Unit 3: Relational Model.

2022/2023

Contents

3.1. Introduction.

First described in **1969** by Edgar F. **Codd**.
All data is represented in terms of **tuples** (=rows), grouped into **relations**.
A database organized in terms of the relational model is a **relational database**.
Main goal: to provide a declarative **method for specifying data and queries**.
Most relational databases use the **SQL** <u>**data definition**</u> **and** <u>**query language**</u>.

3.2. Structure of relational data.

Video: https://youtu.be/ADojtHV2iU4

Database = Set of named **relations** (or **tables**).

Each relation has a set of named **attributes** (or **columns**).

Each **tuple** (or **row**) has a value for each attribute.

Each attribute has a **type** (or **domain**).

Domain: Set of valid values for a column.

Cardinality: Number of rows.

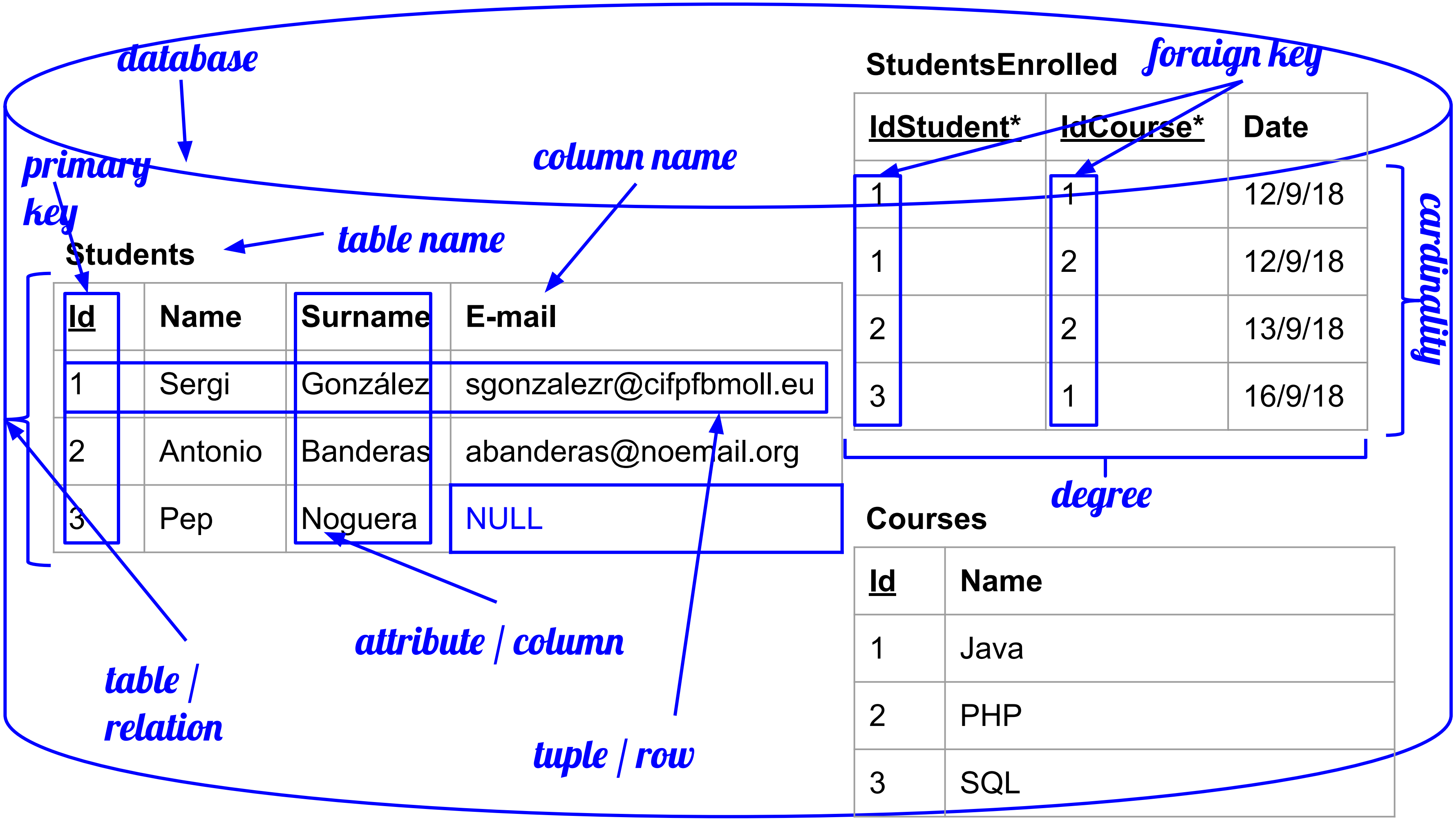
Degree: Number of columns.

Schema: Structural description of a relation in database.

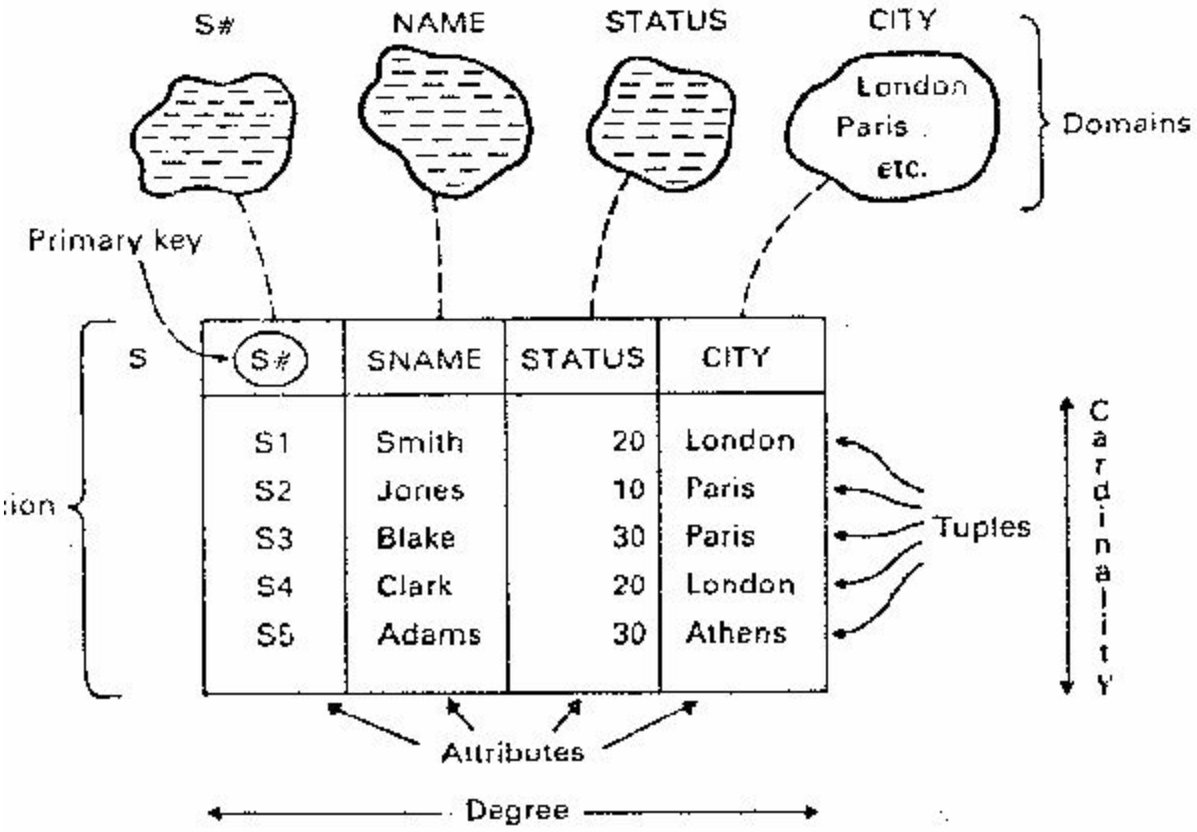
Instance: Actual contents at a given point in time.

NULL: Special value for “unknown” or “undefined”.

Primary Key: Attribute whose value is unique in each tuple. Or set of attributes whose combined values are unique.



database
primary key
Students
table name
column name
Id
Name
Surname
E-mail
1
Sergi
González
sgonzalezr@cifpfbmoll.eu
2
Antonio
Banderas
abanderas@noemail.org
3
Pep
Noguera
NULL
table / relation
attribute / column
tuple / row
StudentsEnrolled
foraign key
IdStudent*
IdCourse*
Date
1
1
12/9/18
1
2
12/9/18
2
2
13/9/18
3
1
16/9/18
cardinality
degree
Courses
Id
Name
1
Java
2
PHP
3
SQL



S#
NAME
STATUS
CITY
London
Paris
etc.
Domains
Primary key
S
S#
SNAME
STATUS
CITY
S1
Smith
20
London
S2
Jones
10
Paris
S3
Blake
30
Paris
S4
Clark
20
London
S5
Adams
30
Athens
Tuples
Cardinality
Attributes
Degree

3.2. Structure of relational data.

Equivalent Database Concepts:

- **Relation** ↔ **Table**
- **Tuple** ↔ **Row or record**
- **Attribute** ↔ **Column or field**
- **Cardinality** ↔ **Number of rows**
- Degree ↔ **Number of columns**
- **Primary key** ↔ **Unique identifier**
- **Foreign key** ↔ **Primary key in other table**
- **Domain** ↔ **Pool of allowed values**

3.3. Domains.

A **domain** is a **collection of values**. One or more attributes get their real values of a domain.

Another definition: A domain is a **set of scalar values**, all of the same type. A scalar value is the smallest semantic unit of information. It is atomic, because if it breaks down it loses its meaning.

Example 1: Domain of NIF# is the set of character strings of length 9.
Example 2: Domain of WEIGHT is the set of small integers less than 10,000.
Example 3: Domain of QTY is the set of integers less than one billion.

Therefore, a "**domain**" is like a "**data type**".

3.3. Domains.

Importance of domains:

To **restrict comparisons** to compatible domains.

To **avoid redundancies** in the definition of attributes.

Queries such as: Relations in the DB with telephone numbers?

3.3. Domains.

Domains can be used **to impose semantic constraints**.

Working with ‘real’ databases you don’t find a constraint like that very often...

- **Example 1**:

```
SELECT P.*, SP.*
FROM P, SP
WHERE P.P# = SP.P# ;
```

  - The comparison in the conditional clause is in the same domain.

- **Example 2**:

```
SELECT P.*, SP.*
FROM P, SP
WHERE P.WEIGHT = SP.QTY ;
```

  - The comparison involves two attributes of different domains, and therefore should not be allowed.

3.4. Relations.

A **relation** consists of two parts: a "**heading**" and a "**body**".

The **<u>heading</u>** consists of a fixed **set of attribute-domain pairs**:

$\{(A_i:D_i)\}$ $(\{(A_1:D_1), (A_2:D_2), \ldots, (A_n,D_n)\})$

with i>0

A_i: set of attribute names

D_i: Domains associated with attributes

The heading is also called the **schema** (of a table).

The number of attribute-domain pairs is called **the degree** of the relation.

3.4. Relations.

A **relation** consists of two parts: a "**heading**" and a "**body**".

The **body** consists of a time-varying set of tuples, where each tuple consists of a **set of attribute-value pairs**:

$\{(A_i{:}v_{ji})\}$ $((A_1{:}v_{j1}), (A_2{:}v_{j2}), \ldots, (A_n{:}v_{jn}))$

with $i > 0$ and $j = 1, 2, \ldots, m$

A_j: set of attribute names

v_{ij}: values assigned to attributes

The body is also called the **instance**. The number of attribute-value pairs is called **cardinality** of the relation.

The body **changes in time**.

Note that **n** is the **degree of the relation** and **m** is the **cardinality of the relation**.

3.4. Relations.

Example of a relation:

header

body

<u>**Id#**</u>	**Name**	**Surname**	**E-mail**
1	Sergi	González	sgonzalezr@cifpfbmoll.eu
2	Antonio	Banderas	abanderas@noemail.org
3	Pep	Noguera	pnoguera@noemail.org
4	Pep	Noguera	pnoguera2@noemail.org

3.4. Relations.

Properties of relations:

1. There are **no duplicate tuples**.
2. **Tuples** are **unordered**.
3. **Attributes** are **unordered**.
4. All **attribute values are atomic** (=not multiple).

Note: That's the difference between a conventional table and a relation.

3.4. Relations.

Kinds of relations:

DBA

Base relations: The real relations. Called "base table" in SQL.

Views: The virtual relations. A view is a named, derived relation.

Snapshots: A snapshot is a real, not virtual, named derived relation.

User

Temporary relations: A nonpermanent named derived relation.

Query results: The final output relation from a specified query. It may not be named and has no permanent existence.

Intermediary results: It’s a relation resulting from some relational expression nested into another larger relational expression.

