>
<tr>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
</tr>
</tbody>
</table>

A table may have several candidate keys.

One candidate key is chosen as the primary key.

Which of the following sets is a candidate key?

- $\{\text{codeD}, \text{nameD}, \text{budget}\}$ — X
- $\{\text{codeD}, \text{nameD}\}$ — X
- $\{\text{codeD}\}$
- $\{\text{nameD}\}$

Gianluca Quercini
Introduction to Databases
Master DSBA 2020 – 2021 18 / 44
Candidate and primary key

Definition (Candidate Key)
A candidate key of a table $R$ is a key $X$ such that no proper subset of $X$ is a key.

<table>
<thead>
<tr>
<th>Department</th>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration</td>
<td>14</td>
<td>codeD</td>
<td>300,000</td>
</tr>
<tr>
<td>Education</td>
<td>25</td>
<td>nameD</td>
<td>150,000</td>
</tr>
<tr>
<td>Finance</td>
<td>62</td>
<td>codeD</td>
<td>600,000</td>
</tr>
<tr>
<td>Human Resources</td>
<td>45</td>
<td>nameD</td>
<td>150,000</td>
</tr>
</tbody>
</table>

Which of the following sets is a candidate key?
- $\{\text{codeD}, \text{nameD}, \text{budget}\}$  
- $\{\text{codeD}, \text{nameD}\}$  
- $\{\text{codeD}\}$ ✓
- $\{\text{nameD}\}$

- A table may have several candidate keys.
- One candidate key is chosen as the primary key.
Candidate and primary key

**Definition (Candidate Key)**

A **candidate key** of a table $R$ is a key $X$ such that no proper subset of $X$ is a key.

<table>
<thead>
<tr>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
</tr>
<tr>
<td>25</td>
<td>Education</td>
<td>150,000</td>
</tr>
<tr>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
</tr>
<tr>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
</tr>
</tbody>
</table>

Which of the following sets is a **candidate key**?

- $\{\text{codeD, nameD, budget}\}$  \(\times\)
- $\{\text{codeD, nameD}\}$  \(\times\)
- $\{\text{codeD}\}$  \(✓\)
- $\{\text{nameD}\}$  \(✓\)

- A table may have several candidate keys.
- One candidate key is chosen as the **primary key**.
Definition (Foreign key)

Let $T_1$ and $T_2$ be two tables. Let $Y$ be a set of columns of $T_1$. $Y$ is a foreign key of $T_1$ on $T_2$ if and only if $Y$ is a key in $T_2$. 

| Employee(codeD) foreign key references Department(codeD) |
|---|---|---|---|
| codeE | first | last | position | salary | codeD |
| 1 | Joseph | Bennet | Office assistant | 55,000 | 14 |
| 2 | John | Doe | Budget manager | 60,000 | 62 |
| 3 | Patricia | Fisher | Secretary | 45,000 | 25 |
| 4 | Mary | Green | Credit analyst | 65,000 | 62 |
| 5 | William | Russel | Guidance counselour | 35,000 | 25 |
| 6 | Elizabeth | Smith | Accountant | 45,000 | 62 |
| 7 | Michael | Watson | Team leader | 80,000 | 14 |
| 8 | Jennifer | Young | Assistant director | 120,000 | 14 |
Foreign keys

- **Simple foreign keys** consist of one column.
- **Composite foreign keys** consist of more than one column.
- The columns of a foreign key **must have the same type** of the columns they reference.

- The columns of a foreign key **do not have to have the same name** as the columns they reference.
- Although in practice it is often the case (simplicity).
Referential integrity

The DBMS uses foreign keys to enforce the **referential integrity constraint**.

**Definition (Referential integrity constraint)**

Referential integrity means that the database must not contain any unmatched foreign key value.

**Example**

An employee **cannot** work in a department that does not exist.
**Question:** Does this database violate the referential integrity constraint?

Employee

<table>
<thead>
<tr>
<th>codeD</th>
<th>first</th>
<th>last</th>
<th>position</th>
<th>salary</th>
<th>codeD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joseph</td>
<td>Bennet</td>
<td>Office assistant</td>
<td>55,000</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>Doe</td>
<td>Budget manager</td>
<td>60,000</td>
<td>62</td>
</tr>
<tr>
<td>3</td>
<td>Patricia</td>
<td>Fisher</td>
<td>Secretary</td>
<td>45,000</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
<td>Green</td>
<td>Credit analyst</td>
<td>65,000</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>William</td>
<td>Russel</td>
<td>Guidance counselour</td>
<td>35,000</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>Elizabeth</td>
<td>Smith</td>
<td>Accountant</td>
<td>45,000</td>
<td>62</td>
</tr>
<tr>
<td>7</td>
<td>Michael</td>
<td>Watson</td>
<td>Team leader</td>
<td>80,000</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>Jennifer</td>
<td>Young</td>
<td>Assistant director</td>
<td>120,000</td>
<td>14</td>
</tr>
</tbody>
</table>

