

II.2414 – Advanced DataBases

Module's responsible:

Hedi YAZID - hedi.yazid@isep.fr

Teachers:

Thibaut de Broca - thibaut@grooptown.com

Alexandra Levchenko - alexandra.levchenko@isep.fr

Thibaut de Broca



- ❖ 2006 - 2011: Student @ ISEP
- ❖ 2010: Intern in startup + self-entrepreneur
- ❖ 2011: Intern Software Engineer @ Google
- ❖ 2011 - 2012: Co-founder startup
- ❖ 2011 - 2015: Tech Lead - Expert @ CGI
- ❖ 2012 - ****: Teacher @ ISEP
- ❖ 2015: Tech Lead - Architect @ Tessarine
- ❖ 2016 - ****: Grootown - Data consulting

















- ❖ More:
- ❖ LinkedIn : <https://fr.linkedin.com/pub/thibaut-de-broca/27/796/890>
- ❖ Github: <https://github.com/tdebroc>
- ❖ Stackoverflow: <https://stackoverflow.com/users/1029722/tdebroc>

Program

1st Part: Relation DB & SQL

2st Part: Big Data & NoSQL

1 	Lecture 1 - Relational & SQL	8 	Lecture - Big Data
2 	Lecture 2 - SQL & PL/SQL	9 	Lecture/Lab - Column DB or Documentdb
3 	Lab 1 - SQL	10 	Lecture - Data Lake
4 	Lecture 3 - Transaction & Normalization	11 	Lecture/Lab - Airflow Scheduler
5 	Lab 2 - SQL & PL/SQL	12 	Lecture/Lab - ELK
6 	Lecture 4 - DB Administration <i>*SQL Test</i>	13 	Lecture/Lab - Apache Spark
7 	Lab 3 - JDBC <i>*SQL Reports to deposit</i>	14 	Lecture/Lab - Graph DB <i>*Big Data report to give</i>

Install Tools before Practical Lessons

Otherwise you'll lose 15 min to 2 hours in the TPs.

- ❖ For Relational Databases:
 - ❖ Install **Oracle Express** (the Database Engine)
 - ❖ Install SQLDeveloper (frontend for querying DB)

- ❖ For Big Data
 - ❖ Install Mongo
 - ❖ Install Cassandra
 - ❖ Install Airflow
 - ❖ Install ELK
 - ❖ Install Neo4J

You'll find all tutorial in Moodle:

<https://moodle.isep.fr/moodle/course/view.php?id=42>

References and Bibliography

❖ Books:

- **Raghu Ramakrishnan , Johannes Gehrke – Database Management Systems – Thirth edition**
- Georges Gardarin – Bases de données – Eyrolles (Livre de poche)
- Andreas Meier- Introduction pratique aux bases de données relationnelles – Collection IRIS
- Big Data: Principles and best practices of scalable realtime data systems, Nathan Marz and James Warren, Manning, 2014
- Principles of Distributed Databases (3rd edition), M. T. Oszu and P. Valduriez, Addison-Wesley Longman, 2011
- Seven databases in seven weeks, Eric Redmond and Jim R. Wilson, Pragmatic Bookshelf, 2012

❖ Internet:

- <http://infolab.stanford.edu/~ullman/fcdb/oracle/or-plsql.html> (PL/SQL)
- http://docs.oracle.com/cd/B28359_01/appdev.111/b28370/toc.htm
- <http://georges.gardarin.free.fr/>
- <http://codex.cs.yale.edu/avi/db-book/db4/index.html>
- <https://martinfowler.com/>

Syllabus

**ANY CHEATING
(COPY/PASTE CODE)
WOULD LEAD TO A
GRADE OF 0/20**

- ❖ Lectures + Tutorials + labs
- ❖ **Instructors**
 - Thibaut de Broca
 - Hedi Yazid
 - Larbi Boubchir
- ❖ **Attendance** is expected, and may be recorded from time to time. Absences for legitimate professional activities and illnesses are acceptable only if prior notice is given to the instructor.
- ❖ **Moodle** contains all course content.
- ❖ **4 Grades: 2 written exams + 1 report + 1 project**
- ❖ **Exams:**
 - Small exam on SQL at the 4th lecture.
 - Final Exam

Relational Labs report

A report with all answers for all Practical Lessons.

Big Data project:

Develop a mini-Datalake fetching 2 data-sources you want, formatting, combining, indexing in ELK to create a dashboard.

Jobs

- ❖ Titles around Database:
 - DBMS Implementor: Builds the system.
 - Database Designer: Establishes Schema.
 - Database Application Developer: Program that operate on database.
 - Database Administrator: Loads Data, keep running smoothly.

- ❖ Any Jobs needs strong skills in Database System:
 - Developers
 - Functional persons
 - Project Manager
 - Entrepreneurs
 - Data Engineer
 - Data analyst/Scientist
 - Analytical Engineer
 - ...

Ex-ISEP



Name: Augustin Rudigoz

Role: Entrepreneur - Founder - CEO @ Mobeye

Database Usage: SQL on MySQL at first then PostgreSQL



Name: Vincent Lebel

Role: Project Manager @ Société Générale

Database Usage: PL/SQL on Oracle



Name: Romain Bourgois

Role: Business Intelligence - Data Scientist / Big Data @ Critéo

Database Usage: noSQL with Spark



Name: You

Role: Any Role

Database Usage: SQL

Database Management Systems (DBMS)



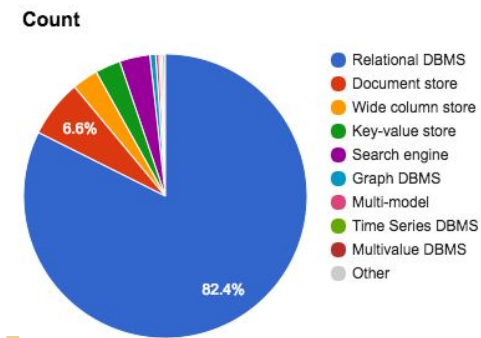
Chapter 1

What Is a DBMS?



- ❖ DBMS: Database Management System
- ❖ A database is an organized collection of data.
- ❖ Models real-world enterprise.
 - Entities (e.g., students, courses)
 - Relationships (e.g., Peter is taking II.2403)
- ❖ A Database Management System (DBMS) is a software package designed to store and manage databases.

DBMS Popularities



402 systems in ranking, January 2023

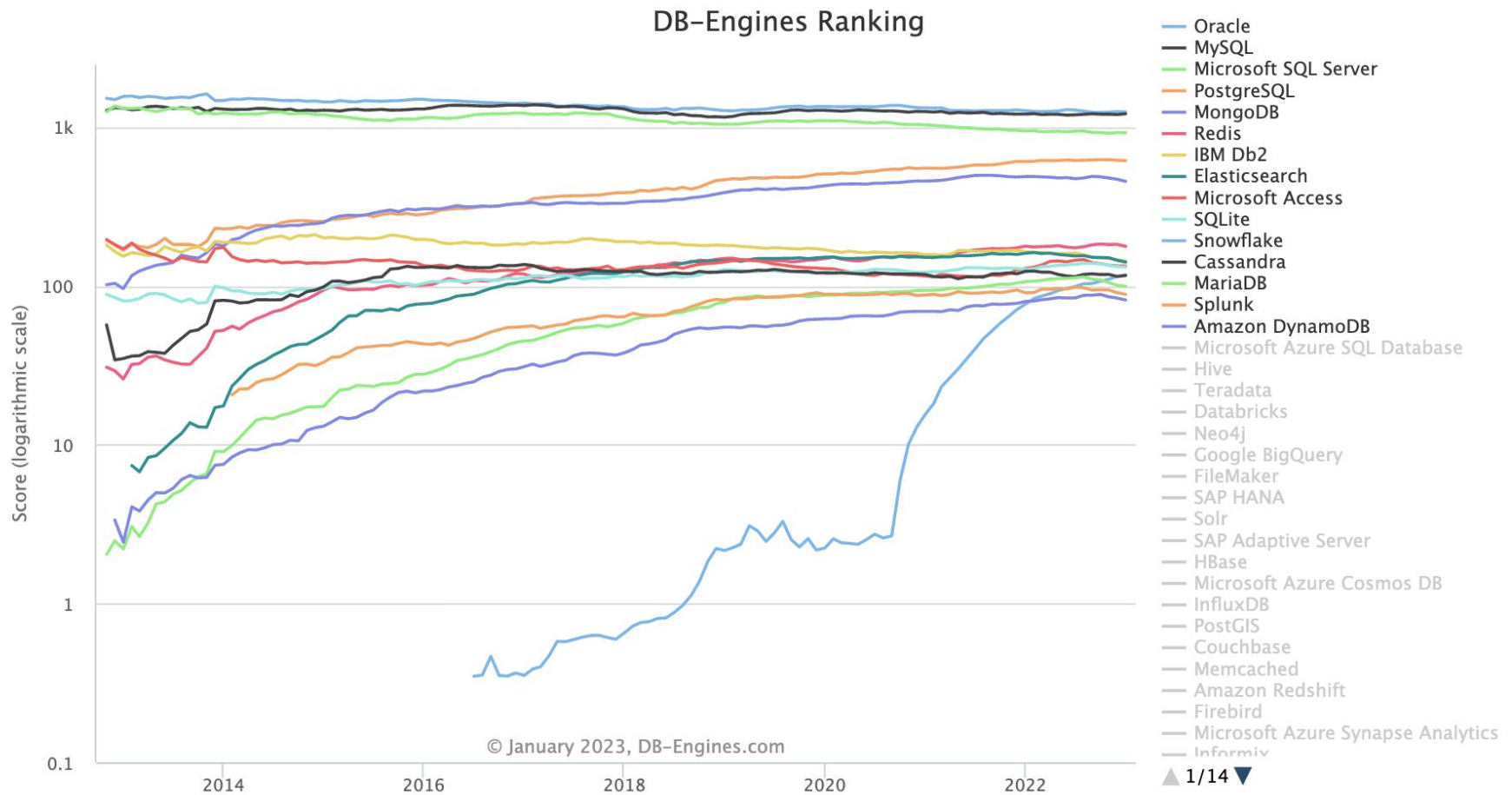
Rank			DBMS	Database Model	Score		
Jan 2023	Dec 2022	Jan 2022			Jan 2023	Dec 2022	Jan 2022
1.	1.	1.	Oracle +	Relational, Multi-model ⓘ	1245.17	-5.14	-21.72
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	1211.96	+12.56	+5.91
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	919.39	-4.96	-25.43
4.	4.	4.	PostgreSQL +	Relational, Multi-model ⓘ	614.85	-3.13	+8.29
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	455.18	-14.15	-33.38
6.	6.	6.	Redis +	Key-value, Multi-model ⓘ	177.56	-5.01	-0.43
7.	7.	7.	IBM Db2	Relational, Multi-model ⓘ	143.57	-3.05	-20.63
8.	8.	8.	Elasticsearch	Search engine, Multi-model ⓘ	141.16	-3.76	-19.59
9.	9.	9.	Microsoft Access	Relational	133.36	-0.47	+4.41
10.	10.	10.	SQLite +	Relational	131.49	-0.94	+4.06
11.	11.	↑ 17.	Snowflake +	Relational	117.26	+2.49	+40.44
12.	12.	↓ 11.	Cassandra +	Wide column	116.31	+1.66	-7.24
13.	13.	↓ 12.	MariaDB +	Relational, Multi-model ⓘ	99.35	-1.57	-7.07
14.	14.	↓ 13.	Splunk	Search engine	88.41	-2.39	-2.04
15.	15.	↑ 16.	Amazon DynamoDB +	Multi-model ⓘ	81.55	-2.29	+1.70
16.	16.	↓ 14.	Microsoft Azure SQL Database	Relational, Multi-model ⓘ	80.37	-1.61	-5.95
17.	17.	↓ 15.	Hive	Relational	74.34	-3.55	-9.11
18.	18.	18.	Teradata	Relational, Multi-model ⓘ	65.43	-0.46	-3.69
19.	19.		Databricks	Multi-model ⓘ	60.82	+0.08	
20.	20.	20.	Neo4j +	Graph	55.84	-1.49	-2.19
21.	21.	↑ 24.	Google BigQuery +	Relational	54.43	-1.26	+8.80



<http://db-engines.com/en/ranking>

DBMS Popularities

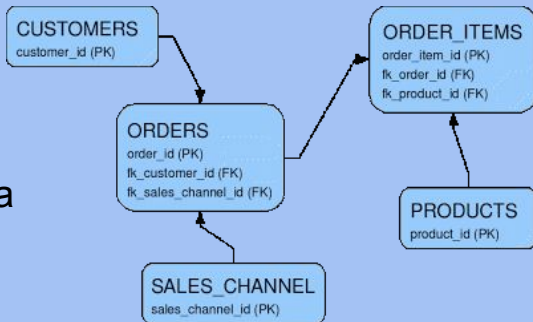
ranking table
January 2023



DBMS Classification

Relational DBMS (10)

Oracle
Mysql
Microsoft SQL server
PostgreSQL
DB2
Access
SQLite
SAP
Teradata
Hive



Here are the first 16th DBMS classified in the 5 most know categories.

