Review of Concepts

Julien Romero - Télécom SudParis

Relational Algebra

General Definitions

- Attribute: Name of a column
  - Give the attributes in the following table

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

General Definitions

- Attribute: Name of a column
    - Give the attributes in the following table: passportID, name, birthdate, height

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

General Definitions

- Attribute: Name of a column
- Domain: Set of values a column can take
  - Suggest domains of the following table

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

General Definitions

- Attribute: Name of a column
- Domain: Set of values a column can take
  - Suggest domains of the following table
    - passportID: String
    - name: String
    - birthdate: Date
    - height: Int

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

General Definitions

- Attribute: Name of a column
- Domain: Set of values a column can take
- Primary key: One or several attributes that identify uniquely a row
    - What is the primary key in the following table?

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

General Definitions

- Attribute: Name of a column
- Domain: Set of values a column can take
- Primary key: One or several attributes that identify uniquely a row
  - What is the primary key in the following table?
    - passportID

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

General Definitions

- Attribute: Name of a column
- Domain: Set of values a column can take
- Primary key: One or several attributes that identify uniquely a row
- Foreign key: One or several attributes that are primary key in another table
  - What is the foreign key in the following table?

voteID	passportID	voteDate	choice
NJ6HI90	JI739HB	22/10/1970	A
NJ6HI90	PC658N	16/01/1980	B
NU9I300	PC658N	27/04/1983	A

General Definitions

- Attribute: Name of a column
- Domain: Set of values a column can take
- Primary key: One or several attributes that identify uniquely a row
- Foreign key: One or several attributes that are primary key in another table
  - What is the foreign key in the following table? passportID

voteID	passportID	voteDate	choice
NJ6HI90	JI739HB	22/10/1970	A
NJ6HI90	PC658N	16/01/1980	B
NU9I300	PC658N	27/04/1983	A

General Definitions

- Attribute: Name of a column
- Domain: Set of values a column can take
- Primary key: One or several attributes that identify uniquely a row
- Foreign key: One or several attributes that are primary key in another table
- Relation Schema: Description of a table composed of the table name, the attribute names and their domains, the primary key, and the foreign keys

General Definitions

- Relation Schema: Description of a table composed of the table name, the attribute names and their domains, the primary key, and the foreign keys
  - Give the relation schema of the following table

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

Citizens

General Definitions

- Relation Schema: Description of a table composed of the table name, the attribute names and their domains, the primary key, and the foreign keys
  - Give the relation schema of the following table
    - Citizens(passportID: String, name: String, birthdate: Date, height: Int)

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

Citizens

General Definitions

- Attribute: Name of a column
- Domain: Set of values a column can take
- Primary key: One or several attributes that identify uniquely a row
- Foreign key: One or several attributes that are primary key in another table
- Relation Schema: Description of a table composed of the table name, the attribute names and their domains, the primary key, and the foreign keys
- Database Schema: Set of all the relation schemas in the database

General Definitions

- Database Schema: Set of all the relation schemas in the database
  - Give the database schema for the following database

voteID	passportID	voteDate	choice
NJ6HI90	JI739HB	22/10/1970	A
NJ6HI90	PC658N	16/01/1980	B
NU9I300	PC658N	27/04/1983	A

Votes

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

Citizens

General Definitions

- Database Schema: Set of all the relation schemas in the database
  - Give the database schema for the following database
    - Citizens(passportID: String, name: String, birthdate: Date, height: Int)
    - Votes(voteID: String, **passportID: String**, voteDate: Date, choice: String)

voteID	passportID	voteDate	choice
NJ6HI90	JI739HB	22/10/1970	A
NJ6HI90	PC658N	16/01/1980	B
NU9I300	PC658N	27/04/1983	A

