Lecture 1 – An introduction to database systems and data modeling

Gianluca Quercini

gianluca.quercini@centralesupelec.fr

Master DSBA 2020 – 2021

CentraleSupélec

What you will learn

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. ▸ Source

- **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

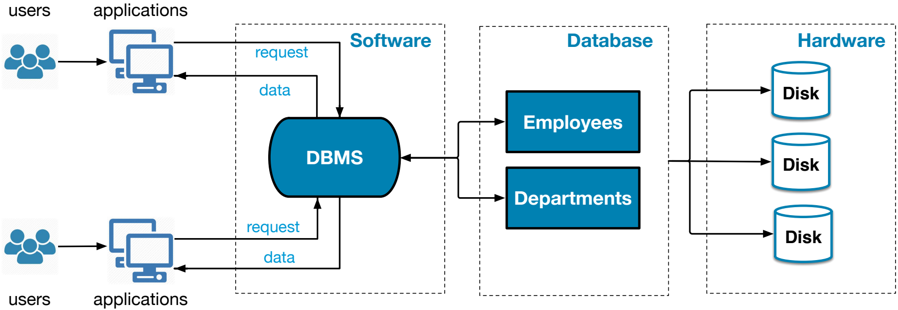
We identify three main components in data:

- **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.

Employees					**Departments**	
First name	**Last name**	**Position**	**Salary**	**Department**	**Name**	**Budget**
Joseph	Bennet	Office assistant	55000	Administration	Administration	300000
John	Doe	Budget manager	60000	Finance	Education	150000
Patricia	Fisher	Secretary	45000	Education	Finance	600000
Mary	Green	Credit analyst	65000	Finance	Human resources	150000
William	Russel	Guidance counselor	35000	Education		
Elizabeth	Smith	Accountant	45000	Finance		
Michael	Watson	Team leader	80000	Administration		
Jennifer	Young	Assistant director	120000	Administration		

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 | **Concurrent access**

Data redundancy | **Data redundancy**

first	last	position	salary
Joseph	Bennet	Office assistant	55,000
John	Doe	Budget manager	60,000
Patricia	Fisher	Credit analyst	45,000

first	last	position	Department
Joseph	Bennet	Office assistant	Administration
John	Doe	Budget manager	Finance
Patricia	Fisher	Secretary	Education

Data inconsistency

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**.

▸ Source

Data model

Definition (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**.

Relational data model

- Proposed by Edgar F. Codd (English computer scientist, 1923-2003) in 1970.

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**.

▸ Source

Structure of the data: tables

☞ 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

Employee

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

Department

codeD	nameD	budget
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

Relational algebra

- The relational model provides some **operators** to manipulate tables.
- These operators are collectively known as the **relational algebra**.

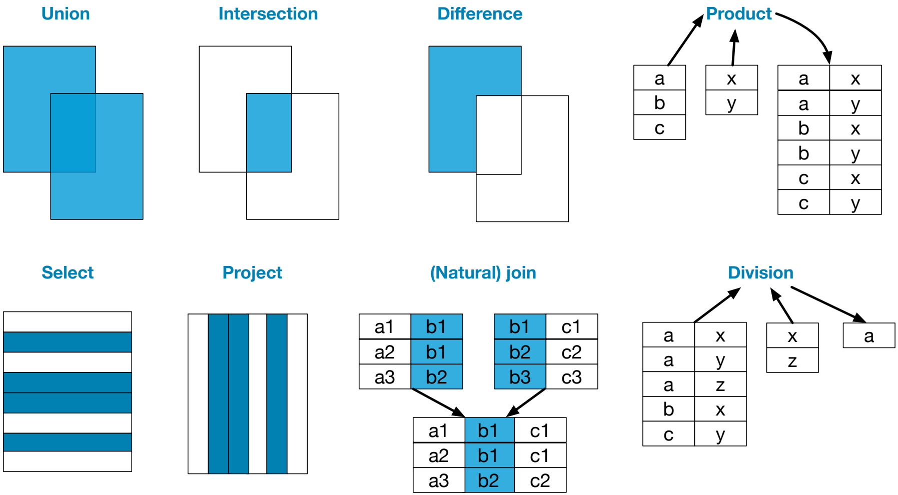
Definition (Relational algebra)

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

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



Source

Integrity constraints

Definition

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

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.**

- The data is consistent if no integrity constraint is violated.