Department

<table>
<thead>
<tr>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
</tr>
<tr>
<td>25</td>
<td>Education</td>
<td>150,000</td>
</tr>
<tr>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
</tr>
<tr>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
</tr>
</tbody>
</table>

Employee(codeD) foreign key references Department(codeD)
**Referential integrity**

**Question:** Does this database violate the referential integrity constraint?

**Answer:** No.

<table>
<thead>
<tr>
<th>codeE</th>
<th>first</th>
<th>last</th>
<th>position</th>
<th>salary</th>
<th>codeD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joseph</td>
<td>Bennet</td>
<td>Office assistant</td>
<td>55,000</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>Doe</td>
<td>Budget manager</td>
<td>60,000</td>
<td>62</td>
</tr>
<tr>
<td>3</td>
<td>Patricia</td>
<td>Fisher</td>
<td>Secretary</td>
<td>45,000</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
<td>Green</td>
<td>Credit analyst</td>
<td>65,000</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>William</td>
<td>Russel</td>
<td>Guidance counselour</td>
<td>35,000</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>Elizabeth</td>
<td>Smith</td>
<td>Accountant</td>
<td>45,000</td>
<td>62</td>
</tr>
<tr>
<td>7</td>
<td>Michael</td>
<td>Watson</td>
<td>Team leader</td>
<td>80,000</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>Jennifer</td>
<td>Young</td>
<td>Assistant director</td>
<td>120,000</td>
<td>14</td>
</tr>
</tbody>
</table>

**Department**

<table>
<thead>
<tr>
<th>codeD</th>
<th>nameD</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Administration</td>
<td>300,000</td>
</tr>
<tr>
<td>25</td>
<td>Education</td>
<td>150,000</td>
</tr>
<tr>
<td>62</td>
<td>Finance</td>
<td>600,000</td>
</tr>
<tr>
<td>45</td>
<td>Human Resources</td>
<td>150,000</td>
</tr>
</tbody>
</table>
Referential integrity

**Question:** Does this database violate the referential integrity constraint?

<table>
<thead>
<tr>
<th>codeE</th>
<th>first</th>
<th>last</th>
<th>position</th>
<th>salary</th>
<th>codeD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joseph</td>
<td>Bennet</td>
<td>Office assistant</td>
<td>55,000</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>Doe</td>
<td>Budget manager</td>
<td>60,000</td>
<td>62</td>
</tr>
<tr>
<td>3</td>
<td>Patricia</td>
<td>Fisher</td>
<td>Secretary</td>
<td>45,000</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
<td>Green</td>
<td>Credit analyst</td>
<td>65,000</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>William</td>
<td>Russel</td>
<td>Guidance counselour</td>
<td>35,000</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>Elizabeth</td>
<td>Smith</td>
<td>Accountant</td>
<td>45,000</td>
<td>62</td>
</tr>
<tr>
<td>7</td>
<td>Michael</td>
<td>Watson</td>
<td>Team leader</td>
<td>80,000</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>Jennifer</td>
<td>Young</td>
<td>Assistant director</td>
<td>120,000</td>
<td>57</td>
</tr>
</tbody>
</table>

**Employee(codeD) foreign key references Department(codeD)**
Referential integrity

**Question:** Does this database violate the referential integrity constraint?

**Answer:** Yes.

<p>| Employee(codeD) foreign key references Department(codeD) |
|-------------------------------|------------------|</p>
<table>
<thead>
<tr>
<th>codeE</th>
<th>first</th>
<th>last</th>
<th>position</th>
<th>salary</th>
<th>codeD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joseph</td>
<td>Bennet</td>
<td>Office assistant</td>
<td>55,000</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>Doe</td>
<td>Budget manager</td>
<td>60,000</td>
<td>62</td>
</tr>
<tr>
<td>3</td>
<td>Patricia</td>
<td>Fisher</td>
<td>Secretary</td>
<td>45,000</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
<td>Green</td>
<td>Credit analyst</td>
<td>65,000</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>William</td>
<td>Russel</td>
<td>Guidance counselour</td>
<td>35,000</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>Elizabeth</td>
<td>Smith</td>
<td>Accountant</td>
<td>45,000</td>
<td>62</td>
</tr>
<tr>
<td>7</td>
<td>Michael</td>
<td>Watson</td>
<td>Team leader</td>
<td>80,000</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>Jennifer</td>
<td>Young</td>
<td>Assistant director</td>
<td>120,000</td>
<td>57</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>codeD</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>62</td>
</tr>
<tr>
<td>45</td>
</tr>
</tbody>
</table>
Referential integrity and write operations

What happens if you want to change the code of a department?
Referential integrity and write operations

- What happens if you want to change the code of a department?
- It depends on how you define the foreign key constraint.
- Many options are possible:
  - **No Action**. The DBMS will block the update.
  - **Cascade**. The DBMS will update the referring values.
  - **Set NULL**. The DBMS will set the referring values to NULL.
  - **Set Default**. The DBMS will set the referring values to their default value.

