Database Management Systems

Lorraine Goeuriot
IUT 1, Université Grenoble Alpes
Auteur du cours : Marie-Christine Fauvet
Université Joseph Fourier, Grenoble – UFR IM^{2}AG

6 janvier 2016

Contenu

1. Introduction
   - Organisation
   - Content of the Module
   - Database Management Systems (DBMS)
2. The Entity-Relationship Model
3. Modelization

Teaching Team

- Lorraine Goeuriot (IUT1 - UGA, LIG)
  Lectures
  lorraine.goeuriot@ujf-grenoble.fr
- Michael Magi (UFR IM2AG - UGA)
  Labs
  michael.magi@ujf-grenoble.fr

Reference : Ramakrishnan R. and Gehrke J. (2000). *Database Management Systems*. http://pages.cs.wisc.edu/~dbbook/

Evaluation

Project (work in pairs)	p
Exam	ex
Final mark	ex*0.5 + p*0.5

Objectives

- Master the relational model and SQL
- Understand Database Management Systems, and their integration within a Web application

Lectures overview :

1. Database design :
   - The Entity-Relationship model
   - The Relational Model
2. A Relational Language : SQL
   - Data Creation/Modification
   - Data Access
3. Web Programming : HTML and PHP

What's a Database ?

A **database** is a collection of data, typically describing the activities of one or more related organizations. For example, a university database might contain information about the following :

- *Entities* such as students, faculty, courses.
- *Relationships* between entities, such as students' enrollment in courses, faculty teaching courses, and the use of rooms for course.

A database contains a structured representation of information :

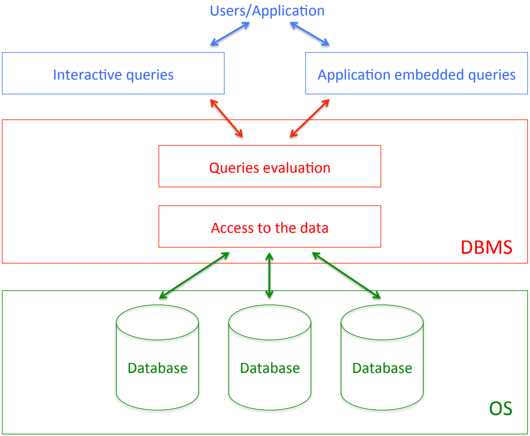
- modelling some real world information
- created for a specific purpose

Examples :

- My address book, containing name, address, and phone number of my friends
- Youtube database : over a billion users, at least 45TB of videos

Database Management Systems (DBMS)

A Database Management System (DBMS) is a software designed to assist in maintaining and utilizing large collections of data.



Advantages of a DBMS

DBMS present the following advantages :

- Data independence (from the application)
- Efficient data access (to store and retrieve)
- Data integrity and security (constant data integrity constraint, classes of users)
- Data administration (organization of data, users...)
- Concurrent access and crash recovery (access for multiple users, system failures)
- Reduced application development time

DBMS characteristics - Summary

DBMS allow to store and access data :

- in a shared context : concurrent access by multiple users
- in an instable context : able to deal with system failures

To DBMS users, managing a database seems :

- Private : it is not affected by other users
- Stable : it is not affected by system problems and failures

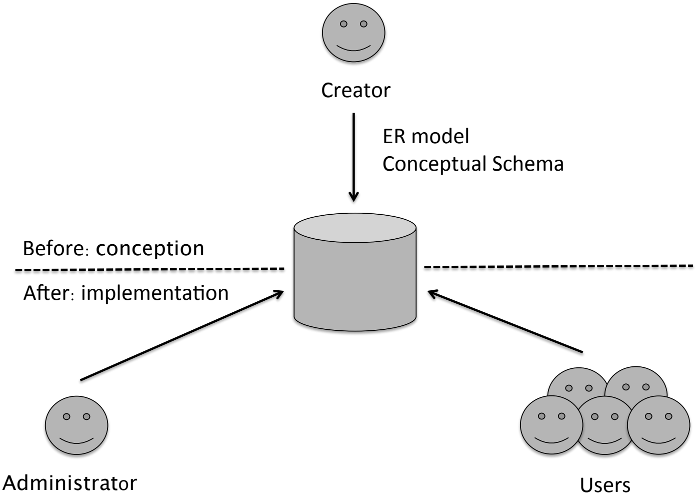
Contenu

1 Introduction

2 The Entity-Relationship Model
- Introduction
- Entity-relationship Model

3 Modelization

Databases Professions (1)