Document Store (1)

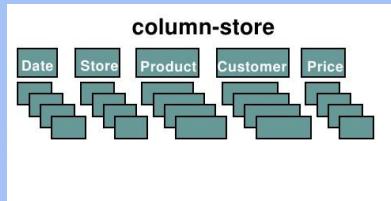
MongoDB

Example of JSON:

```
{
  id : 123
  name: ""
  childs: ["", ""]
}
```

Wide Column Store (2)

Hadoop HBase
Cassandra

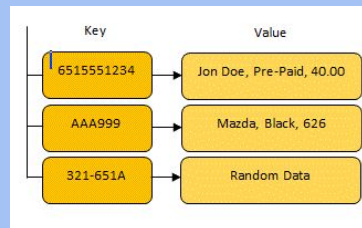


Explanation

<http://bit.ly/1Mcshfd>

Key-Value Store (1)

Redis



Search Engine (2)

ElasticSearch
Solr



Describing and storing data in a DBMS

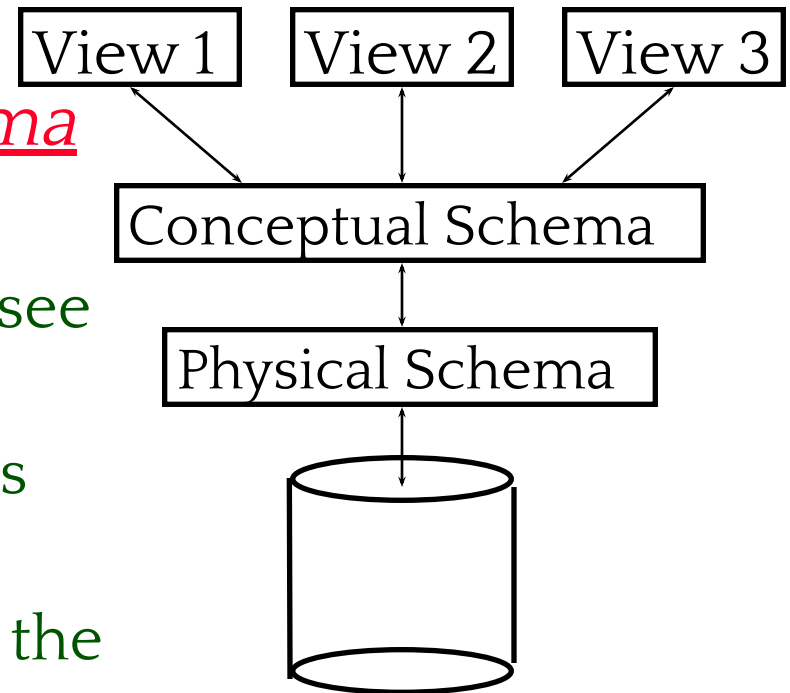
Data Models

- ❖ A *data model* is a collection of concepts for describing data.
- ❖ A *schema* is a description of a particular collection of data, using the given data model.
- ❖ The *relational model of data* is the most widely used model today.
 - Main concept: *relation*, basically a table with rows and columns.
 - Every relation has a *schema*, which describes the columns, or fields.

Levels of Abstraction

- ❖ Many views, single conceptual (logical) schema and physical schema.

- Views describe how users see the data.
- Conceptual schema defines logical structure
- Physical schema describes the files and indexes used.



=> Schemas are defined using DDL (Data definition language).

=> data is modified/queried using DML (Data manipulation language).

Example: University Database

- ❖ Conceptual schema:
 - *Students(sid: string, name: string, login: string, age: integer, gpa: real)*
 - *Courses(cid: string, cname: string, credits: integer)*
 - *Enrolled(sid: string, cid: string, grade: string)*
- ❖ Physical schema:
 - Relations stored as unordered files.
 - Index on first column of Students.
- ❖ External Schema (View):
 - *Course_info(cid: string, enrollment: integer)*

Where data is stored ?

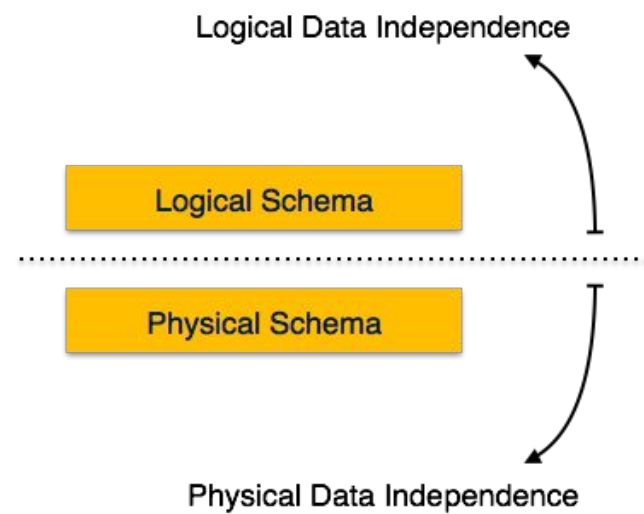
- ❖ It stores Hard on Disks:
 - *Transfers between RAM and disks are high-cost operation compared to in-memory operations.*
- ❖ Why not on RAM ?
 - Costs too much: \$30 will buy you either 4GB of RAM or 2TB of disk today (500x).
 - Volatile: We want data saved between runs.

<http://www.csbio.unc.edu/mcmillan/Media/Comp521F10Lecture13.pdf>

How data is stored ? (Examples)

- ❖ Unordered:
 - *Stores data in order they are inserted: Fast Insertion $O(1)$.*
 - *Longer to retrieve a data: $O(n)$.*
- ❖ Ordered:
 - *Insertion: May have to re-organize: Low efficiency.*
 - *Easier to find data: $O(\log(n))$.*
- ❖ Structured Files:
 - *Heap Files*
 - *Hash Buckets*
 - *B++ Tree: Very common for index.*

Data Independence



- ❖ Applications insulated from how data is structured and stored.
 - ❖ Logical data independence: Protection from changes in *logical* structure of data.
 - ❖ Physical data independence: Protection from changes in *physical* structure of data.
- *One of the most important benefits of using a DBMS!*

Concurrency Control

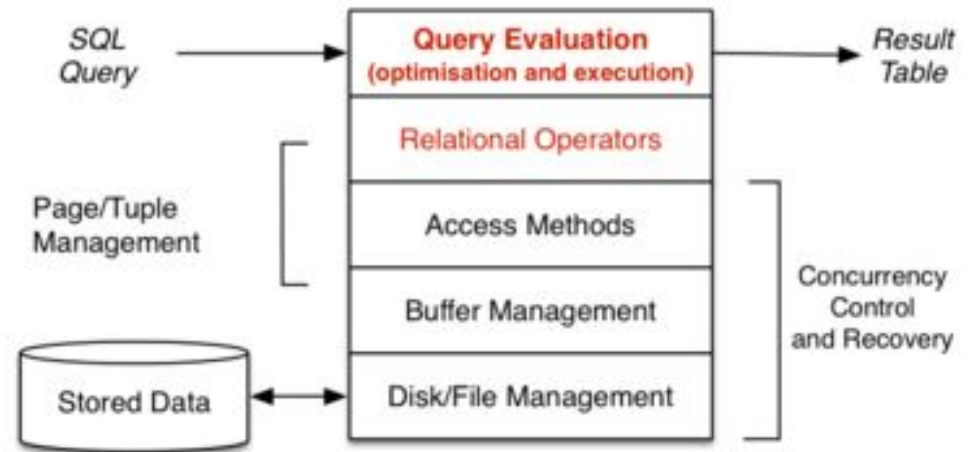
- ❖ Concurrent execution of user programs is essential for good DBMS performance.
 - Because disk accesses are frequent, and relatively slow, it is important to keep the cpu humming by working on several user programs concurrently.
- ❖ Interleaving actions of different user programs can lead to inconsistency.
- ❖ DBMS ensures such problems don't arise: users can pretend they are using a single-user system.

Transaction: An Execution of a DB Program

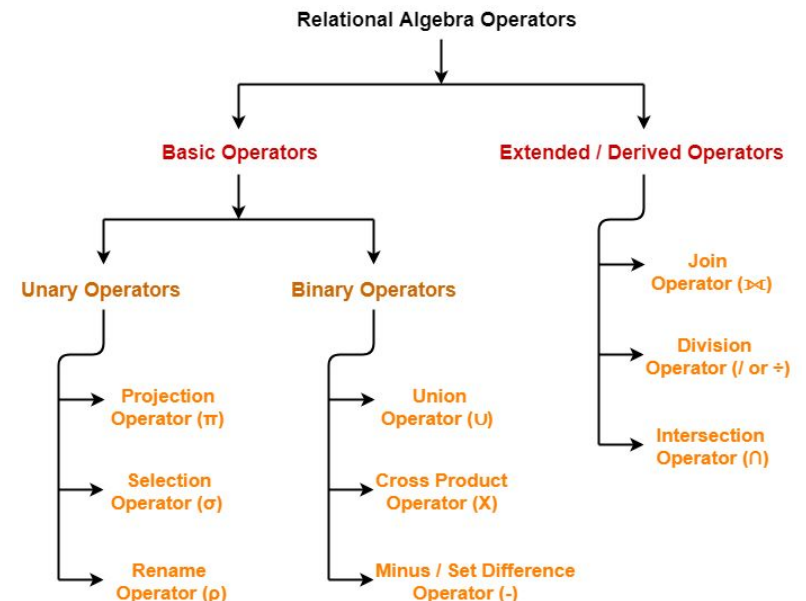
- ❖ Key concept is transaction, which is an *atomic* sequence of database actions (reads/writes).
- ❖ Each transaction, executed completely, must leave the DB in a consistent state if DB is consistent when the transaction begins.
- ❖ Users can specify some simple integrity constraints on the data, and the DBMS will enforce these constraints. (i.e.: UNIQUE, CHECK...)

Structure of a DBMS

- ❖ A typical DBMS has a layered architecture.
- ❖ The figure does not show the concurrency control and recovery components.
- ❖ This is one of several possible architectures; each system has its own variations.



<https://www.cse.unsw.edu.au/~cs9315/18s2/notes/H/notes.html>



Summary

- ❖ DBMS used to maintain, query large datasets.
- ❖ Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
- ❖ Levels of abstraction give data independence.
- ❖ A DBMS typically has a layered architecture.
- ❖ DBMS R&D is one of the broadest, most exciting areas in CS.



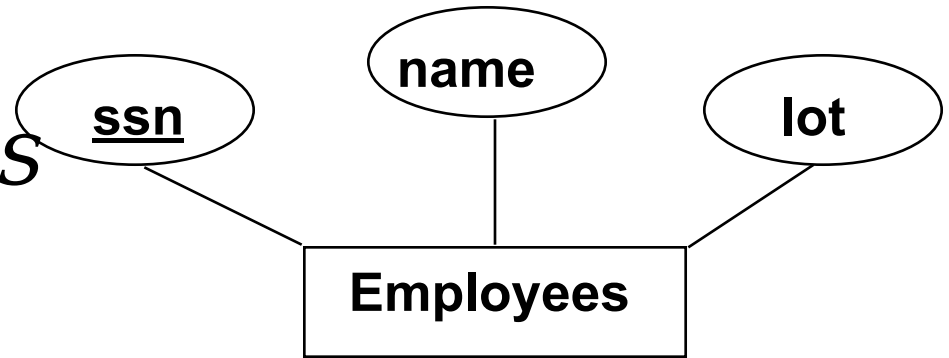
The Entity-Relationship Model

Chapter 2

Overview of Database Design

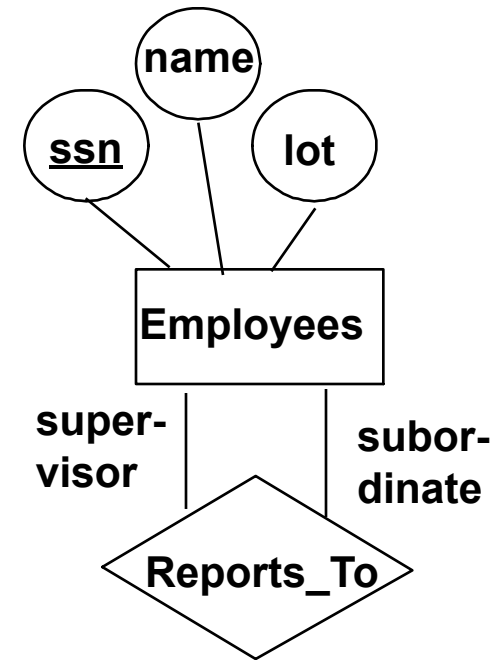
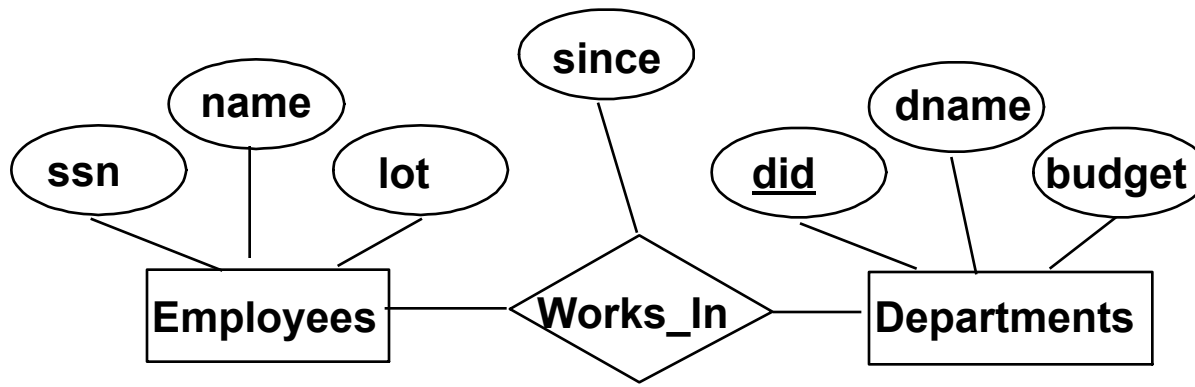
- ❖ Conceptual design: (*ER Model is used at this stage.*)
 - What are the *entities* and *relationships* to model ?
 - What information about these entities and relationships should we store in the database?
 - What are the *integrity constraints* or *business rules* that hold?
 - A database 'schema' in the ER Model can be represented pictorially (*ER diagrams*).
 - Can map an ER diagram into a relational schema.