3.4. Relations.

Base relations

The real relations. Called "base table" in SQL. Autonomous, with name. They are part of the database itself.

```
CREATE TABLE      CUSTOMER(
   CUSTOMER#      DOMAIN (CUSTOMER#)     NOT NULL,
   NIF            DOMAIN (NIF)           NOT NULL,
   CUST_NAME      DOMAIN (NAME)          NOT NULL,
   TELEPHONE      DOMAIN (TELEPHONE),
   PRIMARY KEY (CUSTOMER#));
```

3.4. Relations.

Views

The virtual relations. A view is a named, derived relation. It is represented only by its definition in terms of other relations. It does not have stored data.

```
CREATE VIEW   V_CUSTOMER_INCA
   AS SELECT * FROM CUSTOMER WHERE CITY = 'INCA';
```

3.4. Relations.

Snapshots

A snapshot is a real, not virtual, named derived relation. But unlike the views, it is real, not virtual. It has stored data. It is created in a very similar way to a query. It is only for SELECT queries.

```
CREATE SNAPSHOT S_CUSTOMER_MADRID
      AS SELECT * FROM CUSTOMER WHERE CITY = 'INCA'
REFRESH EVERY DAY;
```

3.4. Relations.

Query results

The final output relation from a specified query. It may not be named and has no permanent existence.

```
SELECT * FROM CUSTOMER WHERE CITY = 'INCA';
```

3.4. Relations.

Intermediary results

It is a relationship (usually without name) resulting from some relational expression nested into another larger relational expression. There is no persistent existence in the database.

```
SELECT DISTINCT CITIES
FROM CUSTOMER
WHERE CUSTOMER# IN
    (SELECT CUSTOMER#
       FROM BILL
       WHERE TOTAL_AMOUNT > 100000);
```

3.4. Relations.

Temporary relations

Temporary relations allow users to store intermediate results rather than having to submit the same query or subquery again and again. Unlike the view, the temporary relation is a real relation in the database which is seen only by the user and which disappears at the end of the user's session. This is especially useful if the query is needed for many other queries, and it is time-consuming to complete it.

```
CREATE TEMPORARY TABLE IF NOT EXISTS tableTemp
AS (SELECT * FROM table1 UNION SELECT * FROM
table2);
```

More samples here.

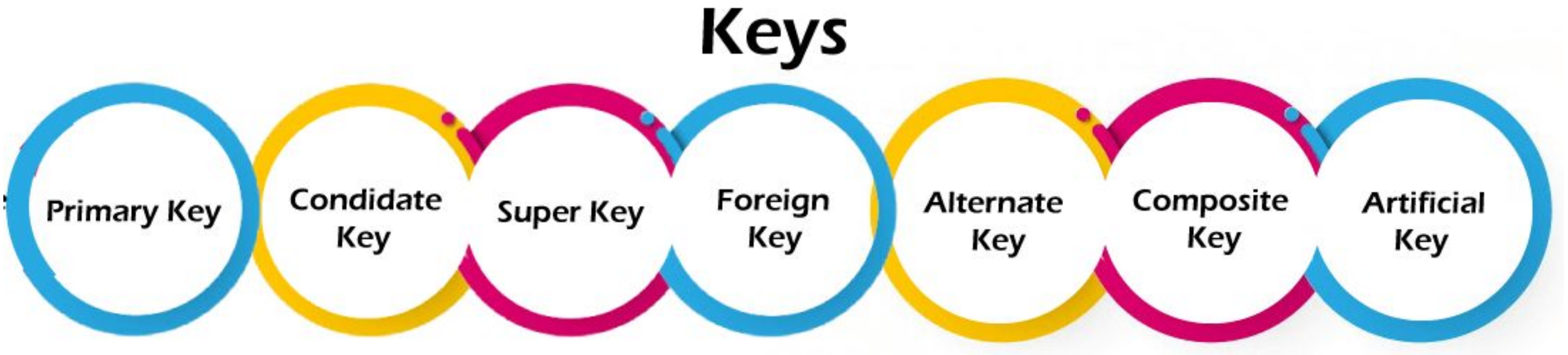
3.5. Key or identifier.

Example: STUDENT (<u>NIF</u>, name, address, telephone, fax, SSN, …)

*Do you remember the concept of **<u>identifier</u>** in the ER model?*

I decided to underline primary keys.

3.5. Key or identifier.



Source: https://www.javatpoint.com/dbms-keys

3.5. Key or identifier.

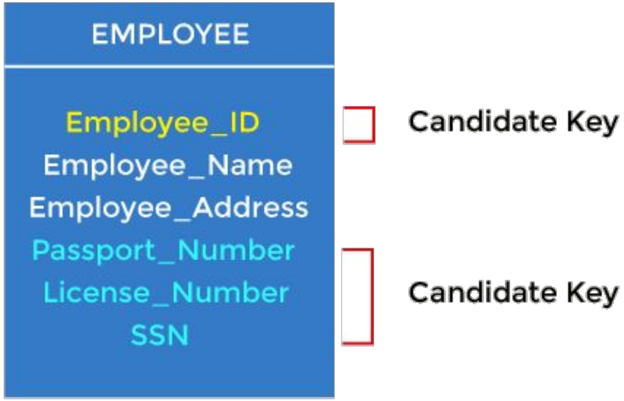
Minimum key

Set of attributes that identify a tuple with the following property:

- If it is composed (formed by more than one attribute) and an attribute is removed, it is no longer a key.

Candidate keys

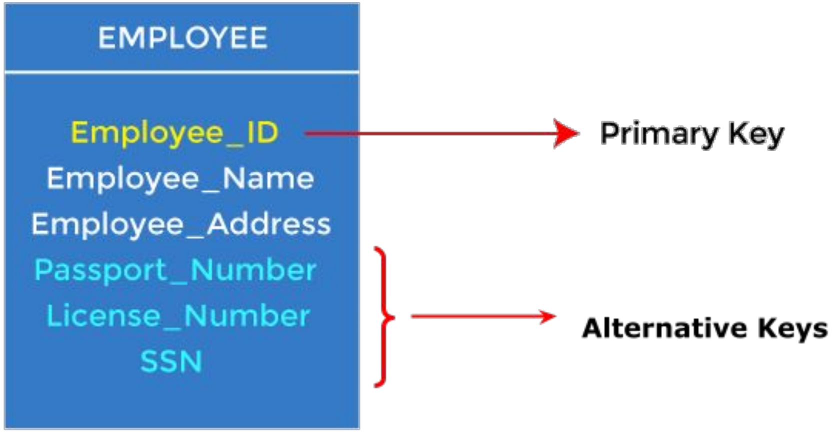
Set of minimum keys.



Source: https://www.javatpoint.com/dbms-keys

3.5. Key or identifier.

Primary key	The chosen one among the candidate keys.
Alternative key	The remaining candidates who are not primary.

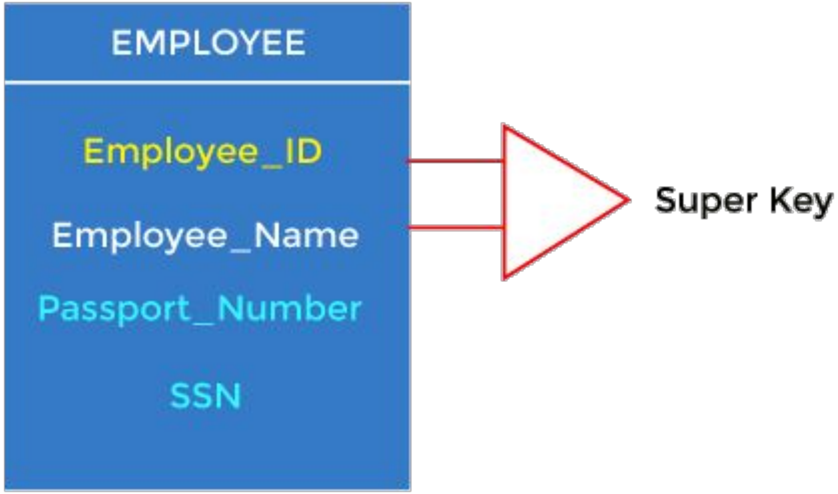


Source: https://www.javatpoint.com/dbms-keys

3.5. Key or identifier.

Super Key

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.

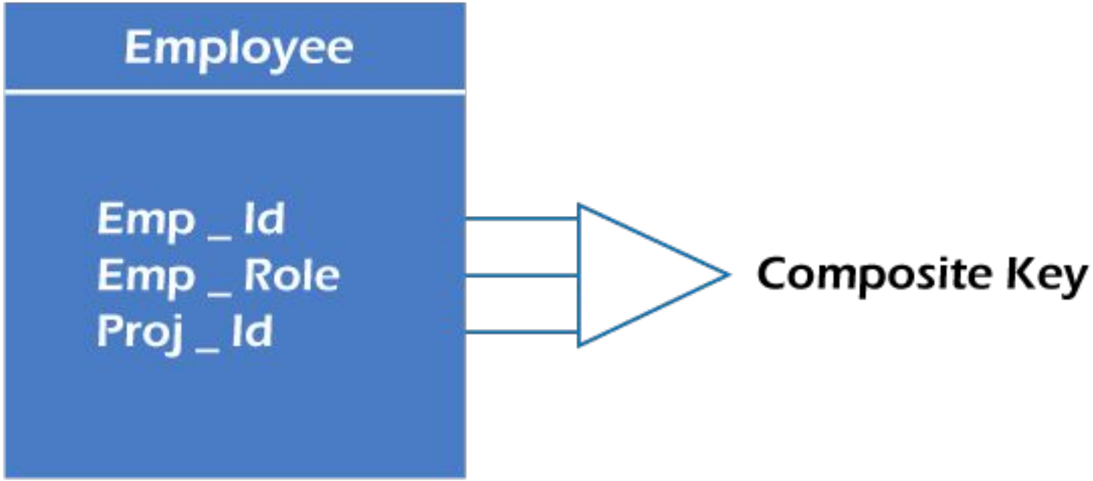


Source: https://www.javatpoint.com/dbms-keys

3.5. Key or identifier.

Composite key

Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.

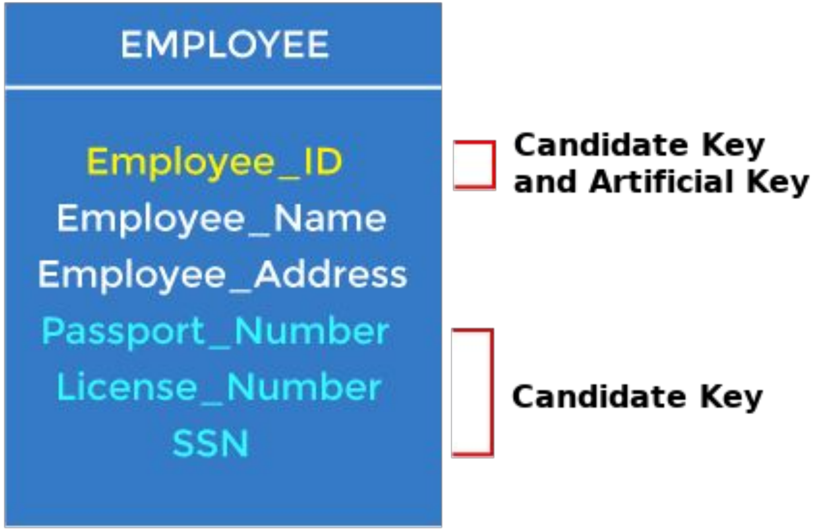


Source: https://www.javatpoint.com/dbms-keys

3.5. Key or identifier.

Artificial key

The key created using arbitrarily assigned data are known as artificial keys. These keys are created when a primary key is large and complex and has no relationship with many other relations. The data values of the artificial keys are **usually numbered in a serial order**.



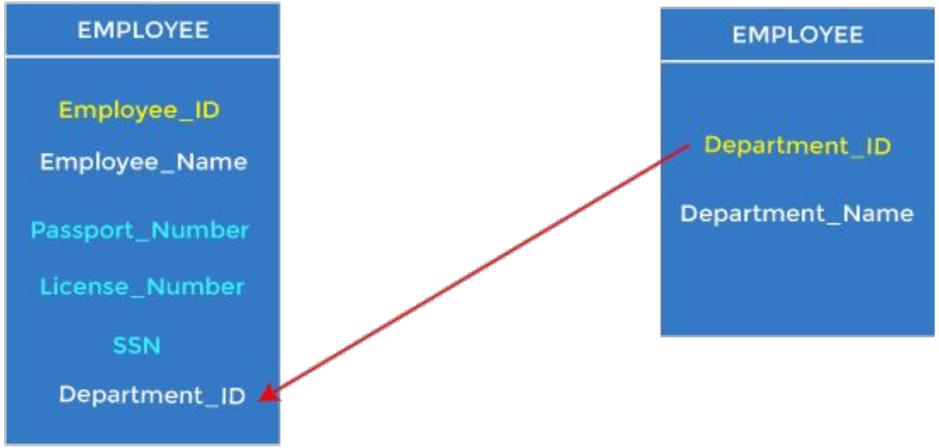
Source: https://www.javatpoint.com/dbms-keys

3.5. Key or identifier.

Foreign key

Set of attributes of a relation that are primary key of another relation.