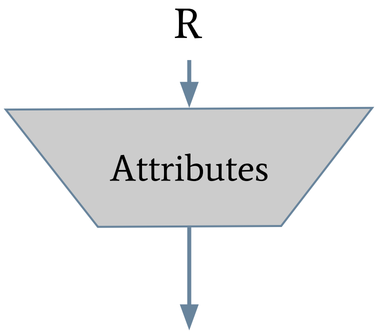
Votes

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

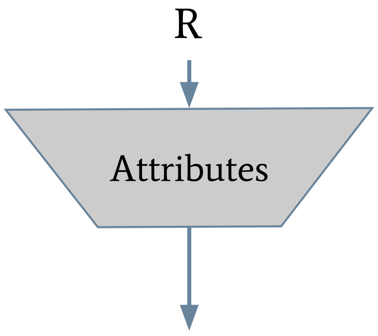
Citizens

Relational Algebra Operations

- Project: Extract the set of given columns

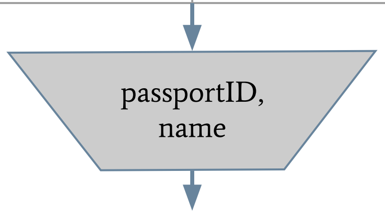


Relational Algebra Operations



- Project: Extract the set of given columns

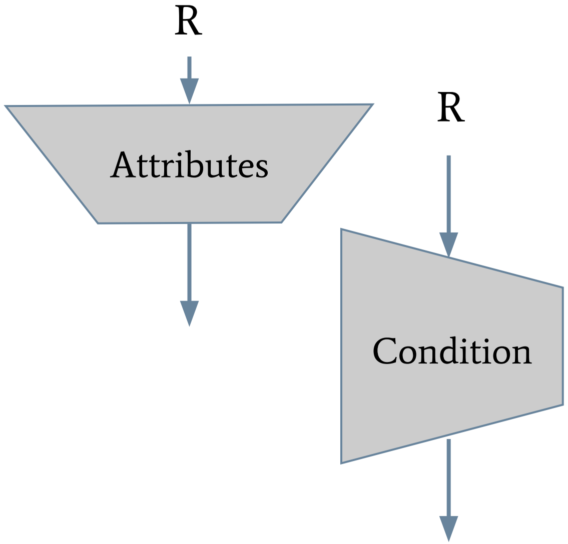
passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177



passportID	name
JI739HB	Bob Dylan
PC658N	Jimi Hendrix

Relational Algebra Operations

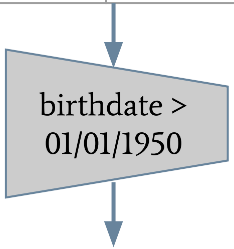
- Project: Extract the set of given columns
- Select: Filter the rows with rows with a condition