Databases Professions (2)

1. Role of the creator
   - Identify needs (with users)
   - Identify data to be stored
   - Choose a structure
   - Create tables and views
2. Role of the administrator
   - Server management
   - Access rights management
   - Coordinate and monitors DB's use
   - Security management
   - Maintenance (system failures, backup...)
3. Role of the user
   - Query the DB
   - Data update
   - Report management

Designing DBs (1)

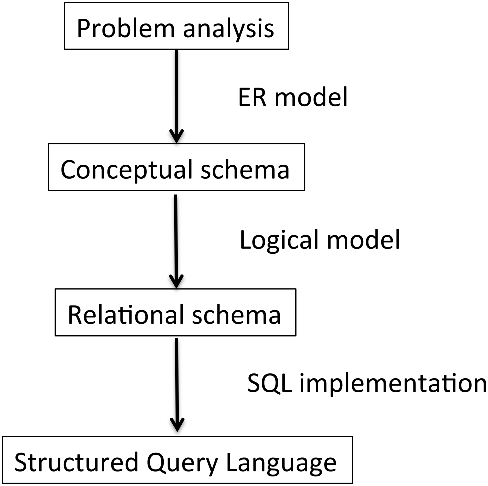
Description $\neq$ data

The structure of the database (schema) should be distinguished from the data (content)

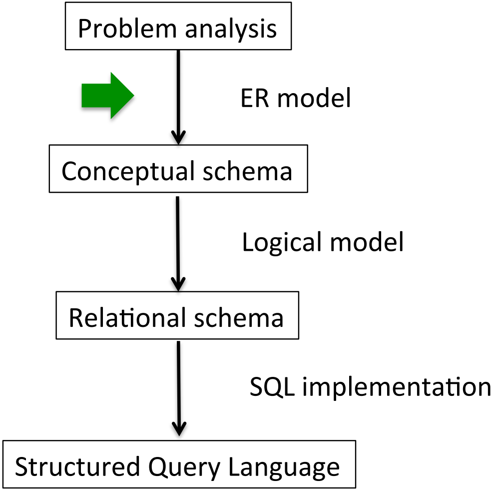
A DB's schema :

- Describes its structure (not its content)
- Is specified at the design stage :
  - Optimization of the size, space, redundancy
  - Optimization of data update

Designing DBs (2)

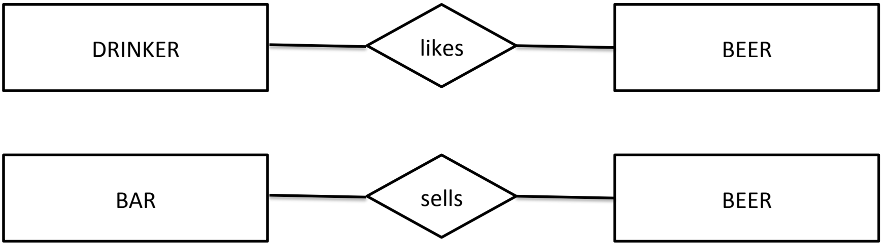


Conception des Bds (2)



Entity-relationship Model

- Static and high-level description of the data model
- Used to describe information needs while establishing the specifications
- Based on 2 concepts : the entities and the relationships



Entities and Relationships

The entities are :

- Modelization of real world things
- Abstract or concrete concepts
- Independent from one another
- Examples :
  - the bar *Australia Hotel*
  - the beer *Pale Ale*
  - *Jean Dupont* on facebook

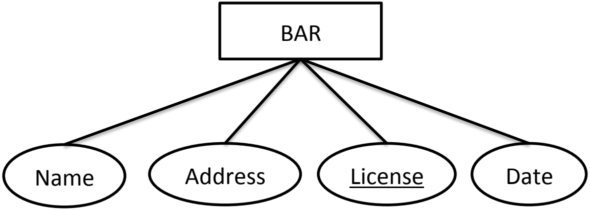
The relationships are :

- A link between 2 entities
- Characterized by a verb
- Examples :
  - a bar sells a beer
  - a facebook user likes a music band
  - a facebook user is friend with another user