ER Model Basics



- ❖ Entity (square): Real-world object distinguishable from other objects. An entity is described (in DB) using a set of attributes (circles).
- ❖ Entity Set: A collection of similar entities. E.g., all employees.
 - All entities in an entity set have the same set of attributes
 - Each entity set has a *key*.
 - Each attribute has a *domain / type*.

ER Model Basics (Contd.)

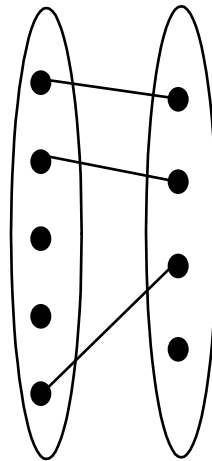
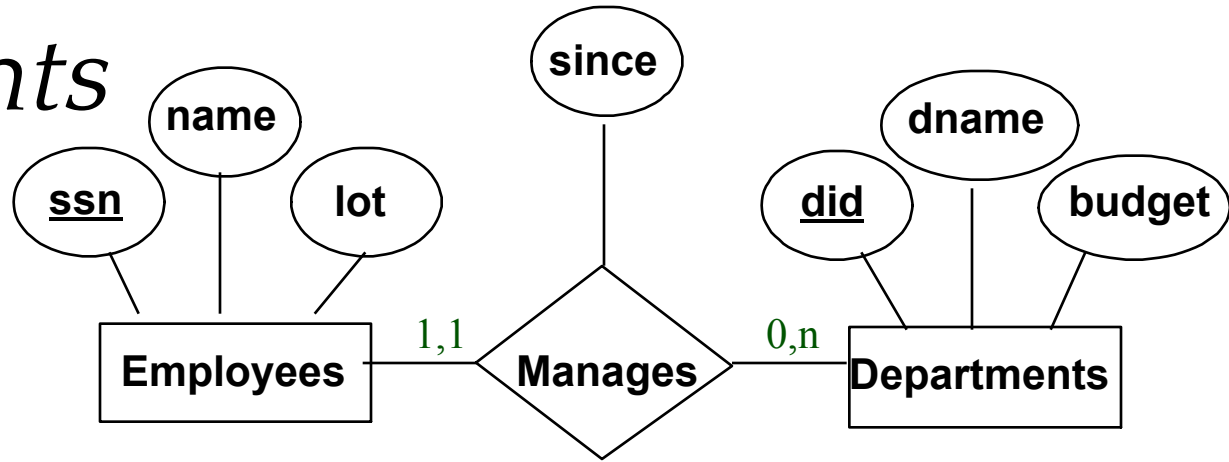


- ❖ Relationship (diamond): Association among two or more entities. E.g., John works in Pharmacy department.
- ❖ Relationship Set: Collection of similar relationships.
 - An n-ary relationship set R relates n entity sets $E_1 \dots E_n$; each relationship in R involves entities e_1, \dots, e_n .
 - Same entity set could participate in different relationship sets, or in different “roles” in same set.

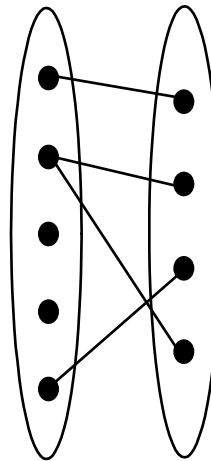
Key Constraints

- ❖ Consider Works_In:
An employee can work in many departments; a dept can have many employees.

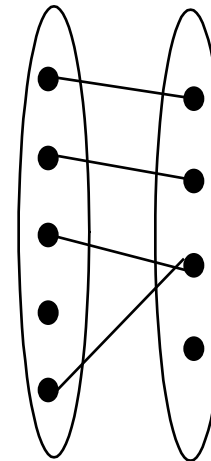
- ❖ In contrast, each dept has at most one manager, according to the key constraint on Manages.



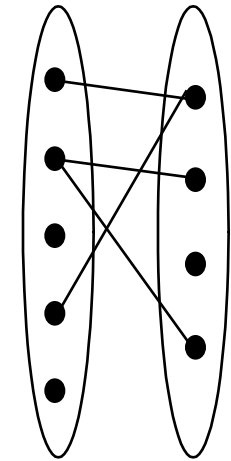
1-to-1



1-to Many



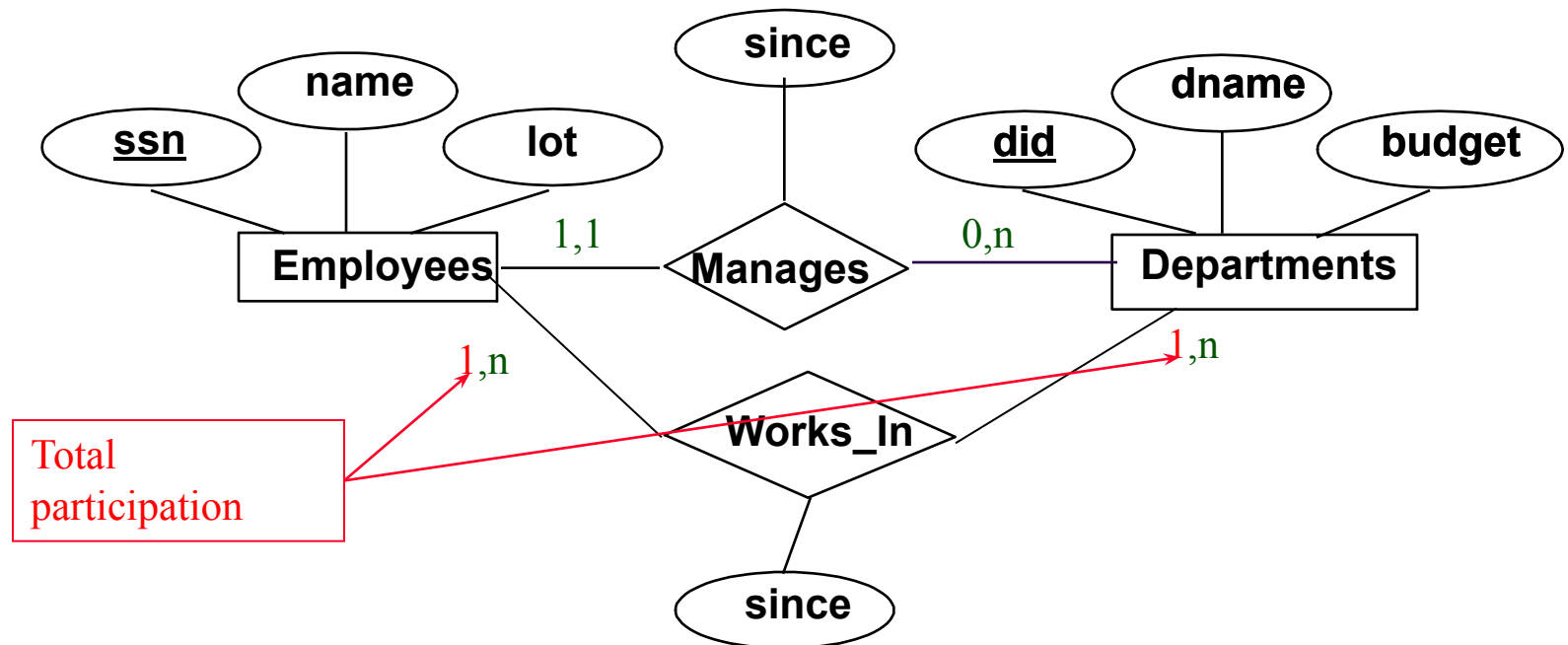
Many-to-1



Many-to-Many

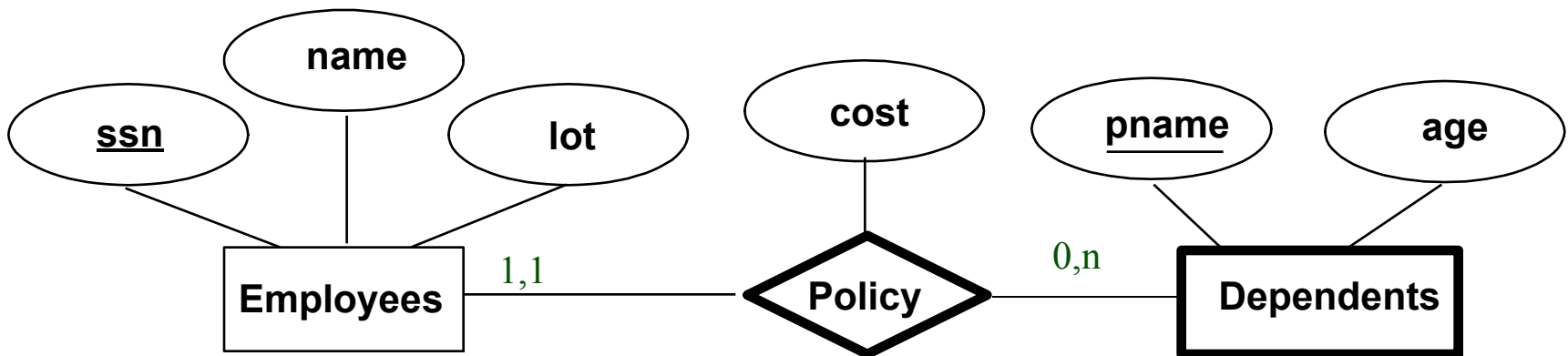
Participation Constraints

- ❖ Does every department have a manager?
 - If so, this is a participation constraint: the participation of Departments in Manages is said to be *total* (vs. *partial*).
 - Every *did* value in Departments table must appear in a tuple of the Manages relation.

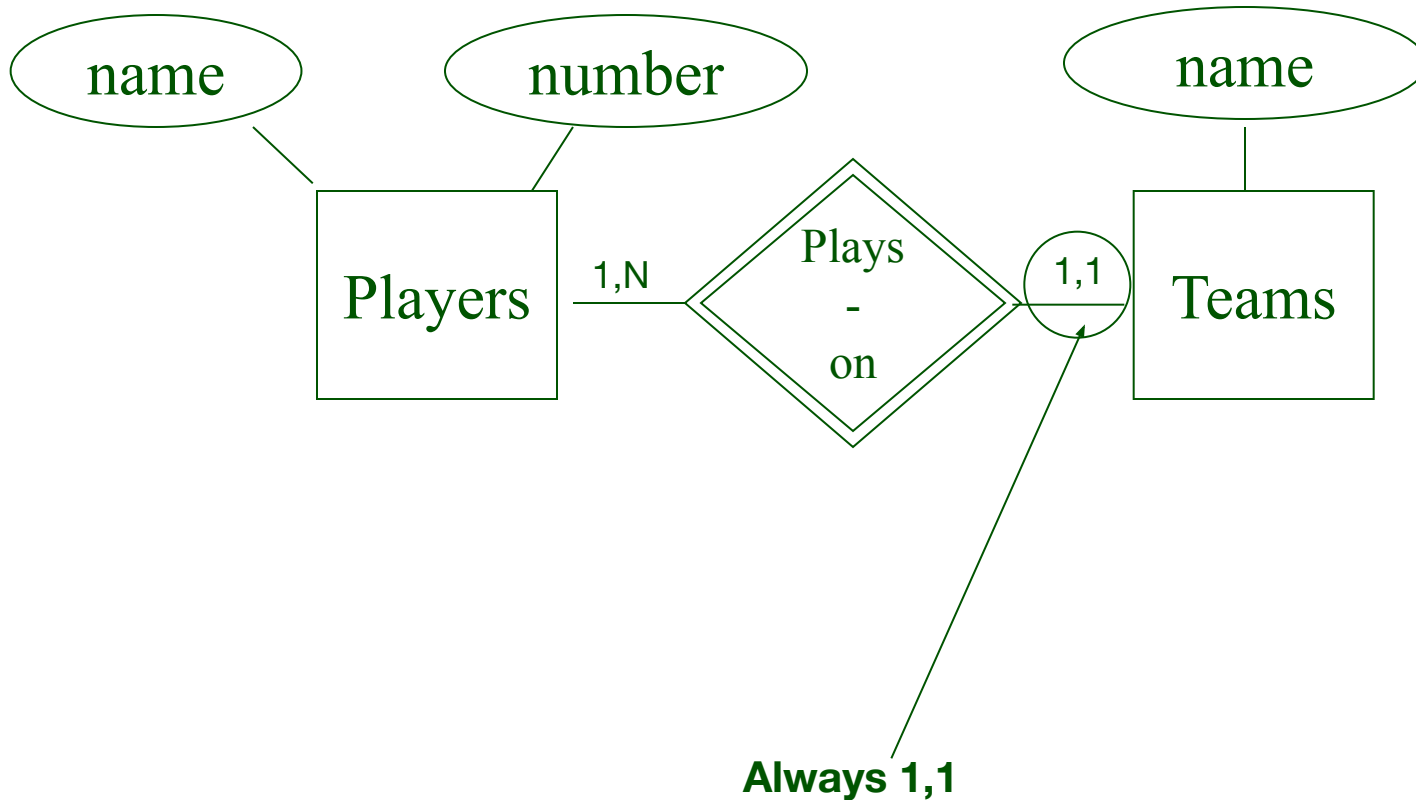


Weak Entities

- ❖ A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
 - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
 - Weak entity set must have total participation in this *identifying* relationship set.