- Similar options apply when **deleting a department**.
Referential integrity and write operations

- What happens if you try to delete the table Department?
Referential integrity and write operations

- What happens if you try to delete the table Department?
- The DBMS won’t allow that.
- Before deleting the table Department you can:
  - Delete the table Employee, or:
  - Remove the foreign key constraint on the table Employee.
Database design

Definition (Database design)

Given some body of data to be represented in a database, database design is the process by which we create a suitable conceptual schema for that data.

Definition (Conceptual schema)

The **conceptual schema** of a database consists of:

- **Entities.** Real-world objects about which we collect data.
- **Attributes.** Properties of the entities.
- **Relationships.** Association among entities.
Database design

Example

We want to design the database of a driving school in Île-de-France.

- The school has several branches across the region.
- The school has several customers and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a driver’s license.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.
- Driver’s licenses are issued by a branch and have a unique license number.
- Driver’s licenses are defined by a category (that limits the types of vehicles that the owner can drive) and has an expiry date.
- A customer can have more than one driver’s license.
We want to design the database of a driving school in Île-de-France.

- The school has several branches across the region.
- The school has several customers and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a driver’s license.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.
- Driver’s licenses are issued by a branch and have a unique license number.
- Driver’s licenses are defined by a category (that limits the types of vehicles that the owner can drive) and has an expiry date.
- A customer can have more than one driver’s license.
We want to design the database of a driving school in Île-de-France.

- The school has several branches across the region.
- The school has several customers and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a driver’s license.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.
- Driver’s licenses are issued by a branch and have a unique license number.
- Driver’s licenses are defined by a category (that limits the types of vehicles that the owner can drive) and has an expiry date.
- A customer can have more than one driver’s license.
Database design

Example

We want to design the database of a driving school in Île-de-France.

- The school has several branches across the region.
- The school has several customers and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a driver’s license.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.
- Driver’s licenses are issued by a branch and have a unique license number.
- Driver’s licenses are defined by a category (that limits the types of vehicles that the owner can drive) and has an expiry date.
- A customer can have more than one driver’s license.
Database design

Example

We want to design the database of a driving school in Île-de-France.

- The school has several branches across the region.
- The school has several customers and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a driver’s license.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.
- Driver’s licenses are issued by a branch and have a unique license number.
- Driver’s licenses are defined by a category (that limits the types of vehicles that the owner can drive) and has an expiry date.
- A customer can have more than one driver’s license.
Database design

We want to design the database of a driving school in Île-de-France.

- The school has several branches across the region.
- The school has several customers and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a driver’s license.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.

- Driver’s licenses are issued by a branch and have a unique license number.
- Driver’s licenses are defined by a category (that limits the types of vehicles that the owner can drive) and has an expiry date.
- A customer can have more than one driver’s license.
We want to design the database of a driving school in Île-de-France.

- The school has several branches across the region.
- The school has several customers and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a driver’s license.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.
- Driver’s licenses are issued by a branch and have a unique license number.
- A customer can have more than one driver’s license.
Database design

Example

We want to design the database of a driving school in Île-de-France.

- The school has several branches across the region.
- The school has several customers and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a driver’s license.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.
- Driver’s licenses are issued by a branch and have a unique license number.
- Driver’s licenses are defined by a category (that limits the types of vehicles that the owner can drive) and has an expiry date.
- A customer can have more than one driver’s license.
Database design

We want to design the database of a driving school in Île-de-France.

- The school has several branches across the region.
- The school has several customers and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a driver’s license.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.
- Driver’s licenses are issued by a branch and have a unique license number.
- Driver’s licenses are defined by a category (that limits the types of vehicles that the owner can drive) and has an expiry date.
- A customer can have more than one driver’s license.
Database design

Example

We want to design the database of a driving school in Île-de-France.

- The school has several branches across the region.
- The school has several customers and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a driver’s license.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.
- Driver’s licenses are issued by a branch and have a unique license number.
- Driver’s licenses are defined by a category (that limits the types of vehicles that the owner can drive) and has an expiry date.
- A customer can have more than one driver’s license.

