Introduction to Databases

Gianluca Quercini

Laboratoire de Recherche en Informatique
CentraleSupélec

2019 – 2020

Plan of the Course

1. Introduction to Database Management Systems (1.5h).
    - **1st Lab Assignment:** *Data Modeling* (1.5h).
2. Using a Relational Database Management System (1.5h).
    - **2nd Lab Assignment:** *Interacting with a MySQL Server* (1.5h).
3. Indexing, Transactions and Stored Procedures in Relational Databases (1.5h).
4. Distributed Databases (1.5h).
5. Introduction to NoSQL DBMS (1h).
6. Using a NoSQL DBMS: MongoDB (1h)
    - **3rd Lab Assignment:** *Using MongoDB* (2h).
7. Graph Databases: Neo4j (1.5h)
    - **4th Lab Assignment:** *Using Neo4j* (3h).
8. Introduction to MapReduce (0.5h).

Evaluation

The evaluation of the course is based on:

- **Group Project** (6h)
  - Groups of 2-3 students.
  - **Goal:** Developing an application that interacts with several DBMS.
  - **Submission:** Source code + a written report.
  - Accounts for **40%** of the final grade.
- **Written examination** (3h).
  - A quiz (1h).
  - Exercises (2h).
  - Accounts for **60%** of the final grade.

Contact Information

- **Email:** gianluca.quercini@centralesupelec.fr
- **Slack** `https://bit.ly/2kNVmJZ` channel #introduction-to-databases

Introduction to Database Management Systems

Storing and Managing Data

- Computer applications need to store and retrieve data.
- Data is kept on **hard disks**.
- The **operating system** provides the **file system** to store and organize the data.
  - Hides the technical details of how data is physically stored.
  - Data is represented as **files**.
- However, the file system lacks many important **data management features**.

Data Management

- **Querying**: how to **efficiently search** data?
- **Updating**: how to handle **inconsistencies** when data is **replicated**?
- **Access Control**: how to assign different **access privileges**?
- **Concurrent Access**: how to deal with different applications trying to access the same file?

A	B	C	D	E	F	G	H
		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			

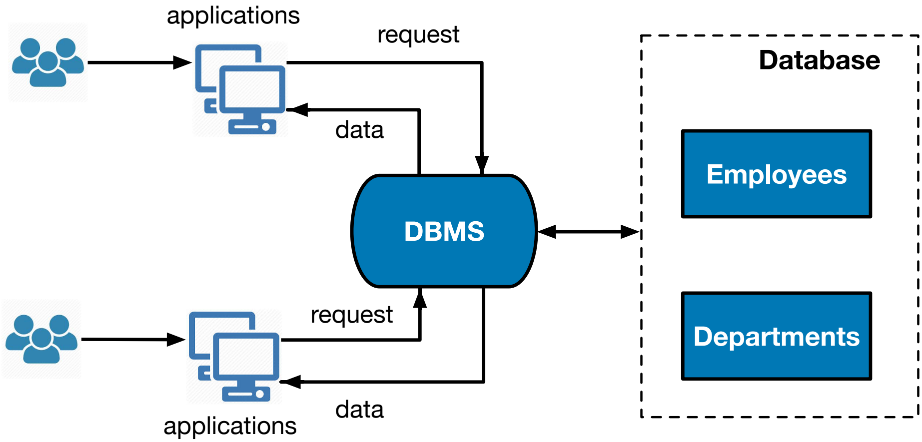
DataBase Management Systems (DBMS)

Definition (DBMS)

A **DataBase Management System** (**DBMS**) is a software application that manages collections of data, or **databases**, by using a **data model**.

- A **data model** is an abstract definition of the objects described by the data.
- A data model defines:
  - **Entities** (e.g., `employee`, `department`).
  - **Attributes**, properties of the entities (e.g., *name*, *salary* . . .).
  - **Relationships**, between sets of entities (employee x works in department d).

DataBase Management Systems (DBMS)



Data Models

Different data models exist:

- **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.

Relational Data Model (Codd, 1970)

- A **database** is a collection of **relations**, or **tables**.

Attributes → (header row); Entities (rows)

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

- **Schema** of a table: set of its attributes.
- The **values** of each attribute have a precise **type**.
- Identification of the entities: notion of **key**.

