UNIT II

The Relational Data Model
The Relational Data Model

Objectives:

- To know the data structures of the relational model: the tuple and the relation, as well as their associated operators.
- To know (basically) how to model the reality using the relational model.
- To be familiar with the algebraic approach for manipulating a database, as well as the logical perspective.
- To know the mechanisms of the relational model needed to express integrity constraints: domain definition and key definition.
- To know additional mechanisms to define constraints and express activity in databases: triggers.
The Relational Data Model

Syllabus:

Introduction
2.1.- The relational data model (algebraic approach).
   2.1.1.- Structures: tuple and relation.
   2.1.2.- Relational Schema: representation of reality.
   2.1.3.- Operators on relations: relational algebra
2.2.- Relational schema: representation of reality
2.3.- The relational data model (logical approach).
   2.3.1.- Logic and databases
   2.3.2.- Logical interpretation of a relational database.
The Relational Data Model

Syllabus: (cont’d.):

2.4.- Integrity constraints.
  2.4.1.- Constraints over attributes: domain and not null.
  2.4.2.- Uniqueness constraints.
  2.4.3.- Notion of primary key. Primary key constraint.
  2.4.4.- Referential integrity: Foreign key constraint.
  2.4.5.- Referential triggered action: action directives.
  2.4.6.- Other mechanisms to represent integrity constraints.
The Relational Data Model

Syllabus: (cont’d.):

2.5.- SQL – The Relational Database Standard.
   2.5.1.- The Data Definition Language (DDL).
   2.5.2.- The Data Manipulation Language (DML).
      2.5.2.1 INSERT, DELETE and UPDATE.
      2.5.2.2 Logical approach in the SELECT clause.
      2.5.2.3 Algebraic approach in the SELECT clause
   2.6.- Derived information: views
      2.6.1.- Notion of view.
      2.6.2.- Applications.
      2.6.3.- Views in SQL.
The Relational Data Model

Syllabus: (cont’d.):

2.7.- Activity mechanisms: triggers.
   2.7.1.- Notion of trigger.
   2.7.2.- Event-Condition-Action (ECA) rules
   2.7.3.- Applications
   2.7.4.- Triggers in SQL.

2.8.- Evolution of the relational model
2.- Introduction to the Relational Data Model

- Historical milestones about the Relational Data Model (RDM):
  - 70’s: Proposed by E. Codd in 1970
  - 80’s: Becomes popular in practice (Oracle, ...). ANSI defines the SQL standard.
  - 90’s: Generalisation and standardisation (SQL’92) and extensions.

Reasons of success:

Simplicity: a database is a “set of tables”.
2.- The RDM: Components and Approaches

RDM = Data structures + Associated operators

Common data structures:
- domains
- attributes
- the tuple
- the relation.

Two Operator Families:
- Algebraic
  - R.A.
- Logical
  - T.R.C
  - D.R.C.
2.- The RDM: Terminology

Data structures:

<table>
<thead>
<tr>
<th>Common Terminology (computing)</th>
<th>RDM Terminology (mathematical)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• data types</td>
<td>• domains</td>
</tr>
<tr>
<td>• fields / columns</td>
<td>• attributes</td>
</tr>
<tr>
<td>• record / row</td>
<td>• tuple</td>
</tr>
<tr>
<td>• table</td>
<td>• relation</td>
</tr>
</tbody>
</table>

They are not *exactly* equivalent
2.1.- The RDM: Algebraic Approach

- The algebraic approach sees tables as sets, and the set of operators working with them as an algebra.
2.1.1.- Notion of tuple

**Tuple schema:**

A tuple schema, \( \tau \), is a set of pairs of the form:

\[
\tau = \{(A_1, D_1), (A_2, D_2), \ldots, (A_n, D_n)\}
\]

where:

\( \{A_1, A_2, \ldots, A_n\} \ (n > 0) \) is the set of attribute names in the schema, necessarily different.

\( D_1, D_2 \ldots, D_n \) are the domains associated with the above-mentioned attributes, which not necessarily have to be different.
2.1.1.- Notion of tuple

Example of **tuple schema**:

\[
\text{Person} = \{(\text{person\_id, integer}), (\text{name, string}), (\text{address, string})\}
\]

where:

\{
\text{person\_id, name, address}
\} is the set of attribute names in the schema.

integer, string, string are the domains which are associated with the attributes.
2.1.1.- Notion of tuple

**Tuple:**

A tuple, $t$, of tuple schema $\tau$ where

$$\tau = \{(A_1, D_1), (A_2, D_2), \ldots, (A_n, D_n)\}$$

is a set of pairs of the form:

$$t = \{(A_1, v_1), (A_2, v_2), \ldots, (A_n, v_n)\}$$

such that $\forall i \ v_i \in D_i$. 
2.1.1.- Notion of tuple

Examples of *Tuple*:

Given the following tuple schema:

```
Person = \{(\text{person}_\text{id}, \text{integer}), (\text{name}, \text{string}), (\text{address}, \text{string})\}
```

We have:

```
t_1 = \{(\text{person}_\text{id}, 2544), (\text{name}, \text{"Joan Roig"}), (\text{address}, \text{"Sueca 15"})\}
```

```
t_2 = \{(\text{person}_\text{id}, \text{"2844F"}), (\text{name}, \text{"R3PO"}), (\text{address}, \text{"46022"})\}
```

```
t_3 = \{(\text{name}, \text{"Pep Blau"}), (\text{person}_\text{id}, 9525), (\text{address}, \text{"dunno!"})\}
```
2.1.1.- Domains

**PROBLEM:** What happens if we don’t know the value a tuple takes in some of its attributes?

*Solution in Programming Languages:* use of special or extreme values (-1, “Empty”, “”, “We don’t know”, 0, “No address”, “---”, ...)

*Solution in the Relational Model:* NULL VALUE (?)

A **Domain** is something more than a datatype:

A domain is a set of elements which always includes the NULL value.
2.1.1.- Tuple Operators

Given tuple: \( t = \{(A_1, v_1), \ldots, (A_i, v_i), \ldots, (A_n, v_n)\} \)

GET:

- \( \text{GET}(t, A_i) = v_i \)

SET:

- \( \text{SET}(t, A_i, w_i) = \{(A_1, v_1), \ldots, (A_i, w_i), \ldots, (A_n, v_n)\} \)

Usual notation

- \( \text{GET}(t, A_i): \quad t.A_i \quad t(A_i) \)
- \( \text{SET}(t, A_i, w_i): \quad t.A_i \leftarrow w_i \quad t(A_i) \leftarrow w_i \)
### 2.1.1.- Example

Given the domain: 

- id_dom: integer
- name_dom, add_dom: string(20)

Tuple schema:

Person = {(person_id, id_dom), (name, name_dom), (address, add_dom)}

Tuples:

- \( t_1 = \{(person\_id, 12345678), (name, \text{“Pepa Gómez”}), (address, \text{“Paz 10”})\} \)
- \( t_2 = \{(name, \text{“Pep Blau”}), (person\_id, 9525869), (address, \text{?}) \} \)

Operations:

\[
\begin{align*}
\text{GET}(t_1, \text{name}) = &\quad \text{“Pepa Gómez”} \\
\text{SET} (t_1, \text{address}, \text{“Colón 15”}) = &\quad \{(person\_id, 12.345.678), (name, \text{“Pepa Gómez”}), (address, \text{“Colón 15”})\} \\
\text{GET} (t_2, \text{address}) = &\quad \text{?}
\end{align*}
\]

*We say that \( t_2\_\text{address} \) is null, not that \( t_2\_\text{address} = \text{null} \).*
2.1.1.- Notion of relation (algebraic)

**Relation**:  
A relation is a set of tuples of the same schema.

**Relation schema**  
A relation schema is the schema of the tuples composing the relation.

**Notation**  
\[ R(A_1: D_1, A_2: D_2, \ldots, A_n: D_n) \]  
Defines a relation \( R \) of schema  
\[ \{(A_1, D_1), (A_2, D_2), \ldots, (A_n, D_n)\} \]
2.1.1.- Properties of a relation

Properties of a relation

- *Degree of a relation*: number of attributes of its schema

- *Cardinality of a relation*: number of tuples that compose the relation.

- *Compatibility*: two relations $R$ and $S$ are compatible if their schemas are identical.
2.1.1.- Example of relation

Example:

A relation of the PERSON schema might be as follows:

\{
  \{(person_id, 1234), (name, “Pepa Gómez”), (address, “Colón 15”)}
, 
  \{(person_id, 2045), (name, “Juan Pérez”), (address, “Cuenca 20”)}
, 
  \{(name, “José Abad”), (person_id, 1290), (address, “Blasco Ibáñez 35”)}
, 
  \{(name, “María Gutiérrez”), (person_id, 35.784.843) (address, “Reina 7”)}
\}