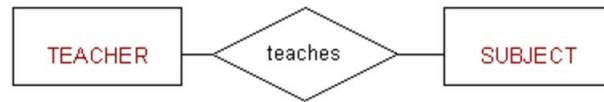


Weak Entity Sets (Cont.)

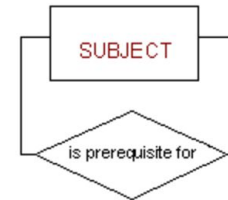


Relationships

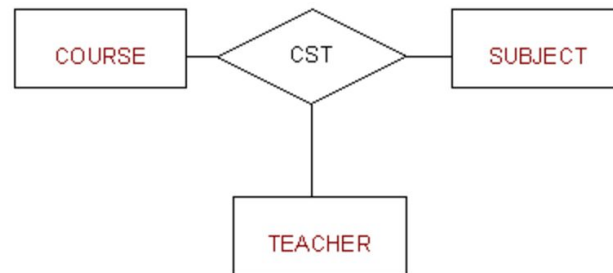
- ❖ A **binary relationship** is when two entities participate and is the most common relationship degree.



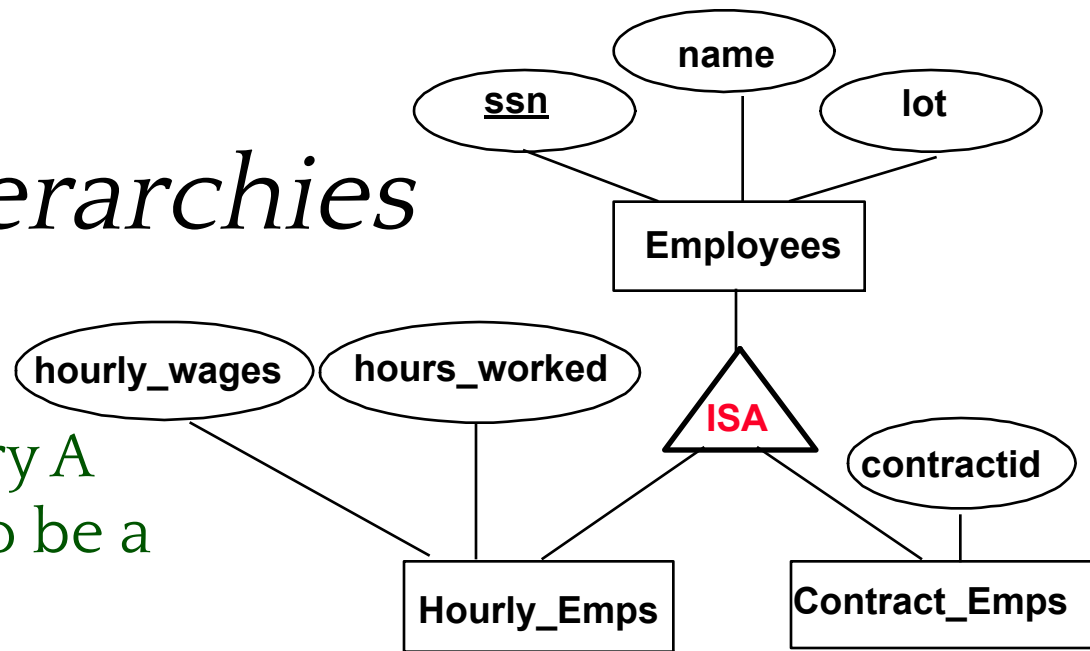
- ❖ A **unary relationship** is when both participants in the relationship are the same entity.



- ❖ A **ternary relationship** is when three entities participate in the relationship.



ISA ('is a') Hierarchies



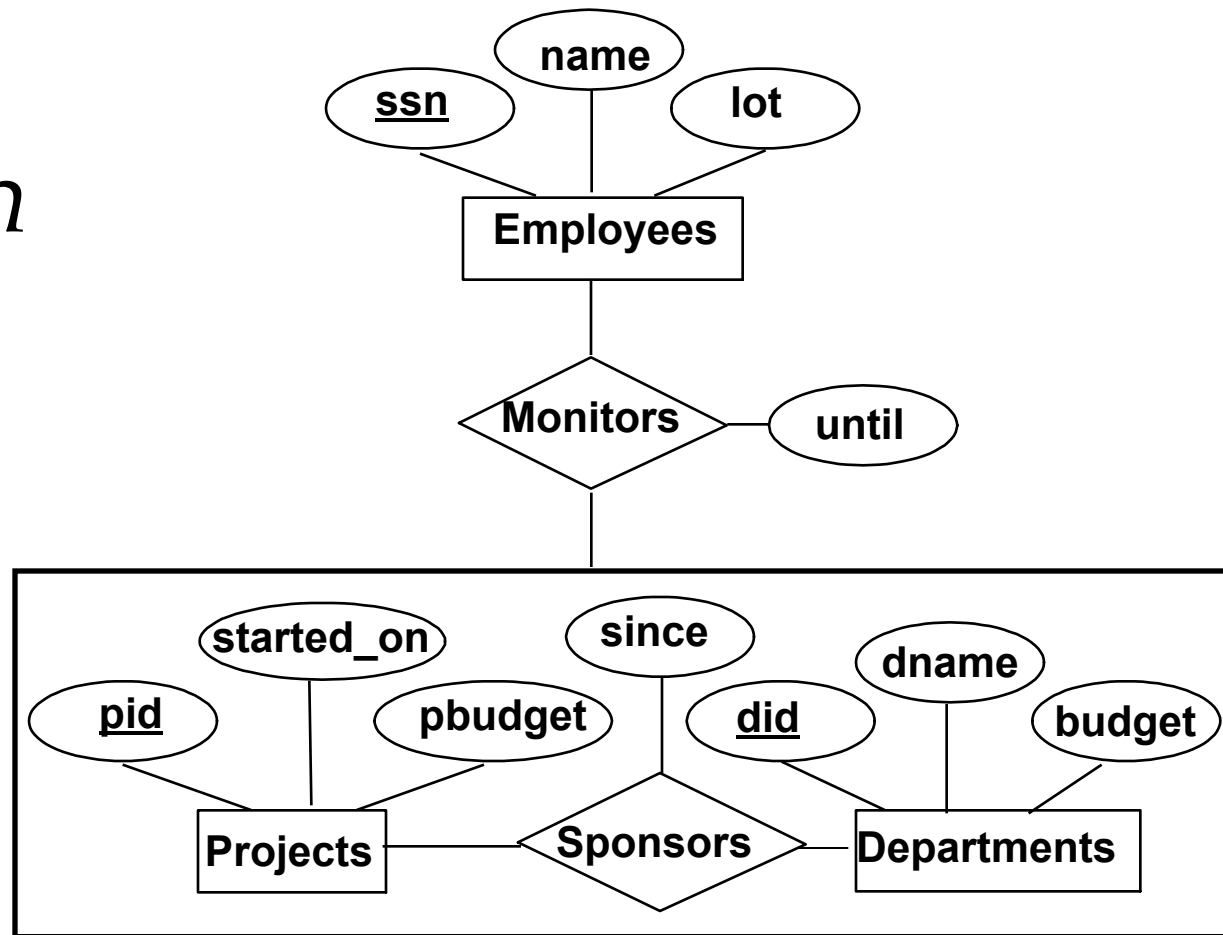
• If we declare A **ISA** B, every A entity is also considered to be a B entity.

- ❖ **Overlap constraints:** Can A be an Hourly_Emps as well as a Contract_Emps entity? (*Allowed/disallowed*)
- ❖ **Covering constraints:** Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (*Yes/no*)
- ❖ Reasons for using ISA:
 - To add descriptive attributes specific to a subclass.
 - To identify entities that participate in a relationship.

Aggregation

- ❖ Used when we have to model a relationship involving (entity sets and) a *relationship set*.

- Aggregation allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.



→ *Aggregation vs. ternary relationship:*

- ❖ Monitors is a distinct relationship, with a descriptive attribute. (i.e., until)
- ❖ Also, can say that each sponsorship is monitored by at most one employee.

Conceptual Design Using the ER Model

❖ Design choices:

- Should a concept be modeled as an entity or an attribute? (e.g., address)
- Should a concept be modeled as an entity or a relationship? (e.g., address)
- Identifying relationships: Binary or ternary?
Aggregation?

Entity vs. Attribute

- ❖ Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- ❖ Depends upon the use we want to make of address information, and the semantics of the data:
 - If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued).
 - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic).

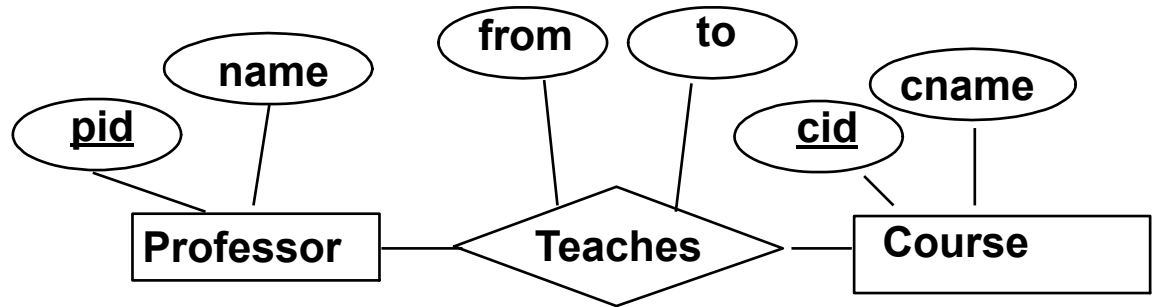
Exercise

A university database contains information about professors (identified by SSN) and courses (identified by courseid). Professors teach courses; each of the following situations concerns the Teaches relationship set. For each diagram, draw an ER diagram that describes it (assuming no further constraints hold).

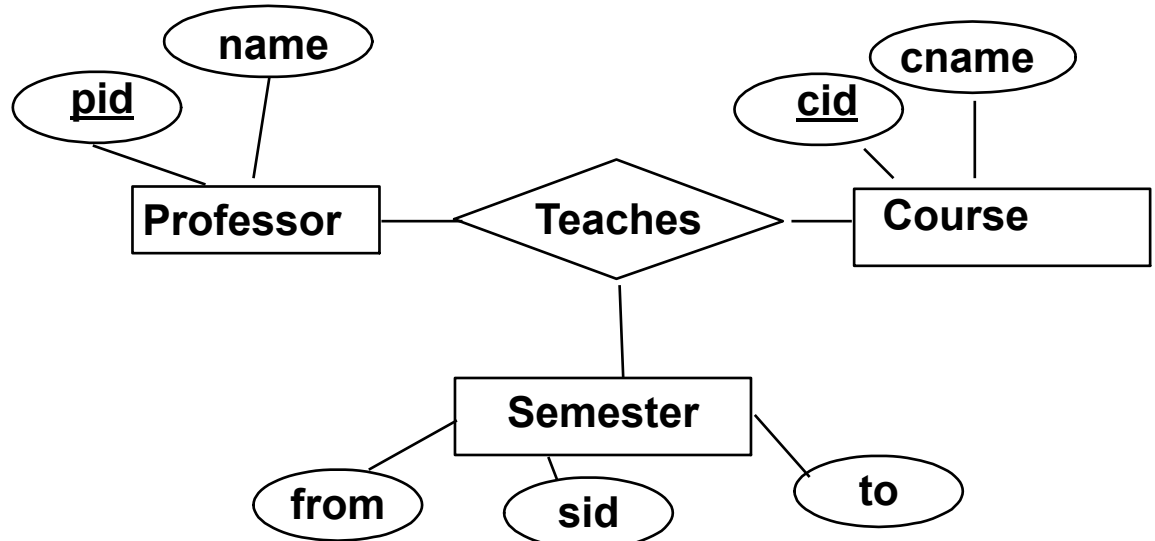
1. Professors can teach the same course in several semesters, and only the most recent such offering needs to be recorded.
2. Professors can teach the same course in several semesters, and each offering must be recorded.