One of the best-known technique for representing the conceptual schema of a database is a **entity-relationship (ER) diagram**.
As their name implies, **ER diagrams** consist of **entities** and **relationships**.

**First**, we need to identify the **entities**.

**Next**, we need to identify the **relationships**.

Both entities and relationships can have **attributes**.

Relationships have **cardinalities**.

- An ER diagram is a **conceptual schema** of a database.
- Eventually, entities and relationships are translated into a collection of tables: the **logical schema** of the database.
Entities

Example

We want to design the database of a driving school in Île-de-France.

- The school has several **branches** across the region.
- The school has several **customers** and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a **driver’s license**.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.
- Driver’s licenses are issued by a branch and have a unique license number.
- Driver’s licenses are defined by a category (that limits the types of vehicles that the owner can drive) and has an expiry date.
- A customer can have more than one driver’s license.
Entities

Customer

Branch

DriverLicense
Relationships

We want to design the database of a driving school in Île-de-France.

- The school has several branches across the region.
- The school has several customers and their personal data is needed.
- Each customer is enrolled in a specific branch.
- Customers must take an exam at any branch to get a driver’s license.
- For each exam, we keep the date and the outcome (pass/fail).
- A customer can take many exams at the same branch.
- Driver’s licenses are issued by a branch and have a unique license number.
- Driver’s licenses are defined by a category (that limits the types of vehicles that the owner can drive) and has an expiry date.
- A customer can have more than one driver’s license.
Relationships

- Customer has a relationship with DriverLicense through "owns" and with Branch through "enrolled".
- DriverLicense has a relationship with Branch through "issues".
- Branch has a relationship with Customer through "take_exam".

Entity-relationship model diagram:

- Customer
- DriverLicense
- Branch

Relationships:
- Customer \(\rightarrow\) DriverLicense: "owns"
- DriverLicense \(\rightarrow\) Branch: "issues"
- Customer \(\rightarrow\) Branch: "take_exam"
- Customer \(\rightarrow\) DriverLicense: "enrolled"
A **cardinality** is expressed as a pair \((min, max)\).

The **minimum cardinality** in \((1, 1)\) means that a customer is enrolled in at least 1 branch.

The **maximum cardinality** in \((1, 1)\) means that a customer is enrolled in at most 1 branch.

The cardinality \((0, n)\) means that a branch can have between 0 and many \((n)\) customers.

The **only** possible values of a cardinality are: 0, 1, \(n\).
Cardinalities

Which cardinalities on the Customer side?

1. (0, n)
2. (1, n)
3. (1, 1)
4. (0, 0)
Cardinalities

Which cardinalities on the Customer side?

1. (0, n) ✓
2. (1, n)
3. (1, 1)
4. (0, 0)
Cardinalities

Which cardinalities on the Branch side?

1. (0, n)
2. (1, n)
3. (1, 1)
4. (0, 0)
Cardinalities

Which cardinalities on the Branch side?

1. (0, n) ✔
2. (1, n)
3. (1, 1)
4. (0, 0)
Cardinalities

Customer

Branch

DriverLicense

enrolled

take_exam

owns

issues

(1,1) (0,n)

(0, n) (0, n)

(1,1)

(1, 1)

Gianluca Quercini
Introduction to Databases
Master DSBA 2020 – 2021 38 / 44
Attributes

- Underlined attributes of an entity are unique identifiers of that entity (primary keys).
- Relationships can have attributes too.
Each entity is translated into a table.
Each attribute of the entity is a column in the corresponding table.
Translating entities and attributes

- **Customer**(`code_customer, first_name, last_name, address, phone_number, birth_date`)
- **Branch**(`branch_name, address`)
- **DriverLicense**(`license_number, category, expiry_date`)

*Underlined columns of a table are part of the primary key of that table.*
Translating relationships

One-to-many relationship. (\(1\), \(1\)) — (\(n\), \(n\))

- Take the **primary key** from Branch (n side) and add it to the attributes of Customer (1 side).
- Customer(\(code\_customer\), first_name, last_name, address, phone_number, birth_date, \(branch\_name\))
- Customer(\(branch\_name\)) **foreign key** to Branch(\(branch\_name\)).
- Same rule applies if the relationship is one-to-one (\(1\), \(1\)) — (\(1\), \(1\))
Many-to-many relationship. \((_, n) \rightarrow (n, n)\)

- The relationship becomes a **table**.
- Attributes of the new table: **primary keys** of the two related entities + **attributes of the relationship**.
- Exam\((\text{code\_customer}, \text{branch\_name}, \text{exam\_date}, \text{outcome})\)
  - \text{code\_customer} **foreign key** to Customer\((\text{code\_customer})\).
  - \text{branch\_name} **foreign key** to Branch\((\text{branch\_name})\).