Degree:
Cardinality:
Compatible with:
### 2.1.1.- Representation of a relation

**Representation of a relation → TABLE**

- tuples are represented as rows
- attributes give name to the column headers

**Example: PERSON relation**

<table>
<thead>
<tr>
<th>Person_id</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>2045</td>
<td>Juan Pérez</td>
<td>Cuenca 20</td>
</tr>
<tr>
<td>1290</td>
<td>José Abad</td>
<td>Blasco Ibáñez 35</td>
</tr>
<tr>
<td>3578</td>
<td>María Gutiérrez</td>
<td>Reina 7</td>
</tr>
<tr>
<td>1234</td>
<td>Pepa Gómez</td>
<td>Colón 15</td>
</tr>
</tbody>
</table>

Row ≈ Tuple

Column ≈ Attribute
2.1.1.- Difference Relation - Table

*The Table is only a Matrix Representation of a Relation*

**TRAITS WHICH DISTINGUISH A RELATION:**

(Derived from the definition of relation as a set of sets)

- There can’t be repeated tuples in a relation (a relation is a set).
- There isn’t a top-down order in the tuples (a relation is a set).
- There isn’t a left-to-right order in the attributes of a relation (a tuple is a set). The name of the attribute must be used to choose.
2.1.1.- Difference Extension - Schema

EXTENSION (data)                                                                 SCHEMA

Tuple

Tuple schema = Relation definition

(Extension of a) relation: set of tuples in a relation

Database:
set of relations

Relation Schema: set of relation definitions which represent an information system

Attention!: DBMSs understand a table as the definition of a relation and not as its content, which eventually changes by applying operators.
2.1.1.- Relation Operators

Operators for the Relation Structure:

• INSERTION
• DELETION
• SELECTION
• PROJECTION
• UNION
• INTERSECTION
• DIFFERENCE
• CARTESIAN PRODUCT
• JOIN

Also part of the R.A.
2.1.1.- Insertion

\[
\text{Insert}(R, t) = R \cup \{ t \}
\]

\(R\) and \(t\) must have the same schema.

Example:

\[
\text{Insert}\{
\{(\text{person\_id}, 12.345.678), (\text{name}, “Pepa Gómez”), (\text{address}, “Colón 15”)}\},
\{(\text{person\_id}, 20.450.120), (\text{name}, “Juan Pérez”), (\text{address}, “Cuenca 20”)}\},
\{(\text{name}, “María Gutiérrez”), (\text{person\_id}, 35.784.843) (\text{address}, “Reina 7”)}\},
\{(\text{name}, “José Abad”), (\text{person\_id}, 12.904.569), (\text{address}, “Blasco Ibáñez 35”)\})
= \\
\{(\text{person\_id}, 12.345.678), (\text{name}, “Pepa Gómez”), (\text{address}, “Colón 15”)}\},
\{(\text{person\_id}, 20.450.120), (\text{name}, “Juan Pérez”), (\text{address}, “Cuenca 20”)}\},
\{(\text{person\_id}, 12.904.569), (\text{name}, “José Abad”), (\text{address}, “Blasco Ibáñez 35”)\},
\{(\text{name}, “María Gutiérrez”), (\text{person\_id}, 35.784.843) (\text{address}, “Reina 7”)}\}
\]

Question: How does insertion affect …?:

degree:

cardinality:
2.1.1.- Deletion

\[
\text{Delete}(R, t) = R - \{ t \}
\]

\(R\) and \(t\) must have the same schema

Example:

Delete\{ 
\{(person_id, 12.345.678), (name, “Pepa Gómez”), (address, “Colón 15”)\},
\{(person_id, 20.450.120), (name, “Juan Pérez”), (address, “Cuenca 20”) \},
\{(person_id, 12.904.569), (name, “José Abad”), (address, “Blasco Ibáñez 35”)\},
\{(name, “María Gutiérrez”), (person_id, 35.784.843) (address, “Reina 7”)\}
\}

\{ (name, “José Abad”), (person_id, 12.904.569), (address, “Blasco Ibáñez 35”)\}

= 
\{ 
\{(person_id, 12.345.678), (name, “Pepa Gómez”), (address, “Colón 15”)\},
\{(person_id, 20.450.120), (name, “Juan Pérez”), (address, “Cuenca 20”) \},
\{(name, “María Gutiérrez”), (person_id, 35.784.843) (address, “Reina 7”)\}
\}

Question: How does deletion affect …?:

Degree:
Cardinality:
2.1.2.- Relational Algebra (R.A.)

R.A.: Set of unitary or binary operators which act upon relations

- They are *closed* operators: the result of applying any R.A. operator over one or two relations is a relation.

- **Set operators:**
  - union,
  - intersection,
  - difference, and
  - Cartesian product.

- **Properly relational operators:**
  - selection,
  - projection,
  - division, and
  - join.