- **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.

Department		
codeD	**nameD**	**budget**
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

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.

Department		
codeD	**nameD**	**budget**
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

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.

Department		
codeD	**nameD**	**budget**
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

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.

Department

codeD	nameD	budget
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

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.

Department		
codeD	**nameD**	**budget**
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

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.

Department

codeD	nameD	budget
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

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.

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.

Department

codeD	nameD	budget
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

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

- {codeD, nameD, budget}
- {codeD, nameD}
- {codeD}
- {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.

Department		
codeD	**nameD**	**budget**
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

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

- {codeD, nameD, budget} ✗
- {codeD, nameD}
- {codeD}
- {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.

Department		
codeD	**nameD**	**budget**
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

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

- {codeD, nameD, budget} ❌
- {codeD, nameD} ❌
- {codeD}
- {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.

Department		
codeD	**nameD**	**budget**
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

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

- {codeD, nameD, budget} ✗
- {codeD, nameD} ✗
- {codeD} ✓
- {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.

Department		
codeD	**nameD**	**budget**
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

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

- {codeD, nameD, budget} ✗
- {codeD, nameD} ✗
- {codeD} ✓
- {nameD} ✓

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

Foreign keys

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

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

Department

codeD	nameD	budget
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

Employee(codeD) foreign key references Department(codeD)

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. ▸ Source

Example

An employee **cannot** work in a department that does not exist.

Referential integrity

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

Employee

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

Department

codeD	nameD	budget
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

Employee(codeD) foreign key references Department(codeD)

Referential integrity

- **Question:** Does this database violate the referential integrity constraint?
- **Answer:** No.

Employee

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

Department

codeD	nameD	budget
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

Employee(codeD) foreign key references Department(codeD)

Referential integrity

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

Employee

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	**57**

Department

codeD	nameD	budget
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

Employee(codeD) foreign key references Department(codeD)

Referential integrity

- **Question:** Does this database violate the referential integrity constraint?
- **Answer:** Yes.

Employee

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	**57**

Department

codeD	nameD	budget
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

Employee(codeD) foreign key references Department(codeD)

Referential integrity and write operations

- What happens if you want to change the code of a department?

Employee					
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

Department		
codeD	**nameD**	**budget**
14	Administration	300,000
25	Education	150,000
62	Finance	600,000
45	Human Resources	150,000

Employee(codeD) foreign key references Department(codeD)

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. ▸ Source

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.

▸ Source

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

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

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

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

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.

- One of the best-known technique for representing the conceptual schema of a database is a **entity-relationship (ER) diagram**.

Entity-Relationship (ER) diagrams

- 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.