Exercise (Answer)

1) Teaches does not allow a professor to teach in a department for two or more semester.



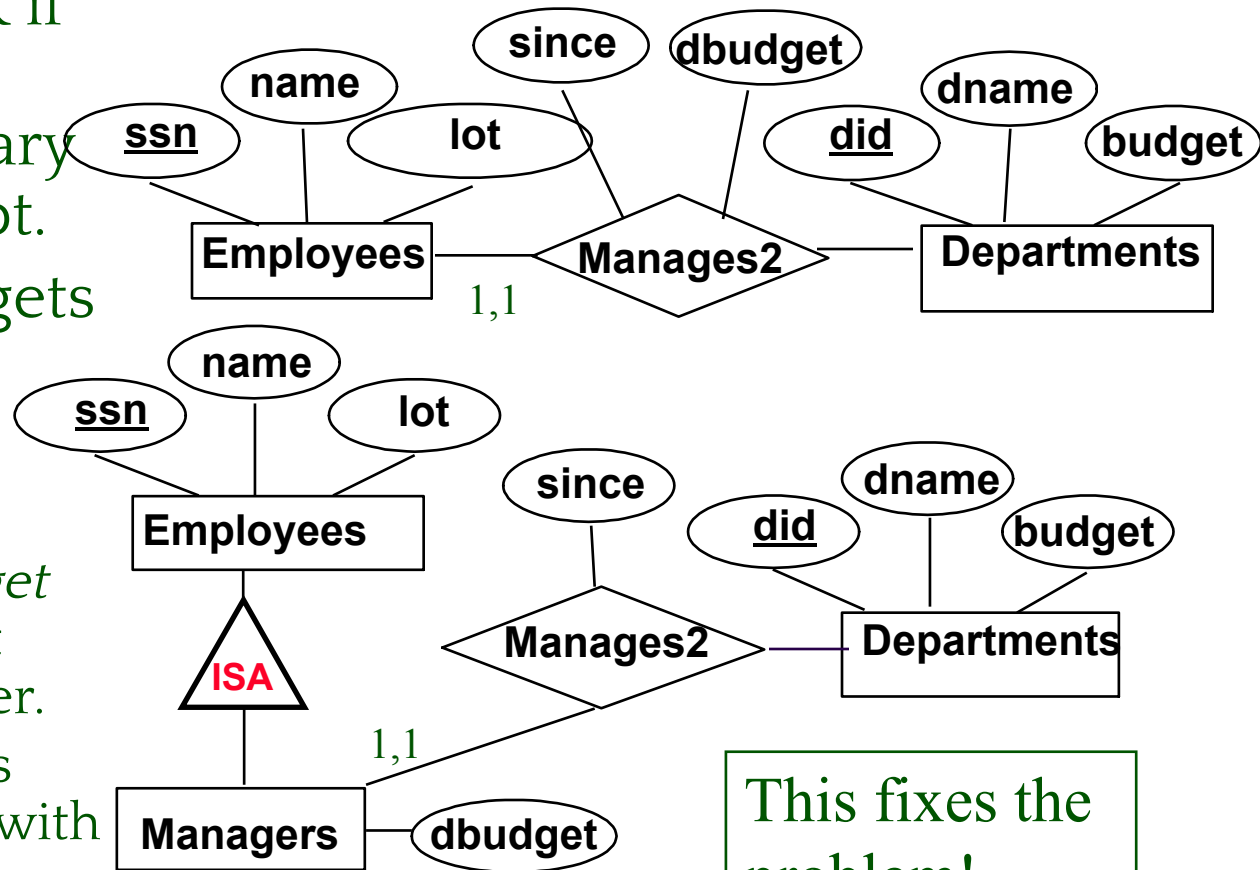
2) We want to record *several values of the descriptive attributes for each instance of this relationship*. Accomplished by introducing new entity set, Semester.



Entity vs. Relationship

- ❖ First ER diagram OK if a manager gets a separate discretionary budget for each dept.
- ❖ What if a manager gets a discretionary budget that covers *all* managed depts?

- **Redundancy:** *dbudget* stored for each dept managed by manager.
- **Misleading:** Suggests *dbudget* associated with department-mgr combination.



This fixes the problem!

Summary of Conceptual Design

- ❖ *Conceptual design follows requirements analysis,*
 - Yields a high-level description of data to be stored
- ❖ ER model popular for conceptual design
 - Constructs are expressive, close to the way people think about their applications.
- ❖ Basic constructs: *entities, relationships, and attributes* (of entities and relationships).
- ❖ Some additional constructs: *weak entities, ISA hierarchies, and aggregation.*
- ❖ Note: There are many variations on ER model.

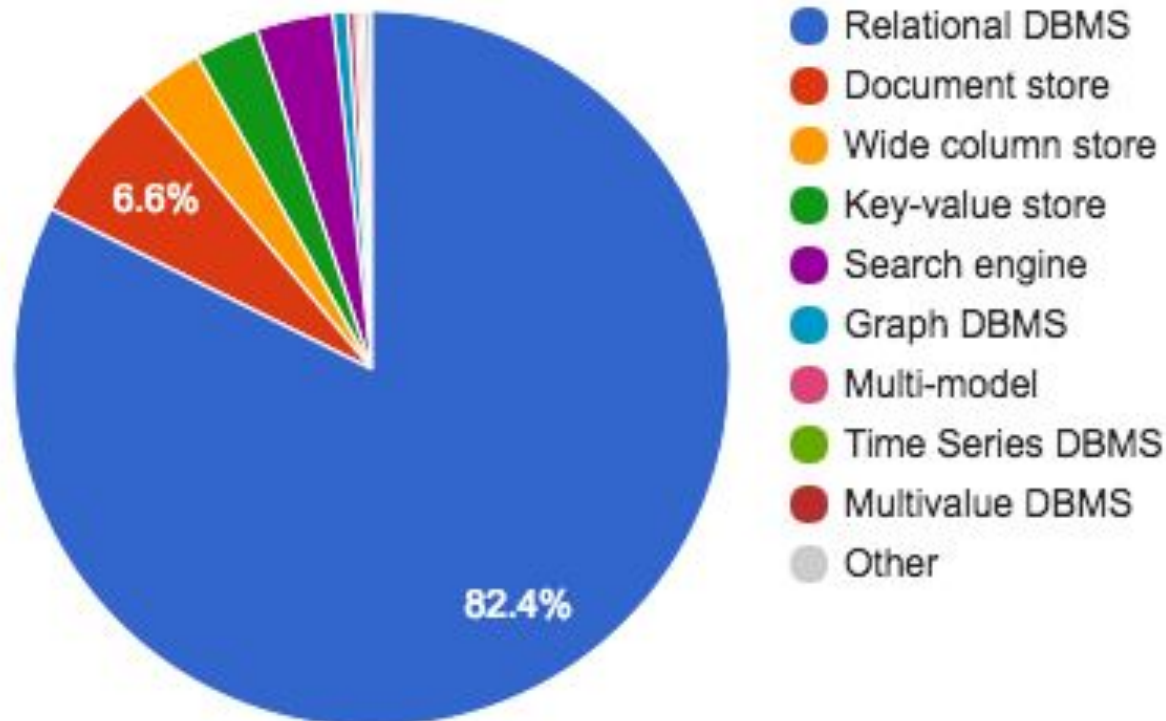
Summary of ER (Contd.)

- ❖ ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
 - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, and whether or not to use aggregation.
- ❖ Ensuring good database design: resulting relational schema should be analyzed and refined further. **FD information and normalization** techniques are especially useful.
 - FD = functional dependency.

ER: Many implementations possible

❖ From rankingdb:

Count

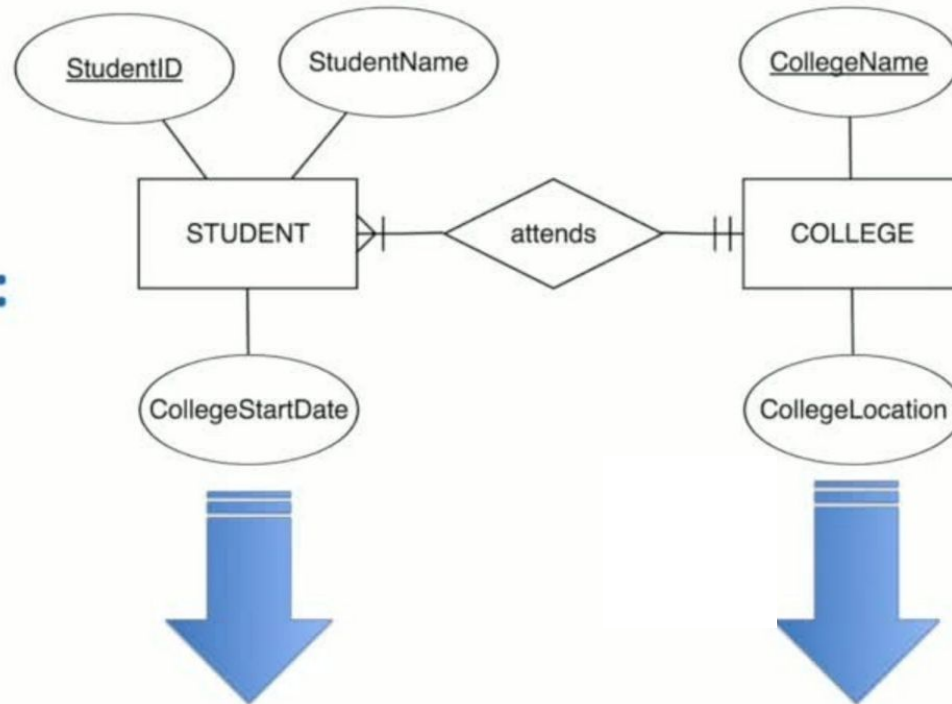


The Relational Model & SQL

Chapter 3

ER to Relational model

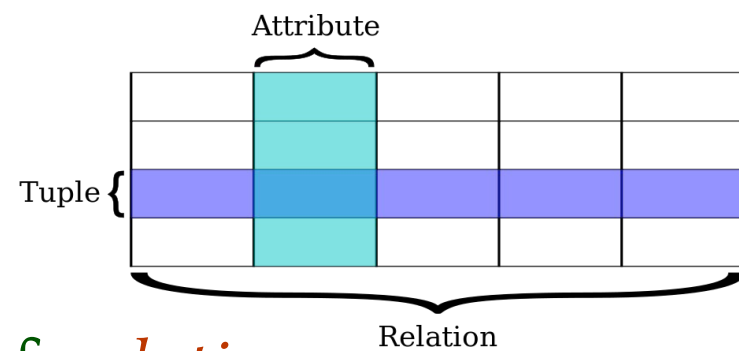
**ER
Diagram:**



**Relational
Schema:**



Relational Database Definition



- ❖ *Relational database*: a set of *relations*
- ❖ *Relation (not relationship!!)*: made up of 2 parts:
 - *Instance*: a *table*, with rows and columns.
#Rows = *cardinality*, #fields = *degree / arity*.
 - *Schema*: specifies name of relation, plus name and type of each column.
 - E.G. Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real).
- ❖ Can think of a relation as a *set* of rows or *tuples* (i.e., all rows are distinct).
 - Tuple: an ordered set of data constituting a record.