- **Special operator:** *rename*
2.1.2.- R.A. (Rename Operator)

Let $R$ be a relation of schema $\{(A_1, D_1), (A_2, D_2), \ldots, (A_n, D_n)\}$. Renaming in $R$ the attributes $A_i, \ldots, A_j$ to $B_i, \ldots, B_j$, denoted as $R((A_i, B_i), \ldots, (A_j, B_j))$, produces a relation which contains each tuple in $R$, but changing their attribute names appropriately.

$$R((A_i, B_i), \ldots, (A_j, B_j)) = \\{ \{(A_1, v_1), \ldots, (B_i, v_i), \ldots, (B_j, v_j), \ldots, (A_n, v_n)\} \mid \{(A_1, v_1), \ldots, (A_i, v_i), \ldots, (A_j, v_j), \ldots, (A_n, v_n)\} \in R \}$$

The schema of the resulting relation is the following:

$$\{(A_1, D_1), \ldots, (B_i, D_i), \ldots, (B_j, D_j), \ldots, (A_n, D_n)\}.$$
Example:
Consider the following schema of a relational database:

River (rcode: rcode_dom, name: name_dom)
Other_Rivers (rcode: rcode_dom, name: name_dom)
Province (pcode: pcode_dom, name: name_dom)
Crosses (pcode: pcode_dom, rcode: rcode_dom)

Question:
How would we rename the relation Crosses so that the attribute 
*pcode* becomes *ProvCode* and *rcode* becomes *RiverCode*?
2.1.2.- R.A. (Rename Operator)

The rename operator is applied over relations.

*NOT over relation schemas*

Example:

Let *Crosses* be a relation represented by the following table:

<table>
<thead>
<tr>
<th>pcode</th>
<th>rcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>r2</td>
</tr>
<tr>
<td>46</td>
<td>r2</td>
</tr>
<tr>
<td>45</td>
<td>r1</td>
</tr>
<tr>
<td>28</td>
<td>r1</td>
</tr>
<tr>
<td>16</td>
<td>r1</td>
</tr>
</tbody>
</table>

*Crosses*(((pcode, ProvCode), (rcode, RiverCode))) =

<table>
<thead>
<tr>
<th>ProvCode</th>
<th>RiverCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>r2</td>
</tr>
<tr>
<td>46</td>
<td>r2</td>
</tr>
<tr>
<td>45</td>
<td>r1</td>
</tr>
<tr>
<td>28</td>
<td>r1</td>
</tr>
<tr>
<td>16</td>
<td>r1</td>
</tr>
</tbody>
</table>
2.1.2.- R.A. (Set Operators)

Union: \( R \cup S \)

Intersection: \( R \cap S \)

Difference: \( R - S \)

Product: \( R \times S \)
Let $R$ and $S$ be two compatible relations with schema $\{(A_1, D_1), \ldots, (A_n, D_n)\}$. The union of $R$ and $S$, denoted by $R \cup S$, is a relation with the same schema as $R$ and $S$, and is composed of all the tuples which belong to $R$, to $S$, or to both relations.

$$R \cup S = \{ t \mid t \in R \lor t \in S\}$$

The union is associative and commutative

$R$ and $S$ must have the same schema
2.1.2.- R.A. (Union Operator)

Example:

```
pcode | Name
44    | Teruel
46    | Valencia
16    | Cuenca
12    | Castellón
```

```
44    | Teruel
46    | Valencia
16    | Cuenca
```

```
45    | Toledo
28    | Madrid
12    | Castellón
```

∪

```
45    | Toledo
28    | Madrid
12    | Castellón
```

=
2.1.2.- R.A. (Difference Operator)

Let $R$ and $S$ be compatible relations with schema $\{(A_1, D_1), \ldots, (A_n, D_n)\}$. The difference between $R$ and $S$, denoted by $R - S$, is a relation with the same schema as $R$ and $S$, and is composed by all the tuples which belong to $R$ and do not belong to $S$.

$$R - S = \{ t | t \in R \land t \notin S \}$$

The difference is neither associative nor commutative.
2.1.2.- R.A. (Difference Operator)

Example:

<table>
<thead>
<tr>
<th>pcode</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>Teruel</td>
</tr>
<tr>
<td>46</td>
<td>Valencia</td>
</tr>
<tr>
<td>16</td>
<td>Cuenca</td>
</tr>
<tr>
<td>12</td>
<td>Castellón</td>
</tr>
<tr>
<td>45</td>
<td>Toledo</td>
</tr>
<tr>
<td>28</td>
<td>Madrid</td>
</tr>
<tr>
<td>12</td>
<td>Castellón</td>
</tr>
</tbody>
</table>

\[
\begin{array}{|c|c|}
\hline
44 & Teruel \\
46 & Valencia \\
16 & Cuenca \\
12 & Castellón \\
45 & Toledo \\
28 & Madrid \\
12 & Castellón \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
16 & Cuenca \\
45 & Toledo \\
28 & Madrid \\
12 & Castellón \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
44 & Teruel \\
46 & Valencia \\
12 & Castellón \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
16 & Cuenca \\
45 & Toledo \\
28 & Madrid \\
12 & Castellón \\
\hline
\end{array}
\]
2.1.2.- R.A. (Intersection Operator)