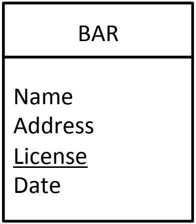
The entities

- The **type** of an entity is an abstraction associated to similar entities (similarly to types in programming languages).
  Examples :
  - the entity *Australia Hotel* has the type *BAR*
  - the entity *Pale Ale* has the type *BEER*
  - the entity *Jean Dupont* has the type *FACEBOOK USER*
- An entity type has *attributes* or *properties*. Example : a *BAR* has the attributes *name*, *address*, *license number* and *open date*
- The **key** of an entity type is one or a set of attributes that uniquely identify each entity from this type. They are usually underlined in the model. Examples : email, student ID, social security number...

Entity Representation



TREE REPRESENTATION



UML REPRESENTATION

The Attributes

Attributes value : each entity has specific values for each attribute

Example : Entity "Australia Hotel"

- Name : *Australia Hotel*
- Address : *The Rocks*
- License : *123456*
- Opening date : *12/01/1940*

Attributes domain : set of possible values.

Example :

- Enumeration : Dom(gender) $=$ {F, M}
- Interval : Dom(mark) $=$ [0, 20]
- Classical types : *string*, *int*, *float*...

Attributes Properties

Simple vs composite : composite attributes can be divided in several simple ones. Example :

- postcode = 38000
- address = (12, "rue de Verdun", 38000, "Grenoble")

Single or multile valued : single-valued attributes have only one value, while multiple-valued ones can have several values. Example :

- age = 25
- diplomas = bac, licence, master

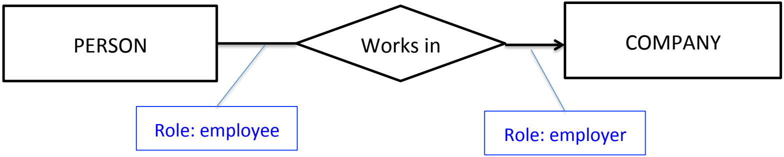
Dérived or not : a derived attribute is computed from another one. Example : the age of a bar can be calculated from its opening date.

Optional vs mandatory : optional attributes are used when the value is NA or unknown. The value becomes *NULL*. Example :

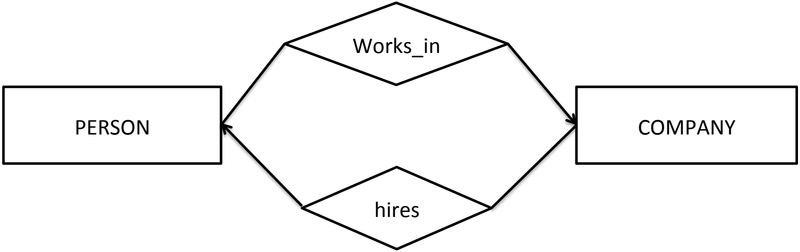
- Maiden name
- Mobile number

Relationship Types

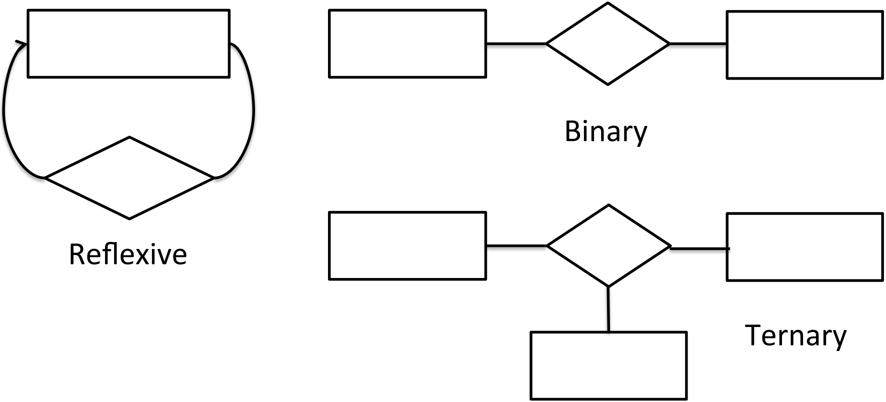
Representation :



Inverse relationship type :