Example Instance of Students Relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

❖ Cardinality = 3, degree = 5, all rows distinct

Relational Query Languages

- ❖ A major strength of the relational model: supports simple, powerful *querying* of data.
- ❖ Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.
 - The key: precise semantics for relational queries.
 - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

The SQL Query Language

- ❖ Developed by IBM (system R) in the 1970s.
- ❖ First version is in 1974
- ❖ Need for a standard since it is used by many vendors
- ❖ Standards:
 - SQL-86 (by American National Standards Institute)
 - SQL-1989 (minor revision)
 - SQL-1992 (major revision)
 - SQL-1999 (major extensions)
 - SQL-2008
 - SQL-2011 (current version)

SQL and Chat GPT



- Already nice demos developed between chatGPT and SQL:

<https://news.ycombinator.com/item?id=34521149>

<https://www.patterns.app/blog/2023/01/18/crunchbot-sql-analyst-gpt/>

<https://blog.langchain.dev/llms-and-sql/>



<https://github.com/cfahlgren1/natural-sql>

<https://huggingface.co/chatdb/natural-sql-7b>

- But scientific paper (22 Jan 2024) <https://arxiv.org/abs/2401.12379> shows high error rate (20%) or high success rate (80% ;-)).
 - Lot of syntax is easier in SQL => OUTER/INNER JOIN, ...
 - ChatGPT may not have the exact answer expected, and this could be very problematic for Operational business queries, financial reports, or any serious production system ... If you want to be 100% sure of the result



=> Use SQL

The SQL Query Language

- ❖ **USED EVERYWHERE:** Even non-relational database (use a subset of) SQL!
 - Big Data / Distributed calcul: Hive / Athena / BigQuery
 - In-memory distributed: Spark
 - Column: Phoenix on HBase
 - Key-Value: N1QL with couchbase
 - Cassandra: CQL
 - DBT: Data Build Tool

SQL - Inspiring projects

❖ Gitql: gitql (to query .git)

🔗 Example Commands

- `select hash, author, message from commits limit 3`
- `select hash, message from commits where 'hell' in full_message or 'Fuck' in full_message`
- `select hash, message, author_email from commits where author = 'cloudson'`
- `select date, message from commits where date < '2014-04-10'`
- `select message from commits where 'hell' in message order by date asc`
- `select distinct author from commits where date < '2020-01-01'`

❖ Textql: textql (for CSV / TSV)

```
1. paul@Pauls-MBP: ~ (zsh)
~ # hello and welcome to textql.
~ cat sample_data.csv
id,name,value,timestamp
1,Paul,5,1440378750
2,Jeff,16,1440378790
3,Dmitri,-3,1440378668
~ textql -sql "select count() from sample_data" sample_data
4
~ # oh right, we have a header row
~ textql -header -sql "select count() from sample_data" sample_data
3
~ # full SQL engine, right at your finger tips
~ textql -header -sql "select max(value) from sample_data" sample_data
16
~ textql -header -sql "select strftime('%H:%M:%S', datetime(timestamp, 'unixepoch')) fr
o
```

The SQL Query Language

- ❖ To find all 18 year old students, we can write:

```
SELECT *  
FROM Students S  
WHERE S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

To find just names and logins, replace the first line:

```
SELECT S.name, S.login
```

Querying Multiple Relations

- ❖ What does the following query compute?

```
SELECT S.name, E.cid  
FROM Students S, Enrolled E  
WHERE S.sid=E.sid AND E.grade="A"
```

Given the following instances of Enrolled and Students:

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ecs	18	3.2
53650	Smith	smith@math	19	3.8

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

we get:

S.name	E.cid
Smith	Topology112

Creating Relations in SQL

- ❖ Creates the Students relation. Observe that the type **(domain)** of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

```
CREATE TABLE Students  
(sid: CHAR(20),  
name: CHAR(20),  
login: CHAR(10),  
age: INTEGER,  
gpa: REAL)
```

- ❖ As another example, the Enrolled table holds information about courses that students take.

```
CREATE TABLE Enrolled  
(sid: CHAR(20),  
cid: CHAR(20),  
grade: CHAR(2))
```

Destroying and Altering Relations

DROP TABLE Students

- ❖ Destroys the relation Students. The schema information *and* the tuples are deleted.

ALTER TABLE Students

ADD COLUMN firstYear: integer

- ❖ The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a *null* value in the new field.

Adding and Deleting Tuples

- ❖ Can insert a single tuple using:

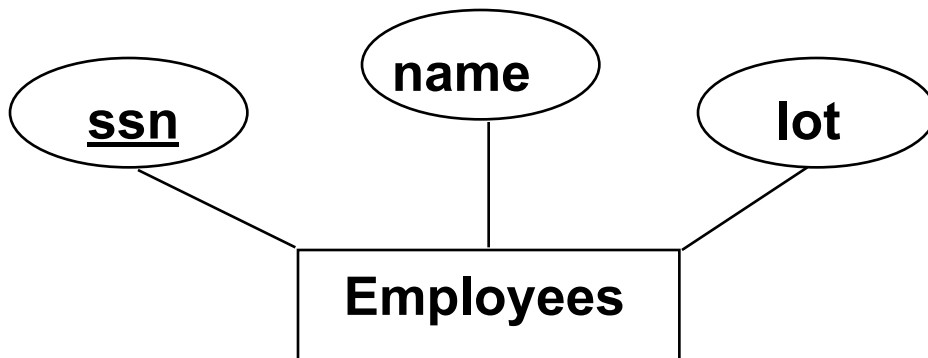
```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
```

- ❖ Can delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

Logical DB Design: ER to Relational

❖ Entity sets to tables:



```
CREATE TABLE Employees  
  (ssn CHAR(11),  
   name CHAR(20),  
   lot INTEGER,  
   PRIMARY KEY (ssn))
```

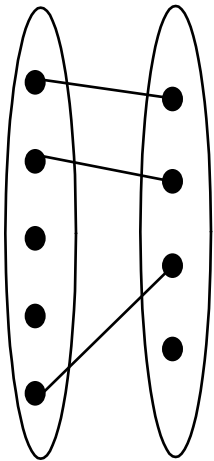
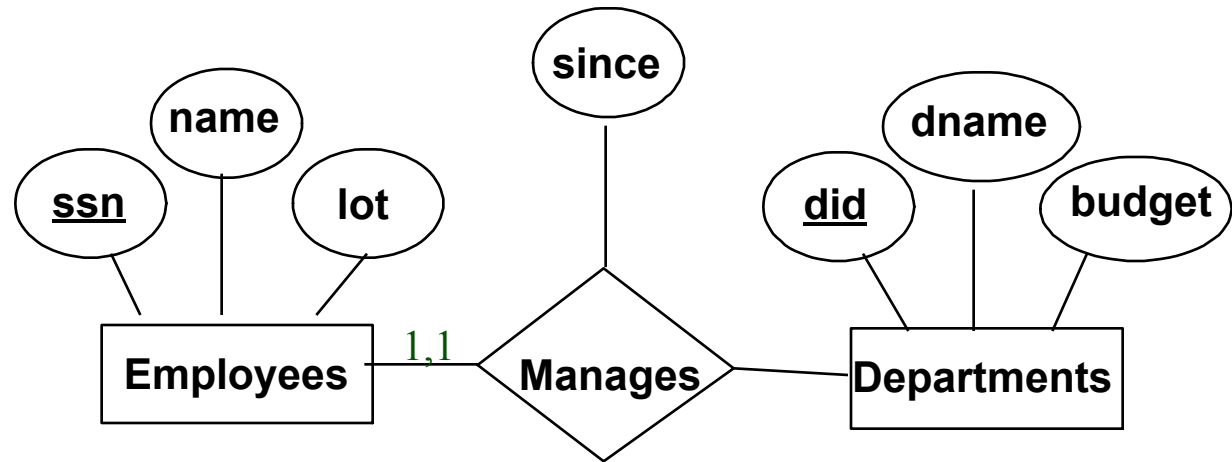
Relationship Sets to Tables

- ❖ In translating a relationship set to a relation, attributes of the relation must include:
 - Keys for each participating entity set (as foreign keys).
 - This set of attributes forms a *superkey* for the relation.
 - All descriptive attributes.

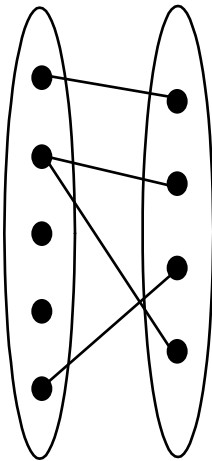
```
CREATE TABLE Works_In(  
    ssn CHAR(11),  
    did INTEGER,  
    since DATE,  
    PRIMARY KEY (ssn, did),  
    FOREIGN KEY (ssn)  
        REFERENCES Employees,  
    FOREIGN KEY (did)  
        REFERENCES Departments)
```

Review: Key Constraints

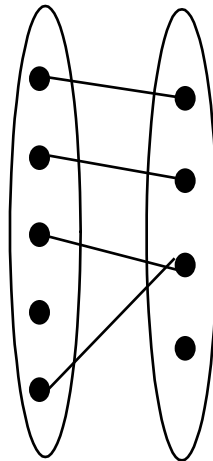
- ❖ Each dept has at most one manager, according to the key constraint on Manages.



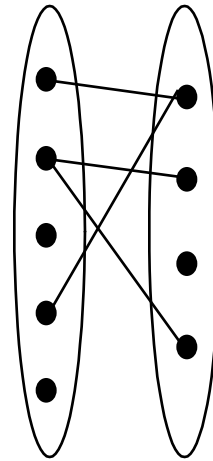
1-to-1



1-to Many



Many-to-1



Many-to-Many

Translation to relational model?

Translating ER Diagrams with Key Constraints

- ❖ Map relationship to a table:
- ❖ Since each department has a unique manager, we could instead combine Manages and Departments.

```
CREATE TABLE Dept_Mgr(  
  did INTEGER,  
  dname CHAR(20),  
  budget REAL,  
  ssn CHAR(11),  
  since DATE,  
  PRIMARY KEY (did),  
  FOREIGN KEY (ssn) REFERENCES Employees)
```

Where do ICs Come From?

- ❖ ICs are based upon the semantics of the real-world enterprise that is being described in the database relations.
 - not null
 - primary key (A_1, \dots, A_n)
 - check (P) , where P is a predicate
- ❖ Key and foreign key ICs are the most common; more general ICs supported too.

```
CREATE TABLE Courses (cid CHAR(10),  
                        cname CHAR ( 10) NOT NULL,  
                        credits INTEGER check (credits >= 0),  
                        PRIMARY KEY (cid)  
                        )
```



Foreign Keys, Referential Integrity

- ❖ Foreign key: Set of fields in one relation that is used to 'refer' to a tuple in another relation. (Must correspond to primary key of the second relation.)
- ❖ E.g. *sid* is a foreign key referring to **Students**:
 - Enrolled(*sid*: string, *cid*: string, *grade*: string)
 - If all foreign key constraints are enforced, referential integrity is achieved, i.e., no dangling references.

Foreign Keys in SQL

- ❖ Only students listed in the Students relation should be allowed to enroll for courses.

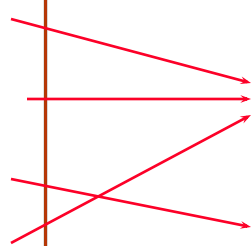
```
CREATE TABLE Enrolled  
  (sid CHAR(20), cid CHAR(20), grade CHAR(2),  
   PRIMARY KEY (sid,cid),  
   FOREIGN KEY (sid) REFERENCES Students )
```

Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8



Enforcing Referential Integrity

- ❖ Consider Students and Enrolled; *sid* in Enrolled is a foreign key that references Students.
- ❖ What should be done if an Enrolled tuple with a non-existent student id is inserted? (*Reject it!*)
- ❖ What should be done if a Students tuple is deleted?
 - Also delete all Enrolled tuples that refer to it.
 - Disallow deletion of a Students tuple that is referred to.
 - Set *sid* in Enrolled tuples that refer to it to a *default sid*.
 - (In SQL, also: Set *sid* in Enrolled tuples that refer to it to a special value *null*, denoting 'unknown' or 'inapplicable'.)
- ❖ Similar if primary key of Students tuple is updated.

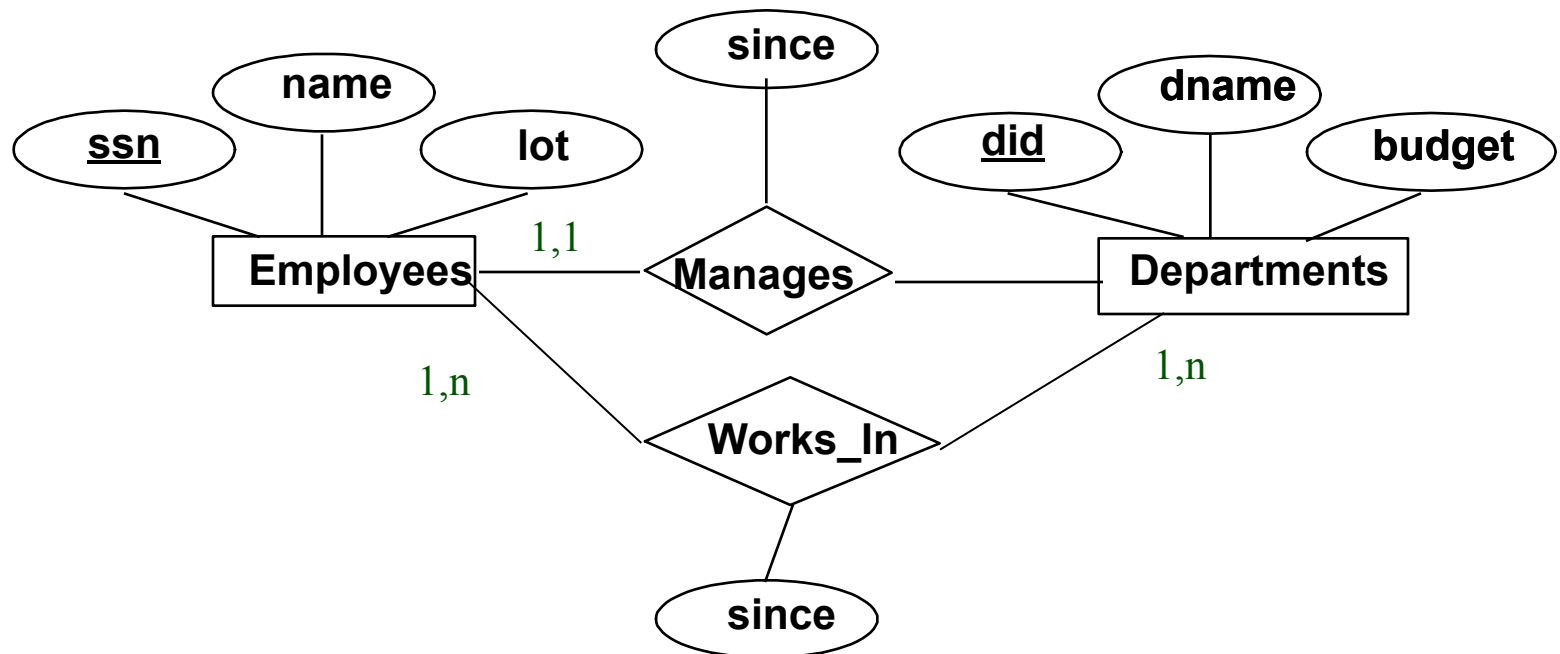
Referential Integrity in SQL

- ❖ SQL/92 and SQL:1999 support all 4 options on deletes and updates.
 - Default is **NO ACTION** (*delete/update is rejected*)
 - **CASCADE** (also delete all tuples that refer to deleted tuple)
 - **SET NULL / SET DEFAULT** (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
(sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid)
REFERENCES Students
ON DELETE CASCADE
ON UPDATE SET DEFAULT )
```