INVOICE (INVOICE_NUMBER, date, NIF*)



I decided to mark the foreign keys with the symbol "".*

Source: https://www.javatpoint.com/dbms-keys

3.6. NULL values.

NULL is a special value that can take an attribute. It can basically mean:

- **Unknown value**
- **Value not applicable**

<u>**Id**</u>	**Name**	**Surname**	**Weight**	**Married**	**WifeName**
1	Sergi	González	74	True	Paquita
2	Antonio	Banderas	71	True	Antonia
3	Pep	Noguera	NULL	False	NULL

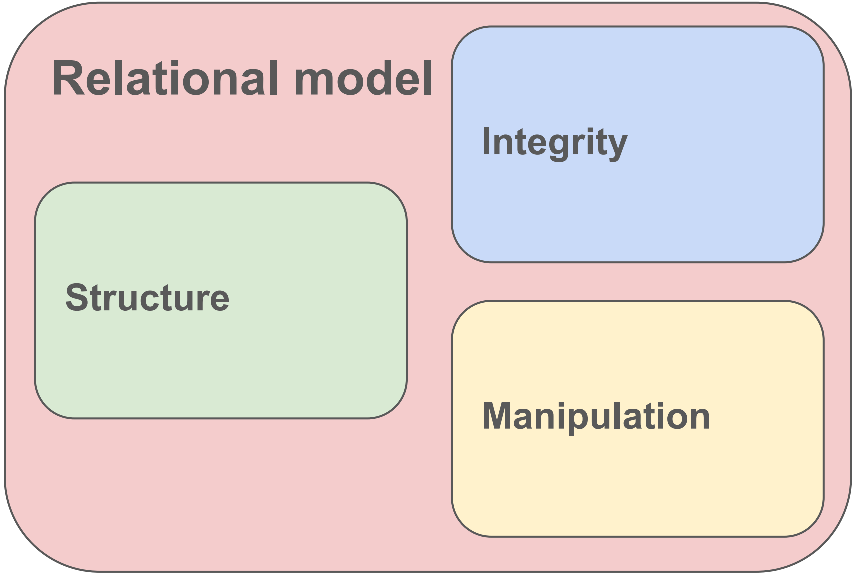
Unknown value

Value not applicable

3.7. Relational integrity.

We said that the relational model deals with three aspects of data:

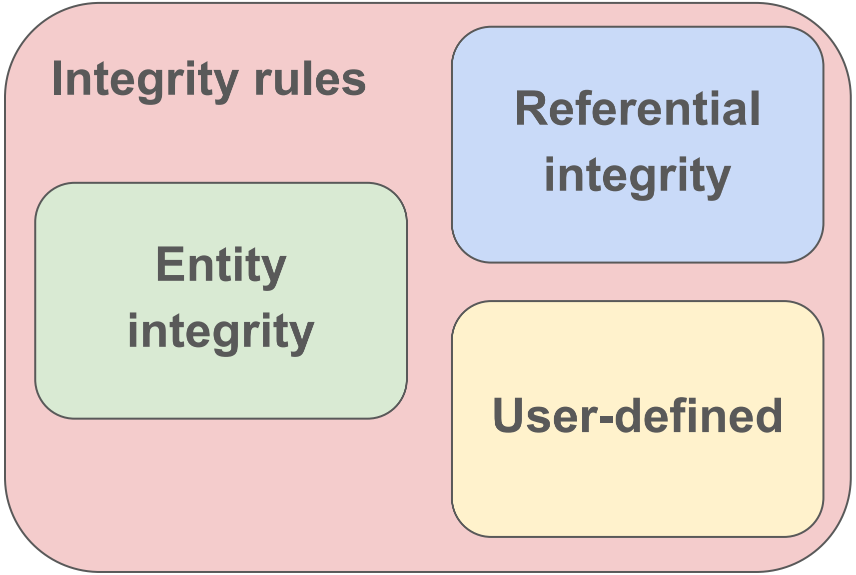
- **Structure**
- **Integrity**
- **Manipulation**



3.7. Relational integrity.

Every relational system must follow the next integrity rules:

- **The entity integrity rule**
- **The referential integrity rule**
- **Additionally: user-defined integrity rules** (that is, specific integrity rules of a specific DB).



3.7. Relational integrity.

THE ENTITY INTEGRITY RULE

No component of the primary key of a base relation is allowed to accept nulls.

Remember that:
• **NULL** may mean "**property does not apply**". For example, the supplier may be a country, in which case the attribute CITY has a null value because such property does not apply.
• **NULL** may mean "**value is unknown**". For example, if the supplier is a person, then a null value for CITY attribute means we do not know the location of this supplier.
• **NULLs cannot be in primary keys**, **but can be in foreign keys**.
• EXAMPLE: DNI_partner can be NULL if you are not married.

3.7. Relational integrity.

THE REFERENTIAL INTEGRITY RULE

The database can not contain inconsistent foreign key values.

In other words, the **valid values of a foreign key** are:

- Existing values in the primary key of reference.
- NULL values.

Another way of saying it:

- The database must not contain any unmatched foreign key values.

Not possible!!

StudentsEnrolled

IdStudent*	**IdCourse***	**Date**
1	1	12/9/18
1	4	12/9/18
2	2	13/9/18
3	1	16/9/18

Students

Id	**Name**	**Surname**	**E-mail**
1	Sergi	González	sgonzalezr@iesfbmoll.org
2	Antonio	Banderas	abanderas@noemail.org
3	Pep	Noguera	pnoguera@noemail.org

Courses

Id	**Name**
1	Java
2	PHP
3	SQL

3.7. Relational integrity.

WHY ARE FOREIGN KEYS IMPORTANT?

- **Foreign-to-primary-key** matching are **the "glue"** which holds the database together.

Another way of saying it:

- Foreign keys provide the "links" between two relations.

Remember that a relation's foreign key can refer to the same relation:

- PERSON (<u>DNI</u>, name, DNI_partner*)

3.7. Relational integrity.

What should the DBMS do if you try to change the primary key referenced by a foreign key?

- **RESTRICT:** It is not allowed to delete the primary key.
- **CASCADE:** Delete the tuple (row) of the foreign keys referencing that primary key.
- **SET NULL:** Set the value of the foreign key to NULL.
- **SET DEFAULT:** Sets the value of the foreign key to a value.
- **Execute a TRIGGER:** Execute a code to do something.

CUSTOMER (<u>NIF</u>, name, address, phone, fax, …)

INVOICE (<u>num_inv</u>, data, …, NIF*)

```
CREATE TABLE INVOICE (
        NUM_INV    DOMAIN (NUM_DOCUMENT),
        DATE DOMAIN (DATE)
        NIF    DOMAIN (NIF),
 PRIMARY KEY (NUM_INV),
 FOREIGN KEY (NIF) REFERENCES
CUSTOMER
        NULLS NOT ALLOWED
        DELETE OF CUSTOMER RESTRICTED
        UPDATE OF CUSTOMER RESTRICTED);
```

3.8. Examples: One-To-Many Relationship.

EMPLOYEES (<u>num</u>, surname, dept_num*)
DEPARTMENTS (<u>num</u>, name)

EMPLOYEES

<u>**num**</u>	**surname**	**dept_num***
401	PITT	40
402	SMITH	10
403	JOHNSON	10
404	WINEHOUSE	20

DEPARTMENTS

<u>**num**</u>	**name**
10	ACCOUNTING
20	RESEARCH
30	PRODUCTION
40	SALES

3.8. Examples: One-To-Many VS One-To-One Relationships.

PRODUCTS (<u>id</u>, name, quantity, price, supplierId*)
SUPPLIERS (<u>id</u>, name, phone)

With SQL, using UNIQUE creating the column, you create a One-To-One Relationship

PRODUCTS

<u>id</u>	name	quantity	price	supplierID*
1001	Pencil 3B	504	0,52	1
1002	Pencil 4B	742	0,63	1
1003	Pencil 5B	201	0,75	2
1004	Pencil 6B	99	0,49	3

SUPPLIERS

<u>id</u>	name	phone
1	ABC SUPPLIES	11223344
2	WEB TRADERS	11223345
3	ZZ COMPANY	11223346

3.8. Examples: Many-To-Many Relationship.

PRODUCTS (id, name, quantity, price)
SUPPLIERS (id, name, phone)
PRODUCTS_SUPPLIERS (productId*, supplierID*)

PRODUCTS_SUPPLIERS

productId*	supplierId*
1001	1
1002	1
1001	2
1001	3

PRODUCTS

id	name	quantity	price
1001	Pencil 3B	504	0,52
1002	Pencil 4B	742	0,63
1003	Pencil 5B	201	0,75
1004	Pencil 6B	99	0,49

SUPPLIERS

id	name	phone
1	ABC SUPPLIES	11223344
2	WEB TRADERS	11223345
3	ZZ COMPANY	11223346

3.8. Examples: Many-To-Many Relationship.

PRODUCTS (id, name, quantity, price)
ORDER (id, date, address)
ORDER_LINES (orderId*, productId*, quantity, price)

ORDER_LINES

orderId*	productId*	quantity	price
1	1001	20	0,52
1	1002	10	0,63
1	1003	12	0,75
2	1001	100	0,52

PRODUCTS

id	name	quantity	price
1001	Pencil 3B	504	0,62
1002	Pencil 4B	742	0,73
1003	Pencil 5B	201	0,85
1004	Pencil 6B	99	0,59

ORDER

id	date	address
1	01-SEP-2019	Caracas, 6
2	01-SEP-2019	Krueger, 13 1-2

3.8. Examples: Reflexive Relationship

PERSON (id, name, surname, motherId*, fatherId*)

id	name	surname	birth_date	motherId*	fatherId*
...	...	...	...	...	..
C127923	MICHAEL	PITT	NULL	C127232	NULL
C127936	LAURA	SMITH	NULL	NULL	NULL
C236182	REBECCA	JOHNSON	12-FEB-1933	C127936	NULL
C127232	JANE	WINEHOUSE	16-DEC-1934	NULL	NULL
C236998	BRADLEY	PITT	16-DEC-1963	C236182	C127923
C236999	DOUGLAS	PITT	02-NOV-1962	C236182	C127923
C245567	CATHERINE	JOHNSON	02-JUL-1975	C236182	NULL

3.8. Examples: Reflexive Relationship Many To Many

SUBJECTS (id, name, year, hours)
PREREQUISITES (subjectId*, subjectIdPre*)

SUBJECTS

id	name	year	hours
1001	Programming	1	230
1002	Databases	1	270
1003	Markup languages	1	190
1004	Accessing Data	2	140
1005	Web Programming	2	150

PREREQUISITES

subjectId*	subjectIdPre*
1004	1001
1004	1002
1005	1001
1005	1003

3.8. Examples: Optionality.

When you create a column, if you use the clause NOT NULL, you can not have NULL values inside that column.

EMPLOYEES

num	surname	dept_num*
401	PITT	NULL
402	SMITH	10
403	JOHNSON	10
404	WINEHOUSE	20

DEPARTMENTS

num	surname
10	ACCOUNTING
20	RESEARCH
30	PRODUCTION
40	SALES

VS

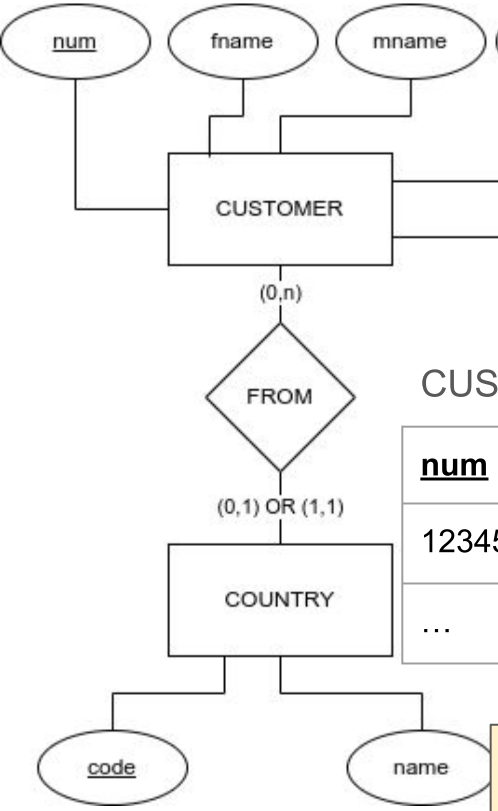
EMPLOYEES

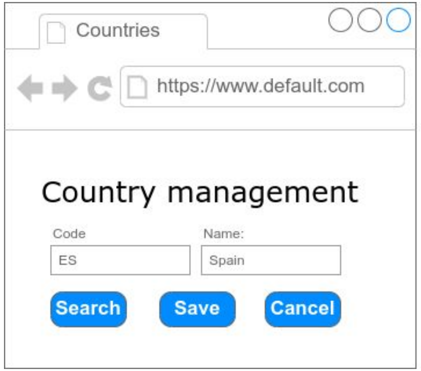
num	surname	dept_num*
401	PITT	40
402	SMITH	10
403	JOHNSON	10
404	WINEHOUSE	20

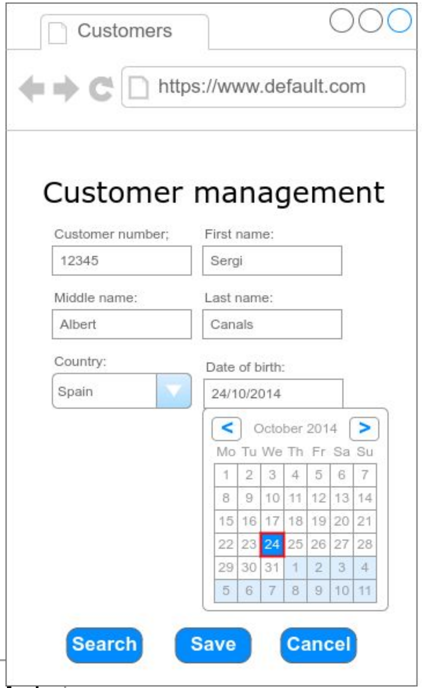
DEPARTMENTS

num	surname
10	ACCOUNTING
20	RESEARCH
30	PRODUCTION
40	SALES

3.8. Examples: Windows design.







CUSTOMERS:

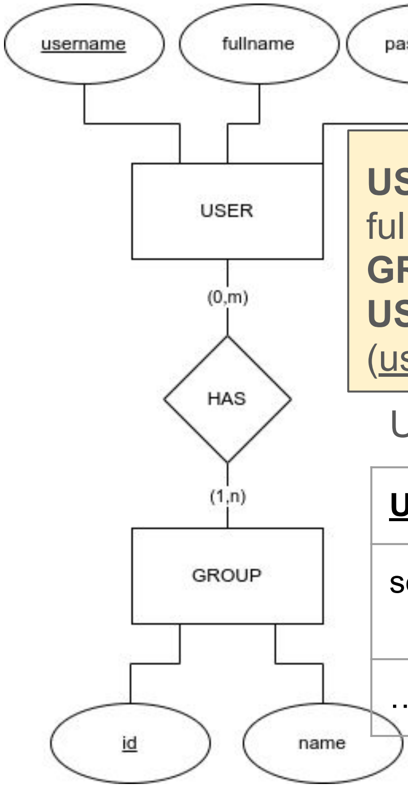
<u>num</u>	fname	mname	lname	date_of_birth	country_code*
12345	Sergi	Albert	Canals	24/10/2014	ES
…					

COUNTRIES:

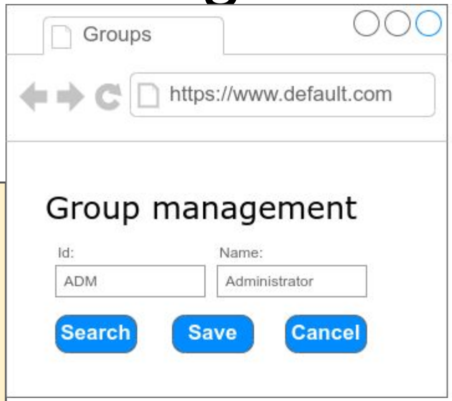
<u>code</u>	name
ES	Spain
PT	Portugal
…	

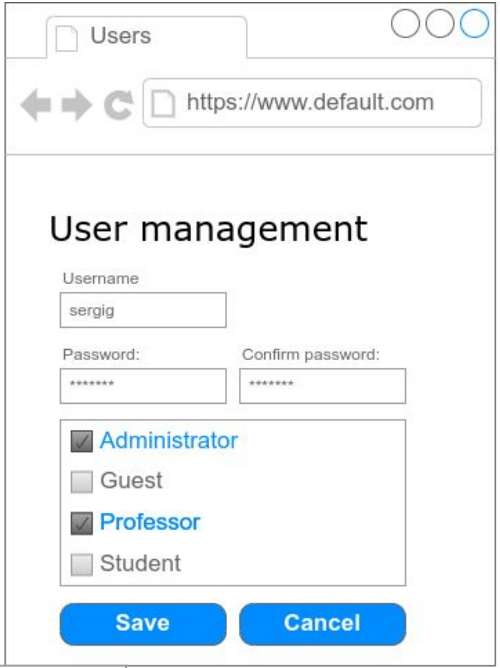
CUSTOMERS (<u>num</u>, fname, mname, lname, date_of_birth, country_code*)
COUNTRIES (<u>code</u>, name)

3.8. Examples: Windows design.



USERS (<u>username</u>, fullname, password)
GROUPS (<u>id</u>, name)
USERS_GROUPS (<u>username*, group_id*</u>)





USERS:

<u>**Username**</u>	**Fullname**	**Password**
sergig	Sergi González	koë3p1mj{
…		

GROUPS:

<u>**id**</u>	**name**
ADM	Administrator
GUE	Guest
PRO	Professor
STU	Student

USERS_GROUPS:

<u>**username***</u>	<u>**group_id***</u>
sergiog	ADM
sergiog	PRO
…	

3.9. Translation of the E-R Model to the Relational Model.

Database Model (E/R)

E/R diagrams

Relational schema

Tables:
row names: attributes
rows: tuples

Physical storage

Complex file organization and index structures.

DBA

DBMS

3.9. Translation of the E-R Model to the Relational Model.

E-R Model	**Relational Model**
entity	relation
relationship	relation (when N:M) or FK + fields in other relation (when 1:N)
associative entity	relation (when N:M) or FK + fields in other relation (when 1:N)
attribute	column
identifier	primary key

3.9. Translation of the E-R Model to the Relational Model.

Many-to-Many (M:N)

- SUBJECT (<u>code</u>, name, hours)
- STUDENT (<u>NIF</u>, name, birth_date)
- SUB_STU (MARK, ?)

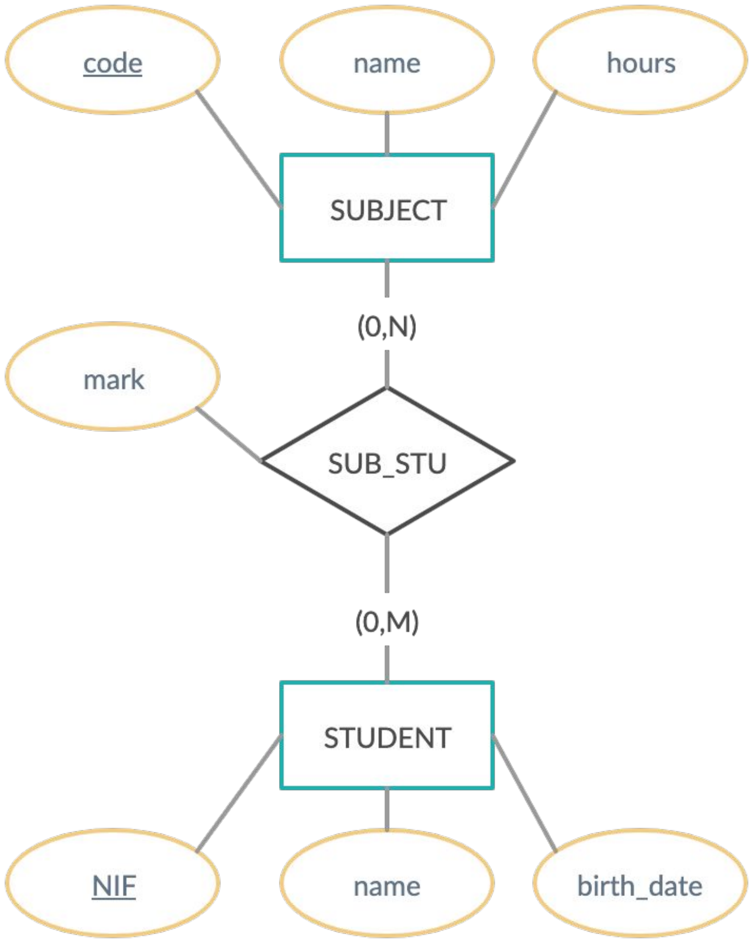
By inheriting as attributes the primary keys:

- SUB_STU (mark, NIF*, code*)

A primary key must be found for the new relation (usually, the union of the two inherited keys is a minimum key):

- SUB_STU (<u> NIF*, code*</u>, mark)

Note: **<u>If you change the relationship “SUB_STU” for a associative entity the procedure is just the same</u>**.



3.9. Translation of the E-R Model to the Relational Model.

Many-to-Many (M:N)

- SUBJECT (<u>code</u>, name, hours)
- STUDENT (<u>NIF</u>, name, birth_date)
- SUB_STU (MARK, ?)

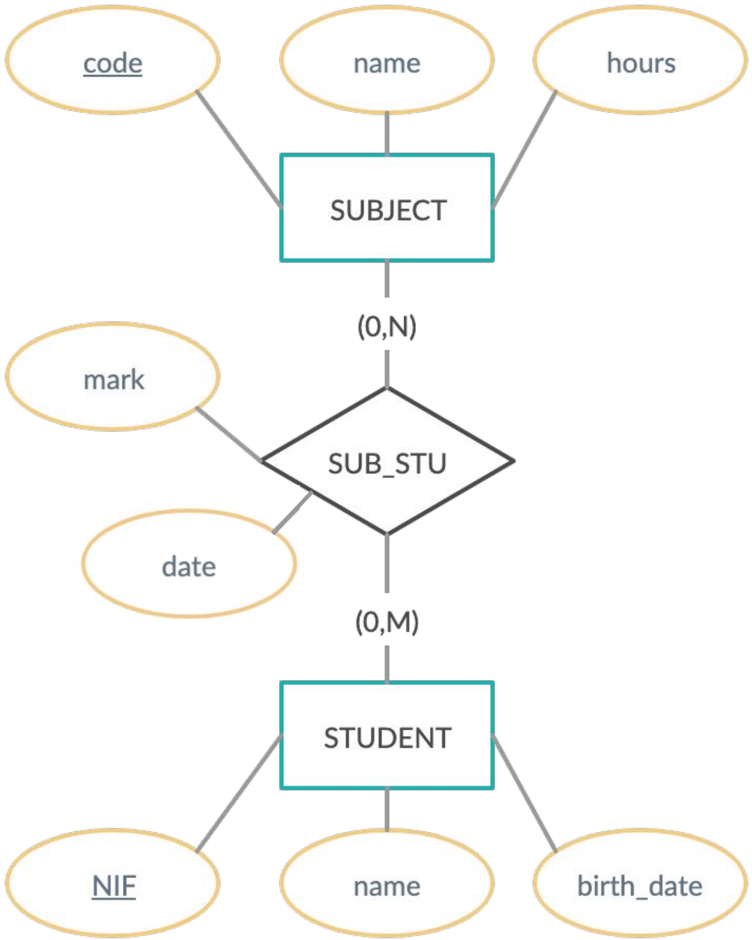
By inheriting as attributes the primary keys:

- SUB_STU (mark, NIF*, code*)

A primary key must be found for the new relation (usually, the union of the two inherited keys is a minimum key):

- SUB_STU (<u> NIF*, code*, date</u>, mark)

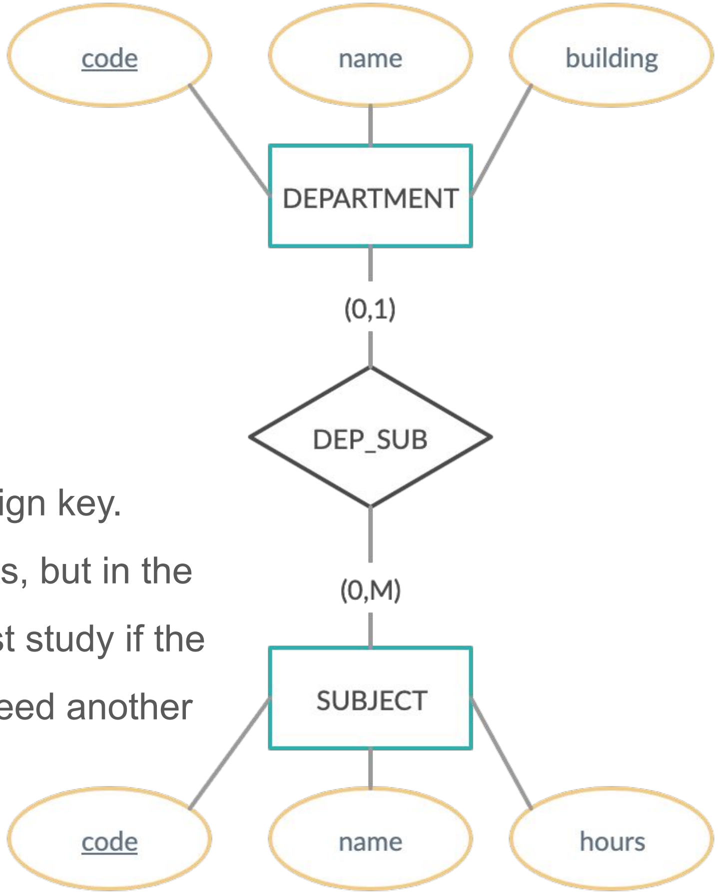
Note: **<u>If you change the relationship "SUB_STU" for a associative entity the procedure is just the same</u>**.



3.9. Translation of the E-R Model to the Relational Model.

One-to-Many (1:N)

- DEPARTMENT (<u>code</u>, name, building)
- SUBJECT (<u>code</u>, name, hours, dept_code*)
- Problems:
  - In case the ratio is 0:N causes NULL values in the foreign key.
  - It is not normal for an interrelation 1:N to have attributes, but in the case that we have attributes in the relationship we must study if the attributes will become part of the "N" relation or if we need another table.

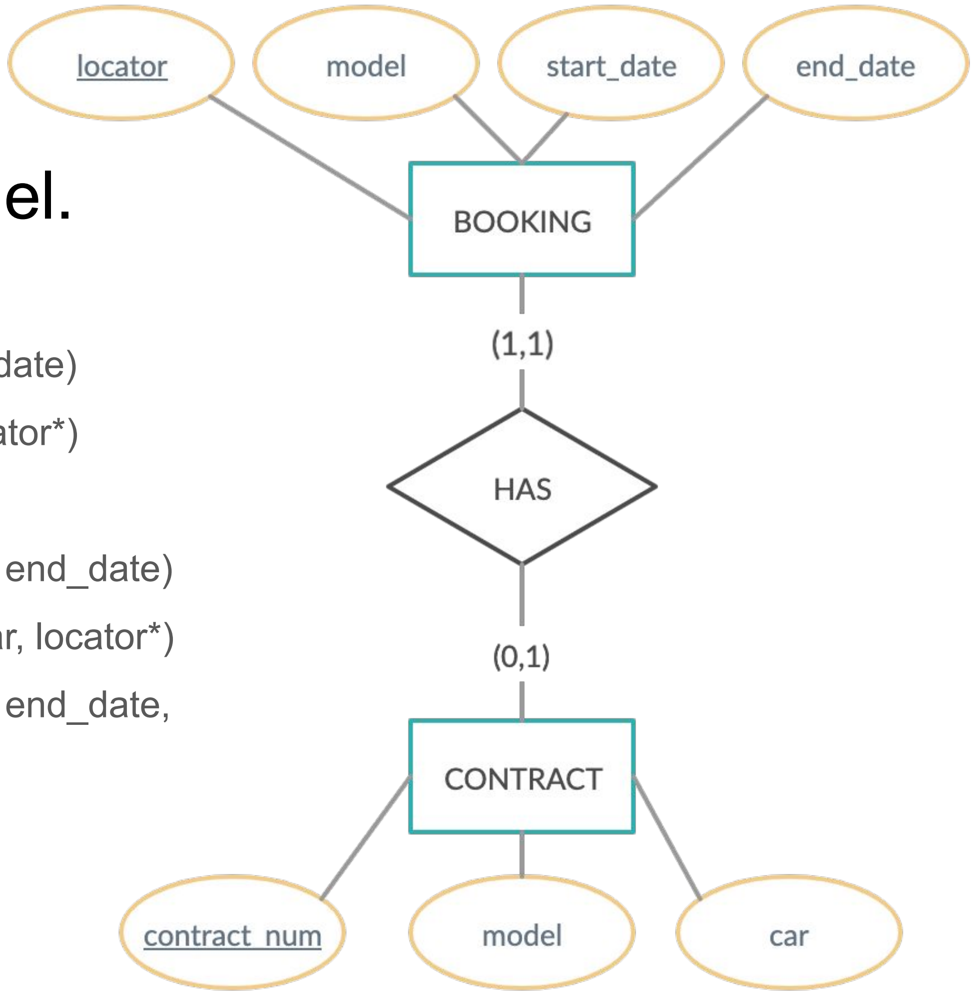


3.9. Translation of the E-R Model to the Relational Model.

One-to-one (1:1)

- BOOKING (<u>locator</u>, model, start_date, end_date)
- CONTRACT (<u>contract_num</u>, model, car, locator*)
- Questions:
  1. BOOKING (<u>locator</u>, model, start_date, end_date)
     CONTRACT (<u>contract_num</u>, model, car, locator*)
  2. BOOKING (<u>locator</u>, model, start_date, end_date, contract_num, model, car)

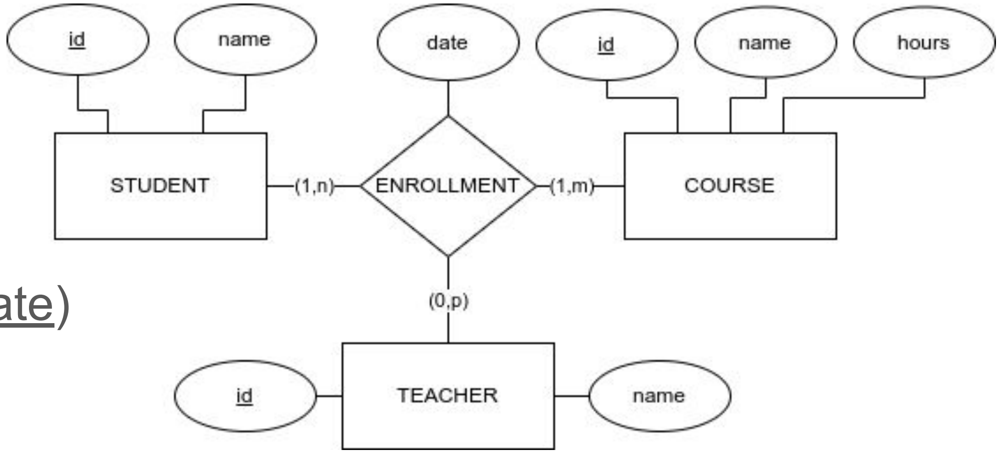
NOTE: In option 2 contract_num is not a foreign key.



3.9. Translation of the E-R Model to the Relational Model.

MULTIPLE RELATIONSHIP:

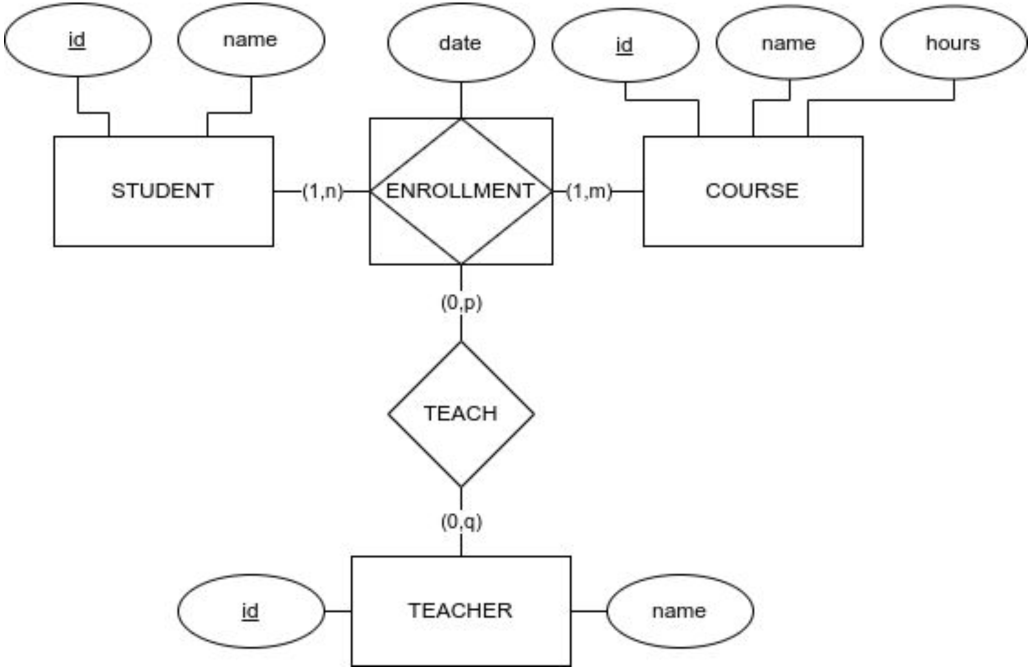
- Students (<u>id</u>, name)
- Courses (<u>id</u>, name, hours)
- Teachers (<u>id</u>, name)
- Enrollment (<u>stuid*, couid*, teaid*, date</u>)



3.9. Translation of the E-R Model to the Relational Model.

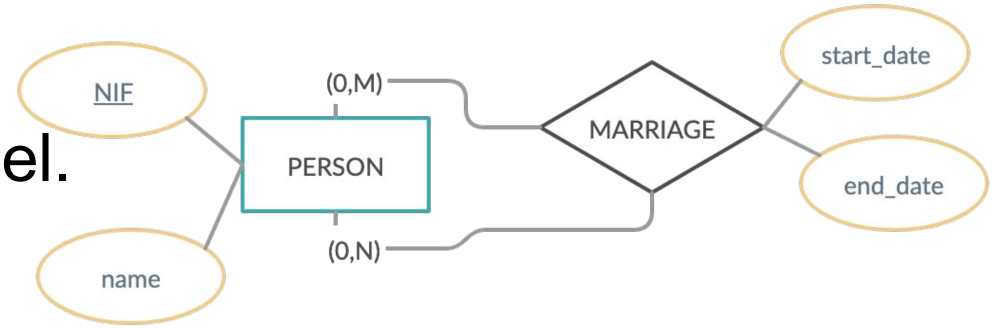
ASSOCIATIVE ENTITY:

- Students (id, name)
- Courses (id, name, hours)
- Teachers (id, name)
- Enrollments (stuid*, couid*, date)
- Teachs (stuid*, couid*, date, teadid*)



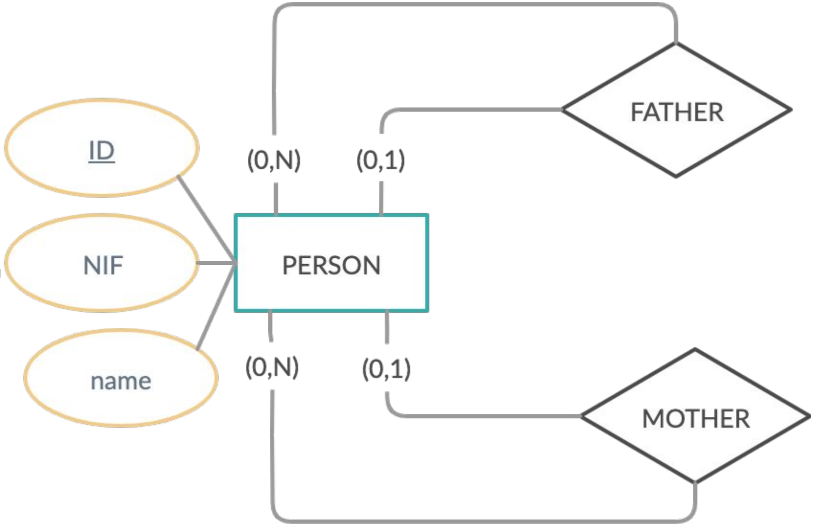
- **PK: Union of foreign keys**.
- If the associative entity has its **own identifier** we must change it for an entity (and multiple relationships).

3.9. Translation of the E-R Model to the Relational Model.



Recursive Relationships:

- PERSON (NIF, name)
- MARRIAGE(NIF*, NIF_partner*, start_date, end_date)



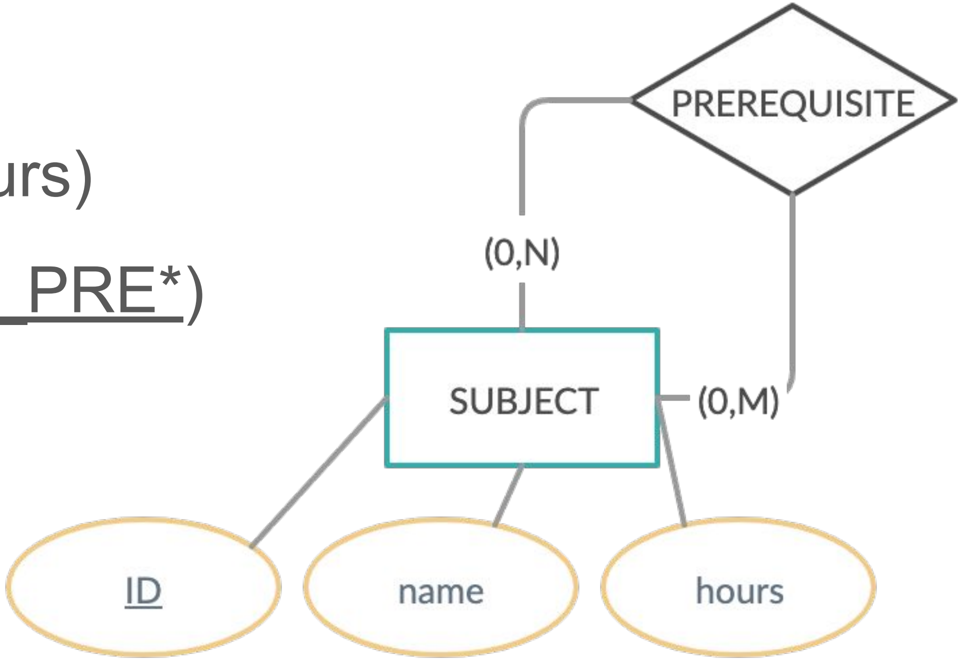
Recursive Relationships:

- PERSON (ID, name, ID_mother*, ID_father*)
  - Can ID_mother/ID_father be NULL?

3.9. Translation of the E-R Model to the Relational Model.

Recursive Relationships:

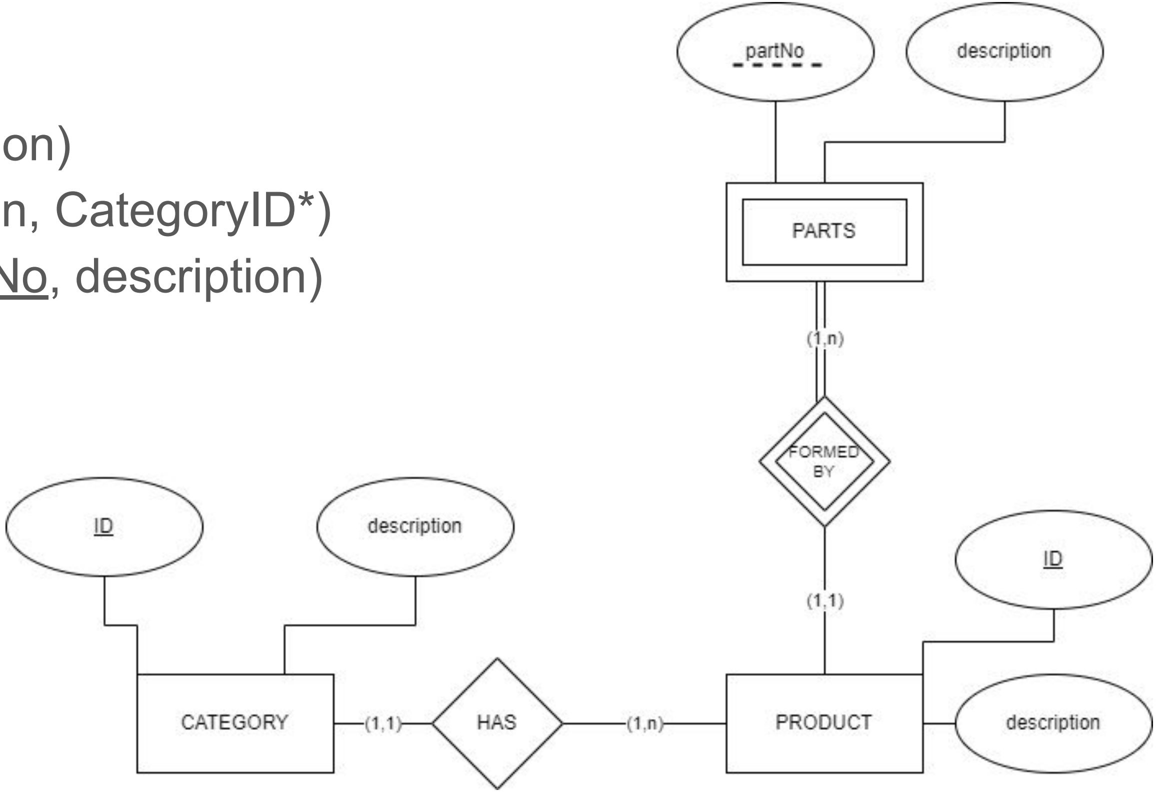
- SUBJECT (<u>ID</u>, name, hours)
- PREREQUISITE (<u>ID*, ID_PRE*</u>)



3.9. Translation of the E-R Model to the Relational Model.

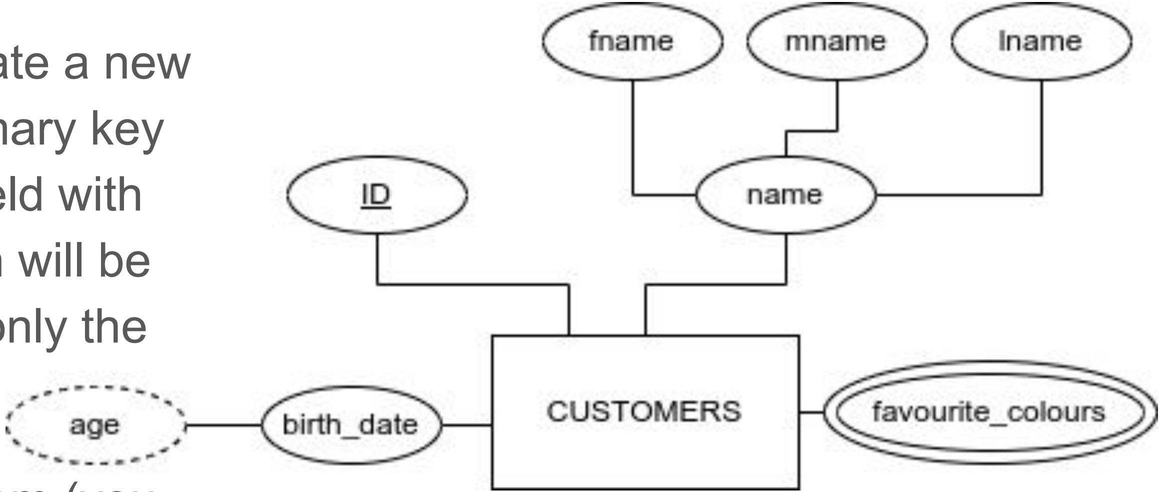
Weak entities:

- CATEGORY (<u>ID</u>, description)
- PRODUCT (<u>ID,</u> description, CategoryID*)
- PARTS (<u>ProductID*, PartNo</u>, description)



3.9. Translation of the E-R Model to the Relational Model.

Multivalued attributes: Create a new relation that includes the primary key of the relation and another field with the attribute (the combination will be the primary key, sometimes only the attribute can be PK).



Derived attribute: Delete them (you can calculate them in a select sentence).

Composited attributes: Decompose them.

CUSTOMERS (<u>ID</u>, fname, mname, lname, birth_date)
FAVOURITE_COLORS (<u>ID*, favourite_colour</u>)

3.9. Translation of the E-R Model to the Relational Model.

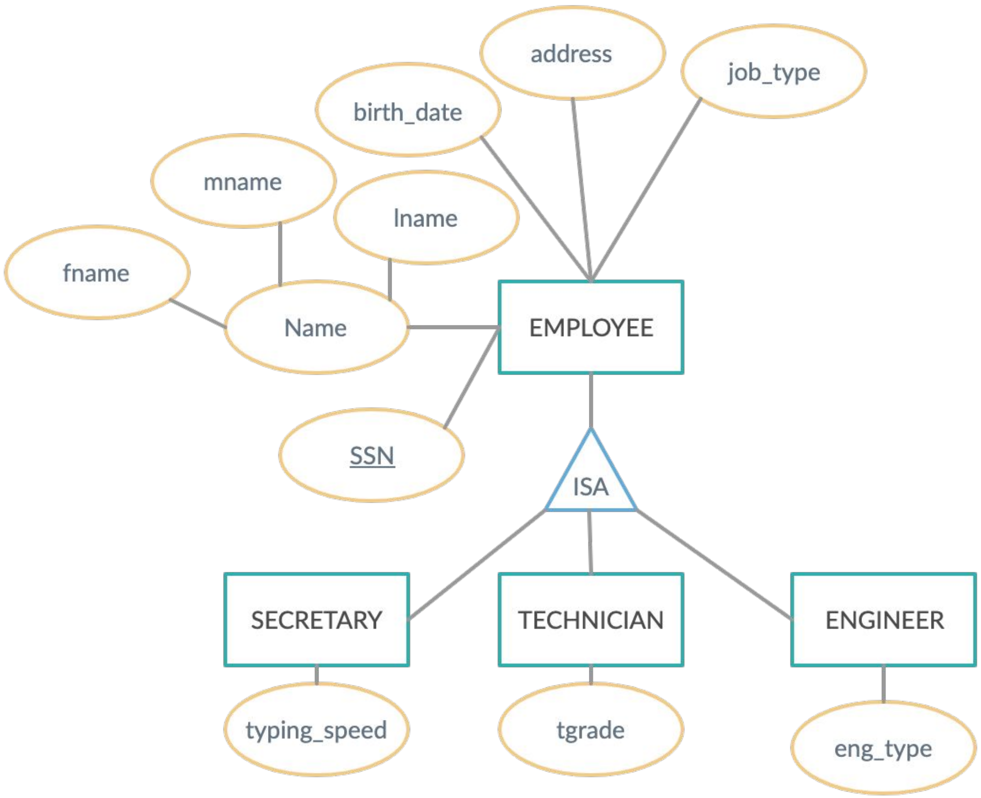
Specialization (we can have employees not secretary, technician, engineer)**:**

EMPLOYEE (<u>SSN</u>, fname, mname, lname, birth_date, address, job_type)

SECRETARY(<u>SSN*</u>, typing_speed)

TECHNICIAN(<u>SSN*</u>, tgrade)

ENGINEER(<u>SSN*</u>, eng_type)

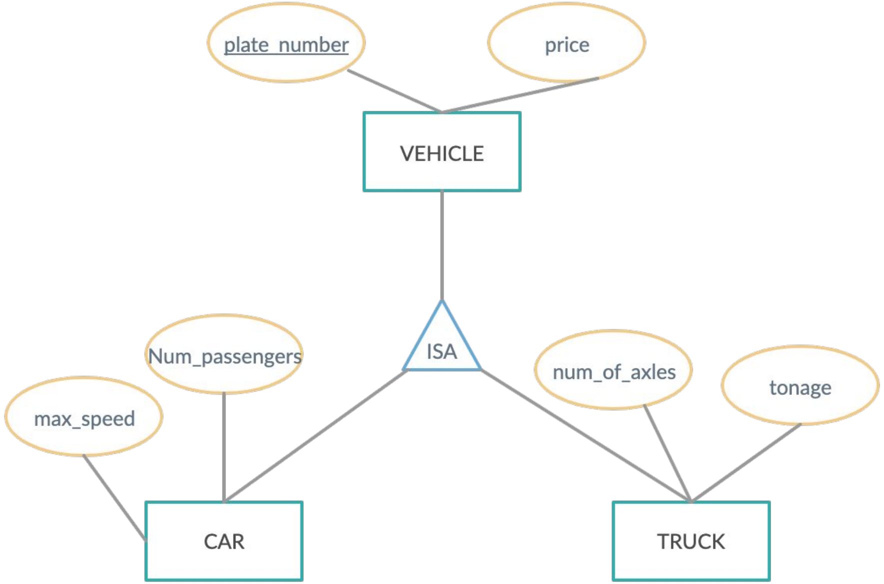


3.9. Translation of the E-R Model to the Relational Model.

Generalization (we don't have vehicles that aren't a truck or a car):

CAR (<u>plate_number</u>, price, max_speed, num_passengers)

TRUCK (<u>plate_number</u>, price, num_of_axles, tonage)

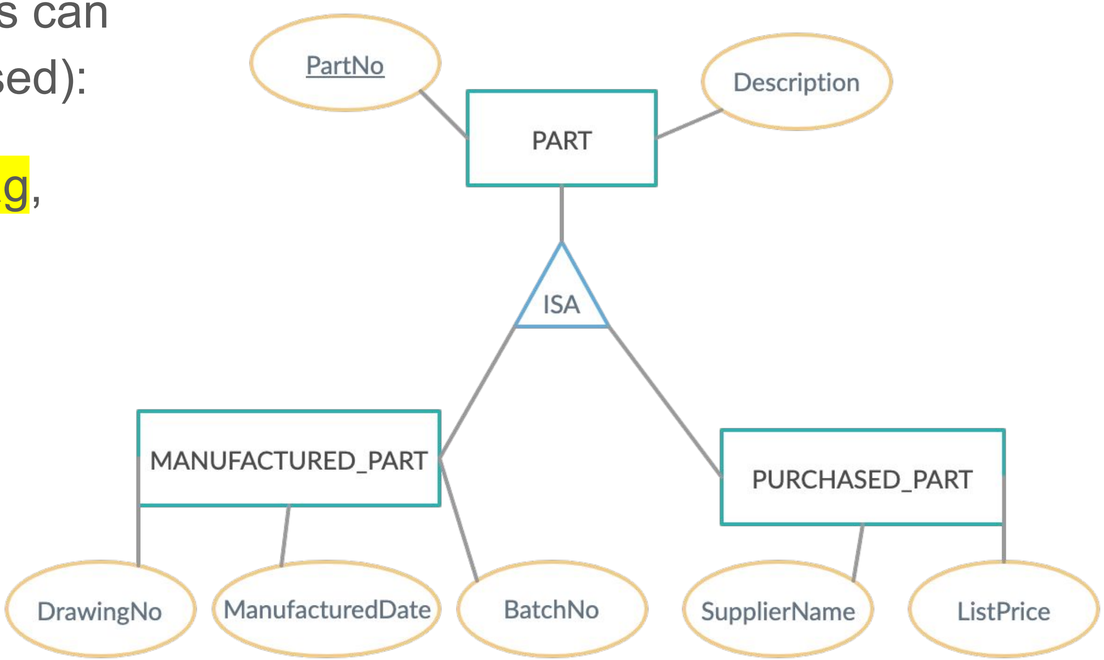


3.9. Translation of the E-R Model to the Relational Model.

Specialization (overlapping parts can be both manufactures or purchased):

PART (<u>PartNo</u>, Description, MFlag, DrawingNo, ManufactureDate, BatchNo, PFlag, SupplierName, ListPrice)

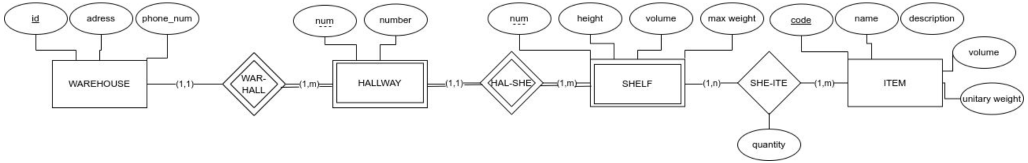
- MFlag is true if the part is manufactured.
- PFlag is true if the part is purchased.



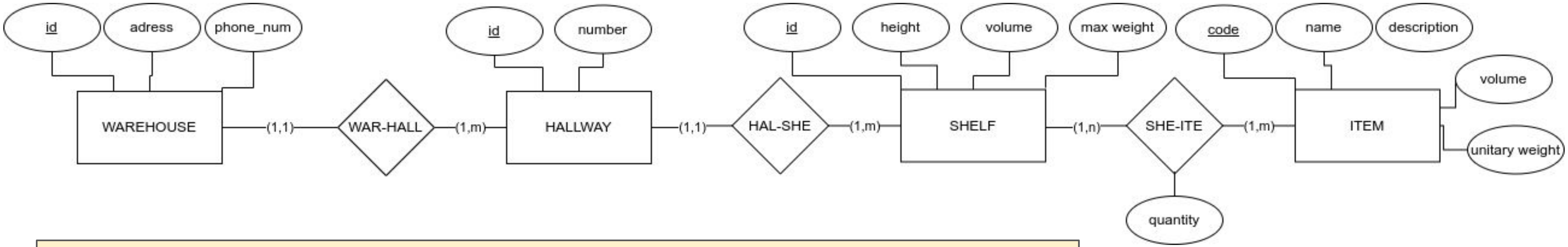
3.9. Translation of the E-R Model to the Relational Model.

See:

- **http://tinman.cs.gsu.edu/~raj/4710/sp03/ch9-part1.pdf**



3.10. Some examples.



WAREHOUSES (id, address, phone_num)
HALLWAYS (id, number, id_warehouse*)
SHELVES (id, height, volume, max_weight, id_hall*)
ITEMS (code, name, description, volume, unitary_weight)
SHE-ITE (id_shelf*, code_item*, quantity)

WAREHOUSES (id, address, phone_num)
HALLWAYS (id_warehouse*, num, number)
SHELVES (id_warehouse*, num_hallway*, num, height, volume, max_weight)
ITEMS (code, name, description, volume, unitary_weight)
SHE-ITE (id_warehouse*, num_hallway*, num*, code_item*, quantity)

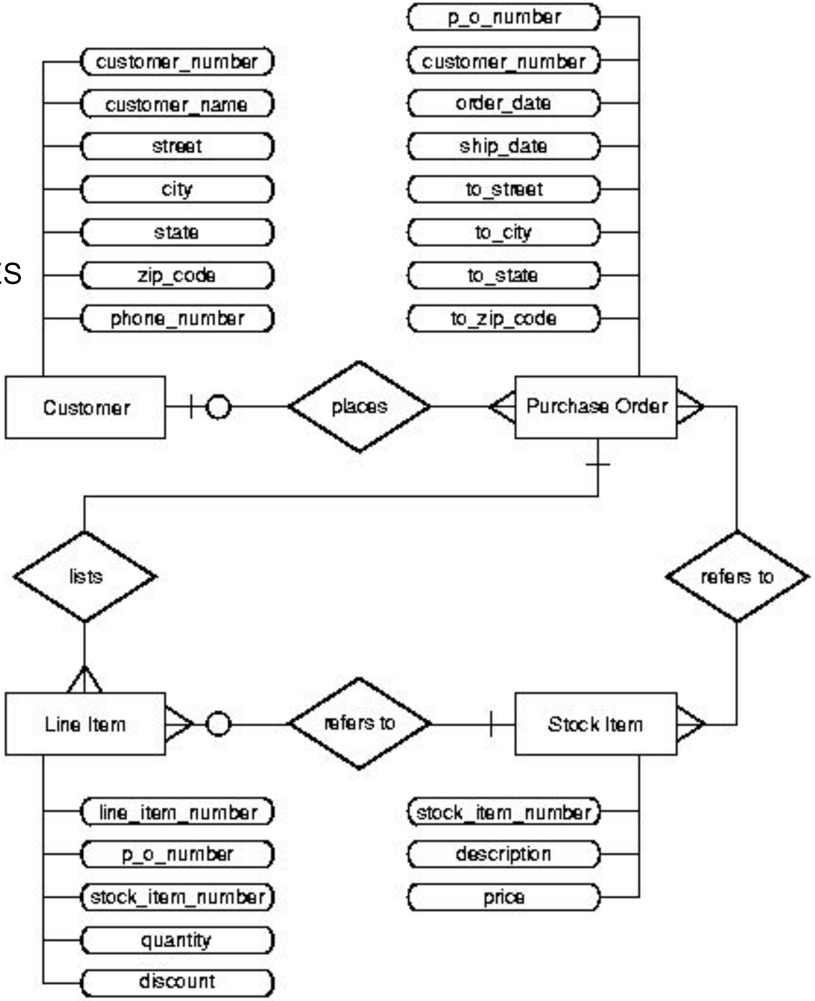
3.10. Some examples.

```
CREATE TABLE Customers (
  CustNo   NUMBER(3) NOT NULL,
  CustName VARCHAR2(30) NOT NULL,
  Street   VARCHAR2(20) NOT NULL,
  City     VARCHAR2(20) NOT NULL,
  State    CHAR(2) NOT NULL,
  Zip      VARCHAR2(10) NOT NULL,
  Phone    VARCHAR2(12),
  PRIMARY KEY (CustNo)
);
```

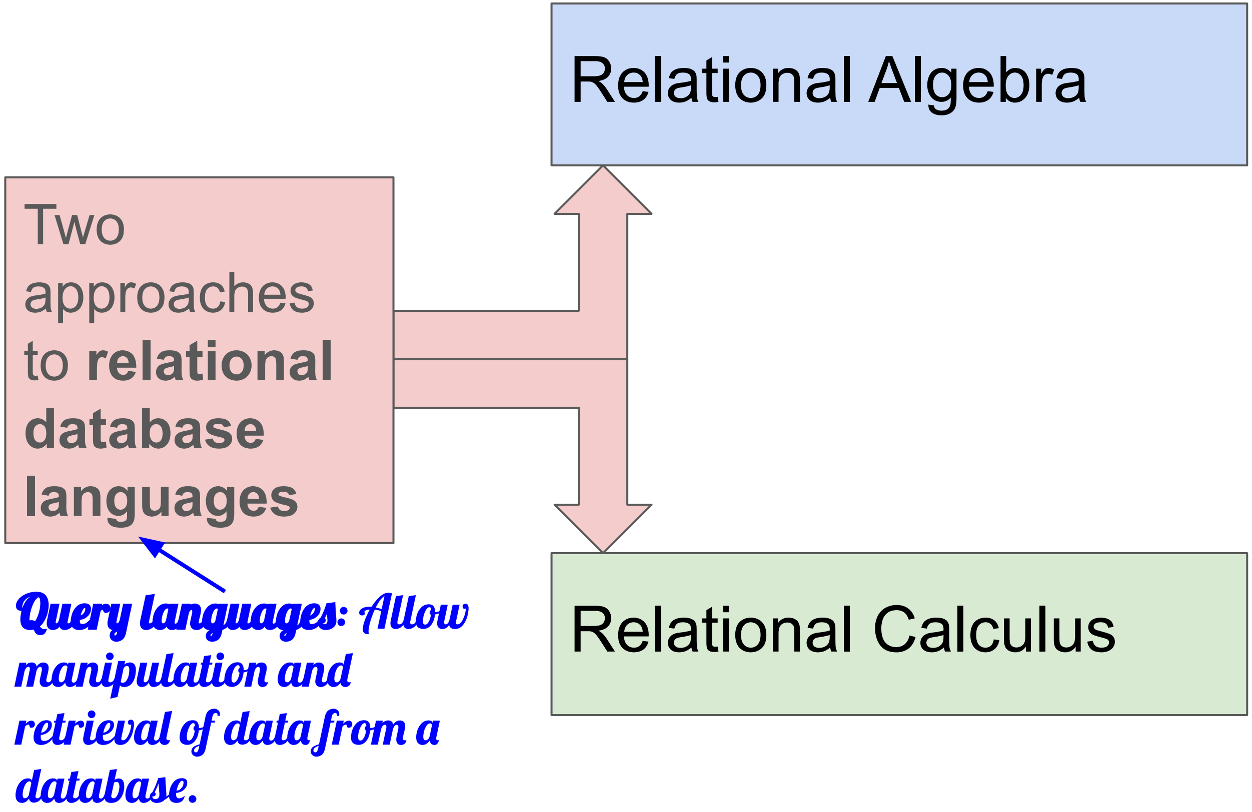
```
CREATE TABLE Orders (
  PONo      NUMBER(5),
  Custno    NUMBER(3) REFERENCES
Customers,
  OrderDate DATE,
  ShipDate  DATE,
  ToStreet  VARCHAR2(20),
  ToCity    VARCHAR2(20),
  ToState   CHAR(2),
  ToZip     VARCHAR2(10),
  PRIMARY KEY (PONo)
);
```

```
CREATE TABLE LineItems (
  LineNo    NUMBER(2),
  PONo      NUMBER(5) REFERENCES Orders,
  StockNo   NUMBER(4) REFERENCES StockItems,
  Quantity  NUMBER(2),
  Discount  NUMBER(4,2),
  PRIMARY KEY (LineNo, PONo)
);
```

```
CREATE TABLE StockItems (
  StockNo     NUMBER(4) PRIMARY KEY,
  Description VARCHAR2(20),
  Price       NUMBER(6,2))
);
```



3.11. What's relational algebra and relational calculus?



3.11. What's relational algebra and relational calculus?

Relational Algebra (RA):

It’s not a specific language of a certain database implementation.

RA are specifications defined by **Codd (1972)** about what should be a **language of manipulation for a RDBMS**.

RA consists of a **collection of operators on relations** (divided into two groups):

Traditional set operators:

- union
- intersection
- difference (not commutative)
- cartesian product

Special relational operators:

- selection
- projection
- join
- division

3.12. Relational algebra.

Closed set:

The result of each of the operations is another relation.

That's the reason why the result of an operation can be converted into an operand:

$R_{final} = R_1 \text{ op } (R_2 \text{ op } (R_3 \text{ op } R_4))$

3.12. Relational algebra.

Compatible relations:

Two relations (or more) are compatible if they have the **same degree** (number of attributes or columns) and if their **attributes are compatible 2 to 2**, that is, that they belong to the same domain (their order is not relevant).

For instance, if we have the relations, A (a, b, c) and B (x, y, z), A and B are two compatible relations if (a, x) (b, y) (c, z) or (a, y) (b, x) (c, z) or ... belong to the **same domain**.

Union, intersection and difference operations can **only** be applied to **compatible relations**.

3.12. Relational algebra.

A	S#	NAME	STATUS	CITY
	S1	Smith	Single	London
	S2	Clark	Married	London

B	S#	NAME	STATUS	CITY
	S1	Smith	Single	London
	S3	Jones	Single	Paris

A **UNION** B

S#	NAME	STATUS	CITY
S1	Smith	Single	London
S2	Clark	Married	London
S3	Jones	Single	Paris

A **INTERSECTION** B

S#	NAME	STATUS	CITY
S1	Smith	Single	London

A **MINUS** B

S#	NAME	STATUS	CITY
S2	Clark	Married	London

B **MINUS** A

S#	NAME	STATUS	CITY
S3	Jones	Single	Paris

3.12. Relational algebra.

A	a	b	c
	a	1	M
	a	2	N
	b	1	M

B	x	y	z
	c	2	P
	d	1	P
	a	2	N
	a	1	M
	a	1	N

A **INTERSECTION** B

a	b	c
a	1	M
a	2	N

A **MINUS** B

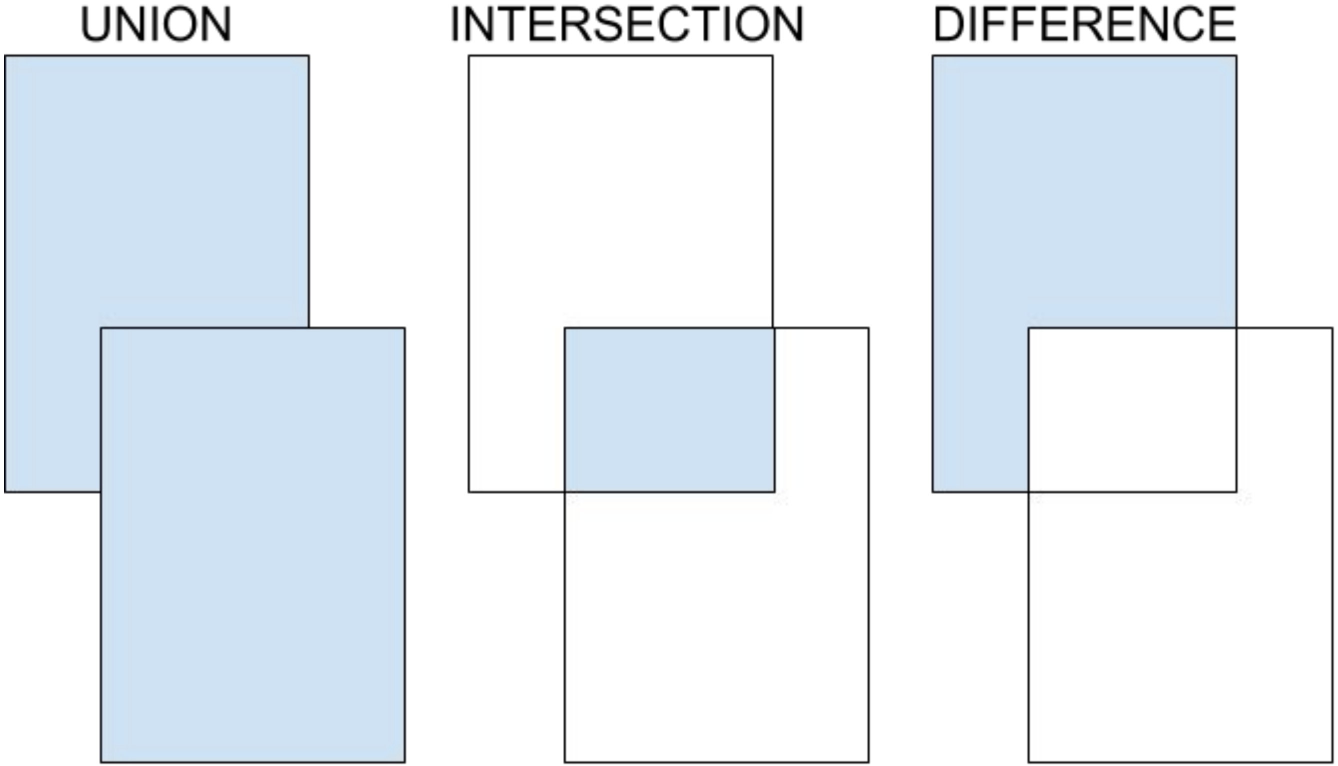
a	b	c
b	1	M

A **UNION** B

a	b	c
a	1	M
a	2	N
b	1	M
c	2	P
d	1	P
a	1	N

B **MINUS** A ?

3.12. Relational algebra.



3.12. Relational algebra.

E	**CITY**
	Paris
	Rio

F	**NAME**
	Gates
	Siegel

F **CARTESIAN PRODUCT** E

NAME	**CITY**
Gates	Paris
Gates	Rio
Siegel	Paris
Siegel	Rio

3.12. Relational algebra.

A	a	b	c
	a	1	M
	a	2	N

B	x	y	z
	c	2	P
	d	1	P
	a	2	N
	a	1	M

A **CARTESIAN PRODUCT** B

a	b	c	x	y	z
a	1	M	c	2	P
a	2	N	c	2	P
a	1	M	d	1	P
a	2	N	d	1	P
a	1	M	a	2	N
a	2	N	a	2	N
a	1	M	a	1	M
a	2	N	a	1	M

3.12. Relational algebra.

A	S#	NAME	STATUS	CITY
	S1	Smith	Single	London
	S2	Clark	Married	London

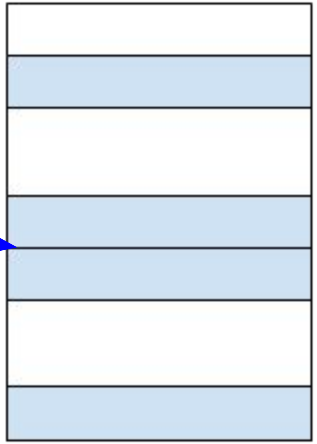
B	S#	NAME	STATUS	CITY
	S1	Smith	Single	London
	S3	Jones	Single	Paris

A WHERE STATUS = 'SINGLE'

S#	NAME	STATUS	CITY
S1	Smith	Single	London

People in relation A who is single!

SELECTION



Horizontal cuts!

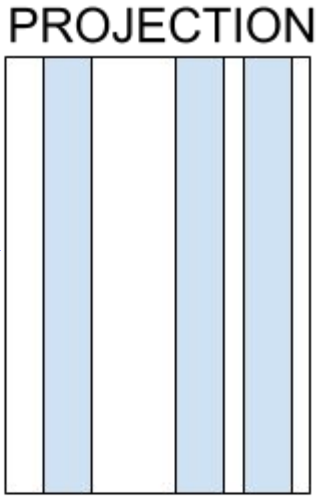
3.12. Relational algebra.

B	x	y	z
	c	2	P
	d	1	P
	a	2	N
	a	1	M
	a	1	N

Projection over relation B (atributes x, y)

x	y
c	2
d	1
a	2
a	1

Vertical cuts!



JOIN: Subset of cartesian product.

3.12. Relational algebra.

A	S#	NAME	STATUS	CITY
	S1	Smith	Single	London
	S2	Clark	Married	London

B	S#	NAME	STATUS	CITY
	S1	Smith	Single	London
	S3	Jones	Single	Paris

C	STATUS	DREAM_CITY
	Single	Paris
	Single	Rio
	Married	Pitt
	Married	Orlando

Cities people dream about

A NATURAL_JOIN C

S#	NAME	STATUS	CITY	DREAM_CITY
S1	Smith	Single	London	Paris
S1	Smith	Single	London	Rio
S2	Clark	Married	London	Pitt
S2	Clark	Married	London	Orlando

(A NATURAL_JOIN C) WHERE CITY=DREAM_CITY

S#	NAME	STATUS	CITY	DREAM_CITY
-				

People who live in dream city

B NATURAL_JOIN C

S#	NAME	STATUS	CITY	DREAM_CITY
S1	Smith	Single	London	Paris
S1	Smith	Single	London	Rio
S3	Jones	Single	Paris	Paris
S3	Jones	Single	Paris	Rio

(B NATURAL_JOIN C) WHERE CITY=DREAM_CITY

S#	NAME	STATUS	CITY	DREAM_CITY
S3	Jones	Single	Paris	Paris

3.12. Relational algebra.

D	NAME	CITY
	Gates	Seattle
	Gates	N.Y.C.
	Siegel	Paris
	Siegel	N.Y.C.
	Siegel	Rio

People who own houses in cities

E	CITY
	Paris
	RIo

People who own houses in cities

D DIVIDE_BY E
Name
Siegel

People who own houses in every dream city

3.12. Relational algebra (examples).

DRUG (<u>drug_id</u>, name, price, lab_id*)

APPLICATION (app_id, name)

APP_DRUG (app_id*, drug_id*)

LABORATORY (lab_id, name)

SALE (sale_num, date, drug_id*, quantity)

3.12. Relational algebra (examples).

1.- Name of applications of drug X:

R1 = SELECTION APP_DRUG (drug_id = X)
R2 = JOIN (R1, APPLICATION) (R1.app_id = APPLICATION.app_id)
R3 = PROJECTION (APPLICATION.name) R2
The same with only a statement:

```
R3 = PROJECTION (APPLICATION.name) (
     JOIN (APP_DRUG, APPLICATION)
     (drug_id = X and
     APP_DRUG.app_id = APPLICATION.app_id)
```

3.12. Relational algebra (examples).

2.- Name of laboratory that provides drug X:

```
R1 = JOIN (DRUG, LABORATORY)
     (drug_id = X and
     DRUG.lab_id = LABORATORY.lab_id)
R2 = PROJECTION (LABORATORY.name) R1
```

The same with only a statement:

```
R2 = PROJECTION (LABORATORY.name) (
     JOIN (DRUG, LABORATORY)
     (drug_id = X and
     DRUG.lab_id = LABORATORY.codi_lab)
```

3.12. Relational algebra (examples).

3.- Name and price of drug provided by Y laboratory, which are for application K:

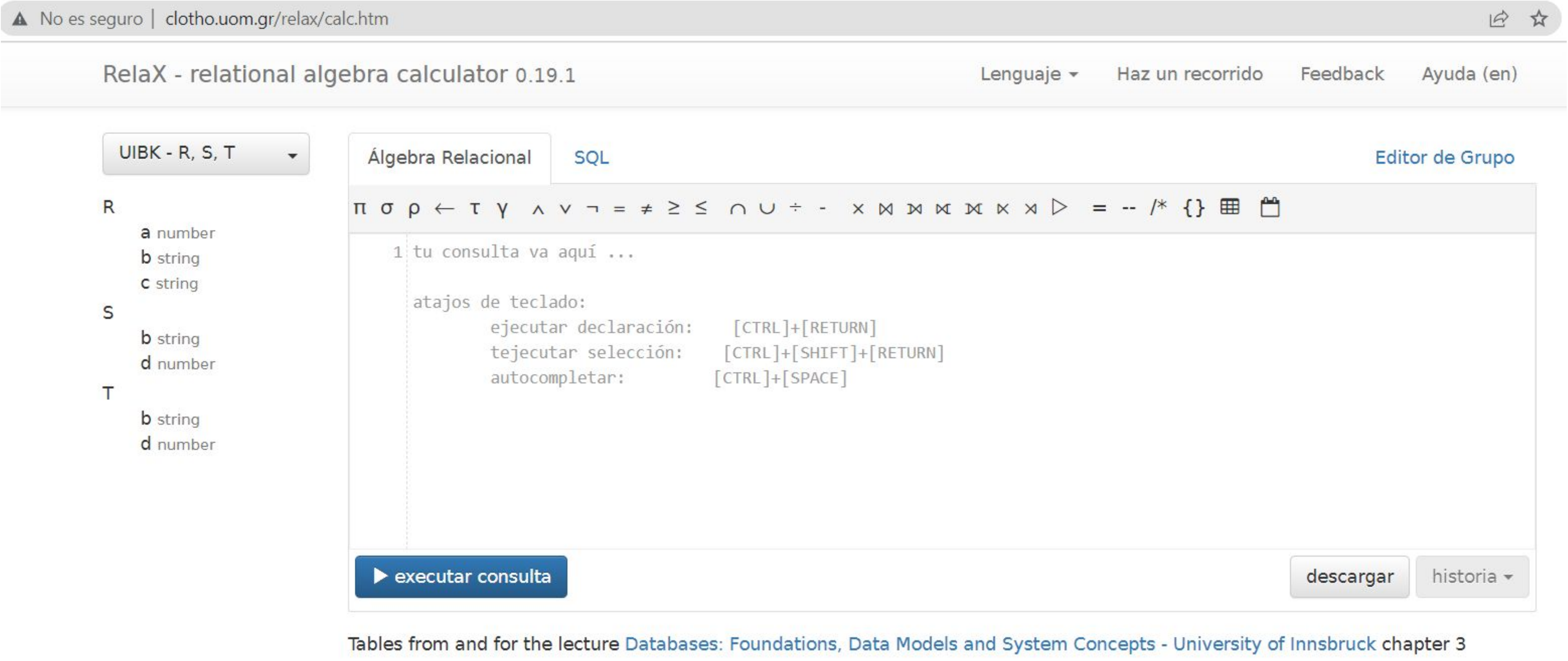
```
R1 = PROJECTION (DRUG.name, DRUG.price)
     JOIN (DRUG, APP_DRUG)
          (lab_id = Y and
          app_id = K and
          DRUG.drug_id = APP_DRUG.drug_id))
```

3.12. Relational algebra (examples).

4.- Name of drugs with a price greater than 1000 and that has not sold any since the date D:

```
R1 = PROJECTION (DRUG_ID) (
     SELECTION SALE (date >= D))
R2 = PROJECTION (drug_id) (
     SELECTION DRUG (price > 1000)
R3 = R2 MINUS R1
R4 = PROJECTION (DRUG.name)
     JOIN (R3, DRUG)
     (R3.drug_id = DRUG.drug_id))
```

3.12. Relational algebra (RelaX).



3.13. Relational calculus.

Alternative to relational algebra as DML.

Relational Algebra oriented to relations but Relational Calculus is oriented to:

tuples

domains

Check this pdf file.

3.13. Relational calculus.

1.- Name of applications of drug X:

```
FOR AD IN APP_DRUG WITH
    (AD.drug_id = X)
    FOR A IN APPLICATION WITH
        (A.app_id = AD.app_id)
        PRINT A.name
    END-FOR
END-FOR
```

3.13. Relational calculus.

2.- Name of laboratory that provides drug X:

```
FOR D IN DRUG WITH
    (D.drug_id = X)
    FOR L IN LABORATORY WITH
        (L.lab_id = D.lab_id)
        PRINT L.name
    END-FOR
END-FOR
```

3.13. Relational calculus.

3.- Name and price of drug provided by Y laboratory, which are for application K:

```
FOR D IN DRUG WITH
    (D.lab_id = Y)
    FOR AD IN APP_DRUG WITH
        (AD.app_id = K)
        PRINT D.name, D.price
    END-FOR
END-FOR
```

3.13. Relational calculus.

4.- Name of drugs with a price greater than 1000 and that has not sold any since the date D:

```
FOR D IN DRUG WITH
      (D.price > 1000 AND
(NOT ANY S IN SALE WITH
      S.drug_id = D.drug_id))
PRINT D.name
END-FOR
```

3.13. Relational calculus.

- More information: http://www.csbio.unc.edu/mcmillan/Media/Comp521F12Lecture05.pdf

Sources.

- http://people.cs.pitt.edu/~chang/156/04reldb.html
- **M. J. Ramos, A. Ramos and F. Montero.** Sistemas gestores de bases de datos. McGrawHill: 1th Edition, 2006.
- **Abraham Silberschatz, Henry F. Korth and S. Sudarshan**. *Database System Concepts.* McGrawHill: 6th Edition, 2010.
- Apunts de la UIB del professor Miquel Manresa (1996).
- https://www.guru99.com/relational-data-model-dbms.html
- https://www.stat.berkeley.edu/~nolan/stat133/Fall05/lectures/dbmsMultiTable.pdf
- https://people.cs.pitt.edu/~chang/156/05algebra.html
- https://www.javatpoint.com/