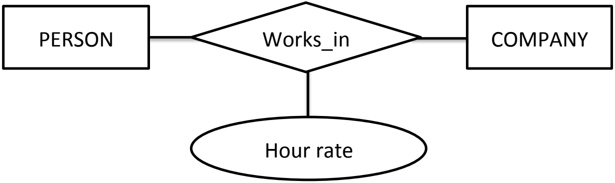


Relationship type Degree



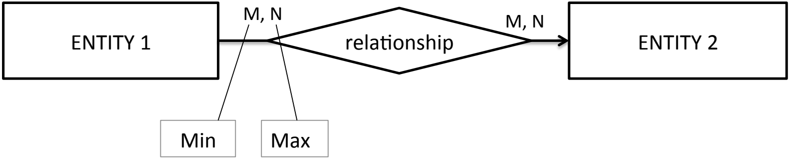
Relationship types : attributes and cardinality

The attributes are properties of the relationship types.

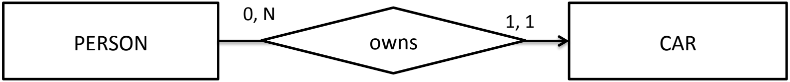


The cardinality characterize the link between the entity and the relationship. It defines the lower and upper bounds (minimum and maximum number of entities taking part in the relationship).

Relationships Cardinality (1)

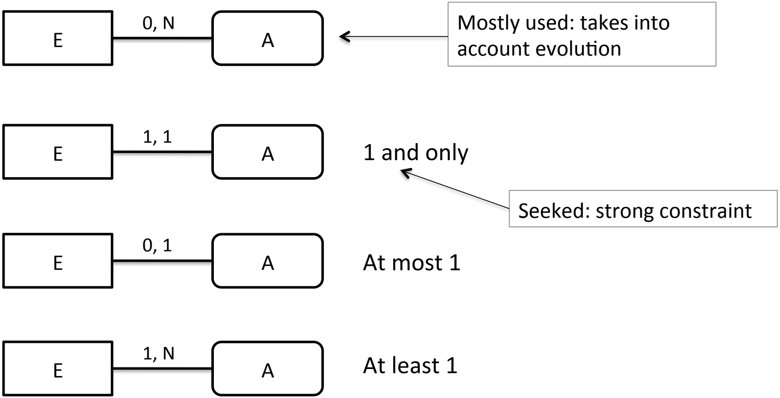


Example

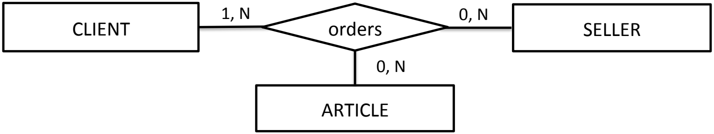


A person can have 0 to N cars, a car has one and only owner.

Relationships Cardinality (2)



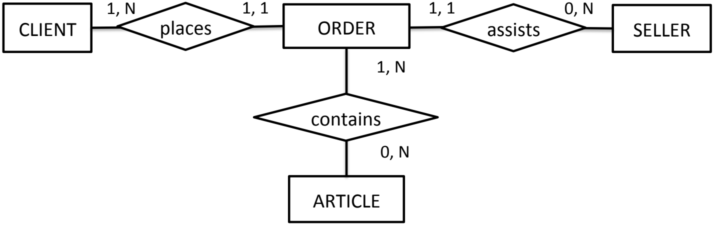
Ternary Relationships



A client must have ordered once, orders are taken through sellers.

N-ary relationships (n>2) should be avoided.

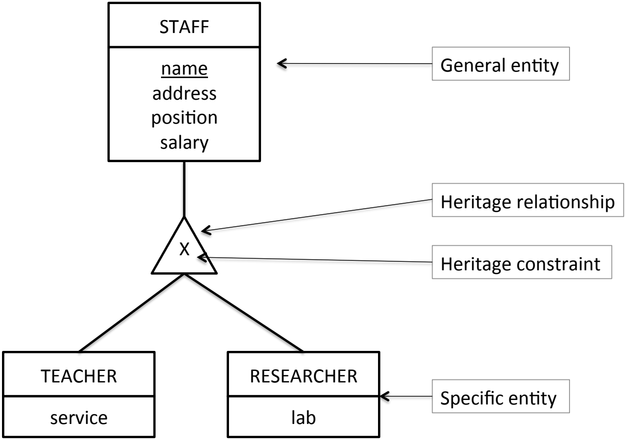
BETTER:



Heritage - Definition

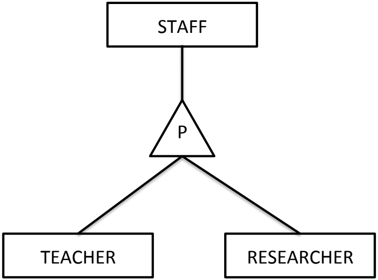
- Heritage allows to *hierarchically organize* entities
- E1 inherits from E2 if E2 is a sort of E1.
  - E1 : general entity
  - E2 : specific entity
- E2 inherits E1 characteristics
- (similarly to OOP)

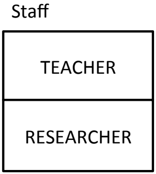
Heritage - Representation



Heritage - Constraints (1)

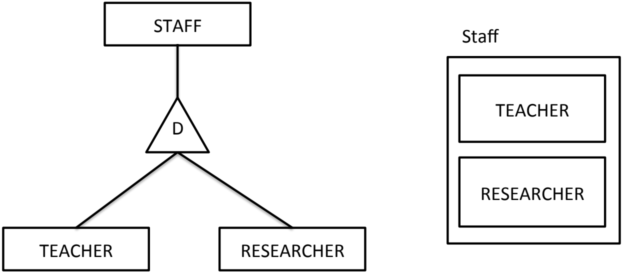
Partition constraint : each entity belonging to the general entity belongs to only one of the specific entity.





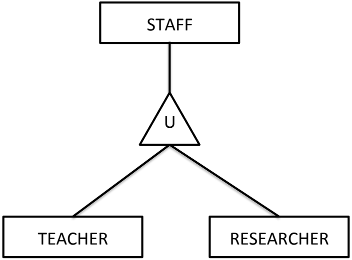
Heritage - Constraints (2)

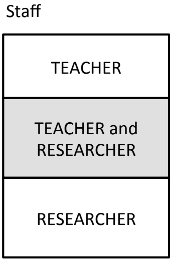
Disjunction constraint : each entity belonging to the general entity belongs to only one of the specific entity or none.



Heritage - Constraints (3)

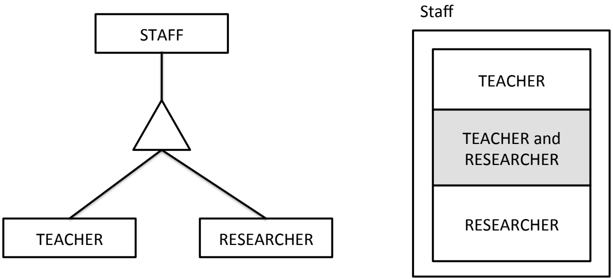
Union constrait : each entity belonging to the general entity belongs to one of the specific entity or both.





Heritage - Constraints (4)

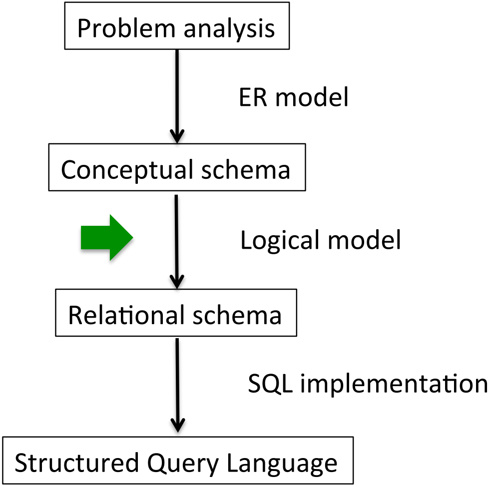
No constraint



Contenu

1. Introduction
2. The Entity-Relationship Model
3. Modelization
   - Modelization of a DB

Designing DBs (2)



Relation - Definition

The main construct for representing data in the relational model is a **relation**. A relation consists of a **relation schema** and a **relation instance** (tuples). The relation instance is a table, and the relation schema describes the column heads for the table.

Example :

Name of the relation

Fields (attributes, columns)

CONSULTATION	NumCons	DateCons
	6	16/12/11
	2	11/12/11
	3	13/12/11

Tupes
(records, rows)

Relation Schema

The **relation schema** :

- is noted $R(X, Y, Z)$
- R is the name of the relation
- X, Y, Z are its attributes

Example :

CONSULTATION(Numcons, DateCons)

Name of the relation *Key* *Attributes*

Database Schema

The **database schema** is the set of relations of the DB.
Example :

- Beers (<u>name</u>, manf)
- Bars (<u>name</u>, addr, license, openDate)
- Drinkers (<u>name</u>, addr, phone)
- Likes (<u>drinker</u>, <u>beer</u>)
- Sells (<u>bar</u>, <u>beer</u>, price)
- Frequents (<u>drinker</u>, <u>bar</u>)

Functional Dependencies

- In the relation $R(X, Y, Z)$, the attribute X functionally determines the attribute Y ($X \rightarrow Y$) if, and only if :
  - each X value is associated with precisely one Y value, independently from Z
  - The existence of $R(X, Y, Z)$ and $R(X, Y', Z')$ implies $Y = Y'$
- Examples :
  - in $Bars$, $name \rightarrow addr, license, openDate$
  - in $Sells$, $bar, beer \rightarrow price$

Functional Dependencies

- Given the following relation R, with the schema $R(A, B, C, D, E)$:

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b2	c2	d2	e1
a2	b1	c3	d3	e1
a2	b1	c4	d3	e1
a3	b2	c5	d1	e1

- Exercise : Which functional dependencies are satisfied by R ?

Functional Dependencies

- Given the following relation R, with the schema $R(A, B, C, D, E)$:

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b2	c2	d2	e1
a2	b1	c3	d3	e1
a2	b1	c4	d3	e1
a3	b2	c5	d1	e1

- Exercise : Which functional dependencies are satisfied by R ?
- Answer :
  $A \rightarrow E$; $B \rightarrow E$; $C \rightarrow ABDE$; $D \rightarrow E$; $AB \rightarrow D$; $AD \rightarrow B$; $BD \rightarrow A$.

Primary Functional Dependency

$X \rightarrow Y$ is a **primary** DF, if and only if : there is no subset of X satisfying a DF with Y.

Examples :

- name, addr, license$\rightarrow$ openDate is not a PDF as

$$name \rightarrow openDate$$

- bar, beer $\rightarrow$ price is a PDF as we do not have :
  - bar $\rightarrow$ price
  - beer $\rightarrow$ price

Relation Key

X is a key in $R(X, Y, Z)$ if and only if :
$X \rightarrow Y, Z$ is a PDF

Examples :

- $name \rightarrow addr, license, openDate$ in $Bars(name, addr, license, openDate)$
- $name \rightarrow addr, phone$ in $Drinkers(name, addr, phone)$
- $bar, beer \rightarrow price$ in $Sells(bar, beer, price)$

Primary Key

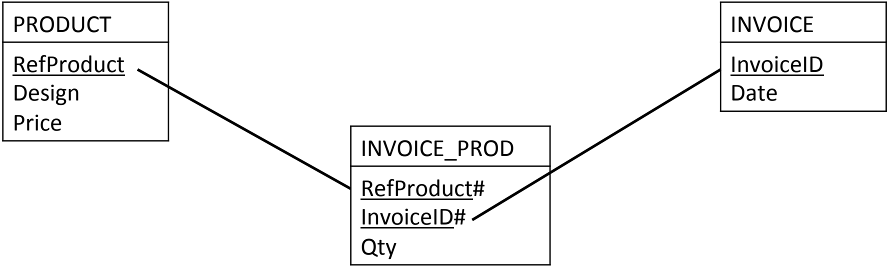
Minimal set of attributes allowing unique identification

- No NULL value
- There should be only one primary key in a relation
- It can sometimes be a sequential number

Foreign Key

- Attribute(s) representing the primary key of another relation
  - Contains only values of the primary key
  - Can be the primary key
  - Should specify if NULL values are allowed
  - Composite if the primary key is composite
- Notation : name followed by # and reference

Foreign Key - Example



InvoiceID references INVOICE.InvoiceID
RefProduct references PRODUCT.RefProduct

Data Integrity - Constraints

- **Integrity constraints** allow to restrict a database update/modification
- It keeps the DB consistent with the schema initially defined
- There are several constraint types :
  - Elementary integrity constraints
  - Intra relation integrity constraints
  - Inter relation integrity constraints
  - Dynamic constraints

Elementary integrity constraints

- Related to a specific attribute
- Allow to restrict :
  - its domain
  - if it can be NULL
- Examples :
  - Registration plate : not NULL, format "XX-000-XX"
  - Year of birth of a user : from 1900 to today
  - User's gender : F or M

Intra- and inter-relation integrity constraints

Intra-relation integrity constraints :

- Specifies dependencies between attributes or entities **within a same relation**
- Key uniqueness is an intra-relation integrity constraint
- Examples :
  - Registration plate is a primary key
  - Departure time $>$ Arrival time

Inter-relation integrity constraints :

- Specifies dependencies between attributes or entities **across different relations**
- Foreign keys : values are restricted through the key values of another relation
- Examples :
  - *bar* and *beer* are foreign keys in relation *Sells*
  - *drinker* and *beer* are foreign keys in relation *Likes*

Dynamic Constraints

- If the value of an attribute gets modified, this change can automatically (or not) be echoed on the value of the attributes it is a foreign key with.
- If an entity is deleted, all the linked entities can automatically (or not) be deleted too.

Modelization of a DB - Process

- Analysis of the problem : existing documents, users needs
- Transformation into elementary steps :

  SUBJET - VERB - COMPLEMENT

- Rules :
  - Subjet $\rightarrow$ entity
  - Verb $\rightarrow$ relationship
  - Complement $\rightarrow$ characteristics of the subject or another entity

Exercise : Modelization of a Library

Analysis :

- Each member has an ID, a first- and a lastname, and a registration date.
- Items borrowed are books and disks ; they all have an ISBN, a title, and an author.
- Authors have an ID, a first- and a lastname.
- Books have a number of pages and a publication number.
- Disks can either be CDs or DVDs and have a printing date.
- When a member borrows an item, the borrowing date and return date are stored.

Exercise : Modelization of a Library

Identification of the entities and the attributes :

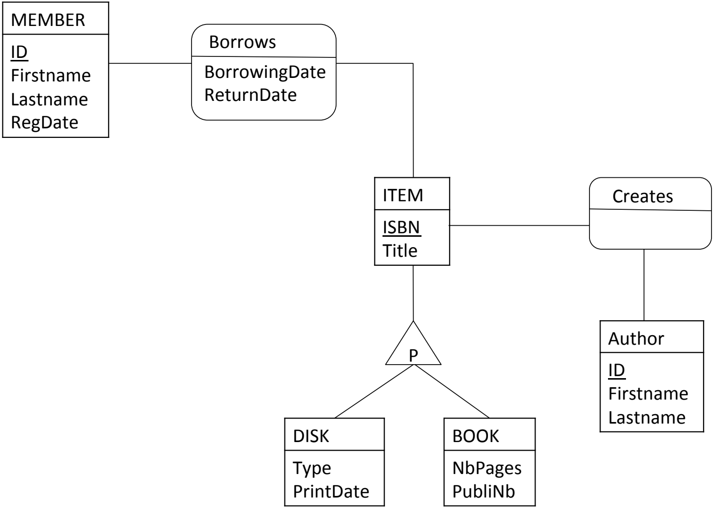
- Each member has an ID, a first- and a lastname, and a registration date.
- Items borrowed are books and disks ; they all have an ISBN, a title, and an author.
- Authors have an ID, a first- and a lastname.
- Books have a number of pages and a publication number.
- Disks can either be CDs or DVDs and have a printing date.
- When a member borrows an item, the borrowing date and return date are stored.

Exercise : Modelization of a Library

Identification of the relationships and their attributes :

- Each member has an ID, a first- and a lastname, and a registration date.
- Items borrowed are books and disks ; they all have an ISBN, a title, and an author.
- Authors creating items an ID, a first- and a lastname.
- Books have a number of pages and a publication number.
- Disks can either be CDs or DVDs and have a printing date.
- When a member borrows an item, the borrowing date and return date are stored.

Exercise : Modelization of a Library



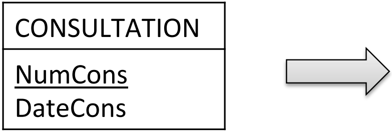
From the ER model to the relational model

- From a graphical representation to a schema
- Simple method following 5 steps

Step 1 - the entities

Creation of a relation for every entity :

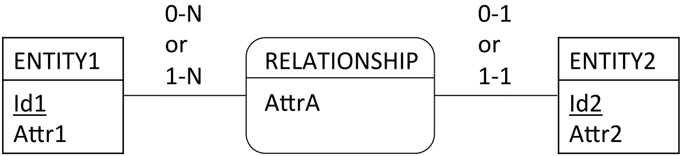
- Single value attributes $\rightarrow$ column
- Identifiers $\rightarrow$ Primary keys



CONSULTATION(NumCons, DateCons)

Step 2 - Simple relationships

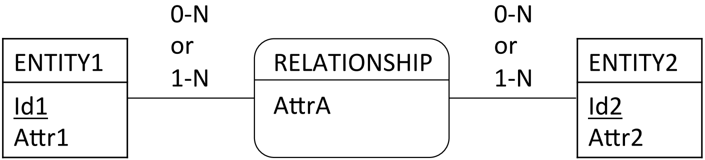
- Simple relationships : with cardinality 0/1-N → 0/1-1



- Keys of *ENTITY*1 and relationship attributes added to ENTITY2
  - ENTITY1 (<u>Id1</u>, Attr1)
  - ENTITY2 (<u>Id2</u>, Attr2, Id1#, AttrA)

Step 3 - Other relationships

- Binary relationships with cardinality 0/1-N $\rightarrow$ 0/1-N :



- Keys of the relationship relation with entity keys and relationship attributes
  - ENTITY1 (<u>Id1</u>, Attr1)
  - ENTITY2 (<u>Id2</u>, Attr2)
  - RELATIONSHIP (Id1#, Id2#, AttrA))

Step 4 - Multiple value attributes

STUDENT
StudentID Name Diplomas

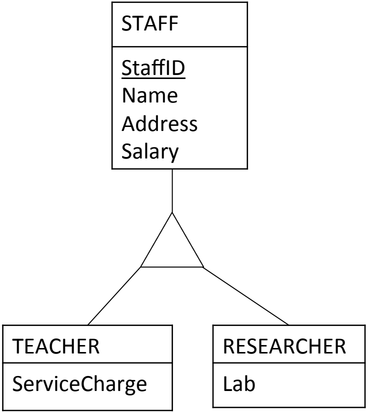
Multiple-valued attribute

Creation of a relation with the entity key (as a foreign key) and the multiple-valued attribute:

- STUDENT (StudentID, name)
- DIPLOMA (StudentID#, Diploma)

Foreign key *Primary key (2 attributes)*

Step 5 - Heritage

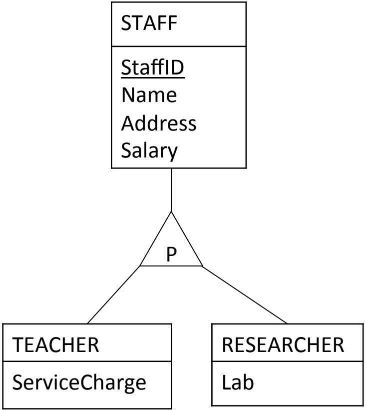


2 methods:

1. Creation of a relation for sub-entities, with the super-entity key:
   - *STAFF (<u>StaffID,</u> Name…)*
   - *TEACHER (<u>StaffID#,</u> ServiceCharge)*
   - *RESEARCHER (<u>StaffID#,</u> Lab)*

2. Creation of a single relation for the super-entity, with sub-entities attributes (possibly NULL)
   STAFF(<u>StaffID,</u> name, …, ServiceCharge, Lab)

Etape 5 - Heritage



In the case of partition constraints, 2 methods:

1. No relation for the super-entity, creation of relations for each sub-entity with super-entity attributes:
   - *TEACHER (<u>StaffID,</u> Name, ..., ServiceCharge)*
   - *RESEARCHER (<u>StaffID,</u> Name, ..., Lab)*

2. Creation of a single relation for the super-entity, with sub-entities attributes (possibly NULL)
   STAFF(<u>StaffID,</u> name, ..., ServiceCharge, Lab)