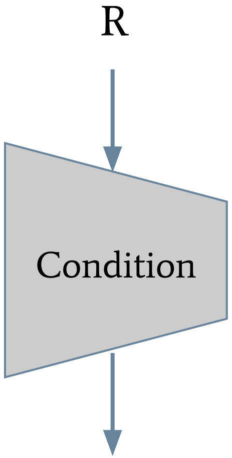
Relational Algebra Operations

- Select: Filter the rows with rows with a condition

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

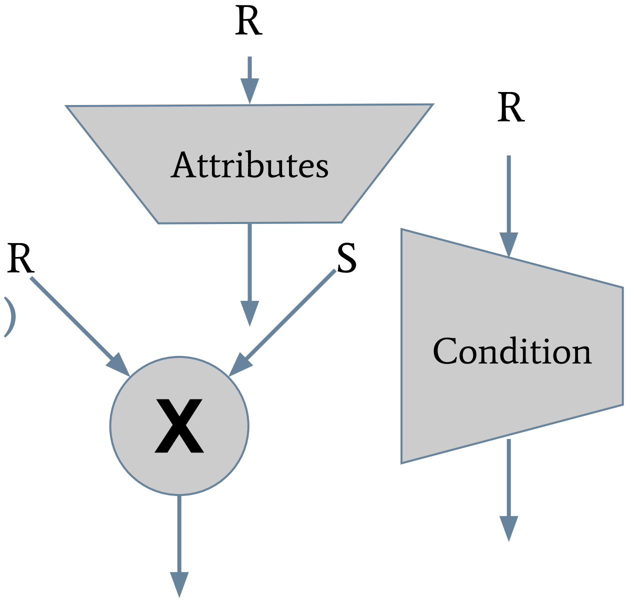


passportID	name	birthdate	height
PC658N	Jimi Hendrix	18/09/1970	177



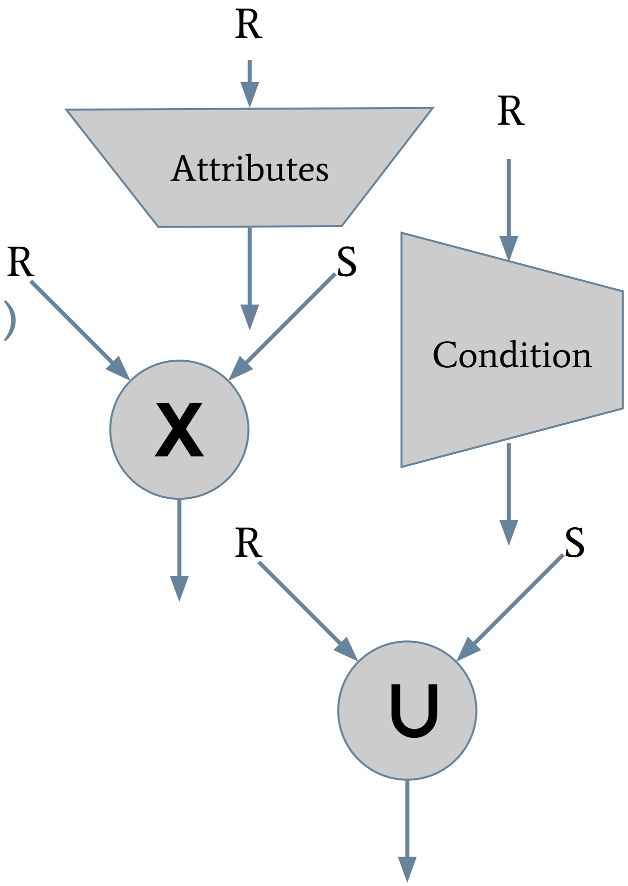
Relational Algebra Operations

- Project: Extract the set of given columns
- Select: Filter the rows with rows with a condition
- Product: Cartesian product (all combinations of rows)



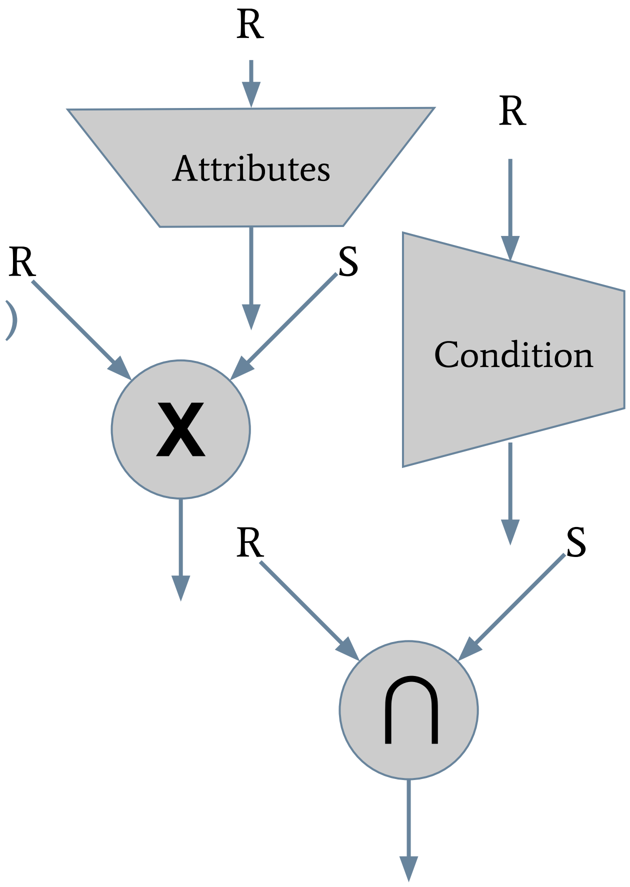
Relational Algebra Operations

- Project: Extract the set of given columns
- Select: Filter the rows with rows with a condition
- Product: Cartesian product (all combinations of rows)
- Set operations (input tables need the same schema!):
  - Union: Concatenation of the tables