Review: Participation Constraints

- ❖ Does every department have a manager?
 - If so, this is a participation constraint: the participation of Departments in Manages is said to be *total* (vs. *partial*).
 - Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)



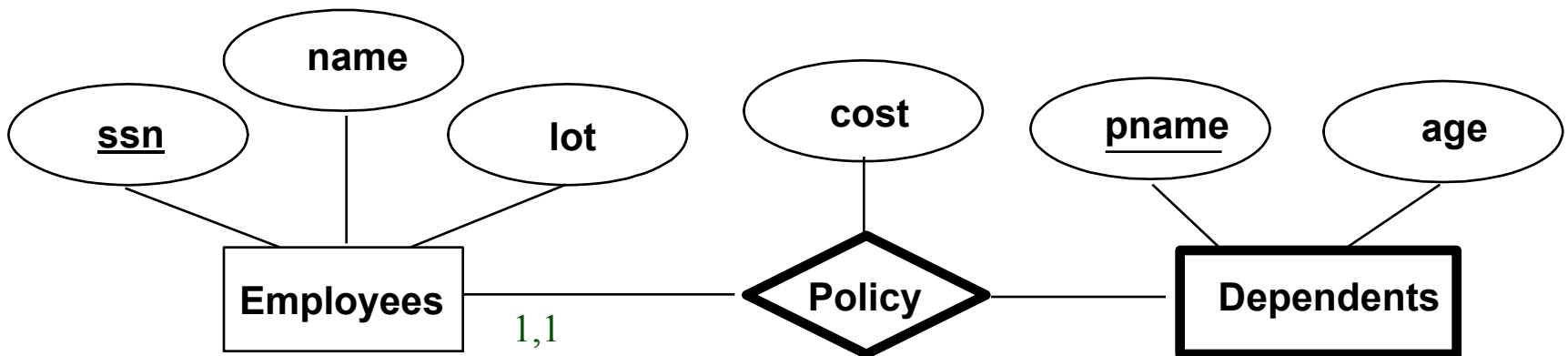
Participation Constraints in SQL

- ❖ We can capture participation constraints involving one entity set in a binary relationship, but little else (without resorting to CHECK constraints).

```
CREATE TABLE Dept_Mgr(  
    did INTEGER,  
    dname CHAR(20),  
    budget REAL,  
    ssn CHAR(11) NOT NULL,  
    since DATE,  
    PRIMARY KEY (did),  
    FOREIGN KEY (ssn) REFERENCES Employees,  
    ON DELETE NO ACTION)
```

Review: Weak Entities

- ❖ A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
 - Owner entity set and weak entity set must participate in a one-to-many relationship set (1 owner, many weak entities).
 - Weak entity set must have total participation in this *identifying* relationship set.



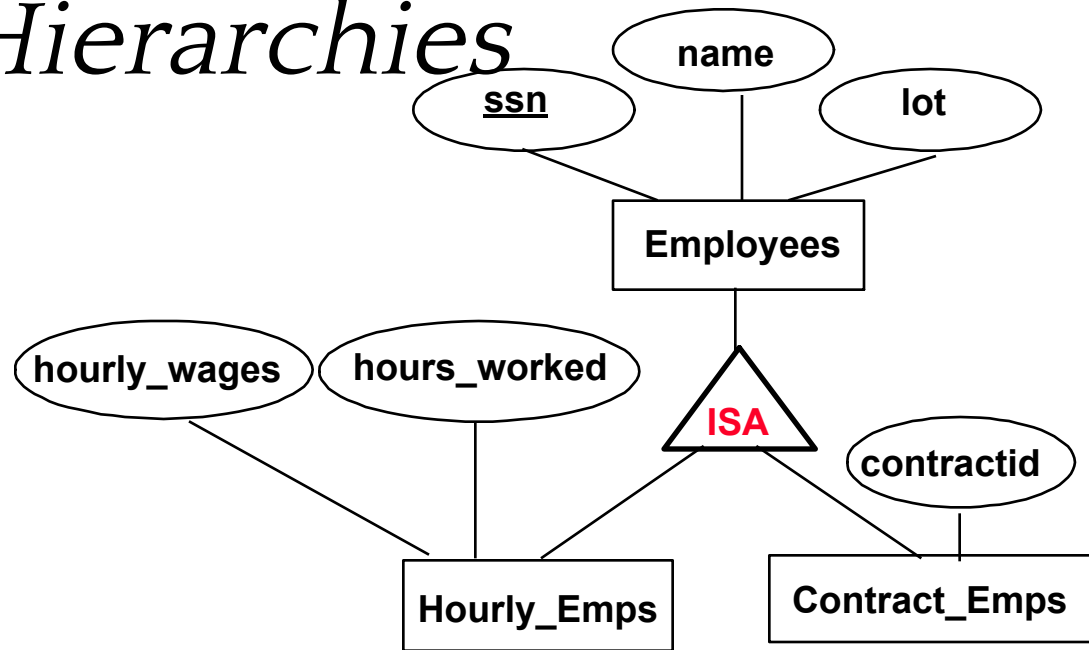
Translating Weak Entity Sets

- ❖ Weak entity set and identifying relationship set are translated into a single table.
 - When the owner entity is deleted, all owned weak entities must also be deleted.

```
CREATE TABLE Dep_Policy (  
    pname CHAR(20),  
    age INTEGER,  
    cost REAL,  
    ssn CHAR(11) NOT NULL,  
    PRIMARY KEY (pname, ssn),  
    FOREIGN KEY (ssn) REFERENCES Employees,  
    ON DELETE CASCADE)
```

Review: ISA Hierarchies

❖ If we declare A **ISA** B, every A entity is also considered to be a B entity.



- ❖ **Overlap constraints:** Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (*Allowed/disallowed*)
- ❖ **Covering constraints:** Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (*Yes/no*)

Translating ISA Hierarchies to Relations

❖ *General approach:*

- 3 relations: Employees, Hourly_Emps and Contract_Emps.
 - *Hourly_Emps*: Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (*hourly_wages*, *hours_worked*, *ssn*); must delete Hourly_Emps tuple if referenced Employees tuple is deleted.
 - Queries involving all employees easy, those involving just Hourly_Emps require a join to get some attributes.

❖ *Alternative: Just Hourly_Emps and Contract_Emps.*

- *Hourly_Emps*: *ssn*, *name*, *lot*, *hourly_wages*, *hours_worked*.
- Each employee must be in one of these two subclasses.

Views

- ❖ A view is just a relation, but we store a *definition*, rather than a set of tuples.

```
CREATE VIEW YoungActiveStudents (name, grade)
AS SELECT S.name, E.grade
FROM Students S, Enrolled E
WHERE S.sid = E.sid and S.age < 21
```

- ❖ Views can be dropped using the **DROP VIEW** command.
 - How to handle **DROP TABLE** if there's a view on the table?
 - DROP TABLE command has options to let the user specify this: RESTRICT/CASCADE.

Views and Security

- ❖ Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
 - Given YoungActiveStudents, but not Students or Enrolled, we can find students who are enrolled, but not the *cid*'s of the courses they are enrolled in.

Views: why?

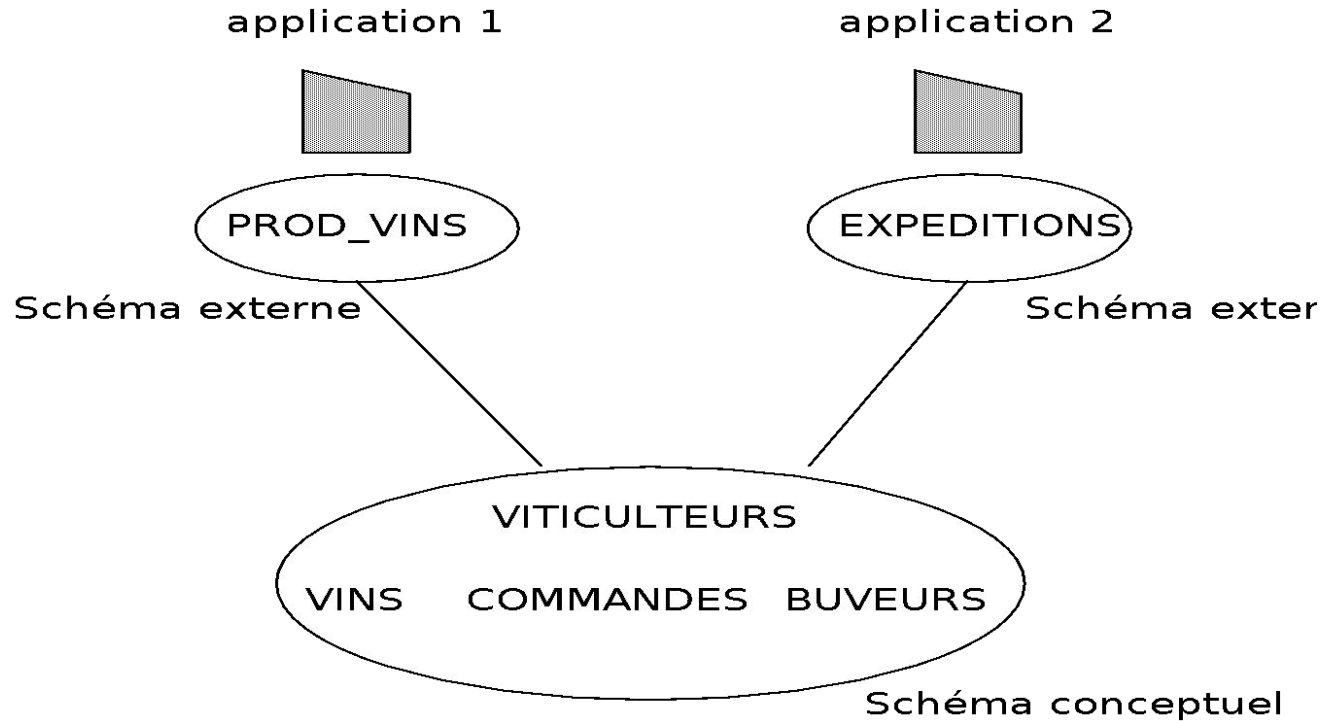
- ❖ The concept of views meets the following requirements:
 - Adaptation to applications
 - Integration of existing applications
 - Dynamic database schema
 - Privacy and Security
 - Decentralized administration of a DB
 - Heterogeneity models
 - Transparency of the locality (in the case of distributed databases)

Views : Create

```
CREATE [ OR REPLACE ] [ FORCE | NOFORCE ] VIEW  
view_name  
(attribute [, attribute ] )  
AS subquery  
[ WITH CHECK OPTION ]  
[ WITH READ ONLY ] ;
```

- ❖ OR REPLACE : recreate the **view** if already exists
- ❖ WITH CHECK OPTION : specify that only the accessible lines in the view can be inserted or modified.
- ❖ WITH READ ONLY specify that no modification (INSERT, UPDATE, ...) can be made on the lines of the view.

Example



VINS (NV, CRU, ANNEE, DEGRE, NVIT)
VITICULTEURS (NVIT, NOM, PRENOM, VILLE)
BUVEURS (NB, NOM, PRENOM, VILLE)
COMMANDE (NB, NV, QTE, DATE)

Example of view (1)

```
CREATE VIEW BUVEURS-1  
AS  SELECT NB, VILLE  
    FROM BUVEURS
```

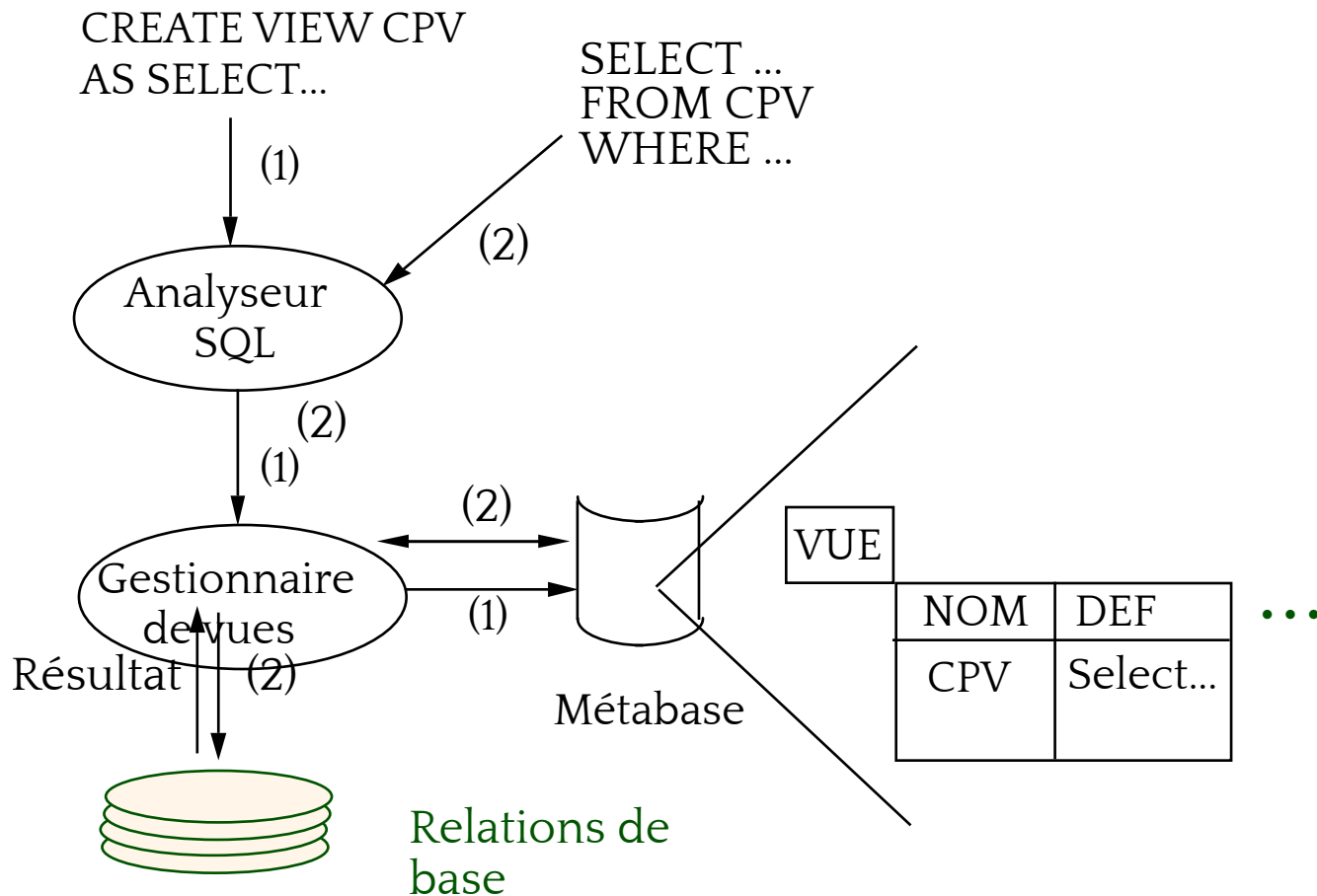
```
CREATE VIEW ACHETE-1  
AS  SELECT NB, NV, QTE, DATE  
    FROM COMMANDE  
    WHERE DATE BETWEEN '01.01.96' AND '31.12.96'
```

```
CREATE VIEW ACHAT-2 (NB, VILLE, CRU, QTE, DATE)  
AS  SELECT B.NB, B.VILLE, V.CRU, C.QTE, C.DATE  
    FROM BUVEURS B, VINS V, COMMANDES C  
    WHERE      C.NB = B.NB AND  
              C.NV = V.NV AND  
              C.DATE BETWEEN '01.01.96' AND '31.12.96'
```

Example of view(2)

```
CREATE VIEW CPV (VILLE, CRU, QTE)
AS SELECT  B.VILLE, V.CRU, SUM (C.QTE)
FROM      BUVEURS B, COMMANDES C, VINS V
WHERE     B.NB = C.NB AND
          C.NV = V.NV AND
          C.DATE BETWEEN '01.01.96' AND '31.12.96'
GROUP BY B.VILLE, V.CRU
```

Declaration and interrogation



Views: querying (1)

Query modification

EXAMPLE :

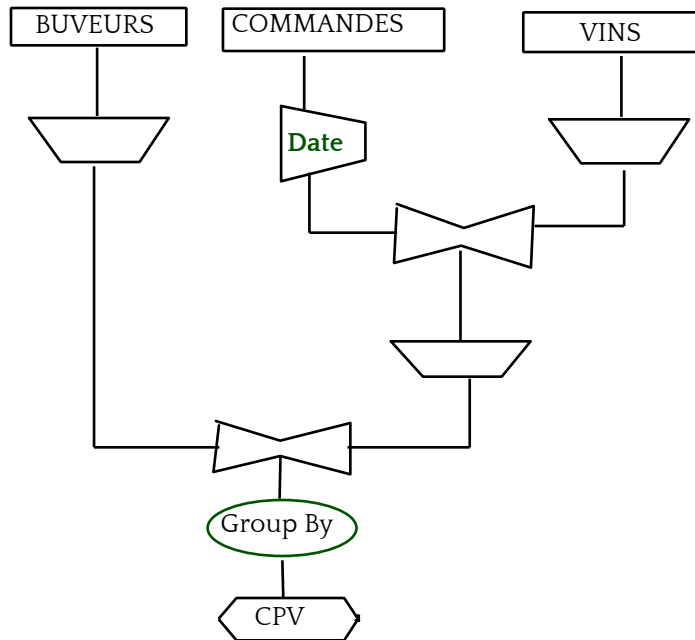
```
SELECT CRU, QTE  
FROM CPV  
WHERE CITY = "PARIS"
```

BECOMES :

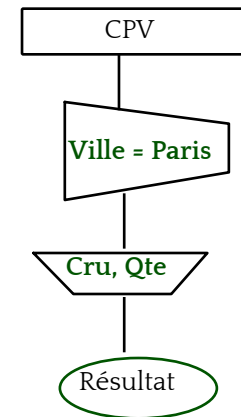
```
SELECT V.CRU, SUM (C.QTE)  
FROM BUVEURS B, COMMANDES C, VINS V  
WHERE B.NB = C.NB AND  
      C.NV = V.NV AND  
      C.DATE BETWEEN '01.01.96' AND '31.12.96' AND  
      B.VILLE = 'PARIS'  
GROUP BY VILLE, CRU
```

Views: querying(2)

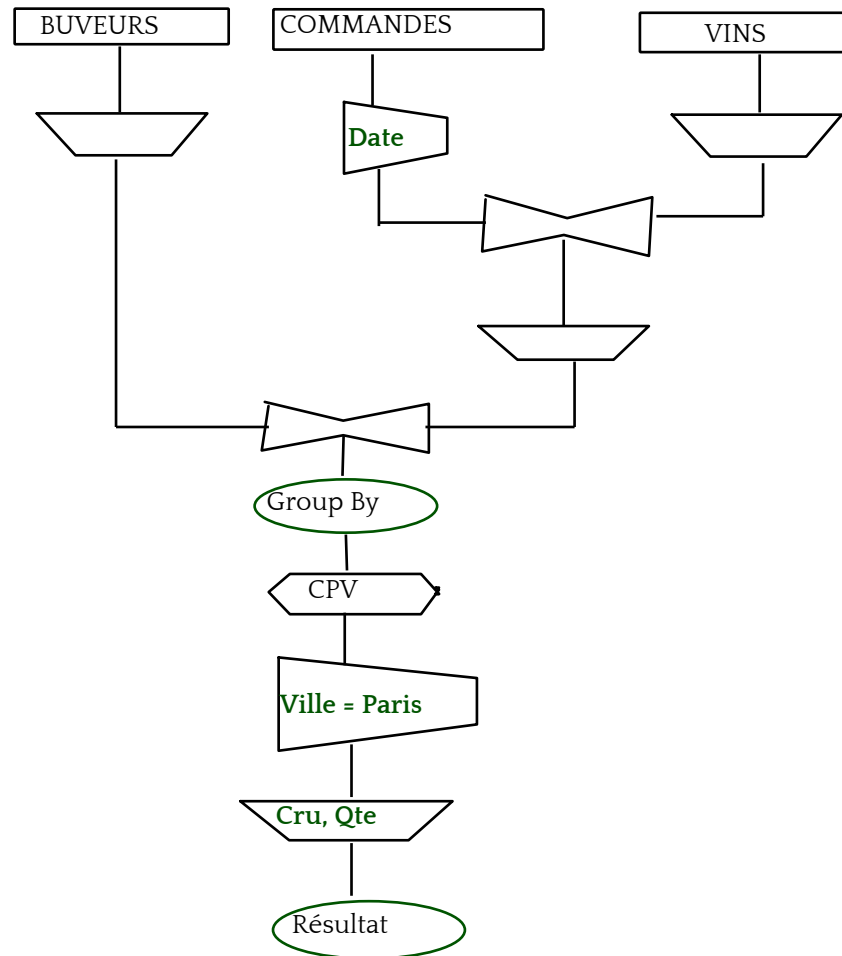
Algebraical tree of CPV view



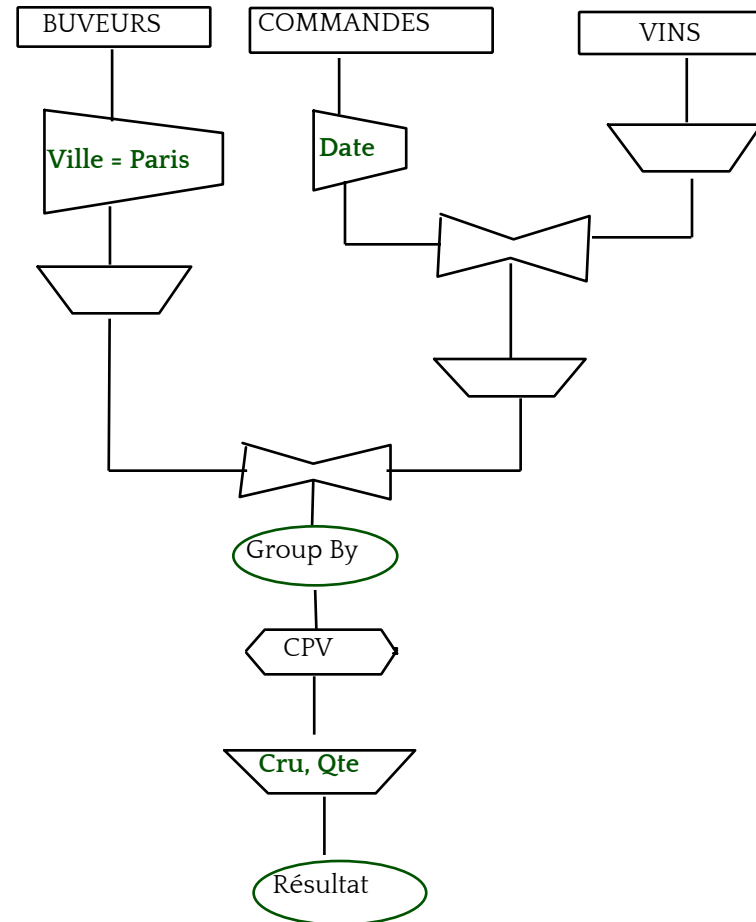
Algebraical tree of the query



Merging the trees



Optimizing the tree



Update views

- ❖ Updates through views are rarely defined
 - Ex: How to report an update on CPV in the tables?
- ❖ To make the report of updates possible, the view definition must meet some requirements
 - SELECT clause must keep the keys of all involved tables
 - SELECT clause must not contain aggregate calculation
 - The definition must not contain a GROUP BY clause or HAVING

Relational Model: Summary

- ❖ A tabular representation of data.
- ❖ Simple and intuitive, currently the most widely used.
- ❖ Integrity constraints can be specified by the DBA, based on application semantics. DBMS checks for violations.
 - Two important ICs: primary and foreign keys
 - In addition, we *always* have domain constraints.
- ❖ Powerful and natural query languages exist.
- ❖ Rules to translate ER to relational model