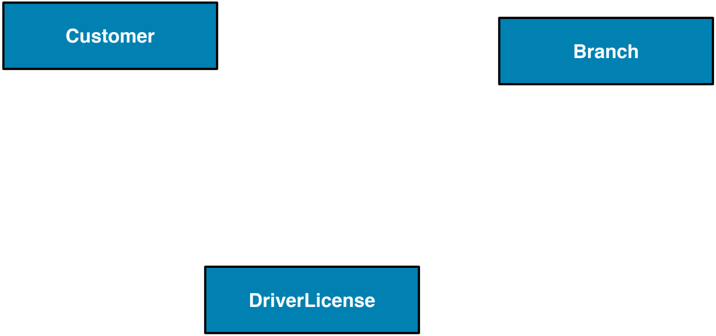
Entities

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



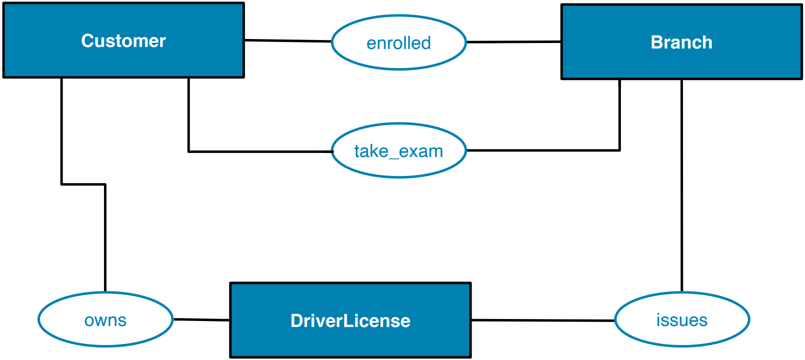
Relationships

Example

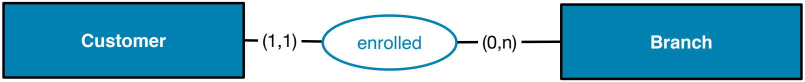
We want to design the database of a driving school in Île-de-France.

- 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**.

Relationships

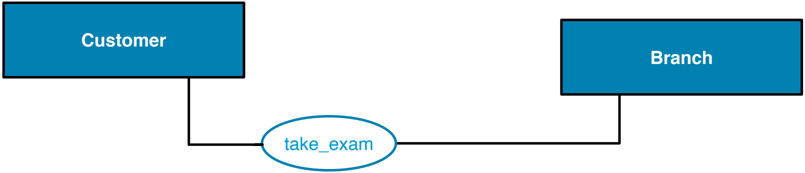


Cardinalities



- 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.

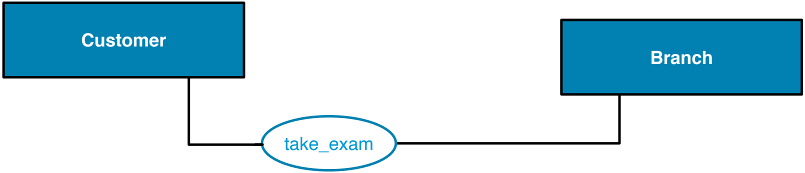
Cardinalities



Which cardinalities on the `Customer` side?

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

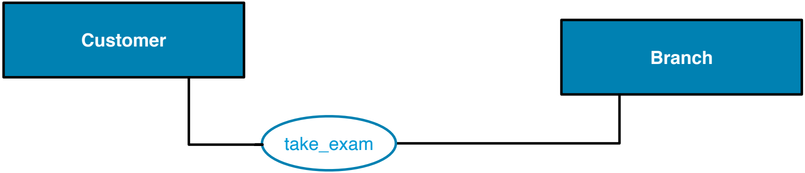
Cardinalities



Which cardinalities on the `Customer` side?

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

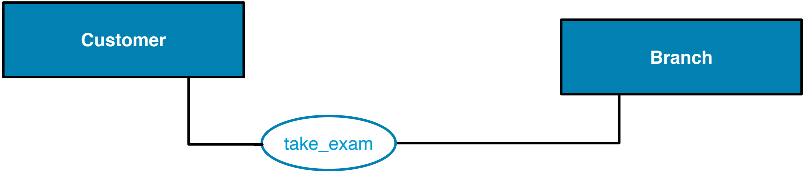
Cardinalities



Which cardinalities on the `Branch` side?

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

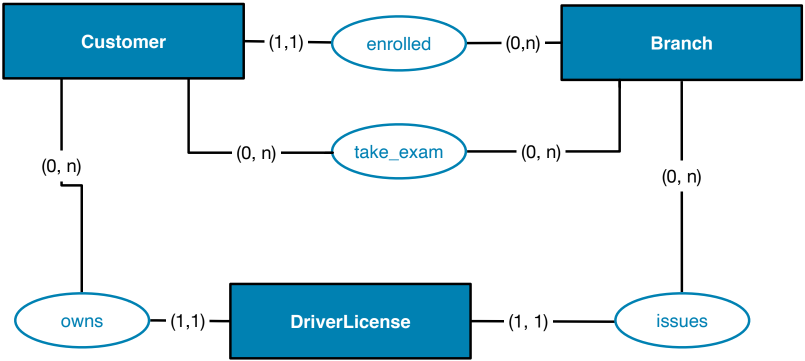
Cardinalities



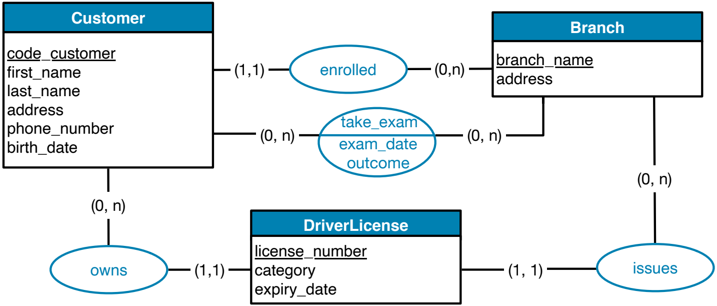
Which cardinalities on the `Branch` side?

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

Cardinalities



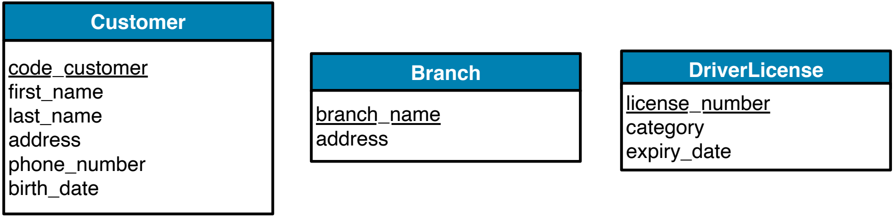
Attributes



☞

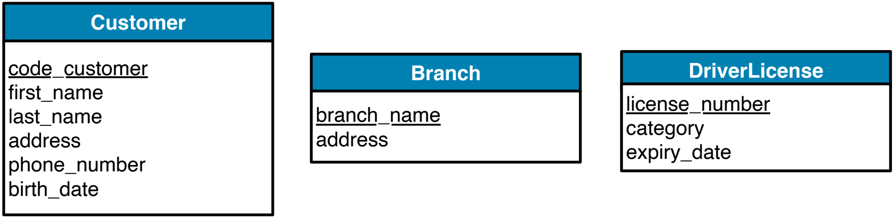
- **Underlined attributes** of an entity are **unique identifiers** of that entity (primary keys).
- Relationships can have attributes too.

From the conceptual to the logical schema



- Each **entity** is translated into a table.
- Each **attribute** of the entity is a **column** in the corresponding table.

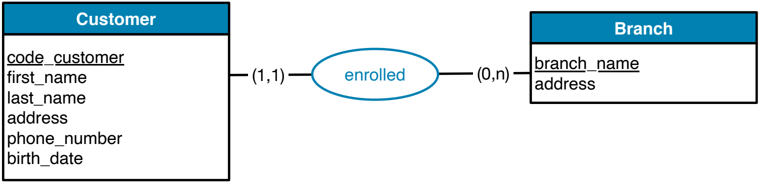
Translating entities and attributes



- 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.

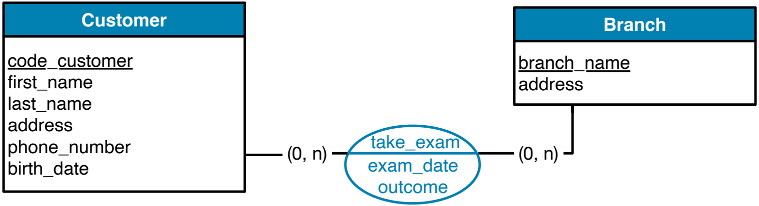
Translating relationships



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

- Take the **primary key** from `Branch` (**n** side) and add it to the attributes of `Customer` (**1** side).
- `Customer`(<u>code_customer</u>, 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**)

Translating relationships



Many-to-many relationship. (_, **n**) — (_, **n**)

- The relationship becomes a **table**.
- Attributes of the new table: **primary keys** of the two related entities + **attributes of the relationship**.
- `Exam`(<u>code_customer</u>, <u>branch_name</u>, <u>exam_date</u>, outcome)
- code_customer **foreign key** to `Customer`(code_customer).
- branch_name **foreign key** to `Branch`(branch_name).

References

- Date, Christopher John. *An introduction to database systems*. Pearson Education India, 2004. Click here