Let $R$ and $S$ be two compatible relations with schema $\{(A_1, D_1), \ldots, (A_n, D_n)\}$. The intersection of $R$ and $S$, denoted by $R \cap S$, is a relation with the same schema as $R$ and $S$, and is composed by all the tuples which belong to $R$ and to $S$.

$$R \cap S = \{ t \mid t \in R \land t \in S \}$$

The intersection is associative and commutative.
2.1.2.- R.A. (Intersection Operator)

Example:

\[
P_{\text{Teruel}} \cap P_{\text{Cuenca}} = P_{\text{Cuenca}}
\]

<table>
<thead>
<tr>
<th>pcode</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>Teruel</td>
</tr>
<tr>
<td>46</td>
<td>Valencia</td>
</tr>
<tr>
<td>16</td>
<td>Cuenca</td>
</tr>
<tr>
<td>12</td>
<td>Castellón</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pcode</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Cuenca</td>
</tr>
<tr>
<td>45</td>
<td>Toledo</td>
</tr>
<tr>
<td>28</td>
<td>Madrid</td>
</tr>
<tr>
<td>12</td>
<td>Castelló</td>
</tr>
</tbody>
</table>
Let $R$ and $S$ be two relations with schemas $\{(A_1, D_1), \ldots, (A_n, D_n)\}$ and $\{(B_1, E_1), \ldots, (B_m, E_m)\}$ respectively such that they do not have any attribute name in common. The Cartesian product of $R$ and $S$, denoted by $R \times S$, is a relation whose schema is the union of the schemas from $R$ and $S$, and is composed by all the tuples which can be constructed by combining one from $R$ and one from $S$.

$$R \times S = \{ \{(A_1, v_1), \ldots, (A_n, v_n), (B_1, w_1), \ldots, (B_m, w_m)\} \mid \{(A_1, v_1), \ldots, (A_n, v_n)\} \in R \text{ and } \{(B_1, w_1), \ldots, (B_m, w_m)\} \in S \}$$

The schema of the resulting relation from $R \times S$ is $\{(A_1, D_1), \ldots, (A_n, D_n), (B_1, E_1), \ldots, (B_m, E_m)\}$. The Cartesian product is associative and commutative.
2.1.2.- R.A. (Cartesian Product Operator)

Example:

<table>
<thead>
<tr>
<th>pcode</th>
<th>ProvName</th>
<th>rcode</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>Teruel</td>
<td>r1</td>
<td>Sénia</td>
</tr>
<tr>
<td>44</td>
<td>Teruel</td>
<td>r2</td>
<td>Túria</td>
</tr>
<tr>
<td>44</td>
<td>Teruel</td>
<td>r3</td>
<td>Xúquer</td>
</tr>
<tr>
<td>46</td>
<td>Valencia</td>
<td>r1</td>
<td>Sénia</td>
</tr>
<tr>
<td>46</td>
<td>Valencia</td>
<td>r2</td>
<td>Túria</td>
</tr>
<tr>
<td>46</td>
<td>Valencia</td>
<td>r3</td>
<td>Xúquer</td>
</tr>
<tr>
<td>16</td>
<td>Cuenca</td>
<td>r1</td>
<td>Sénia</td>
</tr>
<tr>
<td>16</td>
<td>Cuenca</td>
<td>r2</td>
<td>Túria</td>
</tr>
<tr>
<td>16</td>
<td>Cuenca</td>
<td>r3</td>
<td>Xúquer</td>
</tr>
<tr>
<td>12</td>
<td>Castellón</td>
<td>r1</td>
<td>Sénia</td>
</tr>
<tr>
<td>12</td>
<td>Castellón</td>
<td>r2</td>
<td>Túria</td>
</tr>
<tr>
<td>12</td>
<td>Castellón</td>
<td>r3</td>
<td>Xúquer</td>
</tr>
</tbody>
</table>
2.1.2.- R.A. (Relational Operators)