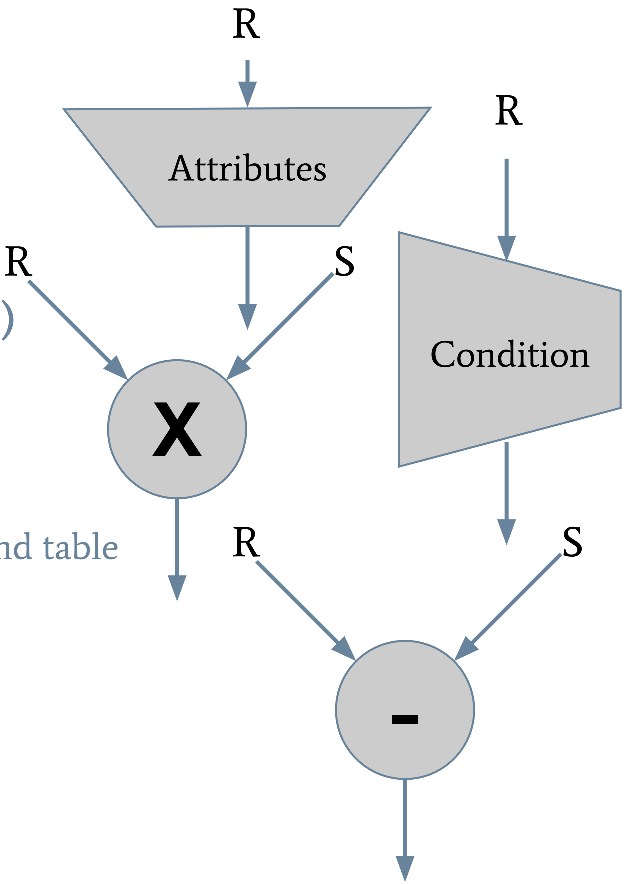
Relational Algebra Operations

- Project: Extract the set of given columns
- Select: Filter the rows with rows with a condition
- Product: Cartesian product (all combinations of rows)
- Set operations (input tables need the same schema!):
  - Union: Concatenation of the tables
  - Intersection: Rows that are in both tables



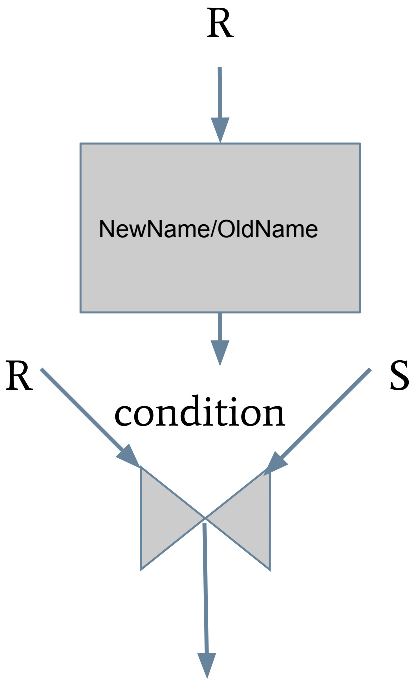
Relational Algebra Operations

- Project: Extract the set of given columns
- Select: Filter the rows with rows with a condition
- Product: Cartesian product (all combinations of rows)
- Set operations (input tables need the same schema!):
    - Union: Concatenation of the tables
    - Intersection: Rows that are in both tables
    - Difference: Removes rows in the first table that are in the second table

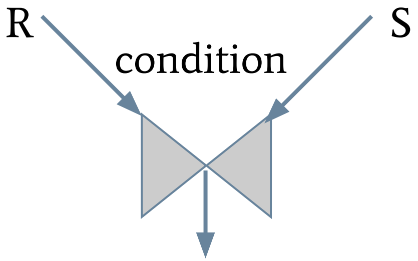


Relational Algebra Operations

- Renaming: Rename an attribute or table
- Join: Link two tables with a given condition
  - Join = Product + Select



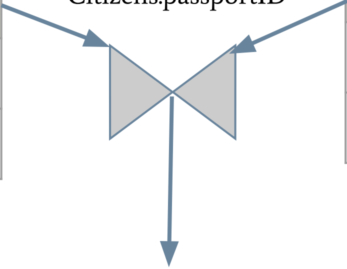
Relational Algebra Operations



- Join: Link two tables with a given condition
  - Join = Product + Select

voteID	passportID	voteDate	choice
NJ6HI90	JI739HB	22/10/1970	A
NJ6HI90	PC658N	16/01/1980	B

Votes


Votes.passportID = Citizens.passportID

passportID	name	birthdate	height
JI739HB	Bob Dylan	24/05/1941	170
PC658N	Jimi Hendrix	18/09/1970	177

Citizens

voteID	Votes.passportID	voteDate	choice	Citizens. passportID	name	birthdate	height
NJ6HI90	JI739HB	22/10/1970	A	JI739HB	Bob Dylan	24/05/1941	170
NJ6HI90	PC658N	16/01/1980	B	PC658N	Jimi Hendrix	18/09/1970	177

Practical Skill

- Know how to read a relational algebra diagram
- Answer a query by combining the operators of the relational algebra

SQL

The Final Template

```
SELECT <list of attributes/columns to select>
FROM <list of tables to consider>
[JOIN <table>
 [ON <join condition>]]*
[WHERE <condition without aggregation>]
[GROUP BY <list of columns used for grouping>
[HAVING <condition with aggregation>]]
[ORDER BY <list of column + ASC or DESC>]
[LIMIT <number of rows>];
```

Important Notes

- The keywords **SELECT**, **FROM**, and **WHERE** match operations in the relational algebra
    - **SELECT** is a projection (!) and attribute renaming
        - **PROJECT**(A1, ..., An in the table R) <-> **SELECT** A1, ..., An **FROM** R;
    - **FROM** is a cartesian product and table renaming
        - **PRODUCT**(R1, ..., Rn) <-> **SELECT** * **FROM** R1, ..., Rn;
    - **WHERE** is a selection
        - **SELECT**(condition in R) <-> **SELECT** * **FROM** R **WHERE** condition;
- <u>Semantics of a SFW query</u>: The execution of a SFW query is equivalent to a cartesian product, followed by a selection, and ending with a projection
- Get element with maximum value (e.g.: oldest date, tallest person): ORDER BY + LIMIT 1

Special SELECT

- **SELECT ***: Select all the columns
- **SELECT DISTINCT**: Select all distinct rows (no repetition)

Equivalence JOIN - WHERE

```
SELECT *
FROM Table1, Table2, Table3
WHERE join_condition1 AND join_condition2
   AND normal conditions
```

Is the same as

```
SELECT *
FROM Table1
JOIN Table2
ON join_condition1
JOIN Table3
ON join_condition2
WHERE normal conditions
```

Aggregation Function

The aggregation functions transform a list of values into a single one. SQL contains **AVG** (average), **COUNT** (number of lines, often combined with **DISTINCT**), **MAX** (maximum), **MIN** (minimum), **SUM**

Conditions with aggregations: Only in **HAVING**!

Practical Skills

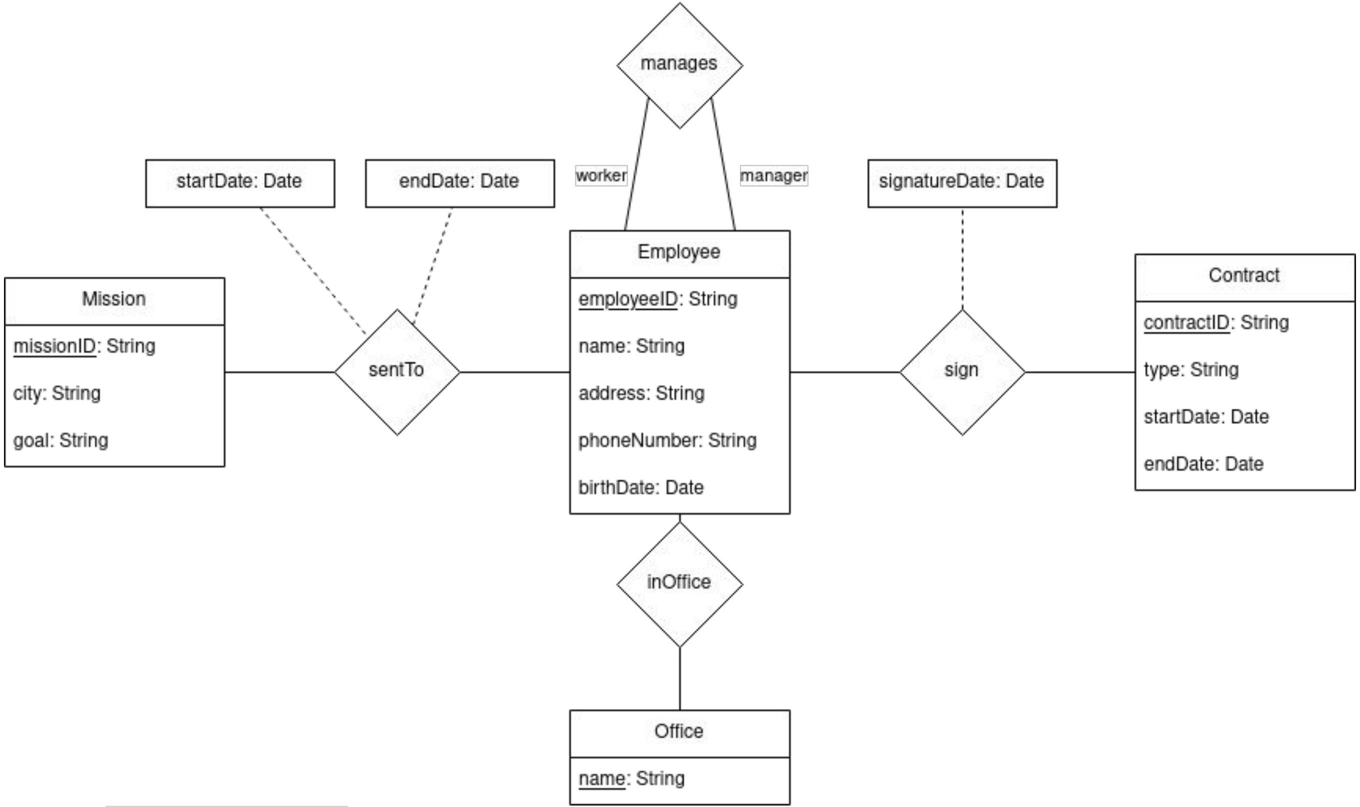
- Conversion from relational algebra to SQL
- Understanding SQL queries
- Answering queries directly in SQL

Database Design

Entity-Relationship Model - Concepts

- Entity: A physical or conceptual “thing” or “object” that can be uniquely identified.
  - It is composed of properties/attributes, which are values (numeric, String, dates, ...) that characterize an entity
  - Represented by a rectangle
- Relationship: Association between entities
  - Relationships can also have properties
- Entity Role: Label on a relationship to clarify what the linked entity represents