Relational Data Model (Codd, 1970)

- Tables are **related** through their attributes.
    - Notion of **foreign key**.

Relationship

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

Notion of Key

Definition (Key)

A set X of columns of a table R is **key** 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.

Relation R		
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}

Notion of Key

Definition (Key)

A set X of columns of a table R is **key** 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.

Relation R		
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}

Notion of Key

Definition (Key)

A set X of columns of a table R is **key** 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.

Relation R		
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}

Notion of Key

Definition (Key)

A set X of columns of a table R is **key** 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.

Relation R		
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}

Notion of Key

Definition (Key)

A set X of columns of a table R is **key** 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.

Relation R		
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}

Notion of Key

Definition (Key)

A set X of columns of a table R is **key** 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.

Relation R		
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}

Candidate 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.

Relation R		
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 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.

Relation R		
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 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.

Relation R		
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 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.

Relation R		
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 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.

Relation R		
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

Foreign keys are used to express the **relationships** between tables.

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.

Relationship

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

Foreign Keys

- The DBMS uses a foreign key to enforce the **referential integrity constraint**.

Example:

- `Employee(codeD)` **foreign key** to `Department(CodeD)`.
- An employee **cannot** work in a department that does not exist.

Data consistency.

- The referential integrity constraint ensures **data consistency** across tables.

Foreign Keys

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

Relationship

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

Foreign Keys

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

Relationship

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

Foreign Key

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

Relationship

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

Foreign Key

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

Relationship

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

Foreign Keys

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

Relationship

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

Foreign Keys

- What happens if you want to change the code of a department?
- It depends on the how you define the foreign key.
- 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.

Foreign Keys

- What happens if you try to **delete** the table `Department`?

Foreign Keys

- 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`.
- The two columns involved in the foreign key can have **different names**.
  - But they have to have the same **type**!
- A foreign key can be a **group** of columns.

Foreign Keys

- 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`.
- The two columns involved in the foreign key can have **different names**.
  - But they have to have the same **type**!
- A foreign key can be a **group** of columns.

Foreign Keys

- 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`.
- The two columns involved in the foreign key can have **different names**.
  - But they have to have the same **type**!
- A foreign key can be a **group** of columns.

Database Design

We want to design the database of a driving school in Île-de-France that has several branches across the region. The school needs to store personal data about each customer, as well as the branch where the customer is enrolled. In order to get a driver's license, a customer must take an exam at any branch of the school. If s/he fails the exam, she can take the exam again any time after a week of the failed exam date. If she passes the exam, the branch where she took the exam will give her a driver license with a unique license number; the license is 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 license.

- **Which tables** do we need to create?
- Is there a **methodology** for designing a database?
- The answer is **Entity-Relationship** (ER) Diagrams.

Database Design

We want to design the database of a driving school in Île-de-France that has several branches across the region. The school needs to store personal data about each customer, as well as the branch where the customer is enrolled. In order to get a driver's license, a customer must take an exam at any branch of the school. If s/he fails the exam, she can take the exam again any time after a week of the failed exam date. If she passes the exam, the branch where she took the exam will give her a driver license with a unique license number; the license is 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 license.

- **Which tables** do we need to create?
- Is there a **methodology** for designing a database?
- The answer is **Entity-Relationship** (ER) Diagrams.

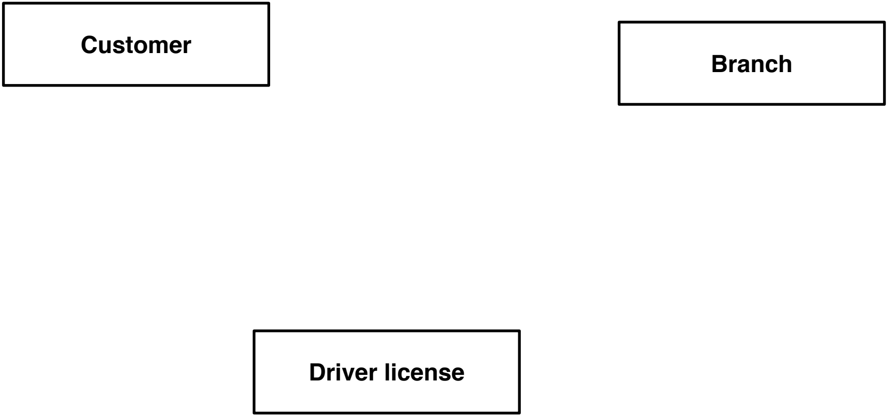
Entity-Relationship (ER) Diagrams

- 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 **logical data model**.
- Eventually, entities and relationships are translated into a collection of tables (the **physical data model**).

Entities

We want to design the database of a driving school in Île-de-France that has several **branches** across the region. The school needs to store personal data about each **customer**, as well as the branch where the customer is enrolled. In order to get a driver's license, a customer must take an exam at any branch of the school. If s/he fails the exam, she can take the exam again any time after a week of the failed exam date. If she passes the exam, the branch where she took the exam will give her a **driver license** with a unique license number; the license is 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 license.

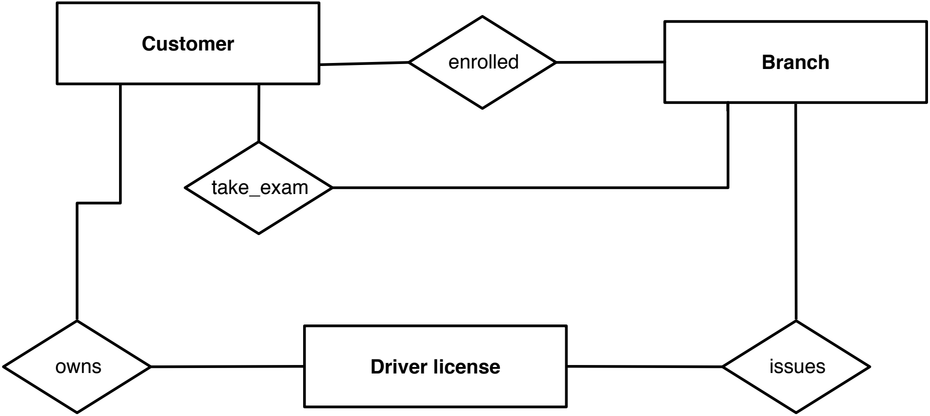
Entities



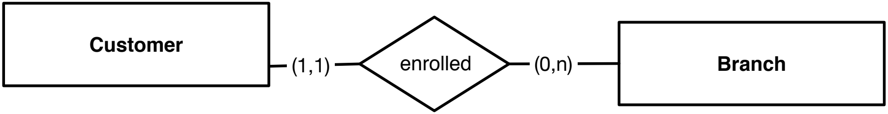
Relationships

We want to design the database of a driving school in Île-de-France that has several branches across the region. The school needs to store personal data about each customer, as well as the branch where the customer **is enrolled**. In order to get a driver's license, a customer must **take an exam** at any branch of the school. If s/he fails the exam, she can take the exam again any time after a week of the failed exam date. If she passes the exam, the branch where she took the exam will **give her** a driver license with a unique license number; the license is 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 license.

Relationships



Cardinalities



- **Cardinalities** are expressed as a pair (min, max).
- The **minimum cardinality** in $(\mathbf{1}, 1)$ means that a customer is enrolled in at least 1 branch.
- The **maximum cardinality** in $(1, \mathbf{1})$ 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?

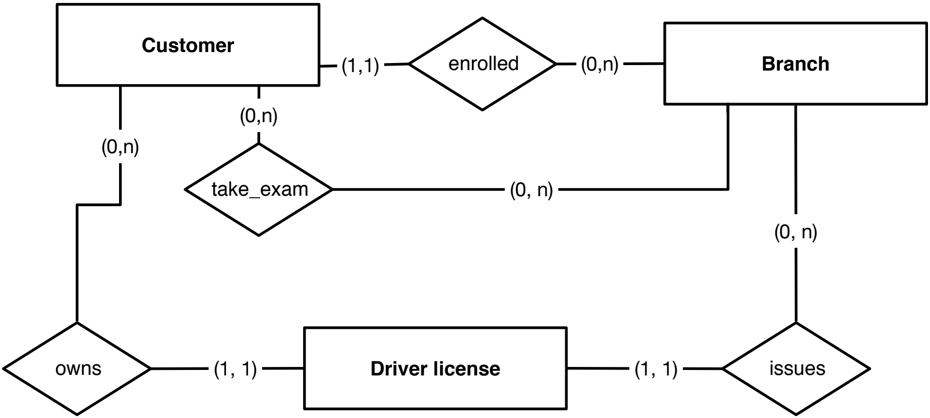
1. (0, n)
2. (1, n)
3. (1, 1)
4. (0, 0)

Cardinalities

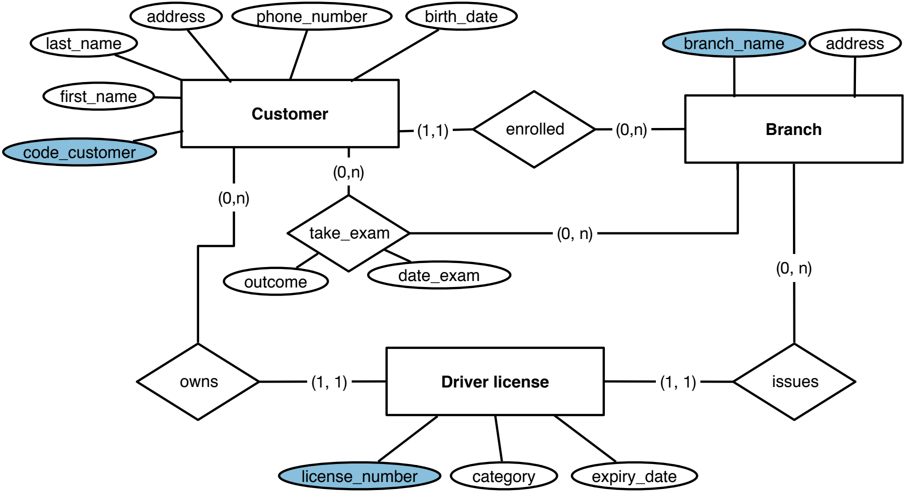
Which cardinalities on the `Branch` side?

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

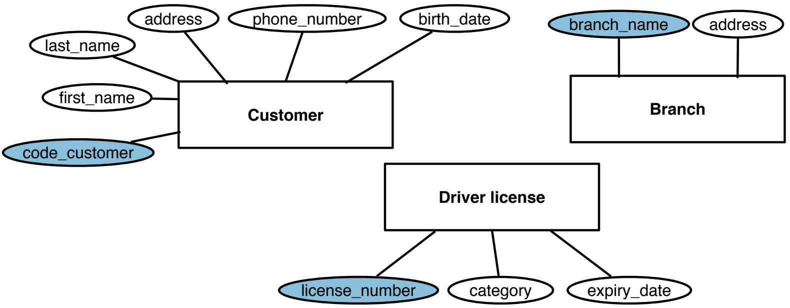
Cardinalities



Attributes

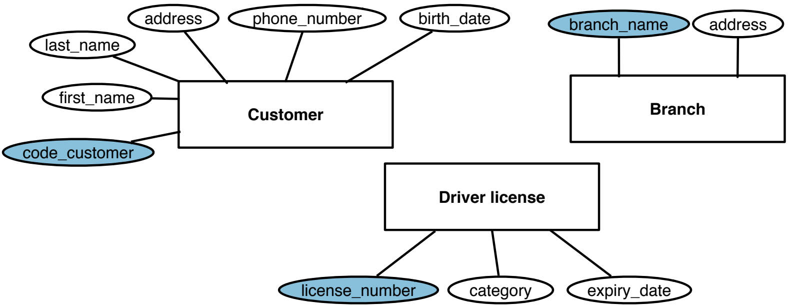

address
phone_number
birth_date
last_name
first_name
code_customer
Customer
(1,1)
enrolled
(0,n)
branch_name
address
Branch
(0,n)
(0,n)
take_exam
(0, n)
outcome
date_exam
(0, n)
owns
(1, 1)
Driver license
(1, 1)
issues
license_number
category
expiry_date

From the Logical to the Physical Model



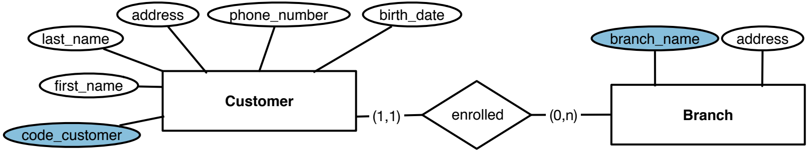
- Each **entity** is translated into a table.
- Each **attribute** of the entity is a **column** in the corresponding table.

From the Logical to the Physical Model



- `Customer`(<u>code_customer</u>, first_name, last_name, address, phone_number, birth_date)
- `Branch`(<u>branch_name</u>, address)
- `DriverLicense`(<u>license_number</u>, category, expiry_date)

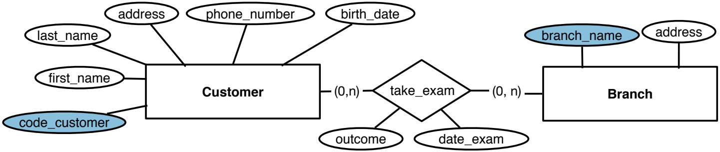
From the Logical to the Physical Model



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**)

From the Logical to the Physical Model



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>date_exam</u>, outcome)
- code_customer **foreign key** to `Customer`(code_customer).
- branch_name **foreign key** to `Branch`(branch_name).

Normalization

- **Normalization**: process by which **redundancy** is eliminated or reduced to a minimum.
- Redundancy **wastes** disk space.
  - Huge concern in the 70s (cost of 1 GB ˜$300k).
  - Less of a concern today (cost of 1 GB ˜$0.019).
- Redundancy might cause **data inconsistencies**.
  - Number of telephone of an employee in two rows.
  - When we update a row, we need to update the second.
- **Normalization theory**: based on the definition of six **normal forms**.
  - Each normal form adds some constraints to the previous.
- We'll only see the first three normal forms.

First Normal Form (1NF)

Definition (First Normal Form (1NF))

A **table** is in **first normal form** if it meets the following conditions:

1. Each row is unique (identified by a **primary key**).
2. There are **no duplicate columns**.
3. Each cell contains a **single value** (lists are not allowed).

First Normal Form (1NF)

Question. The table Book is not in 1NF. Why?

Example

```
Book (book_id, isbn, title, type, year, authors,
      publisher_name, publisher_country)

("book-001", 978-0321197849, "An introduction to database systems",
 "Hardcover", 1999, "Christopher J. Date, UK", "Addison-Wesley", "US" )

("book-001", 978-8178082318, "An introduction to database systems",
 "Paperback", 1999, "Christopher J. Date, UK", "Addison-Wesley", "US" )

("book-002", 978-3446215337, "Data Mining",
 "Paperback", 2001, "Ian H. Witten, New Zealand"; "Frank Eibe, Germany",
 "Hanser Fachbuch", "Germany" )
```

First Normal Form (1NF)

Question. The table `Book` is not in 1NF. Why?

- **Answer.** Column *authors* contains a list of values.

Example

```
Book (book_id, isbn, title, type, year, authors,
      publisher_name, publisher_country)

("book-001", 978-0321197849, "An introduction to database systems",
 "Hardcover", 1999, "Christopher J. Date, UK", "Addison-Wesley", "US" )

("book-001", 978-8178082318, "An introduction to database systems",
 "Paperback", 1999, "Christopher J. Date, UK", "Addison-Wesley", "US" )

("book-002", 978-3446215337, "Data Mining",
 "Paperback", 2001, "Ian H. Witten, New Zealand"; "Frank Eibe, Germany",
 "Hanser Fachbuch", "Germany" )
```

First Normal Form (1NF)

- We create a new table `BookAuthor` that contains the authors of the books.
- The two tables `Book` and `BookAuthor` are in 1NF.
- But there is still a lot of **redundancy**. **Where**?

Example

```
Book       (book_id, isbn, title, type, year, publisher_name,
            publisher_country)
BookAuthor (isbn, year, aut_id, aut_name, aut_country)
```

First Normal Form (1NF)

- We create a new table BookAuthor that contains the authors of the books.
- The two tables Book and BookAuthor are in 1NF.
- But there is still a lot of **redundancy**. **Where**?
  - title of the book is repeated for each edition.
  - same goes for the publisher name and country.
  - the author country is repeated for each authored book.

Example

Book (book_id, isbn, title, type, year, publisher_name, publisher_country)
BookAuthor (isbn, year, aut_id, aut_name, aut_country)

Functional Dependencies

Definition (Functional Dependency)

Given a relational table, a set of attributes $Y = \{Y_1, \ldots, Y_n\}$ is **functionally dependent** on a set of attributes $X = \{X_1, \ldots, X_m\}$, which is denoted as $X \rightarrow Y$, if any value $(x_1, \ldots, x_m)$ of X always implies value $(y_1, \ldots, y_n)$ of Y.

- Meaning of a functional dependency $X \rightarrow Y$:
  - if any two rows have the same values of X
  - then they have the same values of Y.

Functional Dependencies

Example

```
Book       (book_id, isbn, title, type, year, publisher_name,
            publisher_country)
BookAuthor (isbn, year, aut_id, aut_name, aut_country)
```

Functional dependencies in table `Book`:

- $book\_id \rightarrow title$
- $isbn \rightarrow book\_id, type, publisher\_name$
- $publisher\_name \rightarrow publisher\_country$

Functional dependencies in table `BookAuthor`:

- $aut\_id \rightarrow aut\_name, aut\_country$

Second Normal Form (2NF)

Definition (Prime attribute)

A **prime** column is one that belongs to at least one candidate key. Conversely, a **non-prime** column is one that does not belong to any candidate key.

Definition (Second Normal Form (2NF))

A table is in **second normal form** if it meets the following conditions:

- The table is in 1NF.
- All non-prime columns depend on all the columns of each candidate key.

Second Normal Form (2NF)

- Candidate key in table `Book`: {*isbn*, *year*}.
- Table `Book` is not in 2NF:
  - *isbn* → *title*; *isbn* → *type*; *isbn* → *publisher_name*.

- **Question:** table `BookAuthor` is not in 2NF. Why?

Example

```
Book        (book_id, isbn, title, type, year, publisher_name,
             publisher_country)
BookAuthor (isbn, year, aut_id, aut_name, aut_country)
```

- How do we turn this database into 2NF?

Second Normal Form (2NF)

- Candidate key in table `Book`:$\{isbn, year\}$.
- Table `Book` is not in 2NF:
  - $isbn \rightarrow title$; $isbn \rightarrow type$; $isbn \rightarrow publisher\_name$.
- **Question:** table `BookAuthor` is not in 2NF. Why?

Example

```
Book       (book_id, isbn, title, type, year, publisher_name,
            publisher_country)
BookAuthor (isbn, year, aut_id, aut_name, aut_country)
```

- How do we turn this database into 2NF?

Second Normal Form (2NF)

- Candidate key in table `Book`: $\{isbn, year\}$.
- Table `Book` is not in 2NF:
  - $isbn \rightarrow title$; $isbn \rightarrow type$; $isbn \rightarrow publisher\_name$.
- **Question:** table `BookAuthor` is not in 2NF. Why?
  - **Answer.** $isbn \rightarrow aut\_name$; $isbn \rightarrow aut\_country$.

Example

```
Book       (book_id, isbn, title, type, year, publisher_name,
            publisher_country)
BookAuthor (isbn, year, aut_id, aut_name, aut_country)
```

- How do we turn this database into 2NF?

Second Normal Form (2NF)

- The following tables are in 2NF.
- Still redundant data
  - *publisher_name* $\rightarrow$ *publisher_country*

.

Example

Book	(book_id, <u>isbn</u>, title, type, publisher_name, publisher_country)
PublicationYear	(<u>isbn</u>, <u>year</u>)
BookAuthor	(<u>isbn</u>, <u>aut_id</u>)
Author	(<u>aut_id</u>, aut_name, aut_country)

Third Normal Form (3NF)

Definition (Third Normal Form (3NF))

A table is in **third normal form** if it meets the following conditions:

- The table is in 2NF.
- No non-prime column depends on non-prime columns.

The table `Book` is not in 3NF. Why?

Example

```
Book            (book_id, isbn, title, type, publisher_name,
                 publisher_country)
```

Third Normal Form (3NF)

Example

```
Book             (book_id, title)
                 f.d.: book_id --> title
BookEdition      (isbn, book_id, type, publisher_name)
                 f.d.: isbn --> book_id, type, publisher_name
Publisher        (publisher_name, publisher_country)
                 f.d.: publisher_name --> publisher_country
PublicationYear  (isbn, year)
                 f.d.: none
BookAuthor       (isbn, aut_id)
                 f.d.: none
Author           (aut_id, aut_name, aut_country)
                 f.d.: aut_id --> aut_name, aut_contry
```

Boyce-Codd Normal Form

Another way to define 3NF

In a table in third normal form, a non-key column must provide a fact about the key, use the whole key and nothing but the key (so help me Codd!) — William Kent, 1983

- 3NF still leaves the door open to data redundancy.

Definition (Boyce-Codd Normal Form)

A table is in **Boyce-Codd normal form** (BCNF) if, for every functional dependency $X \rightarrow Y$, X is a key.

Boyce-Codd Normal Form

Example

`Course (course_name, major_name, lecturer_id)`

- A teacher only teaches one course. ($lecturer\_id \rightarrow course\_name$).
- A course can be taught in two different majors.
- Candidate keys are $\{course\_name, major\_name\}$ and $\{lecturer\_id, major\_name\}$.
- Table `Course` is not in BCNF ($lecturer\_id$ is not a superkey).
- The following example is in BCNF.

Example

`Course`	(<u>`course_name`</u>, `lecturer_id`)
`Lecturer`	(<u>`lecturer_id`</u>, <u>`major_name`</u>)

Fourth Normal Form

Definition (Fourth Normal Form)

A table is in **fourth normal form** (4NF) if it does not contain two or more independent multi-valued facts about an entity.

Example

`Employee (emp_id, emp_skill, emp_language)`

- *emp_skill* and *emp_language* are **multi-valued**.
  - An employee can have many skills and speak many languages.
- *emp_skill* and *emp_language* are **independent**.
  - Skills do not depend on the spoken languages.
- **Uncertainty** as to how the values are stored in the table.

Fourth Normal Form

Disjoint format

```
(1, "programming", NULL)
(1, "network admin", NULL)
(1, NULL, "French")
(1, NULL, "English")
(1, NULL, "German")
```

Random mix format

```
(1, "programming", "French")
(1, "network admin", "English")
(1, "network admin", "German")
```

Cross-product format

```
(1, "programming", "French")   (1, "network admin", "French")
(1, "programming", "English")  (1, "network admin", "English")
(1, "programming", "German")   (1, "network admin", "German")
```

Tables in 4NF

EmployeeSkill (<u>emp_id</u>, emp_skill)
EmployeeLanguage (<u>emp_id</u>, emp_language)

Fifth Normal Form

Definition (Fifth Normal Form)

A table is in **fifth normal form** (5NF) if its information content cannot be reconstructed from several smaller tables.

Example

`SaleRepresentative (agent, company, product)`

1. Companies make products.
2. Agents represent companies.
3. An agent representing a company sells all products made by that company.

The columns *company* and *product* are not independent. The table is in 4NF.

Fifth Normal Form

Content of the table SaleRepresentative

```
("Smith", "Ford", "car")      ("Jones", "Ford", "car")
("Smith", "Ford", "truck")    ("Jones", "Ford", "truck")
("Smith", "GM", "car")        ("Brown", "Toyota", "car")
("Smith", "GM", "truck")      ("Brown", "Toyota", "bus")
```

Tables in 5NF

AgentCompany (<u>agent</u>, <u>company</u>)
CompanyProduct (<u>company</u>, <u>product</u>)

Content of AgentCompany

```
("Smith", "Ford")
("Smith", "GM")
("Jones", "Ford")
("Brown", "Toyota")
```

Content of CompanyProduct

```
("Ford", "car")
("Ford", "truck")
("GM", "car")
("GM", "truck")
("Toyota", "car")
("Toyota", "bus")
```