Selection

Join

Projection

Division

\[
\begin{array}{ccc}
\text{a1} & \text{b1} & \text{c1} \\
\text{a2} & \text{b1} & \text{c2} \\
\text{a3} & \text{b2} & \text{c3}
\end{array}
\ \bowtie
\ \begin{array}{ccc}
\text{a1} & \text{b1} & \text{c1} \\
\text{a2} & \text{b2} & \text{c2} \\
\text{a3} & \text{b3} & \text{c3}
\end{array}
= \begin{array}{ccc}
\text{a1} & \text{b1} & \text{c1} \\
\text{a2} & \text{b1} & \text{c2} \\
\text{a3} & \text{b2} & \text{c2}
\end{array}
\]

\[
\begin{array}{ccc}
a & x \\
a & y \\
b & x \\
c & y
\end{array}
\div
\begin{array}{ccc}
x \\
y
\end{array}
= \begin{array}{ccc}
a
\end{array}
\]
Let $R$ be a relation with schema \{(A_1, D_1),\ldots, (A_n, D_n)\} and let \{A_i, A_j,\ldots, A_k\} be a subset of the attribute names in $R$ with $m$ elements ($1 \leq m \leq n$). The projection of $R$ over \{A_i, A_j,\ldots, A_k\}, denoted by $R[A_i, A_j,\ldots, A_k]$, is a relation which is defined as follows:

\[
R[A_i, A_j,\ldots, A_k] = \{ \{(A_i, v_i), (A_j, v_j),\ldots, (A_k, v_k)\} \mid \exists t \in R \text{ such that } \{(A_i, v_i), (A_j, v_j),\ldots,(A_k, v_k)\} \subseteq t \}
\]

The relation schema of $R[A_i, A_j,\ldots, A_k]$ is

\[
\{(A_i, D_i), (A_j, D_j),\ldots,(A_k, D_k)\}.
\]
2.1.2.- R.A. (Projection Operator)

Example:

Let R be

<table>
<thead>
<tr>
<th>person_id</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.450.120</td>
<td>Juan Pérez</td>
<td>Cuenca 20</td>
</tr>
<tr>
<td>12.904.569</td>
<td>José Abad</td>
<td>Blasco Ibáñez 35</td>
</tr>
<tr>
<td>35.784.843</td>
<td>María Gutiérrez</td>
<td>Reina 7</td>
</tr>
<tr>
<td>12.345.678</td>
<td>Pepa Gómez</td>
<td>Colón 15</td>
</tr>
</tbody>
</table>

R[person_id, address] =

<table>
<thead>
<tr>
<th>person_id</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.450.120</td>
<td>Cuenca 20</td>
</tr>
<tr>
<td>12.904.569</td>
<td>Blasco Ibáñez 35</td>
</tr>
<tr>
<td>35.784.843</td>
<td>Reina 7</td>
</tr>
<tr>
<td>12.345.678</td>
<td>Colón 15</td>
</tr>
</tbody>
</table>
2.1.2.- R.A. (Join Operator)

Let $R$ and $S$ be two relations with schemas \{(A_1, D_1),\ldots, (A_n, D_n), (B_1, E_1),\ldots,(B_m,E_m)\}$ and \{(B_1,E_1),\ldots,(B_m,E_m), (C_1,F_1),\ldots,(C_p,F_p)\}, respectively, in such a way that $B_1,\ldots, B_m$ are the common attributes in both schemas. The join of $R$ and $S$, denoted by $R \bowtie S$, is a relation which contains all the tuples which can be constructed by combining a tuple from $R$ with another from $S$ such that they have the same value for every common attribute name.

\[
R \bowtie S = \{(A_1, v_1),\ldots,(A_n, v_n),(B_1, w_1),\ldots,(B_m, w_m), (C_1, y_1),\ldots,(C_p, y_p)\} | \\
\{ (A_1, v_1),\ldots, (A_n, v_n), (B_1, w_1),\ldots, (B_m, w_m) \} \in R \land \\
\{ (B_1, w_1),\ldots, (B_m, w_m), (C_1, y_1),\ldots, (C_p, y_p) \} \in S 
\]

The join operator is associative and commutative. The resulting relation schema is \{(A_1, D_1),\ldots, (A_n, D_n), (B_1, E_1),\ldots, (B_m, E_m), (C_1, F_1),\ldots, (C_p, F_p)\}. 
2.1.2.- R.A. (Join Operator)

Example:

<table>
<thead>
<tr>
<th>pcode</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>Teruel</td>
</tr>
<tr>
<td>46</td>
<td>Valencia</td>
</tr>
<tr>
<td>16</td>
<td>Cuenca</td>
</tr>
<tr>
<td>12</td>
<td>Castellón</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pcode</th>
<th>rcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>r1</td>
</tr>
<tr>
<td>46</td>
<td>r2</td>
</tr>
<tr>
<td>30</td>
<td>r2</td>
</tr>
<tr>
<td>20</td>
<td>r1</td>
</tr>
<tr>
<td>44</td>
<td>r3</td>
</tr>
<tr>
<td>12</td>
<td>r1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pcode</th>
<th>name</th>
<th>rcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>Teruel</td>
<td>r1</td>
</tr>
<tr>
<td>46</td>
<td>Valencia</td>
<td>r2</td>
</tr>
<tr>
<td>44</td>
<td>Teruel</td>
<td>r3</td>
</tr>
<tr>
<td>12</td>
<td>Castellón</td>
<td>r1</td>
</tr>
</tbody>
</table>
2.1.2.- R.A. (Join Operator)

More examples:

<table>
<thead>
<tr>
<th>pcode</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>Terol</td>
</tr>
<tr>
<td>46</td>
<td>València</td>
</tr>
<tr>
<td>16</td>
<td>Conca</td>
</tr>
<tr>
<td>12</td>
<td>Castelló</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pcode</th>
<th>rcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>r1</td>
</tr>
<tr>
<td>50</td>
<td>r2</td>
</tr>
<tr>
<td>30</td>
<td>r2</td>
</tr>
</tbody>
</table>

\[=\]

<table>
<thead>
<tr>
<th>pcode</th>
<th>Name</th>
<th>rcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>Terol</td>
<td>44</td>
</tr>
<tr>
<td>44</td>
<td>Terol</td>
<td>50</td>
</tr>
<tr>
<td>46</td>
<td>València</td>
<td>44</td>
</tr>
<tr>
<td>46</td>
<td>València</td>
<td>50</td>
</tr>
<tr>
<td>16</td>
<td>Conca</td>
<td>44</td>
</tr>
<tr>
<td>16</td>
<td>Conca</td>
<td>50</td>
</tr>
</tbody>
</table>
2.1.2.- R.A. (Division Operator)

Let $R$ and $S$ be two relations with schemas $\{(A_1, D_1), \ldots, (A_n, D_n), (B_1, E_1), \ldots, (B_m, E_m)\}$ and $\{(B_1, E_1), \ldots, (B_m, E_m)\}$ respectively. The division of $R$ by $S$, denoted by $R \div S$, is a relation defined as follows:

$$R \div S = \{ \{(A_1, v_1), \ldots, (A_n, v_n)\} \mid \forall s \in S \ (s = \{(B_1, w_1), \ldots, (B_m, w_m)\}) \rightarrow \exists t \in R \text{ and } t = \{(A_1, v_1), \ldots, (A_n, v_n), (B_1, w_1), \ldots, (B_m, w_m)\} \}$$

The schema of $R \div S$ is $\{(A_1, D_1), \ldots, (A_n, D_n)\}$. The division operator is neither associative nor commutative.
Let $R$ be a relation of schema $\{(A_1, D_1), \ldots, (A_n, D_n)\}$. The selection in $R$ with respect to the condition $F$, denoted by $R \text{ WHERE } F$, is a relation of the same schema $R$, which is composed by all the tuples in $R$ such that condition $F$ holds.

$$R \text{ WHERE } F = \{ \ t \mid t \in R \text{ and } F(t) \text{ has value } true \}$$

What is the condition $F(t)$ like?

How is $F(t)$ evaluated?
2.1.2.- R.A. (Selection Operator)

What is the condition $F$ like?

Types of Comparison:
- Null($A_i$)
- $A_i \alpha A_j$
- $A_i \alpha a$

where $\alpha$ is a comparison operator ($<$, $>$, $\le$, $\ge$, $=$, $\neq$), $A_i$ and $A_j$ are attribute names and $a$ is a value from the domain associated with attribute $A_i$, different from the null value.

The Conditions are constructed from comparisons, using parentheses and logical operators ($\lor$, $\land$, $\neg$).
2.1.2.- R.A. (Selection Operator)

How is the condition $F(t)$ evaluated?