Entity-Relationship Model - Concepts

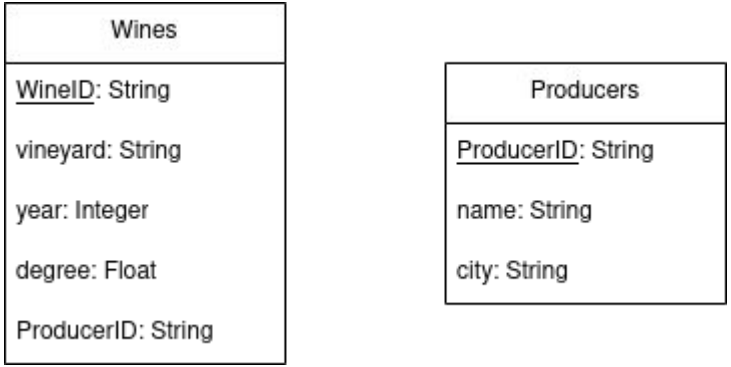


Cardinality

- Indicate in how many relations each entity is enrolled.
- Three types:
  - One-to-one
  - One-to-many
  - Many-to-many

Redundancy

- An attribute is redundant is it appears in two tables, but in fact represents a relationship.



Wines

WineID: String

vineyard: String

year: Integer

degree: Float

ProducerID: String

Producers

ProducerID: String

name: String

city: String

From E/R To Database Schema

- The translation of entities is straightforward:
  - We use the same table names, attributes, and primary keys
- For the translation of a relationship:
  - We create a new table
  - We add the attributes specific to the relationship **AND** the primary keys of all the entities.
  - We need to think about what the primary key is (depends on the cardinality)

Practical Skills

- Know how to interpret an E/R diagram and answer questions about it.
- Know how to convert an E/R diagram into a database schema.

RDBMS

Table Creation

```
CREATE TABLE table_name
(
    attribute1 data_type1,
    attribute2 data_type2,
    attribute3 data_type3,
    attribute4 data_type4,
    PRIMARY KEY (attribute1, attribute2),
   [FOREIGN KEY(column_name) REFERENCES other_table(other_column)]

)
```

Table Deletion

DROP TABLE table_name;

```
INSERT INTO table
VALUES (‘value 1', 'value 2', …),
       (‘value 1', 'value 2', …),
       (‘value 1', 'value 2', …)
```

Update table

```
UPDATE table
SET attribute1 = 'new value 1', attribute2 = 'new value 2’
WHERE condition
```

Delete Row

```
DELETE FROM table
WHERE condition
```

NoSQL

ACID Properties

- Atomicity: A transaction either completely succeeds or completely fails
- Consistency: The database respects the integrity constraints before and after each transaction
- Isolation: If two transactions happen at the same time, it is like they happen sequentially
- Durability: If a transaction is successful, the modifications are permanent, even if the system fails

Scalability

- Adapt to the change of amount of work
- Two types:
  - Vertical scaling: Replace machines by more powerful ones
  - Horizontal scaling: Add more (cheap) machines

NoSQL

- Properties
  - Non-relational: Not only tables
  - Distributed: Can be on several machines, all around the world
  - Scalable: Store and query large amount of data
  - Available: Even if a machines crashes, continues working
- Four types:
  - Tabular: Like relational databases
  - Key-Value: A table with two columns, a key and a value. Can only search by key.
  - Document: Store structured documents like JSON
  - Graph: Store graphs composed of nodes and relationships

JSON

- A file format to structure data.
- A JSON file is either a value, a list of value, or an association of value. Values are standard types (string, integer, float) and JSON files.
- We can access a value by specifying its path from the root
  - myJSON[0][“candidat”][“speciality”][1][“domain”]

We can translate a database into JSON documents, but it creates redundancy.

Graphs

- A graph links nodes (that represent entities) with edges (that represent relationships)
- We can translate our relational database into a graph by identifying the tables that represent entities and the tables that represent relationships.

Exam

Exam

- Tuesday April, 2nd, 2023, 2:30am-6:30am
- D0010-11
- No document allowed
- At least half the point will pure knowledge (MCQ or closed question)
- Practical questions like in the labs