Null Value $\Rightarrow$ Need for a Trivalued Logic $\{T, F, \text{undefined}\}$:

- if $F$ is of the form $A_i \alpha A_j$ then $F(t)$ is evaluated as undefined if at least one of $A_i$ or $A_j$ has null value in $t$; otherwise it is evaluated to the certainty value of the comparison $t(A_i) \alpha t(A_j)$;

- if $F$ is of the form $A_i \alpha a$ then $F(t)$ is evaluated as undefined if $A_i$ has null value in $t$; otherwise it is evaluated to the certainty value of the comparison $t(A_i) \alpha a$; and

- if $F$ is of the form $\text{null}(A_i)$ then $F(t)$ is evaluated as true if $A_i$ has null value in $t$; otherwise it is evaluated to false.
2.1.2.- R.A. (Selection Operator)

**Trivalued Logic:** (Truth tables for the logical connectives $\land$, $\lor$ and $\neg$)

<table>
<thead>
<tr>
<th>G</th>
<th>H</th>
<th>$F = G \land H$</th>
<th>$F = G \lor H$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>undefined</td>
<td>false</td>
<td>false</td>
<td>undefined</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>undefined</td>
<td>false</td>
<td>undefined</td>
</tr>
<tr>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>true</td>
<td>undefined</td>
<td>undefined</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>undefined</td>
<td>true</td>
<td>undefined</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G</th>
<th>$F = \neg G$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>
2.1.2.- R.A. (Selection Operator)

<table>
<thead>
<tr>
<th>person_id</th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>20450120</td>
<td>Juan Pérez</td>
<td>Cuenca 20</td>
</tr>
<tr>
<td>12904569</td>
<td>José Abad Blasco Ibáñez</td>
<td>Blasco Ibáñez 35</td>
</tr>
<tr>
<td>?</td>
<td>María Gutiérrez</td>
<td>Reina 7</td>
</tr>
<tr>
<td>12345678</td>
<td>Pepa Gómez</td>
<td>Colón 15</td>
</tr>
</tbody>
</table>

Example:

Let R be

Operations:

\[ R \text{ where } \neg (\text{name} = \text{“Juan Pérez”}) \land (\text{person_id} > 12500500) = \]

\[ R \text{ where } \neg (\text{person_id} > 12500500) = \]

\[ R \text{ where } (\text{person_id} \leq 12500500) \lor (\text{person_id} > 12500500) = R ? \]
2.1.2.- Summary of Operators

- INSERTION
- DELETION
- RENAMING
- SELECTION
- PROJECTION
- UNION
- INTERSECTION
- DIFFERENCE
- CARTESIAN PRODUCT
- JOIN
- DIVISION

Relational Algebra
2.2.- Representation of Reality

- For each object in reality about which we want to have information we define a relation whose attributes denote the properties of interest for these objects (code, name, …) in such a way that each tuple which is present in this relation must be interpreted as a particular instance of an object;

- In order to represent associations between objects we use explicit references through attributes which identify each object.
2.2.- Representation of Reality

EXAMPLE 1:

• Reality: *Dishes and menus in a restaurant.*

• Database schema:

  Menu(menu_name: d4, price: d2)
  Is_composed_of(dish_name: d5, menu_name: d4)
  Dish(dish_name: d5, calories: d6, wine_id: d8, cook_name: d7)
  Wine(wine_id: d8, wine_name: d11, year: d13, colour: d14)
  Cook(name: d7, age: d9, country: d10)
  Intervenes(ing_name: d1, dish_n: d5, quantity: d15)
  Ingredient(ing_name: d1, price: d2, description: d3)
2.2.- Representation of Reality

CARDINALITY/MULTIPLICITY between two objects A and B:

Generic Notation

\[ R(A(\min_A, \max_A), B(\min_B, \max_B)) \]

- Each tuple in \( A \) requires a \( \min_A \) of corresponding tuples in \( B \), but at most \( \max_A \).
- Each tuple in \( B \) requires a \( \min_B \) of corresponding tuples in \( A \), but at most \( \max_B \).

Example:

A wine may appear in many dishes but a dish must have one and only one wine.
2.2.- Representation of Reality

Between two relations only one maximum can be greater than one.

Be careful! This implies that:

In the relational model, the cardinalities several to several (many to many) can only be obtained through an intermediate table.

Example: a dish may have many ingredients. Similarly, an ingredient may appear in many dishes. *We need the table: intervenes.*
2.2.- Representation of Reality

INTUITIVE REPRESENTATION (Ms. Access)

If this cardinality is $\infty$, then it corresponds to $(0, \infty)$
If this cardinality is 1, then it corresponds to $(0, 1)$
2.2.- Representation of Reality

EXAMPLE 1:
2.1.- Exercises: R.A. Queries

EXAMPLE 2. RESTAURANT:

• Obtain the name of the dishes with less than 2,000 calories:

• Obtain the name of the cook of the dishes with white wine.

• Obtain all the information from the dishes cooked by Russian cooks:
2.1.- Exercises: R.A. Queries

EXAMPLE 2. RESTAURANT (Contd.):

• Obtain the age of the cook whose dishes only have old vintage wines (year < 1982).

• Obtain the name of the menus without egg:

• Obtain the name of the most expensive ingredient: