Introduction: DBMS

Bipin C. DESAI

Grading:

As per the administrative slides

Only random parts (which could be all or some or none) of the assignments could be graded!

The final letter grade would be based on the traditional conversion scheme; e.g, A’s would be assigned to numerical total marks well above high 80s

Office hours: As in CrsMgr
For common questions answers would be posted on the announcements/FAQ entries for the course page in CrsMgr page. Pl. read/re-read these before sending an email. Pl. send email in text
DO NOT SEND DUPLICATE emails

Tutors, Lab Instructors : As announced on CrsMgr

Account for course manager has been created and the ID/PW have been emailed!
If you have not seen it, look in your spam/thrash folder.
Late registration will have to wait for their records to be exported to CrsMgr.
For the students who haven’t provided an email to the Concordia’s SIS, you need to find out your ID/PW!
Sign in to CrsMgr: change your PW & update email address

Course Manager System(CrsMgr):
https://confsys.encs.concordia.ca/CrsMgr

If you are officially registered in the course, you would be registered in CrsMgr.
CrsMgr would be used for administrating on-line quizzes, managing the grades, submission of assignments and reports, scheduling of demos and peer evaluation for each group marked entity.

Your grade and the feedback could be viewed in CrsMgr.

Please read the announcements for this course regularly in CrsMgr.
Answers to common questions as well additional instructions
for the course would be posted in the FAQ section.
Look at it before sending out emails.
No emails will be answered if the instructions are already posted.
Pl. send email in text – DOES NOT MEAN TEXT MESSAGES

Sign in to CrsMgr and change your email adr. to the ENCS email.

Form a team of 4 in the first class or before the deadline:

Choose a leader: the leader signs into CrsMgr and joins a new group

- update his/her email to **ENCS** computer account
- ~~insert a password for group DB~~ (would be generated automatically):

Each member of the team then joins this group.

For each member of the team:

- update his/her email to **ENCS** computer account

Above steps could be done by using and uploading a text file!

If you do not join a group, youwill be put randomly in any group to create groups with up to 4 members

Note: Assignments and Projects (warm up and main) would be done by this group with a single submission per group to be uploaded by the group(team) leader.

Upload the assignment/project to the course manager system https://confsys.encs.concordia.ca/CrsMgr

ONLY the group leader could upload a group submission.

Software

On the ENCS system:

MySQL/MariaDB (Oracle is no longer supported by AITS!)
PHP
HTML
CGI (security) – defunct: each group have their own system

On you own(for WinX or Linux)

Download and install Mariadb(opensource version of MySQL)
https://mariadb.org/
Install PHP: http://www.php.net/
Install Web Server: http://www.apache.org/

Remember the demo would be on the ENCS system so you need to port your code/database Your system must also work on a Linux platform.
AITS uses MySQLm

Look into LAMP or its port to Windows; the following may have changed!

https://ampps.com/download
https://codebriefly.com/how-to-setup-apache-php-mysql-on-windows-10/

If you use other database engines:

NO HELP from us

Only MySQL is supported by ENCS's AITS

You are free to use any database engine.

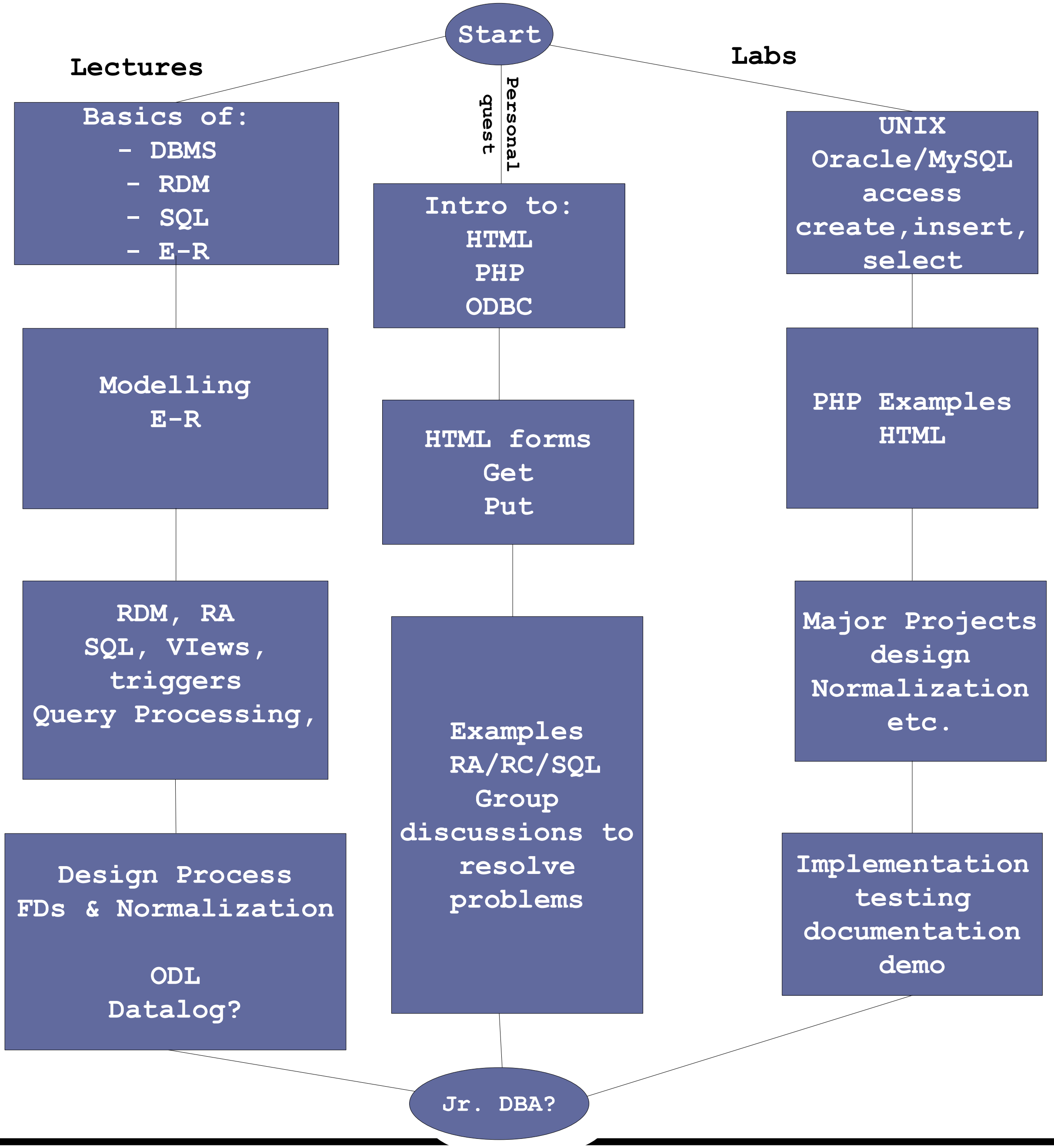
However, you are on your own to download and install the database on your own system;

You also need to make arrangements with your teammates so that the work is coordinated.

You need to install PhP and Apache servers as well.

The database system applications you develop for the projects **must** be compatible with Linux (optionally WinX) and ported to one of our system for the demo.

Remember the demo would be on the ENCS system so you need to port your projects!



Start

Lectures

Basics of:
- DBMS
- RDM
- SQL
- E-R

Modelling
E-R

RDM, RA
SQL, VIews,
triggers
Query Processing,

Design Process
FDs & Normalization

ODL
Datalog?

Personal quest

Intro to:
HTML
PHP
ODBC

HTML forms
Get
Put

Examples
RA/RC/SQL
Group
discussions to
resolve
problems

Labs

UNIX
Oracle/MySQL
access
create,insert,
select

PHP Examples
HTML

Major Projects
design
Normalization
etc.

Implementation
testing
documentation
demo

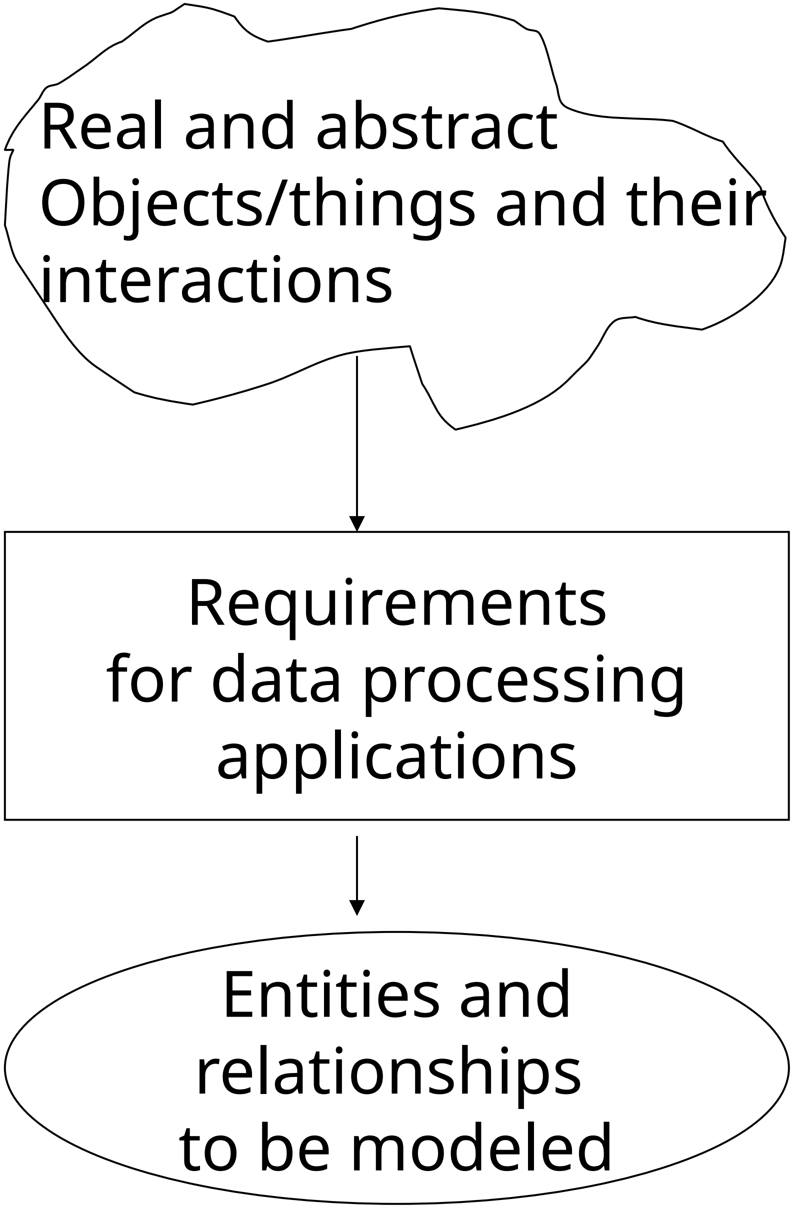
Jr. DBA?

Modeling techniques (E-R, ODL)

Basic relational model

Design of database applications

Database programming (MySQL, SQL, PHP, HTML, CSS, Javascript)



Real and abstract Objects/things and their interactions

Requirements for data processing applications

Entities and relationships to be modeled

DBMS! What is it?

Database is an integrated data collection (Logically consistent and persistent)

It is derived from the model of a set of applications for a real world enterprise.

DBMS is a software package designed to make managing almost any database.

DBMS offers: data independence, efficiency, integrity, security, concurrency, recovery

Why Database?

Information Age: 30-40% of world trade and growing

Web(Unstructured data) and .com

Digital Library

Human Genome Project

Email,
Entertainment
OSN,
Shopping

Day to day operation of Mama/Papa Store

List of titles, artist, and download site of shared files.

Information about employees, departments, projects, etc. in an organization

Information about students, courses, enrollments, professors, etc. in an educational institute

Information about books, videos, albums, members, etc. in a library

DBMS is a complex set of software packages:

- create new databases, store and manage data
- provide application development and support environment

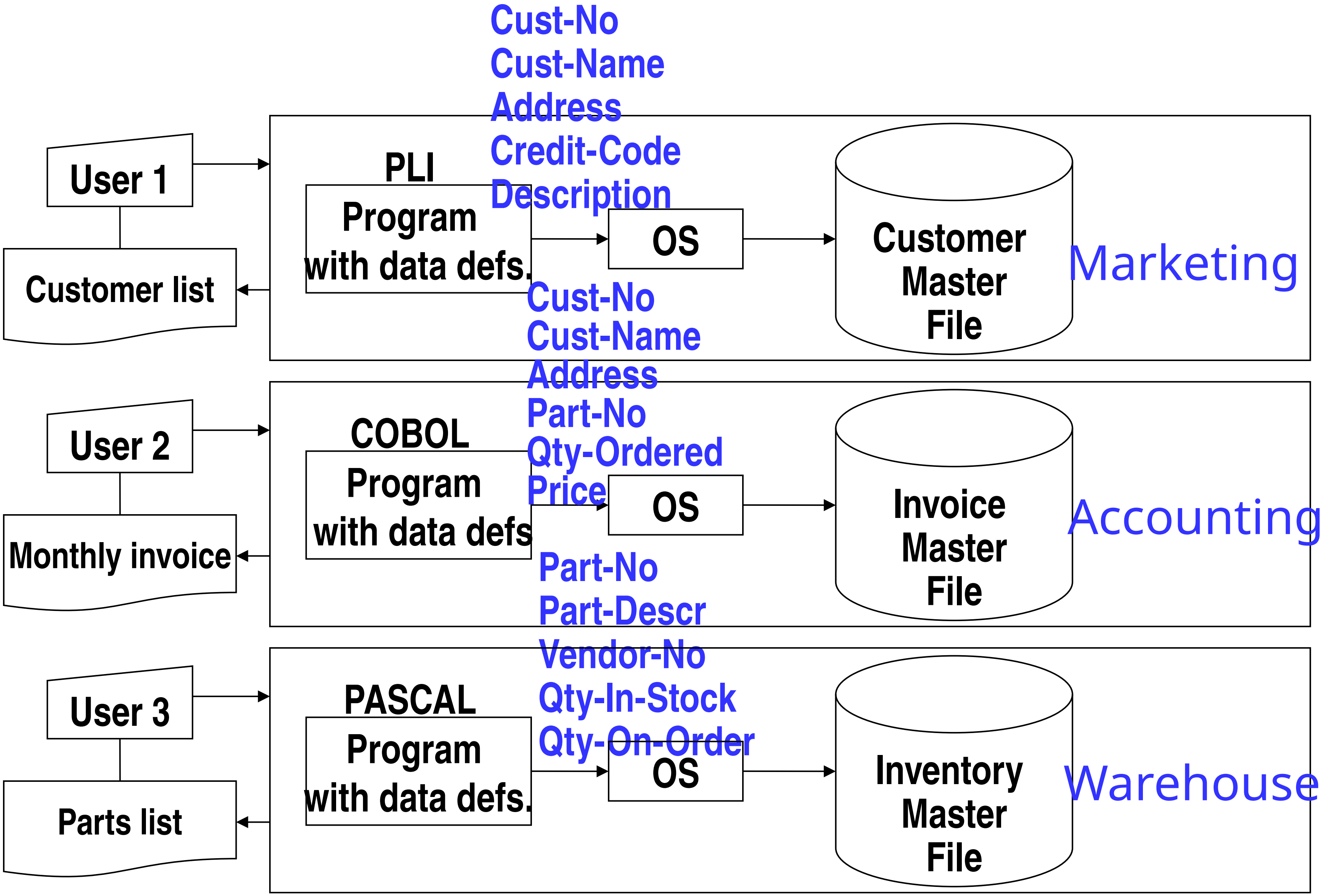
Application Support: Gives developers tools to build applications for using the data. Allows easy method for users to query and modify the data

Persistent storage: Support the storage of data

Transaction management: Controls concurrent access to data from many users

Supports the ACID properties .

Atomicity Concurrency Integrity Durability



User 1

Customer list

PLI

Program with data defs.

Cust-No
Cust-Name
Address
Credit-Code
Description

OS

Customer Master File

Marketing

User 2

Monthly invoice

COBOL

Program with data defs

Cust-No
Cust-Name
Address
Part-No
Qty-Ordered
Price

OS

Invoice Master File

Accounting

User 3

Parts list

PASCAL

Program with data defs.

Part-No
Part-Descr
Vendor-No
Qty-In-Stock
Qty-On-Order

OS

Inventory Master File

Warehouse

Pros & Cons of file based system

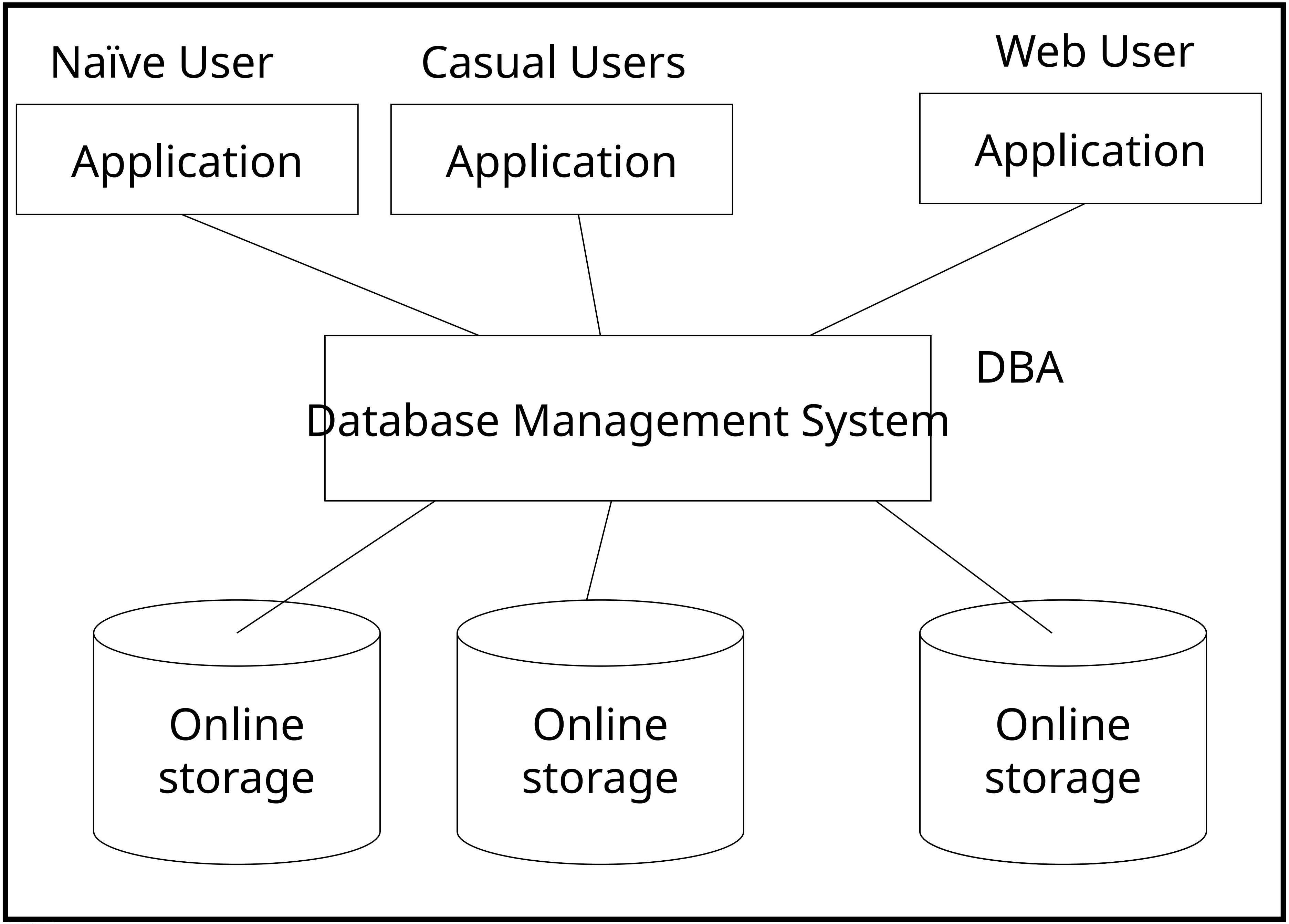
Sharing not possible data definition is “locked” in application programs which “owns” the file and the data in it

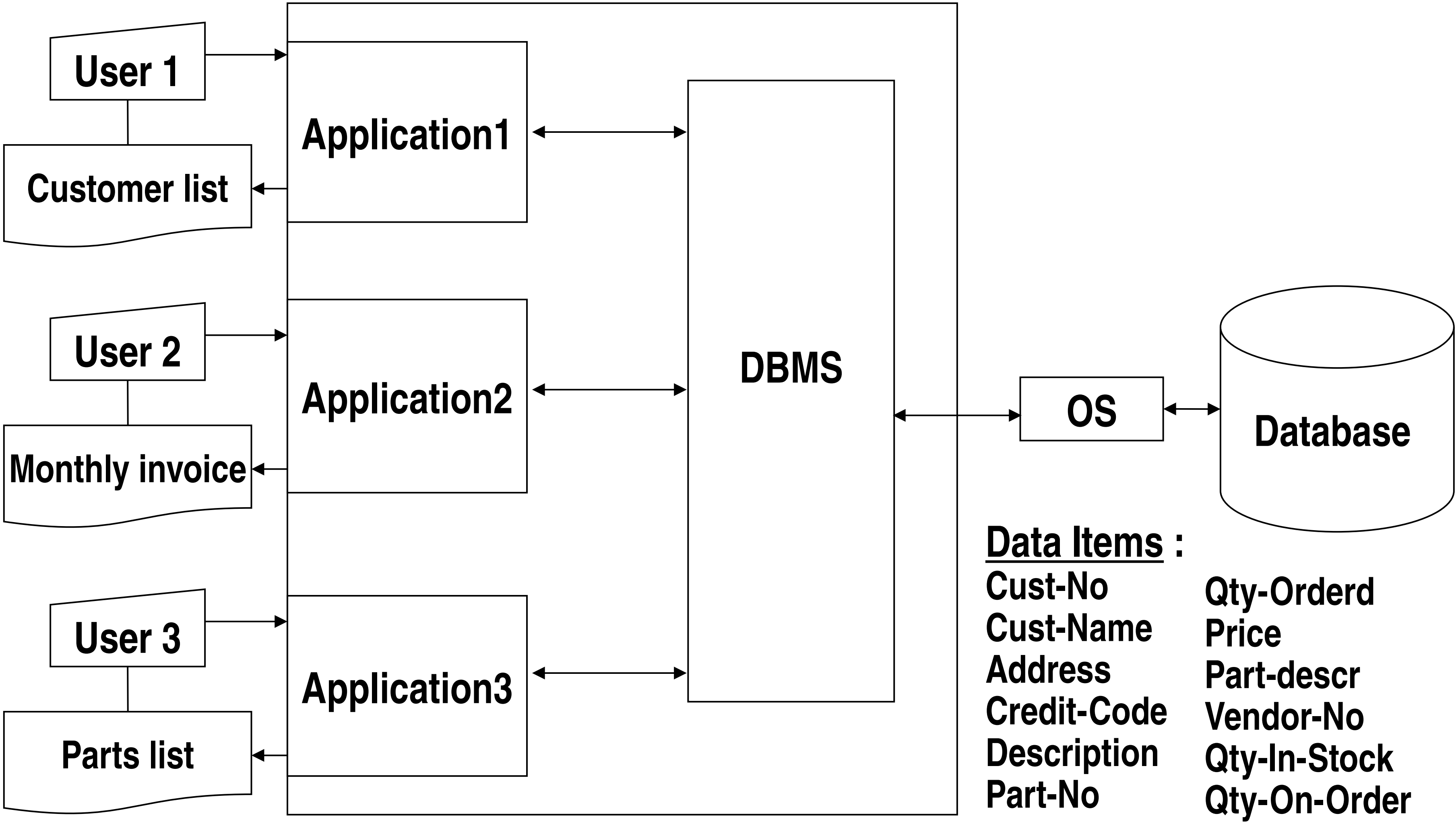
Redundancy of data: Same data is duplicated perhaps in slightly different format over various files

Multiple updates: Changes have to be made to all files containing the same data. ***Possibility of inconsistency***

Waste of storage space:

Reliability and better local control


Naïve User
Casual Users
Web User
Application
Application
Application
DBA
Database Management System
Online storage
Online storage
Online storage



Data Items :

Cust-No
Cust-Name
Address
Credit-Code
Description
Part-No

Qty-Orderd
Price
Part-descr
Vendor-No
Qty-In-Stock
Qty-On-Order

Pros & Cons of DBMS

Reduce data redundancy and avoiding inconsistency

Provide Concurrent access

Offer Centralized control

- security(appropriate authorization and its control),
- integrity(constraints and their enforcements)
- reliability(backups and replication)

Data abstraction and independence

First Step: Data Models

- Data Model: concept to describe data
- Schema: description of a collection of data using a specific data model
- Relational Model: Based on the concept of **relation***(table with rows and columns)*

A Data Model is a collection of concepts for describing

- Entities(objects) and relationships among them
- Expressing the semantics and constraints from the real world

Object-Based Modeling Techniques

- Entity-Relationship (ER) Model
- Object-Oriented (OO) Model

Record-Based Models

- Hierarchical Model: used by earliest DBMS – IBM's IMS
- Network Model: second generation DBMS - DBTG
- Relational Model: the first based on theory - relations

  (RA, RC, Datalog)

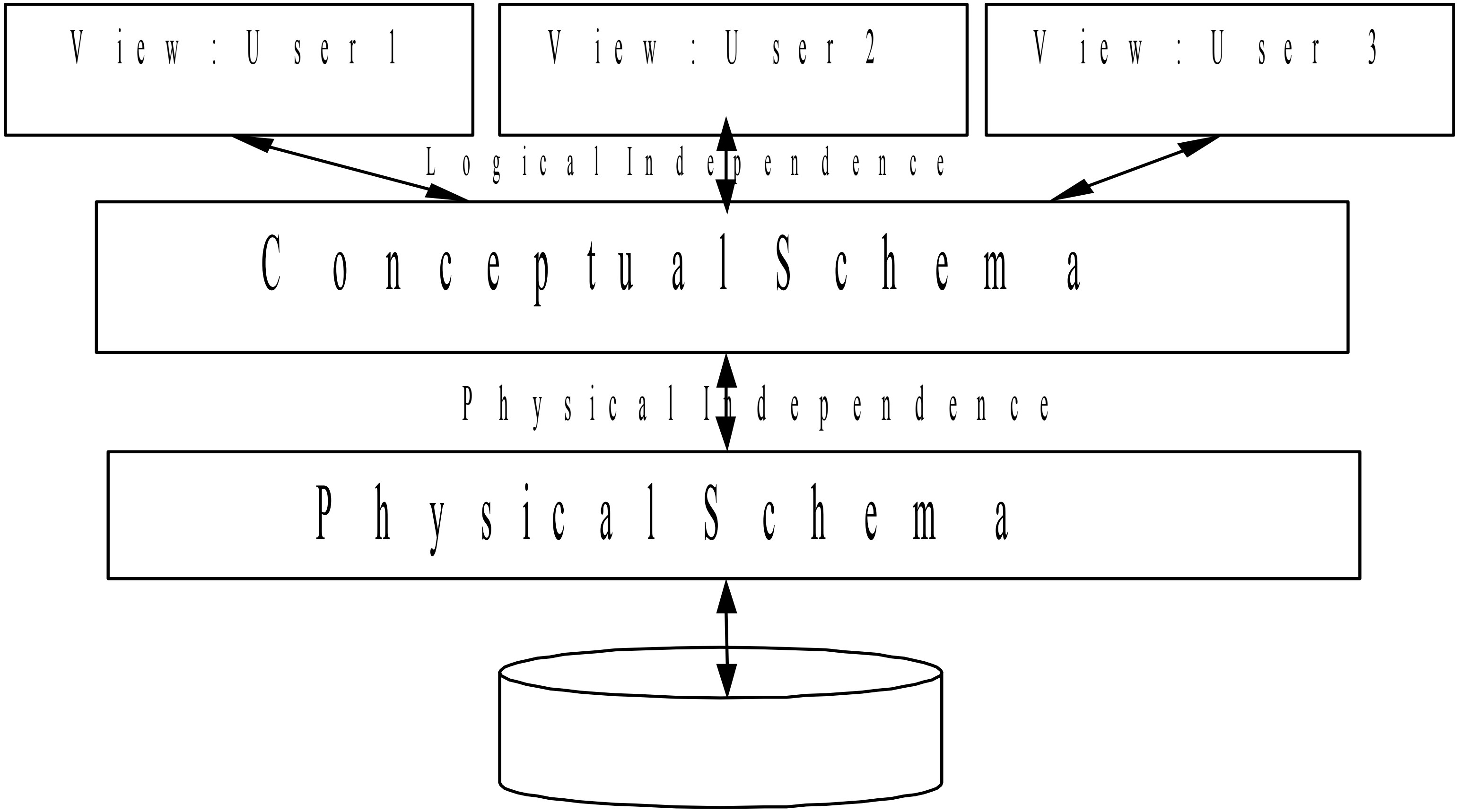
Employee Name
Employee Phone Number

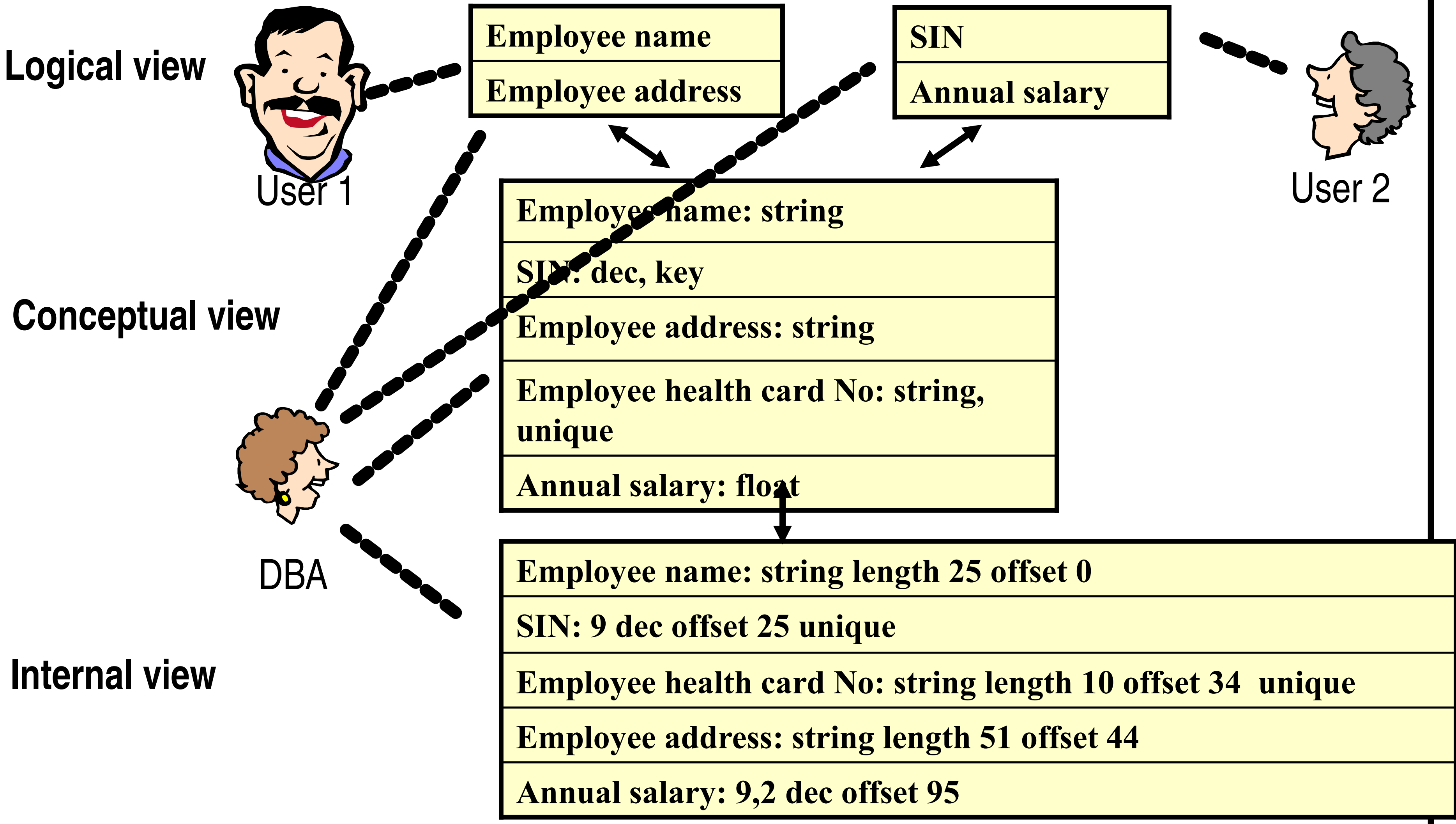
Employee Name
Employee SIN
Employee Salary

Employee Name
Employee Phone Number
Employee SIN
Employee Address
Employee Annual Salary
Employee YTD Salary

Employee Name string
Employee Phone Number digits
Employee SIN digits
Employee Address string
Employee Annual Salary money units
Employee YTD Salary money units

Three level Concepts

- 



Logical view

User 1

Employee name

Employee address

SIN

Annual salary

User 2

Conceptual view

Employee name: string

SIN: dec, key

Employee address: string

Employee health card No: string, unique

Annual salary: float

DBA

Internal view

Employee name: string length 25 offset 0

SIN: 9 dec offset 25 unique

Employee health card No: string length 10 offset 34 unique

Employee address: string length 51 offset 44

Annual salary: 9,2 dec offset 95

Three levels & Independence

- User View: How users view data - derived from conceptual view-
- Conceptual Schema: Logical structure of the database
- Physical Schema: The actual files and indices used
- Schema defined using DDL

Data Independence: modify definition of schema at one level without affecting a schema definition at a higher level.

Logical Data Independence: modify logical schema without causing application programs to be rewritten

- adding new fields to a record or changing the type of a field

Physical Data Independence: modify physical schema without causing logical schema or applications to be rewritten

- changing file structure from sequential to direct access

University Database

- External Schema:

Course_Enrol(C#:char, Number:int);

- Conceptual Schema:

Student(S#, Name, Dept)

Course(C#, Cname, Credits)

Enrollment(C#, S#, grade)

- Physical Schema:

files, indexed on S#, C#, etc

A database schema is a description of a particular collection of data, using a given data model

Part of a schema for a university. database in relation model would contain among others, the following:

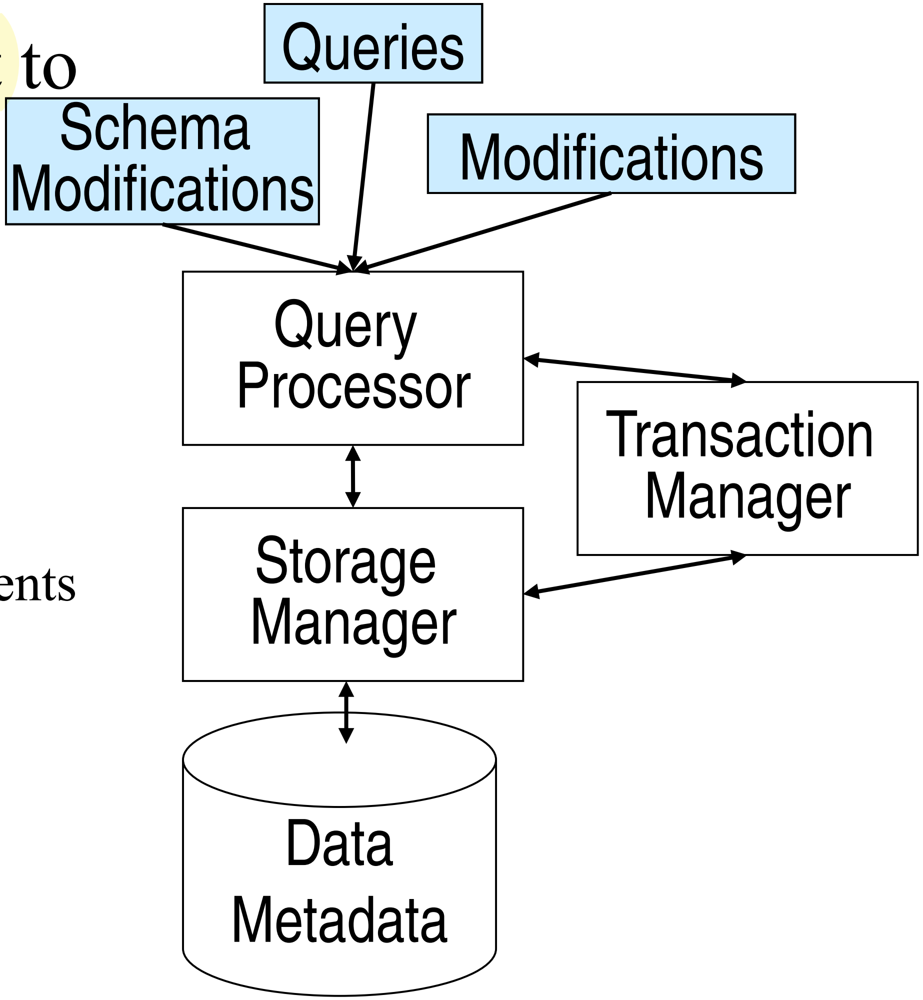
Students (sid, name, department, dob, address)

An instance of a database schema is the actual content of the database at a particular point in time

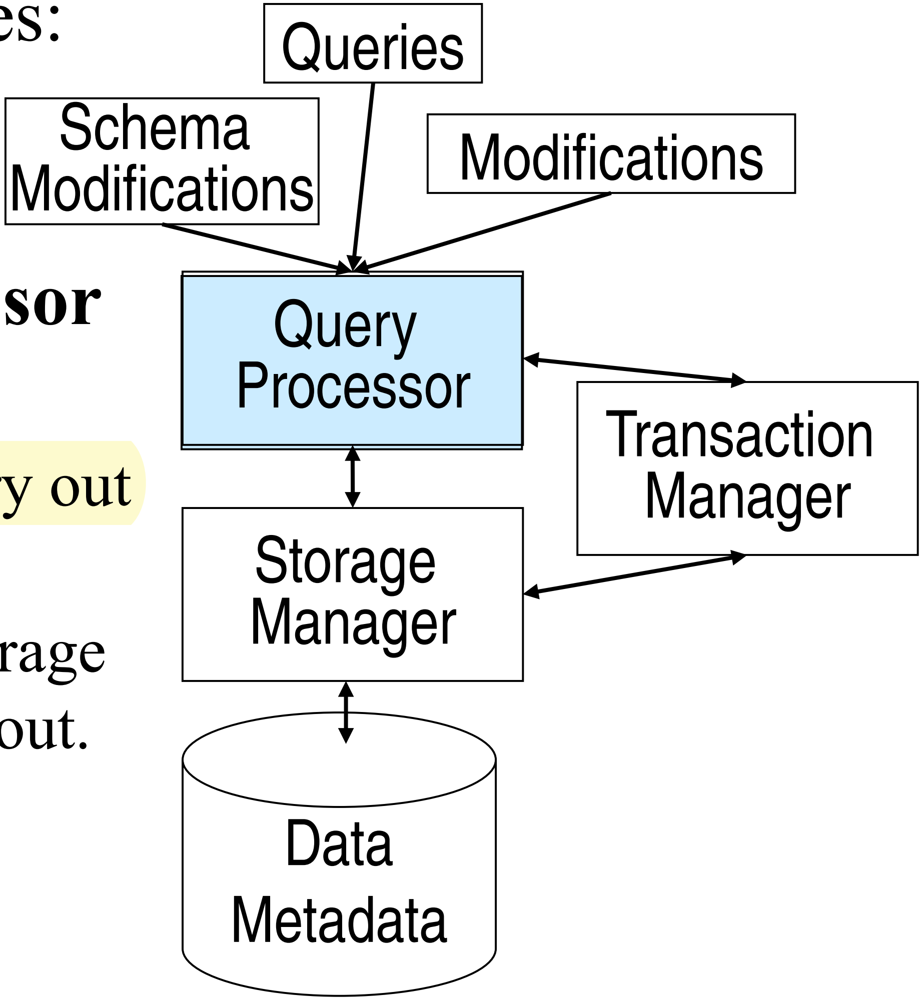
sid	name	department	dob	address
1112223	John Smith	CS	12-01-82	22 Pine, #1203
2223334	Ali Brown	EE	31-08-73	2000 St. Marc
3334445	Youwong Li	CS	23-11-79	1150 Guy

The Architecture of a DBMS

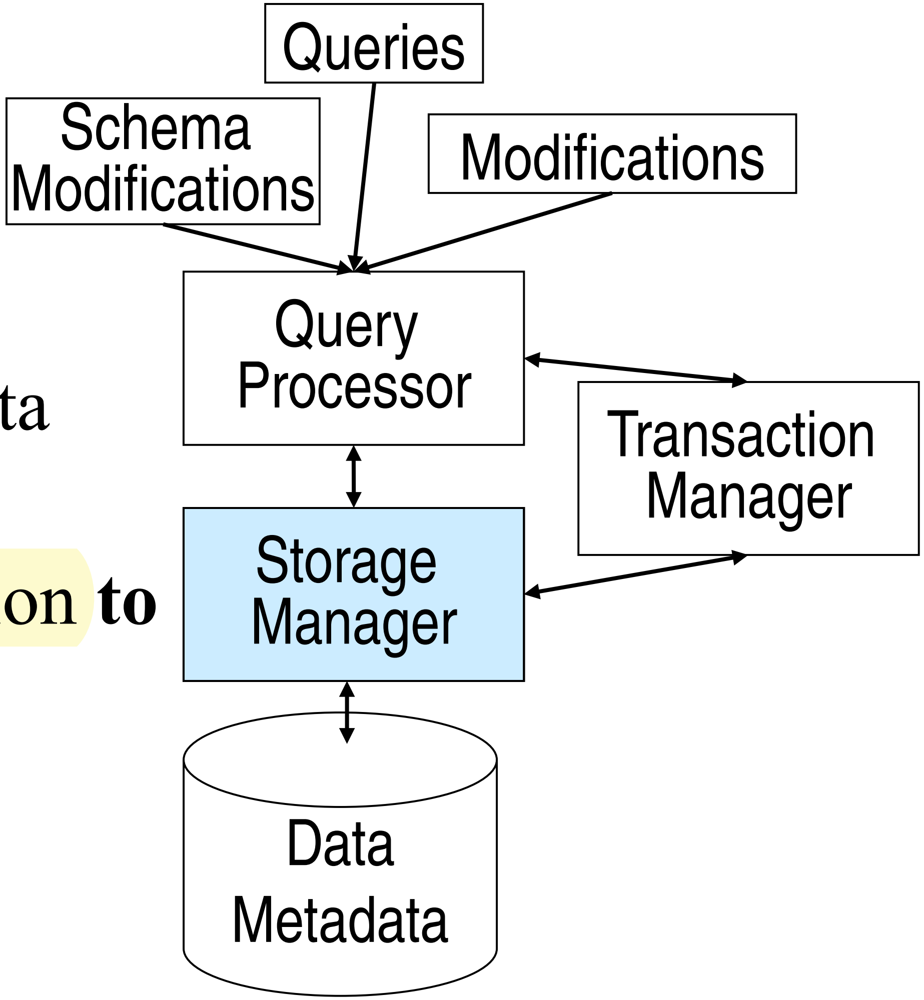
- There are 3 types of input to DBMS:
  - Access via queries
  - Updates to data
  - Updates to model
    - □ Initial database creation,
    - □ addition to schema components
    - □ schema modifications



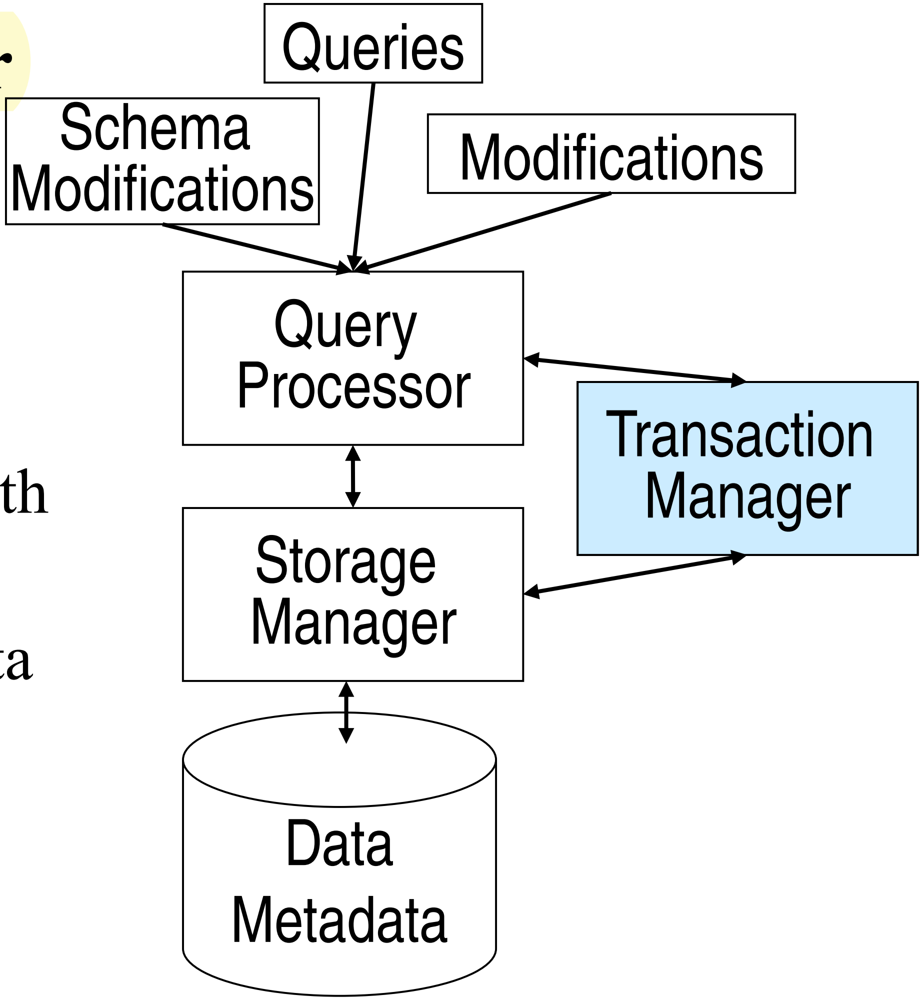
- The **query processor** handles:
  - Queries
  - Updates
- The job of the **query processor** which includes an optimizer
  - To find the "best" way to carry out a requested operation
  - To issue commands to the storage manager that will carry them out.



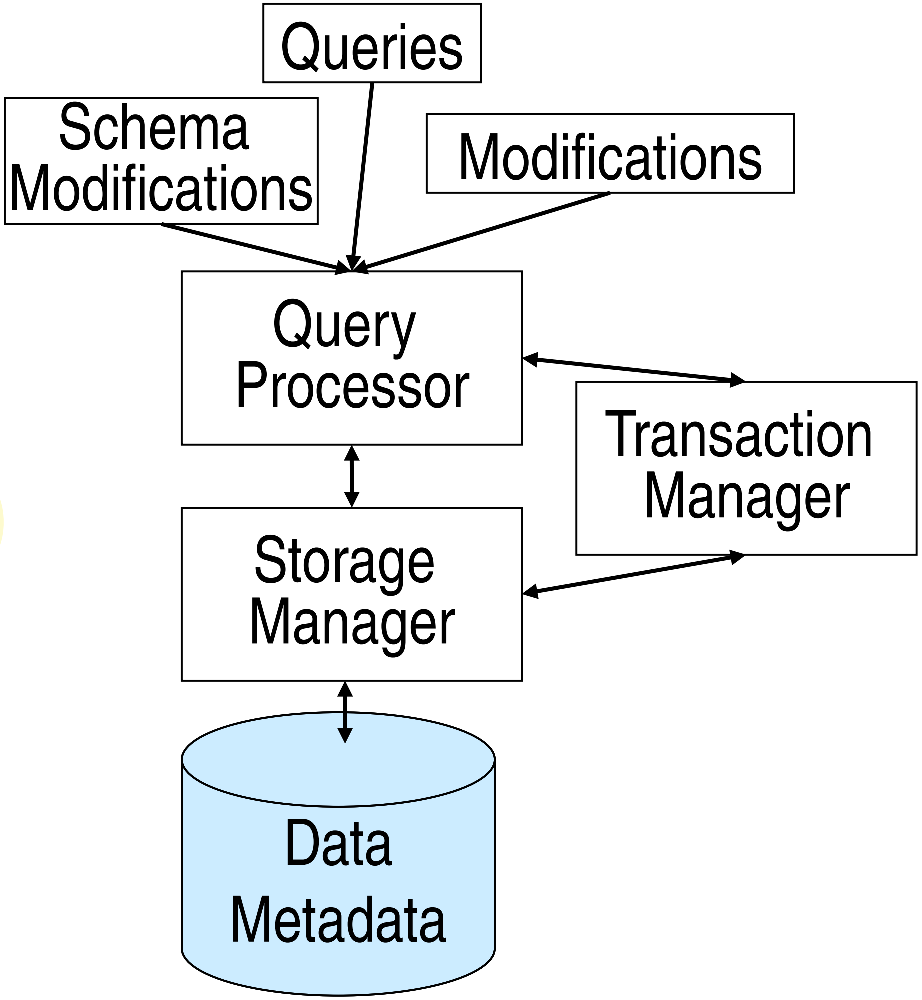
- The job of the **storage manager** is
  - To obtain requested information **from** the data storage
  - To modify the information **to** the data storage when requested.



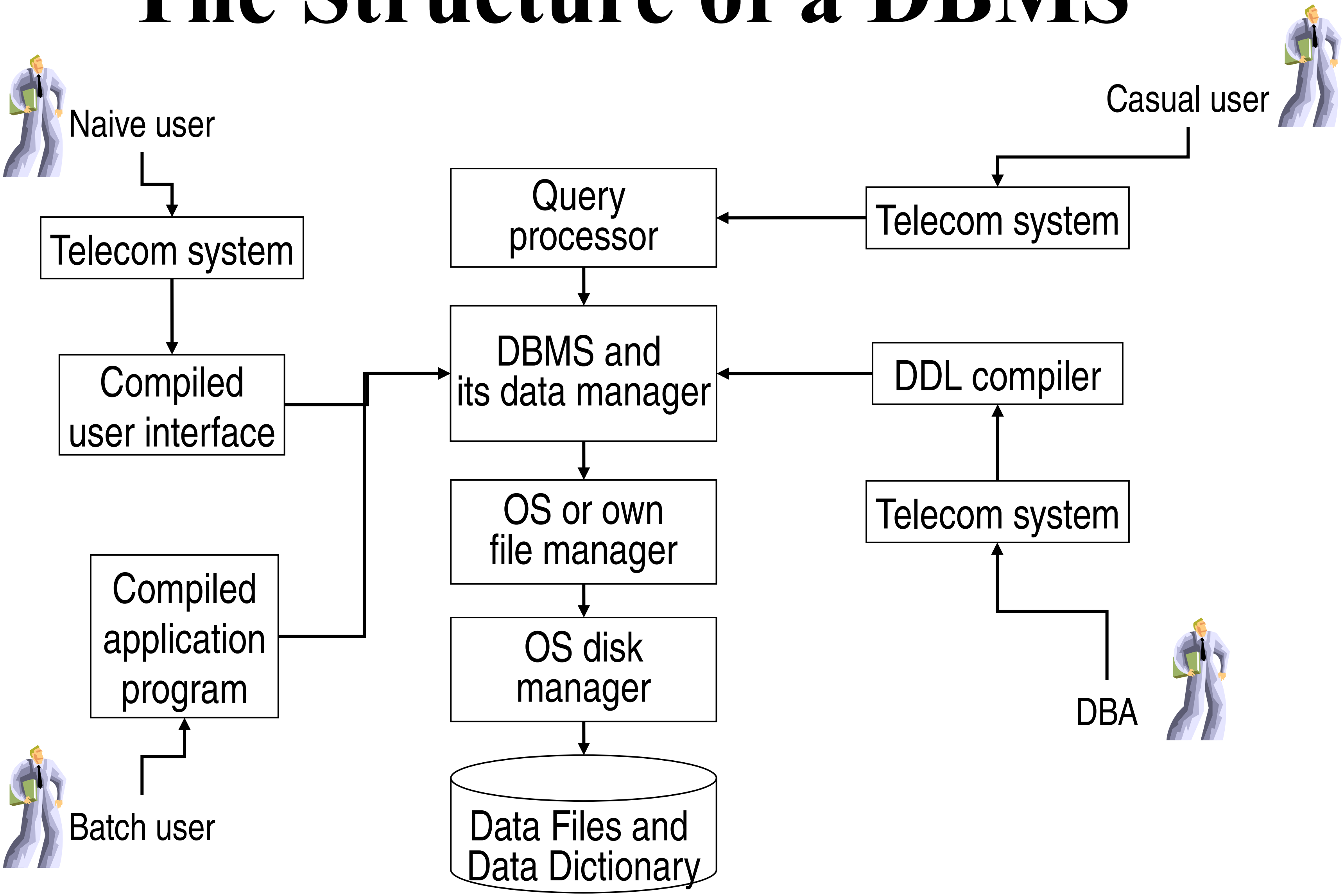
- The **transaction manager** is responsible for the **enforcing ACIDity**
  - several concurrent transactions(one or more queries) do not interfere with each other
  - the system will not lose data even if there are failures (*done through Recovery subsystem*)



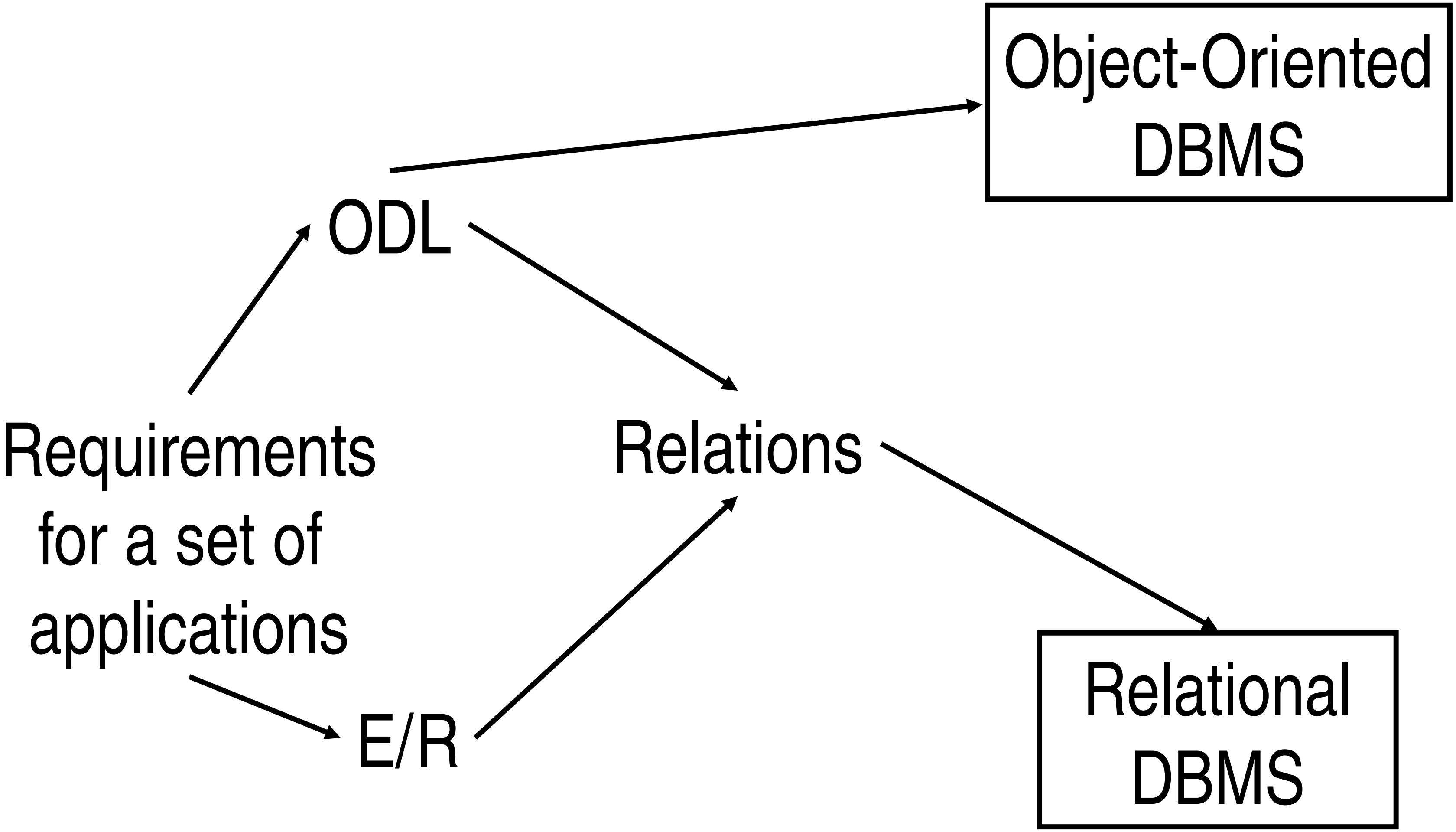
- Database contents include:
  - Metadata for the DBMS and one or more databases
  - Data belonging to one or more databases
  - Access aids such as indices and statistics



The Structure of a DBMS



Database Design Process
and
Conceptual Design



Object-Oriented DBMS

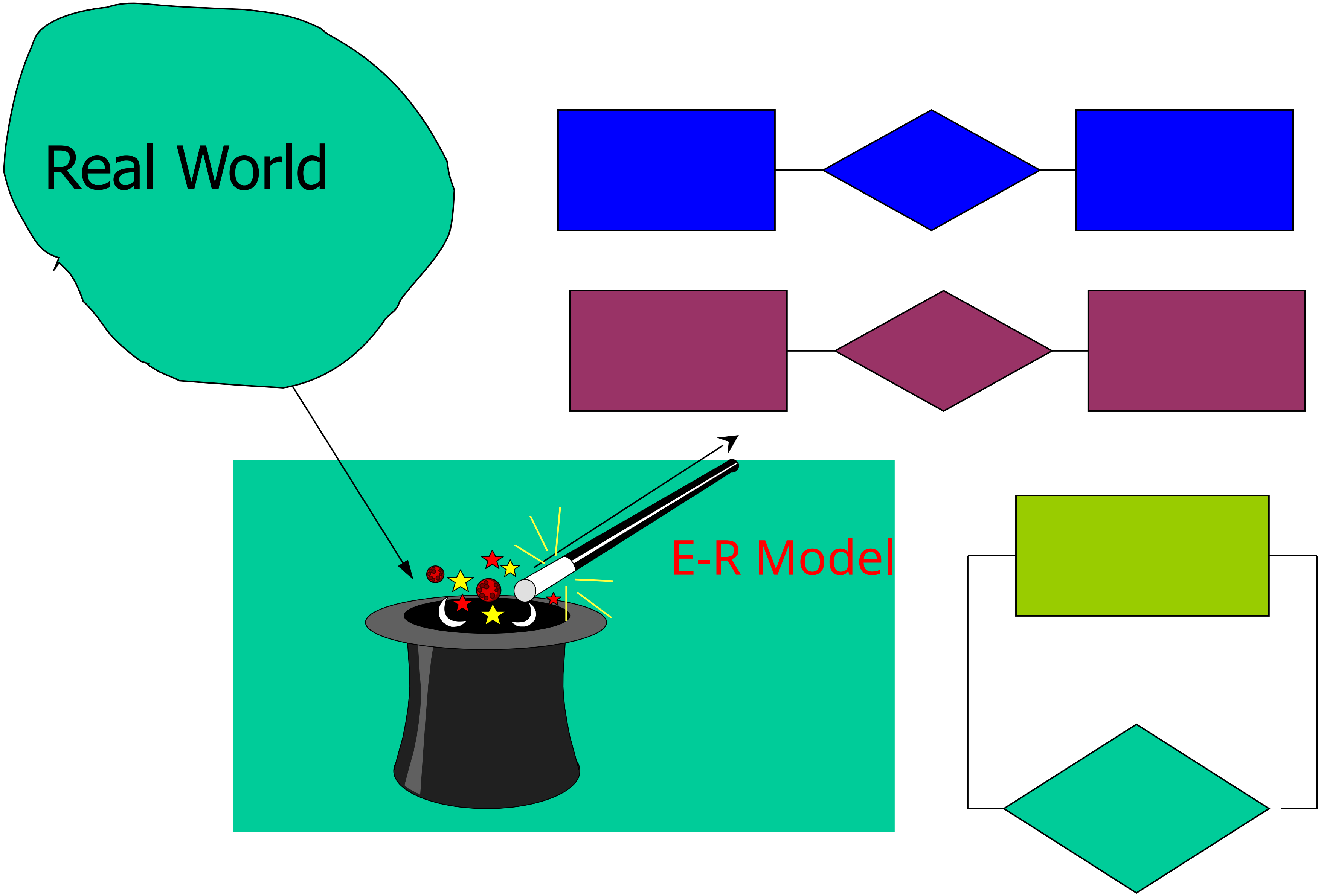
ODL

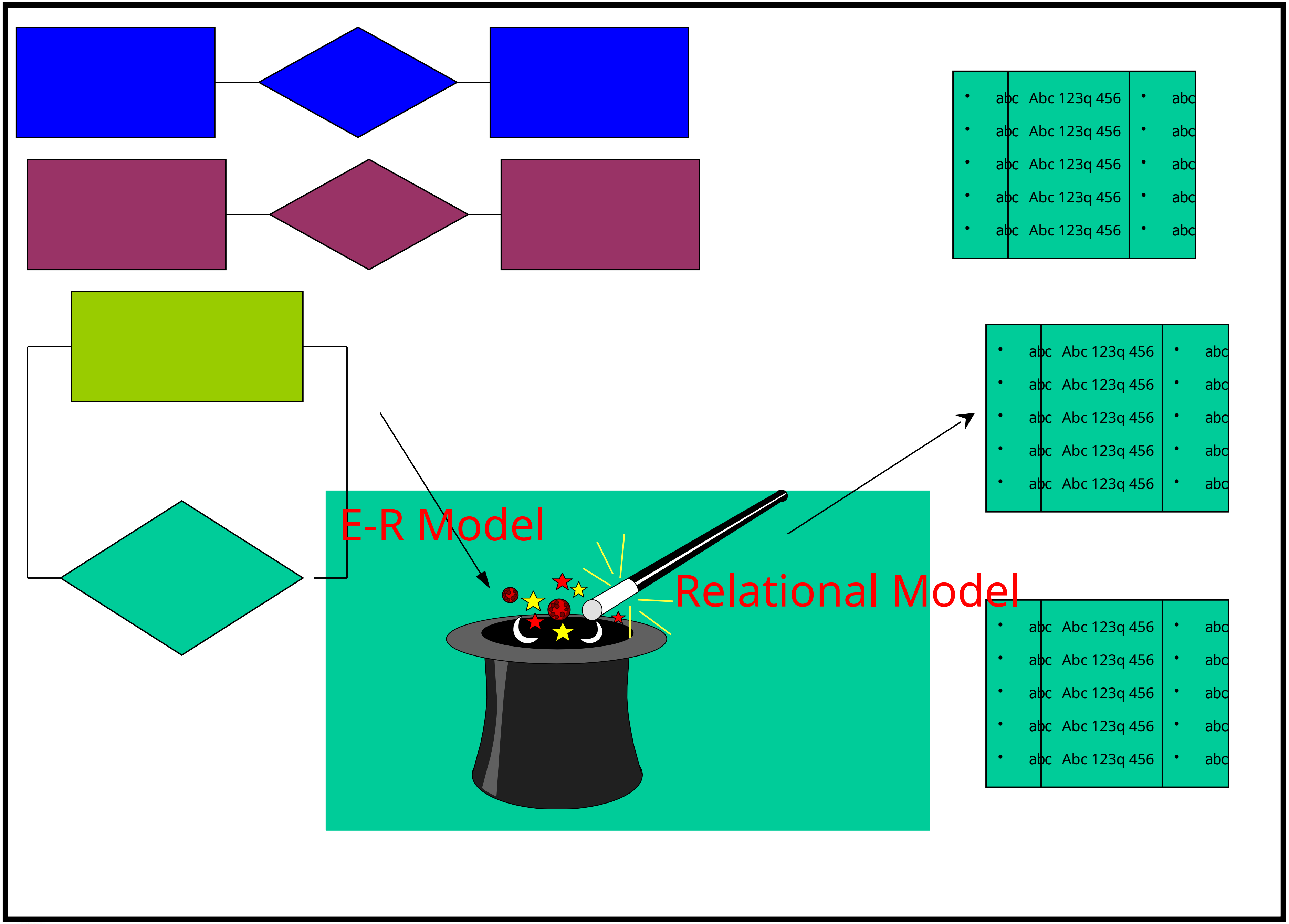
Relations

Requirements for a set of applications

E/R

Relational DBMS


Real World
E-R Model



E-R Model

Relational Model

•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc

•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc

•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc
•	abc	Abc 123q 456	•	abc

Relational Model

In this model, the data is organized in relations (tables)

Relational database schema: DDL component of SQL

Data Definition Language

set of table names

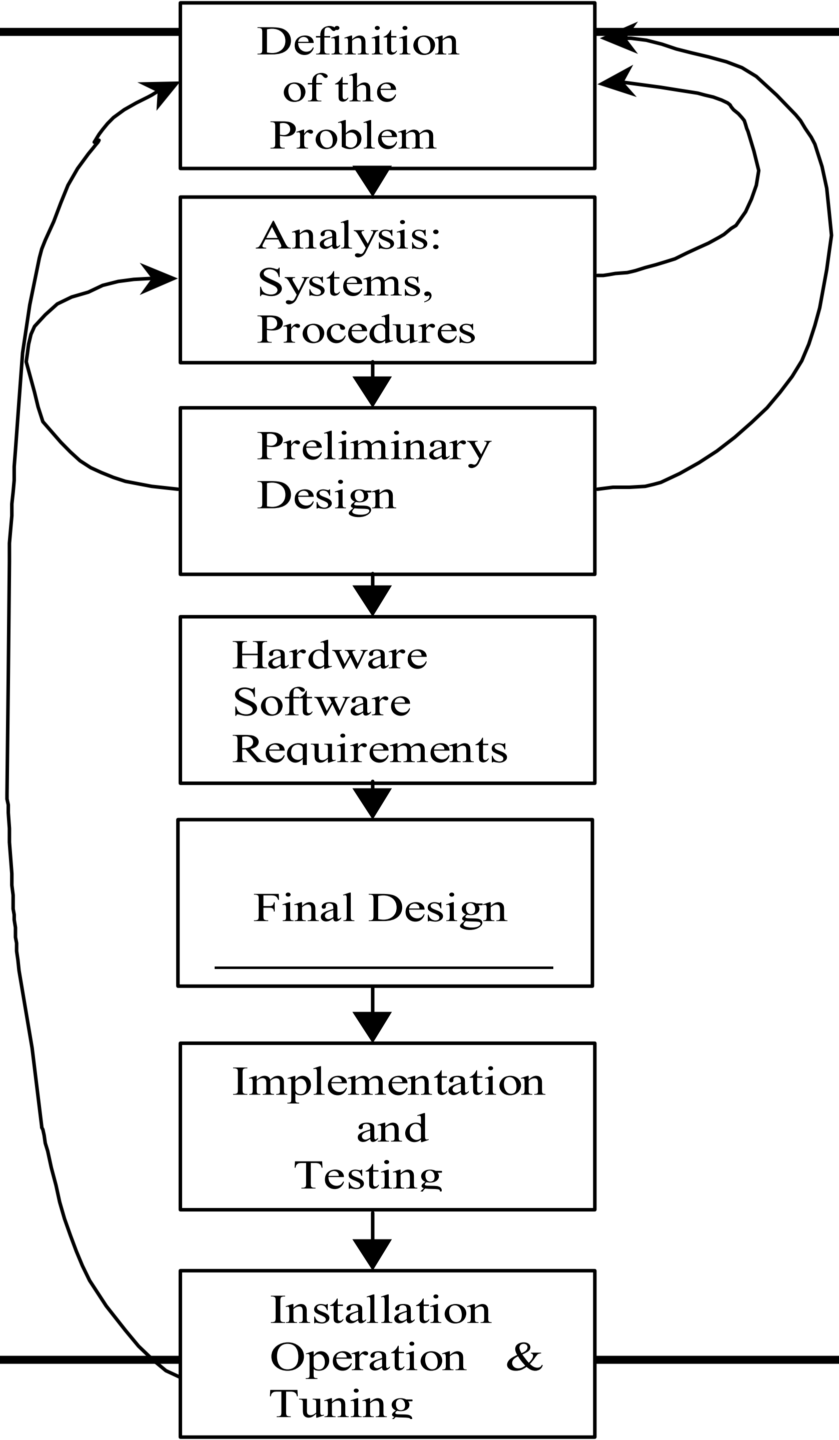
list of attributes for each table and their properties

Examples of tables from a university database:

Student : stud_number, name, address, program

Department: name, budget_code, room, phone

Course: name, number, credits



Definition of the Problem

Analysis: Systems, Procedures

Preliminary Design

Hardware Software Requirements

Final Design

Implementation and Testing

Installation Operation & Tuning

Database Design Process

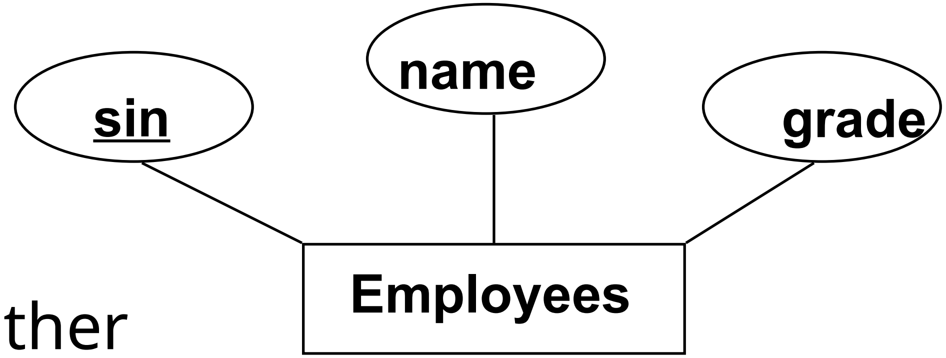
- *Definition of the problem*
- *Study underlying applications(Procedure Manuals, Interviews etc.)*
    - What are the *entities* and *relationships* involved?
    - What details about them should be in the database?
    - What are the *procedures, business rules, constraints*?
    - Who are the users? What do they need?
- *Preliminary Conceptual design*:
    - ER Model

Entity-Relationship Model

Database Design Process

- *Software/Hardware Requirements*
  - UML for software design Unified Modeling Language
- *Final Design: Schema Refinement: (Normalization)*
  - Check relational schema for redundancies and related anomalies.
  - External Schemas, indices, views, access methods
- *Application programs, forms, reports, user interfaces*
- *Implementation and testing*
- *Installation and Tuning:*
  - Data Distribution, Physical re-design
  - Performance, Security, Backup & Recovery.

ER Model



- *Entity*: Real-world object distinguishable from other objects.
  - An entity is described using a set of *attributes.*
- *Entity Set*: A collection of similar entities.
  - All entities in an entity set have the same set of attributes.
  - Each entity set has a *key.*
  - Each attribute has a *domain.*
  - Can map entity set to a relation easily.

```
CREATE TABLE Employees
    (sin CHAR(9),
    name CHAR(25),
    grade INTEGER,
    PRIMARY KEY  (sin))
```

```
mysql> CREATE TABLE Employees
              (sin CHAR(9),
       name CHAR(25),
       grade INTEGER,
       PRIMARY KEY  (sin));
Query OK, 0 rows affected (0.00 sec)
mysql> show tables;
+------------------+
| Tables_in_db11s  |
+------------------+
| Employees        |
+------------------+
mysql> desc Employees;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| sin   | char(9)  | NO   | PRI |         |       |
| name  | char(25) | YES  |     | NULL    |       |
| grade | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

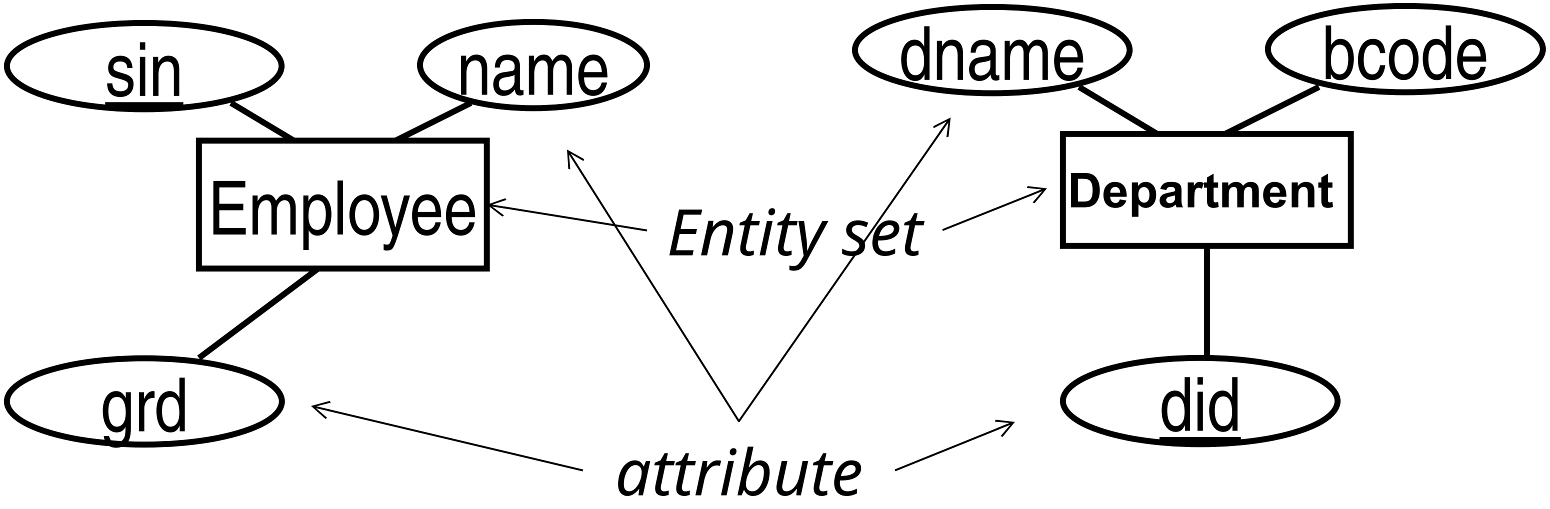
Note: size of integer is defaulted to 11

The Extra field contains any additional information that is available about a given column.
The value is auto_increment for columns that have the AUTO_INCREMENT attribute and empty otherwise.

```
CREATE TABLE Department
   (did mediumint not null  auto_increment,
    dname CHAR(16),
    bcode char(12),
    PRIMARY KEY  (did));
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> desc Department;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| did   | mediumint(9) | NO   | PRI | NULL    | auto_increment |
| dname | char(16)     | YES  |     | NULL    |                |
| bcode | char(12)     | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

Entities and entity sets



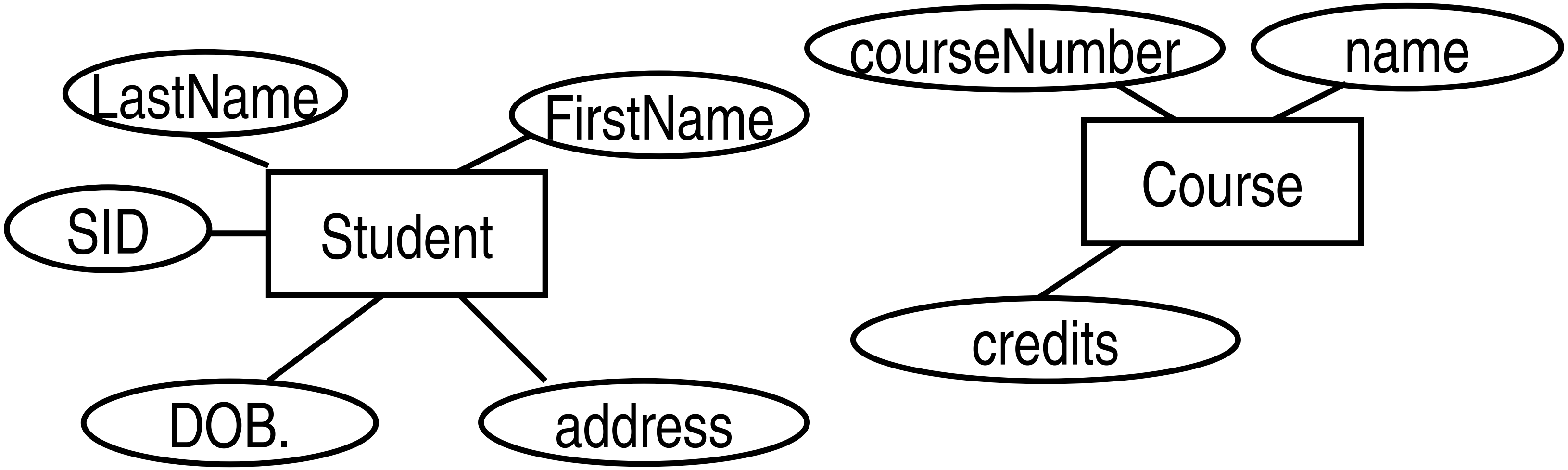
All employees, and departments have the same set of properties(attributes)
To distinguish one instance of an entity in an entity set from others, we introduce an identifying attribute
This is the primary key and it is underlined

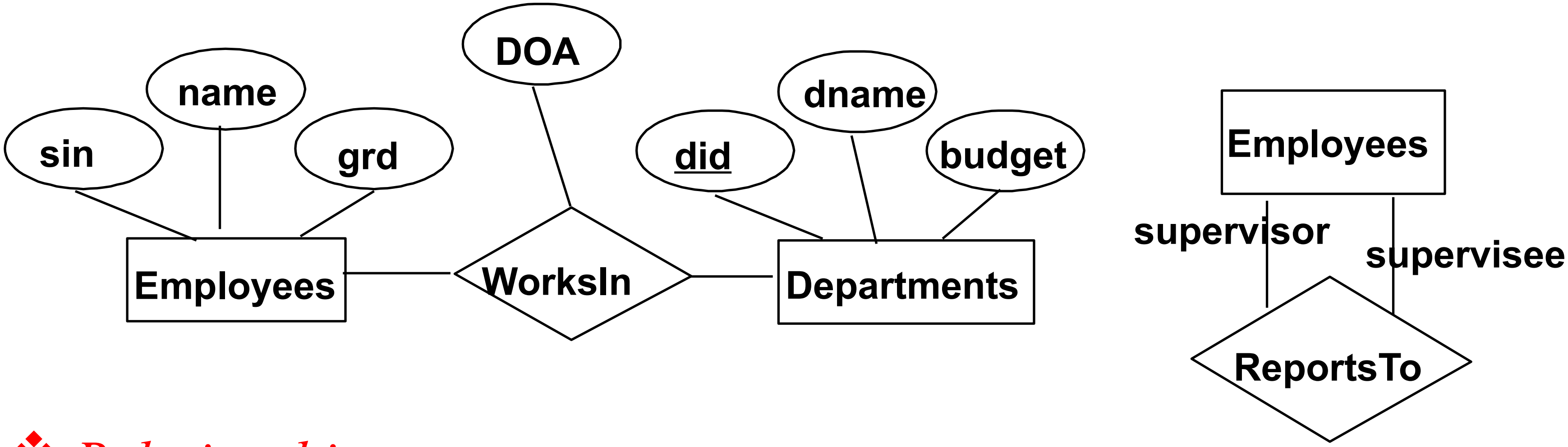
Entity – Real world object distinguishable from other objects of the same type

Entity Set -- A collection of similar entities: all have same set of properties

ODL:

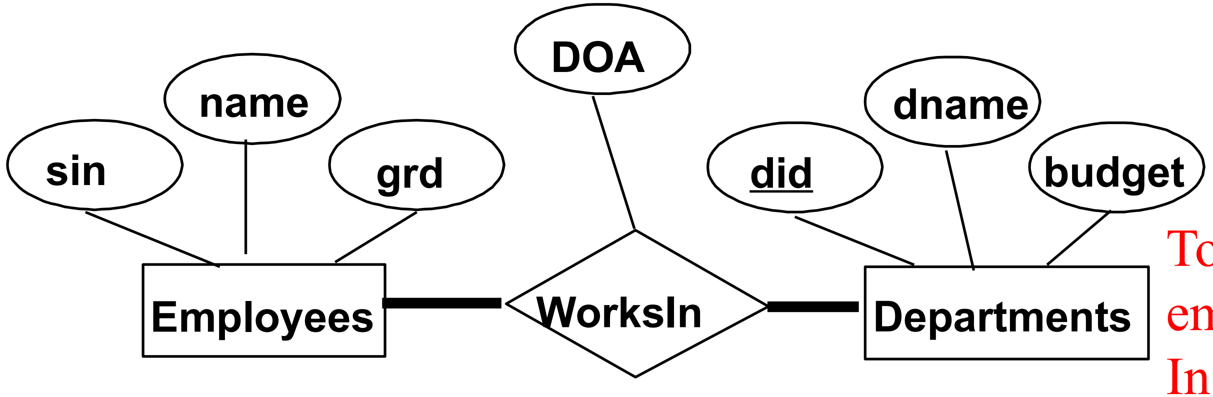
Object corresponds to entity **Class** corresponds to entity set



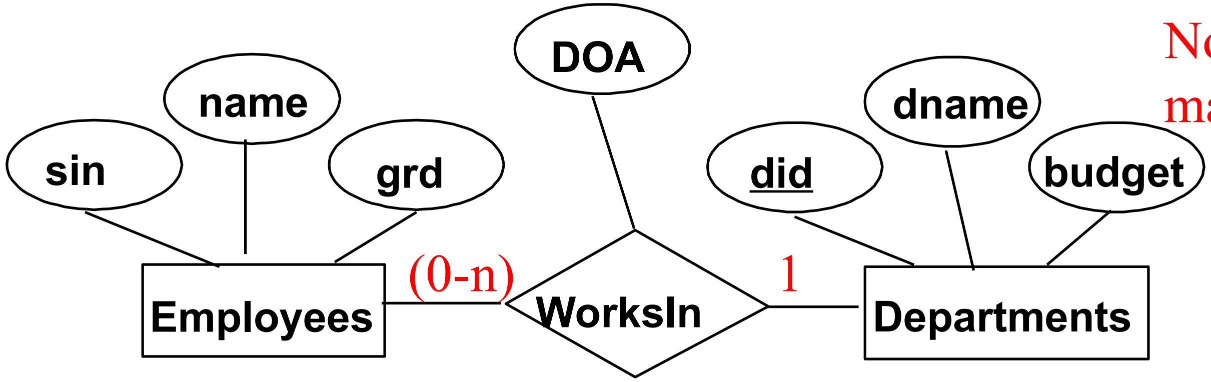


- *Relationship*:
  - Association among 2 or more entities.
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R expresses an association among n entity sets E1 ... En; each relationship in R involves entities e1 ∈ E1, ..., en ∈ En

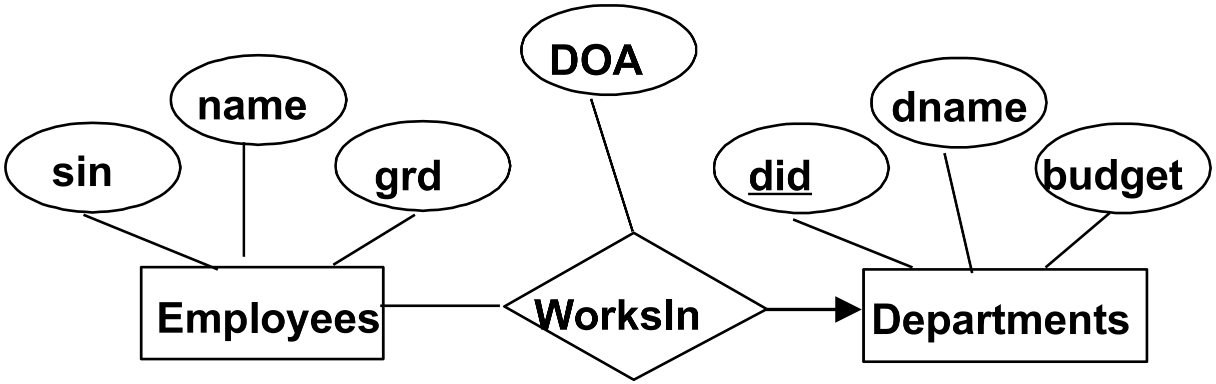
Same entity set could participate in different relationship sets, or in different "roles" in same set.



Total participation of all employees & departments In the WorksIn relationship
No Employee or Department may exist without being related

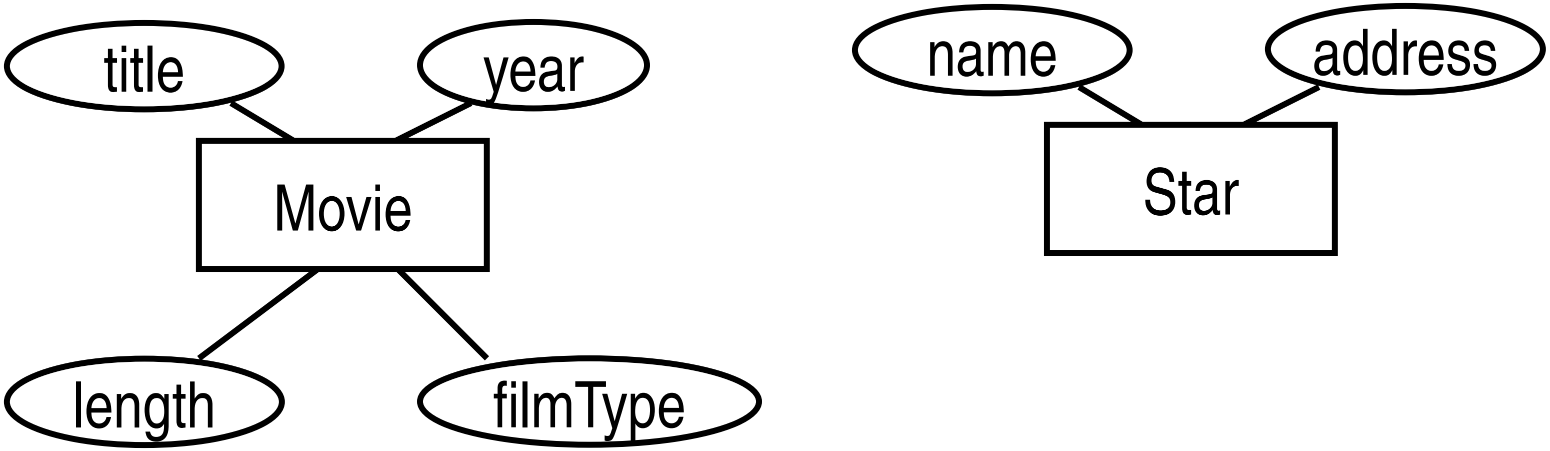


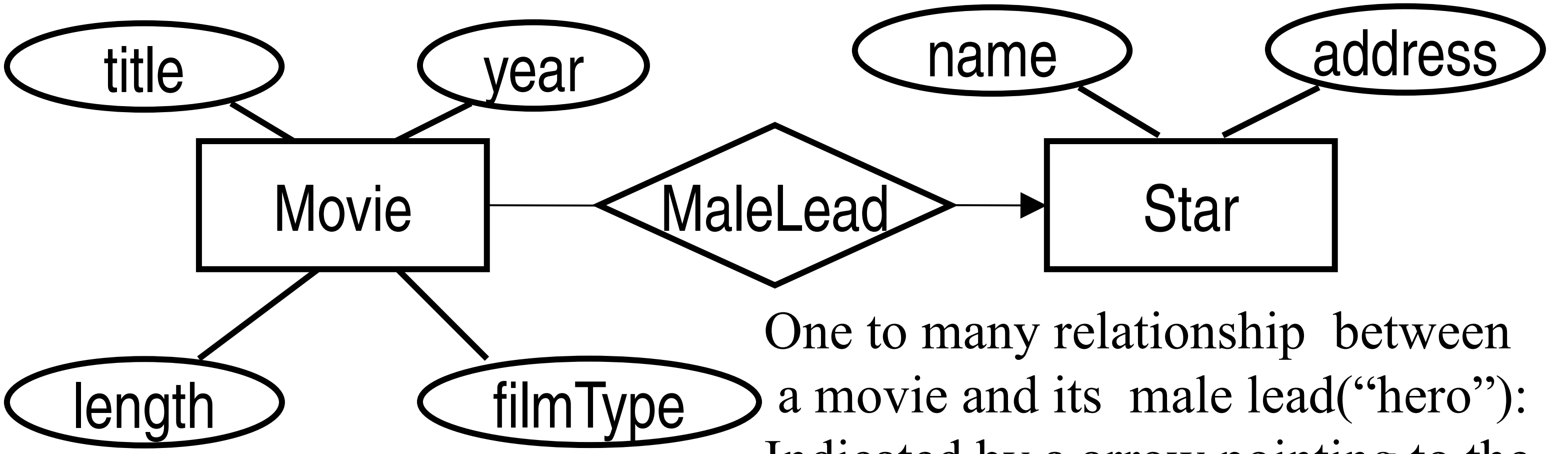
Many to one relationship: many employees in a department;
but an employee is assigned to only one department



Alternate way of showing a many-to-one relationship

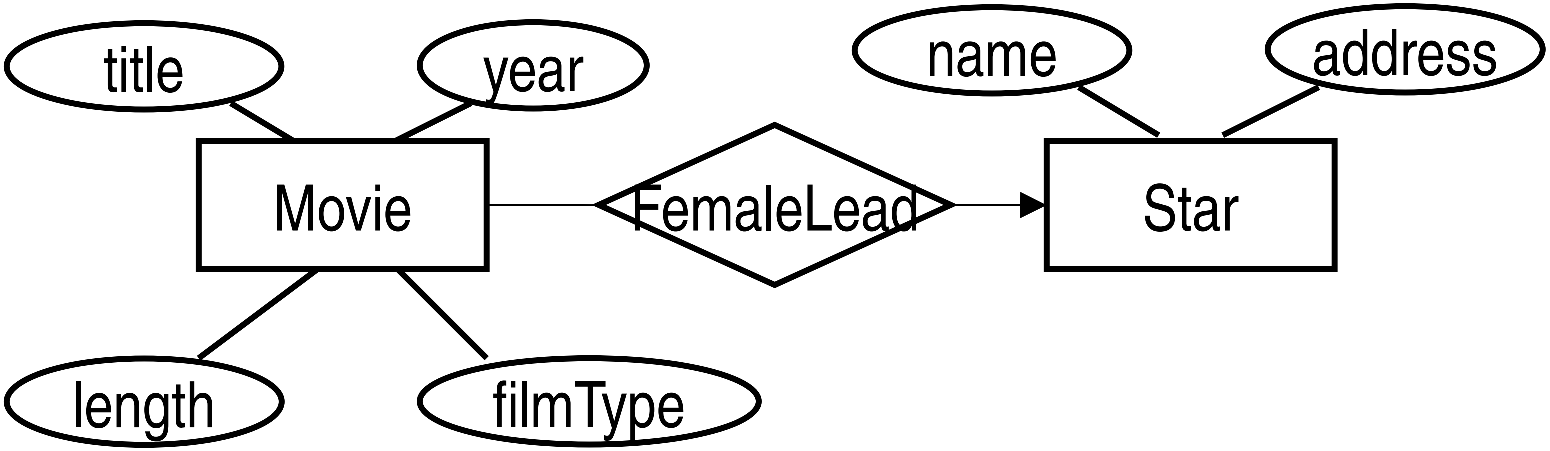
Entities and entity sets

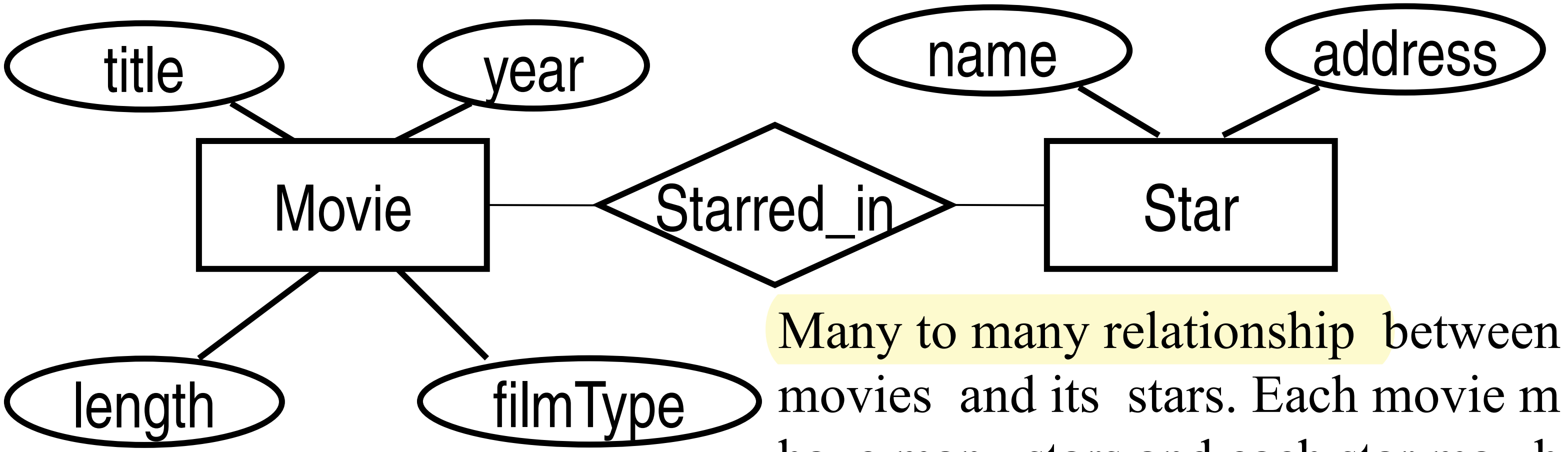




One to many relationship between a movie and its male lead("hero"): Indicated by a arrow pointing to the "one side" – **A movie has but one main role, The star may be a lead in many movies**

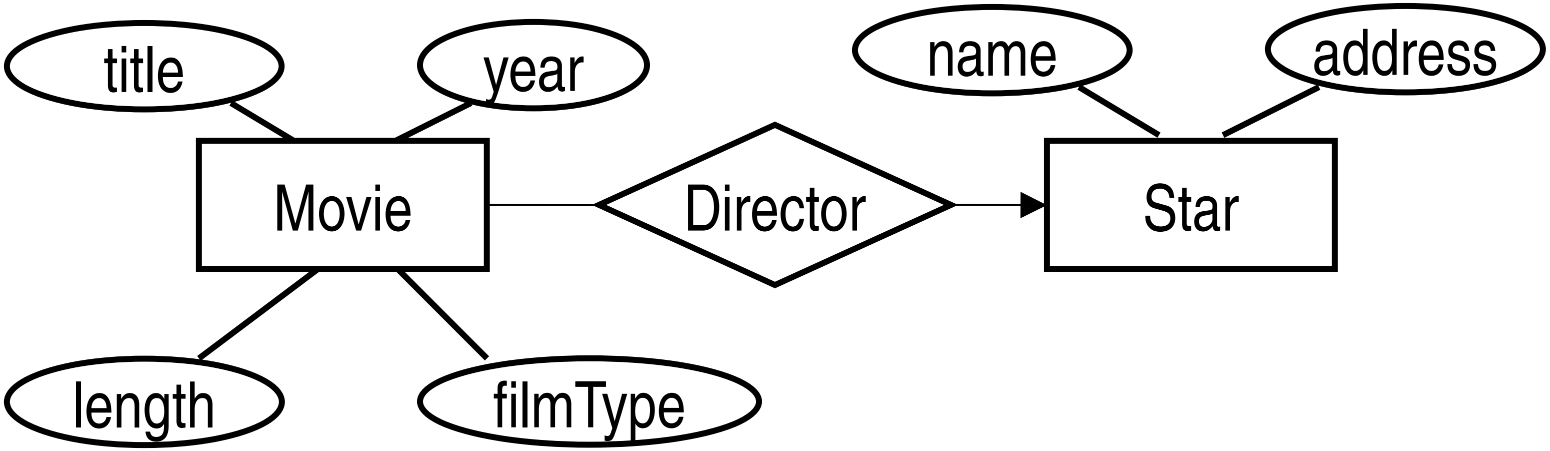
Entities and entity sets





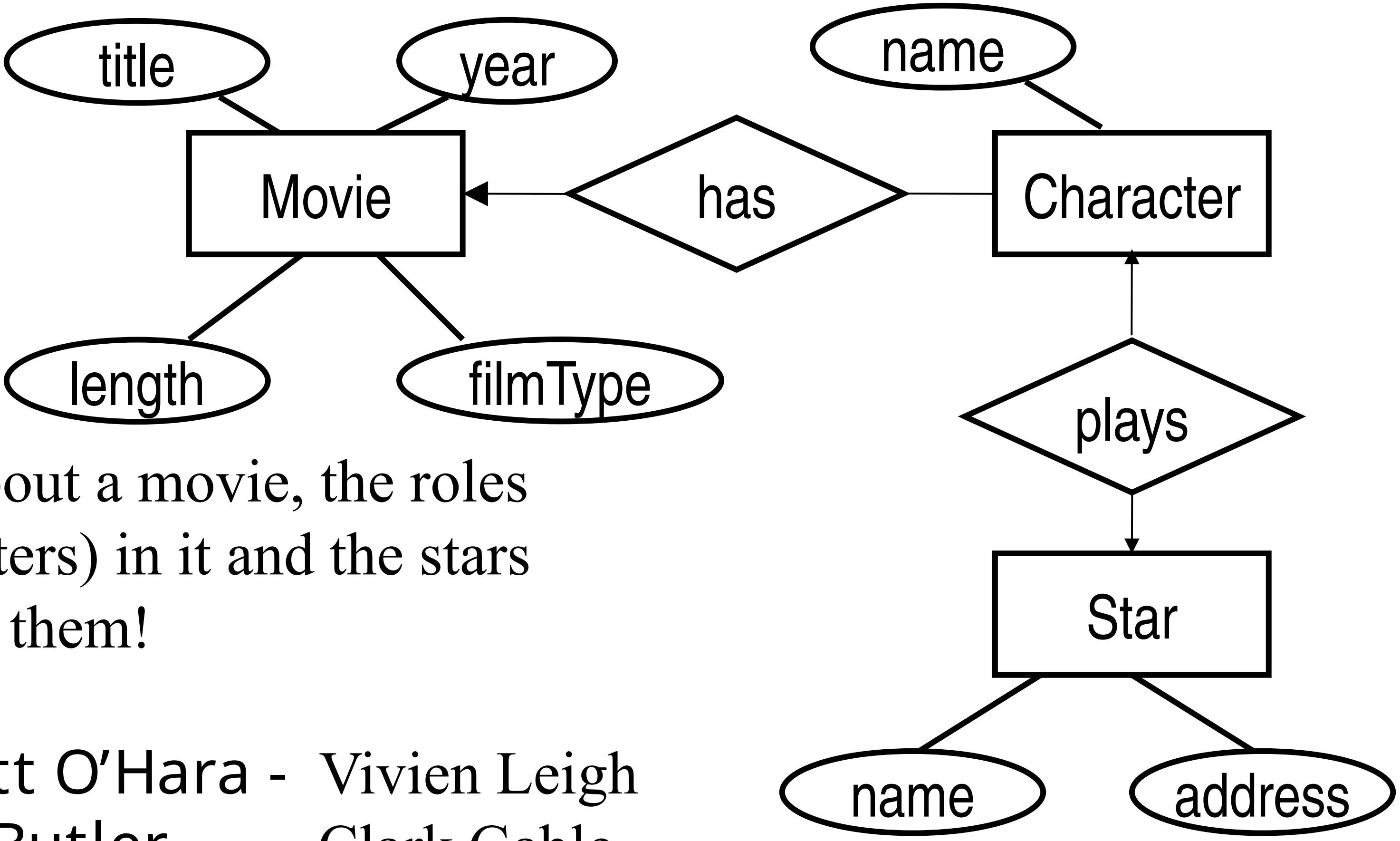
Many to many relationship between movies and its stars. Each movie may have many stars and each star may have featured in many movies. Indicated by no arrows on the connecting lines.

Entities and entity sets



How about a movie and the roles(characters) in it and the stars playing them!

Entities and entity sets



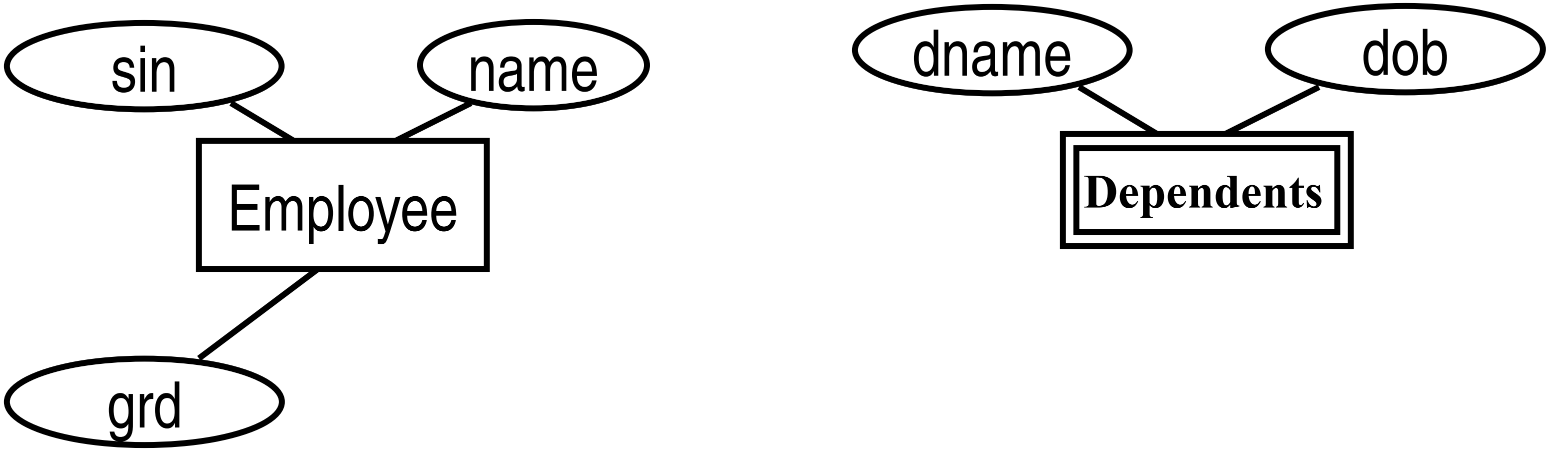
How about a movie, the roles (characters) in it and the stars playing them!

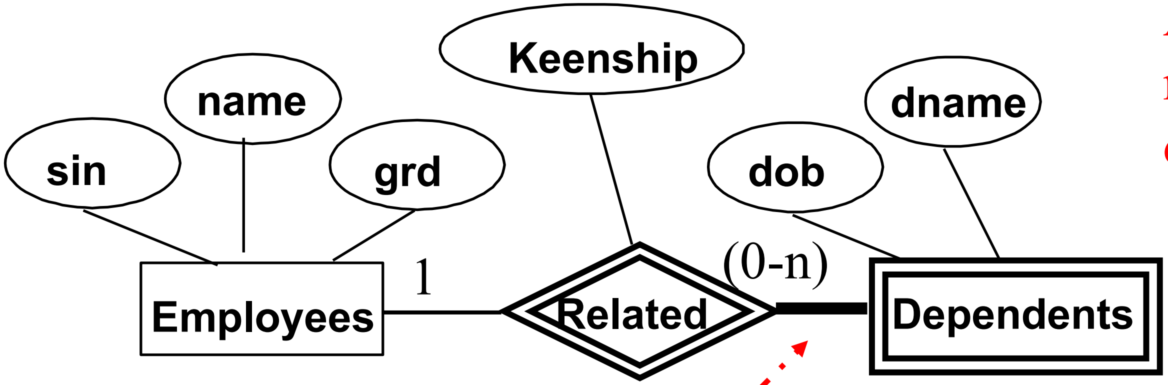
Scarlett O'Hara - Vivien Leigh
Rhett Butler - - Clark Gable

Alex Guinness plays eight members of the D'Ascoyne family in Kind hearts and coronets(1949)

Matt Damon played the lead in the *Bourne* triology.

Entities and entity sets





An employee may have 0 to n dependents

Total participation

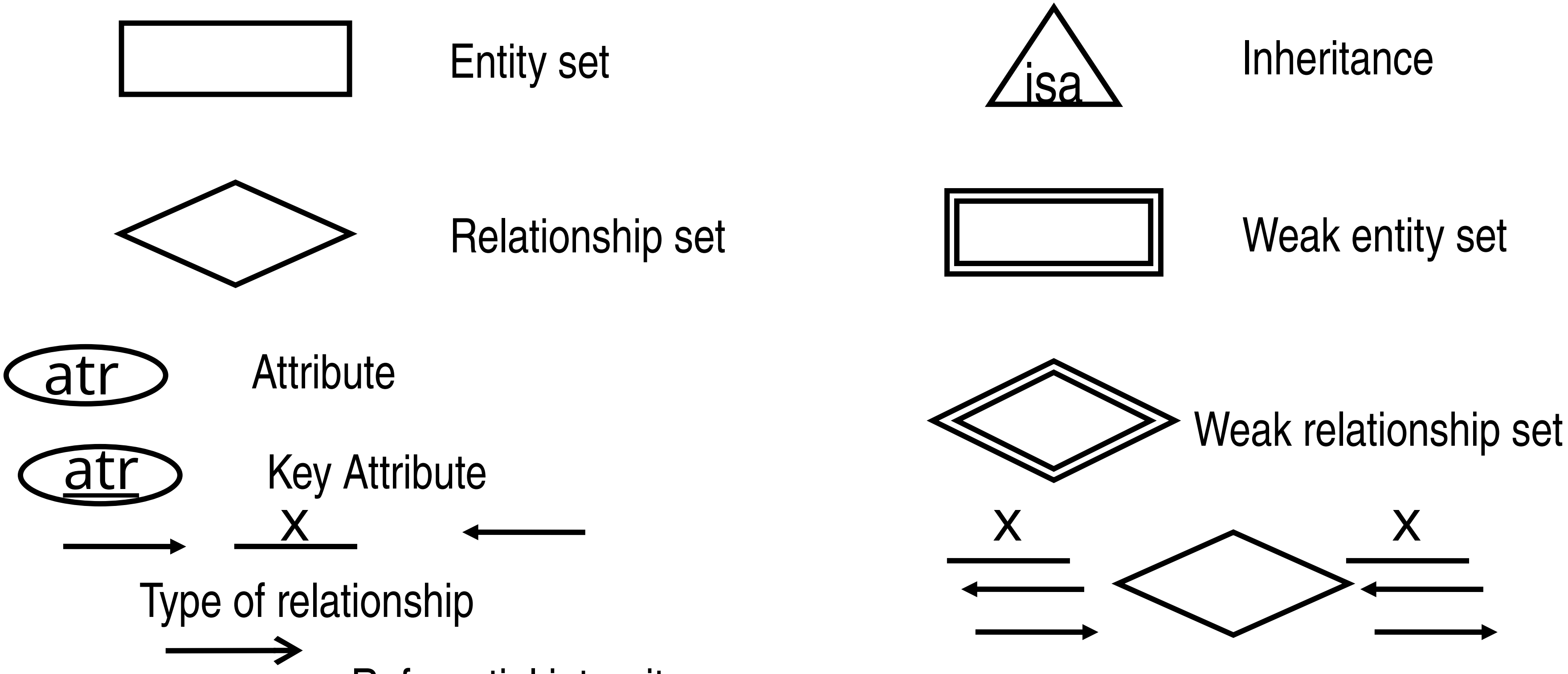
All dependents must be related to some employee(but only one!)

E/R model is a *graphical* approach to database modeling

E/R is widely used in database design

E/R model grew out of modeling application database

No standard for E/R diagrams: a number of variations



Entity set

Inheritance

Relationship set

Weak entity set

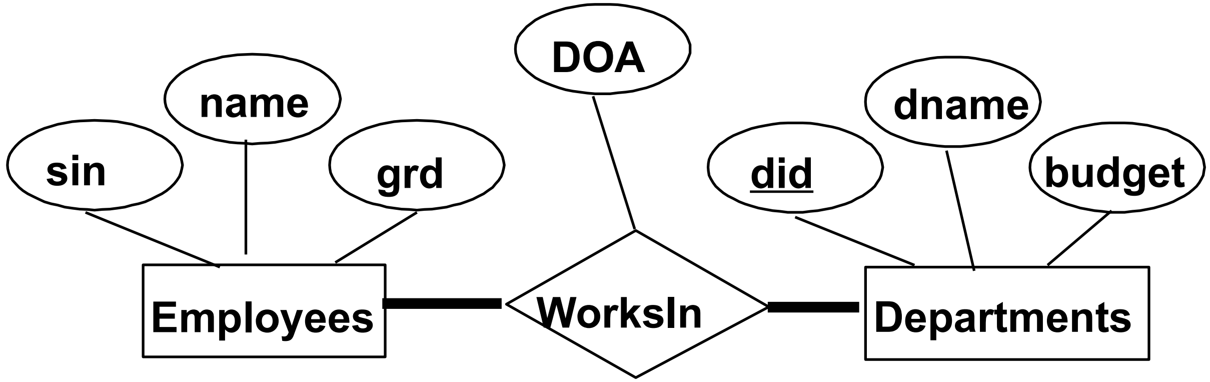
Attribute

Weak relationship set

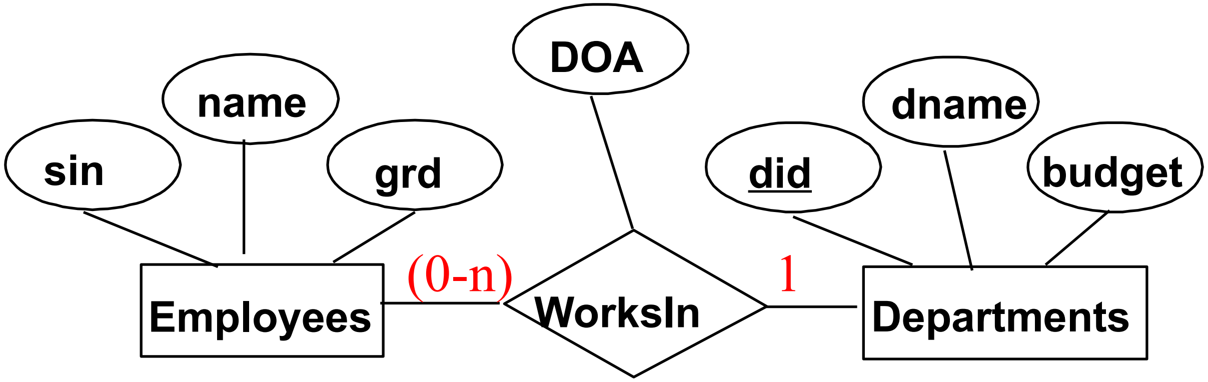
Key Attribute

Type of relationship

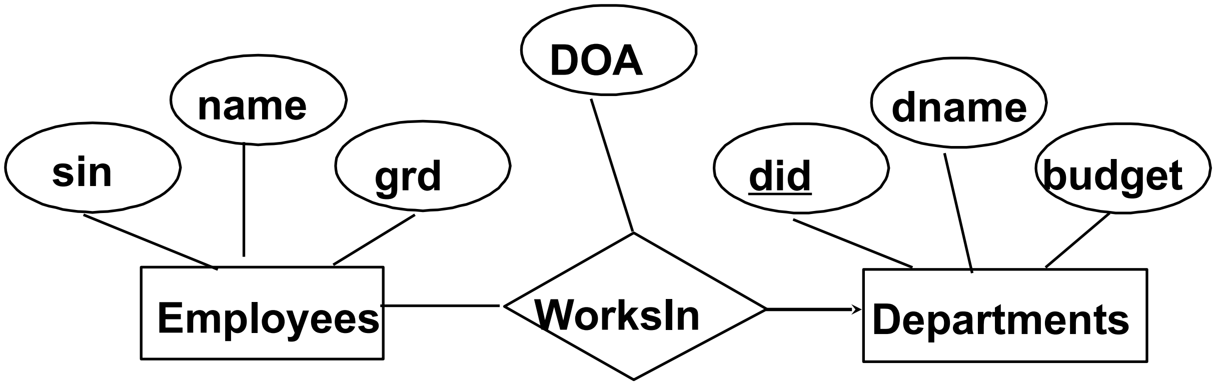
Referential integrity



Total participation of all employees & departments In the WorksIn relationship

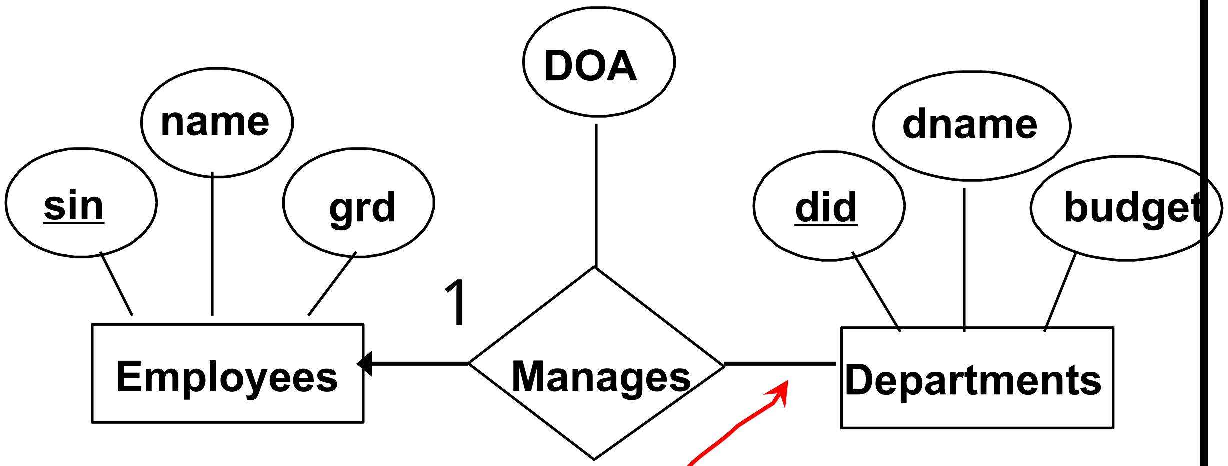


Many to one relationship: many employees in a department; but an employee is assigned to only one department



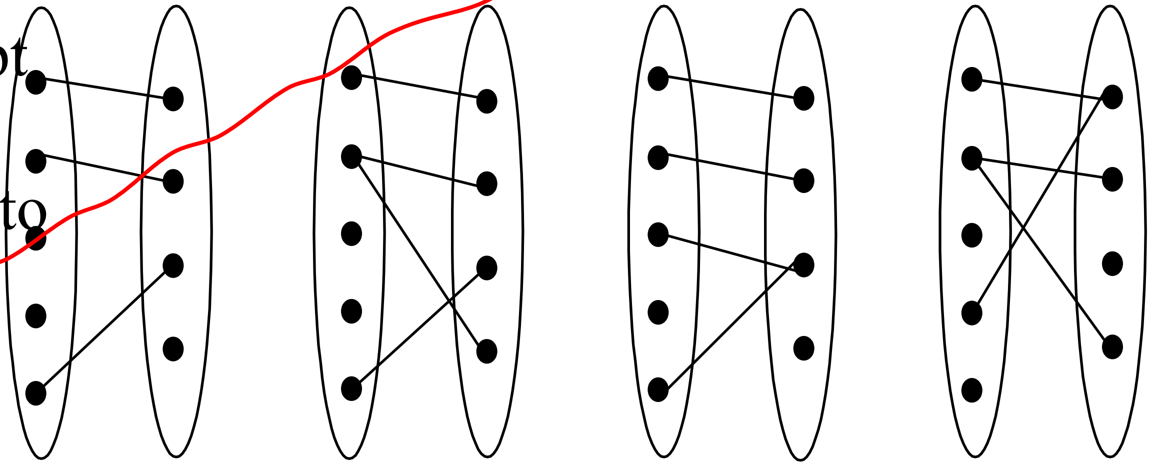
Alternate way of showing a many-to-one relationship

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.

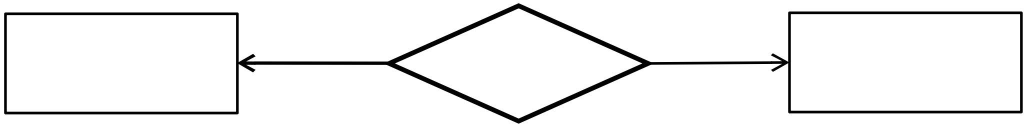
A department can have only one manager,
an employee could manage many departments.



- Relationship sets can have *attributes*
- 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 superkey for the relation.
  - All descriptive attributes.

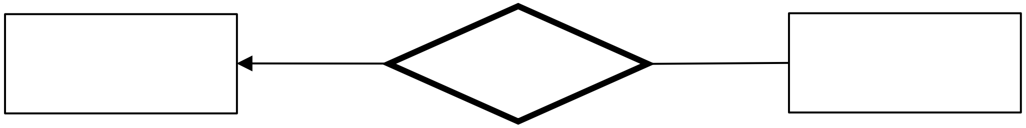
```
CREATE TABLE WorksIn
  (sin  CHAR(9),
  did  INTEGER,
  DOA  DATE,
  PRIMARY KEY (sin, did),
  FOREIGN KEY (sin)
        REFERENCES Employees,
  FOREIGN KEY (did)
        REFERENCES Departments)
```

If a relationship is
1-to-1 primary key is from either of t
the other side is a foreign k



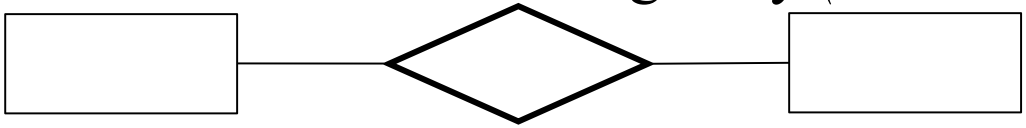
If a binary relationship between two entity sets is 1-to-1,

- the primary key of the relationship is the key of the entity from either of the '1' side, the other side is a foreign key*(would be unique)*



If a binary relationship between two entity sets is 1-to-many,

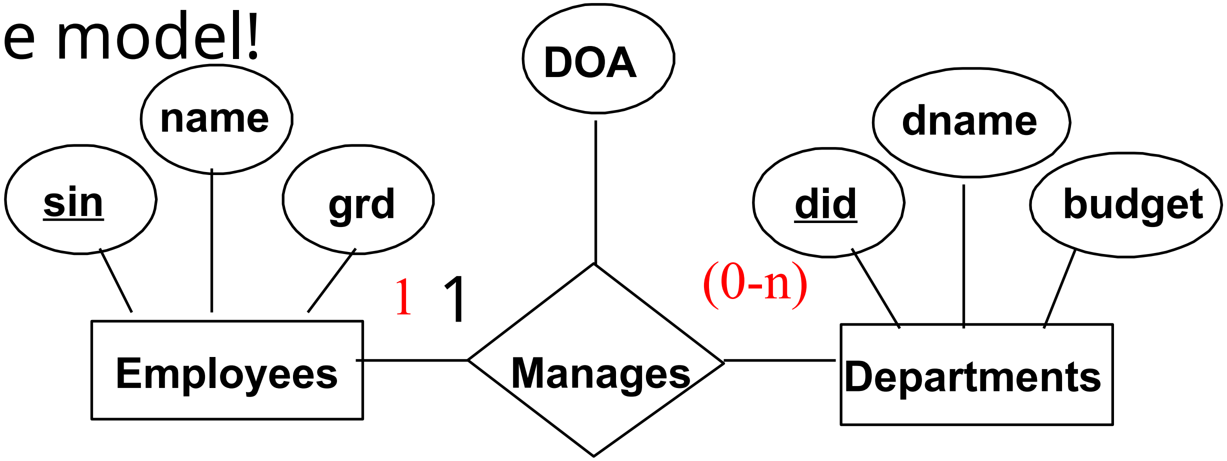
- the primary key of the relationship is from the 'm' side, the '1' side is the foreign key *(would be unique)*

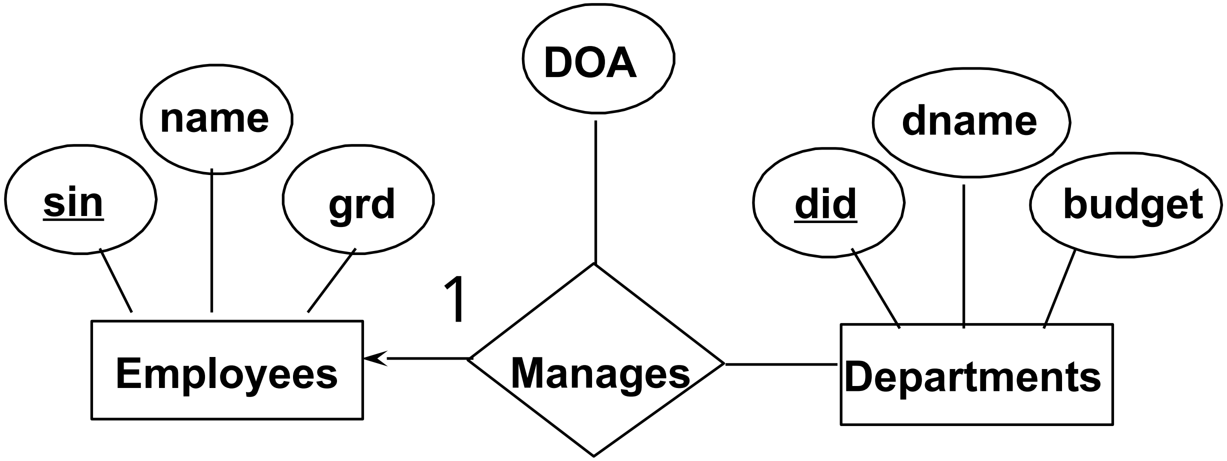


If a binary relationship between two entity sets is m-to-n,

- the primary key is composite, consisting of the primary key of the entities from each side of the relationship

Alternate methods of showing the same model!





- Map relationship to a table:
  - Note that did is the key now!
  - Separate tables for Employees and Departments.
- Since each department has a unique manager, we could instead combine Manages and Departments.

```
CREATE TABLE  Manages
 ( sin  CHAR(9),
  did  INTEGER,
  DOA DATE,
  PRIMARY KEY  (did),
  FOREIGN KEY (sin) REFERENCES Employees,
  FOREIGN KEY (did) REFERENCES Departments)
```

```
CREATE TABLE  DeptMgr
  (did  INTEGER,
  dname  CHAR(20),
  budget  REAL,
  sin  CHAR(9),
  DOA  DATE,
  PRIMARY KEY  (did),
  FOREIGN KEY (sin) REFERENCES Employees)
```

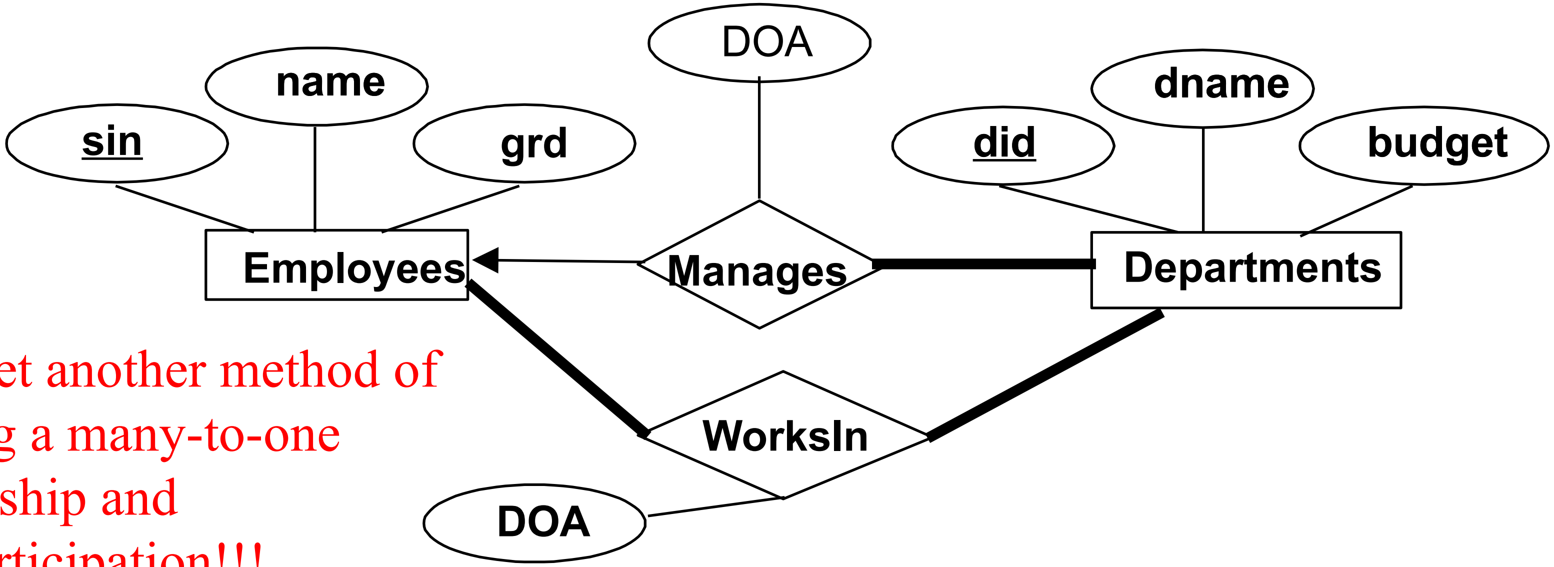
Null for Dept. w/o manager!

Participation Constraints

- Every department has a manager (a business rule) ⇒ *participation constraint*:
  - The participation of Departments in Manages is *total*

(all instances of Department must have a manager; participation of Employees is *partial i.e., not all employees are managers*).

Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *sin* value!)



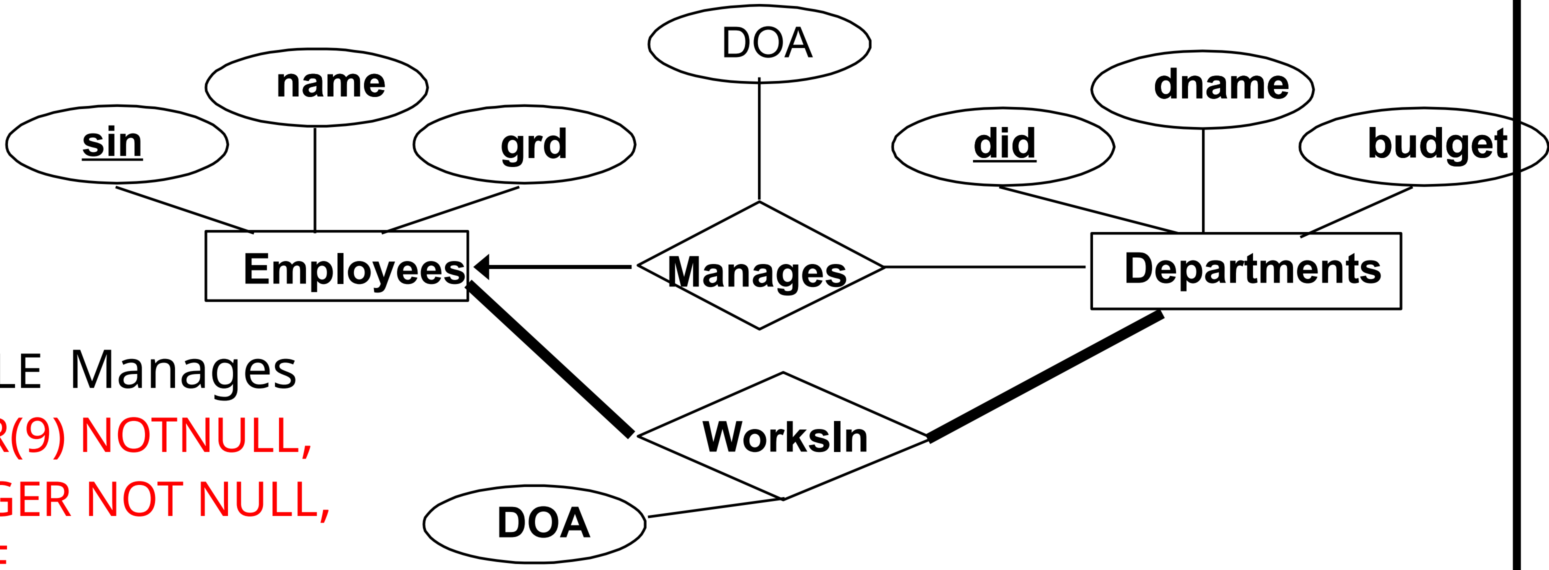
Note: yet another method of showing a many-to-one relationship and total participation!!!

Participation Constraints: SQL

- A participation constraints involving one entity set in a binary relationship, can be expressed as follows without resorting to CHECK constraints.

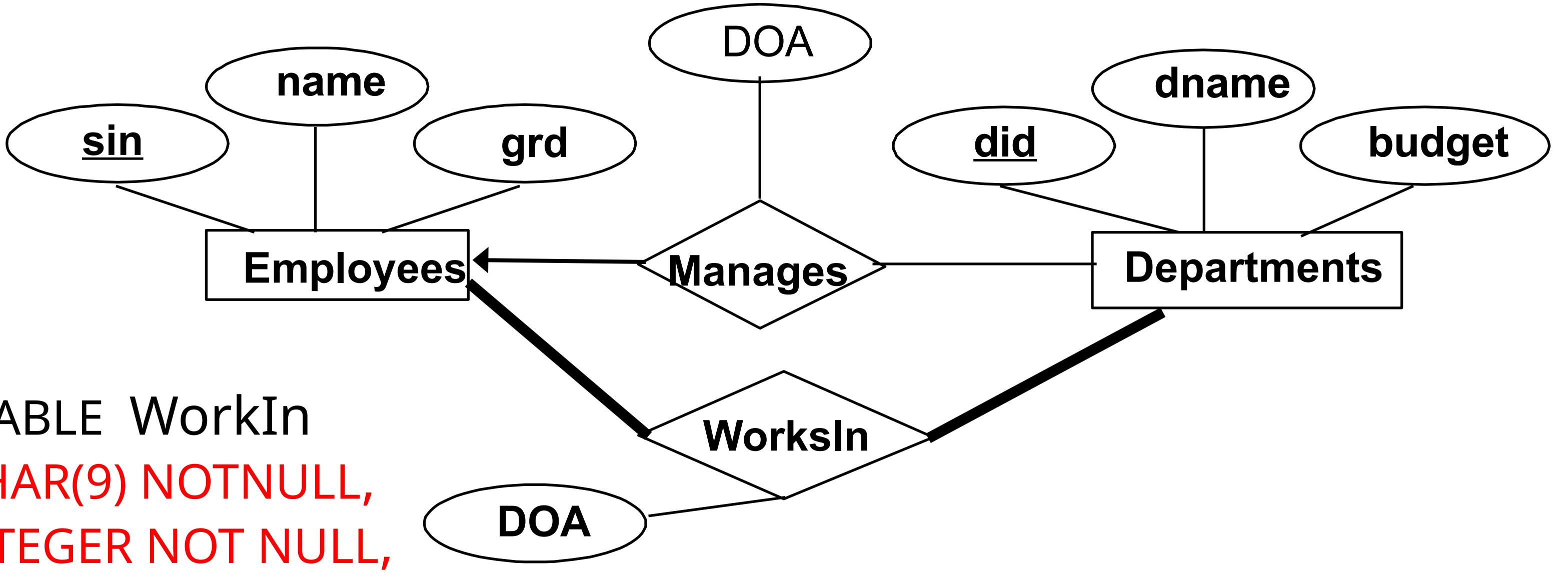
```
CREATE TABLE  DeptMgr
   (did  INTEGER,
   dname  CHAR(20),
   budget  REAL,
   sin  CHAR(9) NOT NULL,
   DOA  DATE,
   PRIMARY KEY  (did),
   FOREIGN KEY  (sin) REFERENCES Employees,
      ON DELETE NO ACTION)
```

Every department must have a manager!



```
CREATE TABLE  Manages
  ( sin  CHAR(9) NOTNULL,
   did  INTEGER NOT NULL,
   DOA DATE,
   PRIMARY KEY  (did),
   FOREIGN KEY (sin) REFERENCES Employees,
   FOREIGN KEY (did) REFERENCES Departments)
```

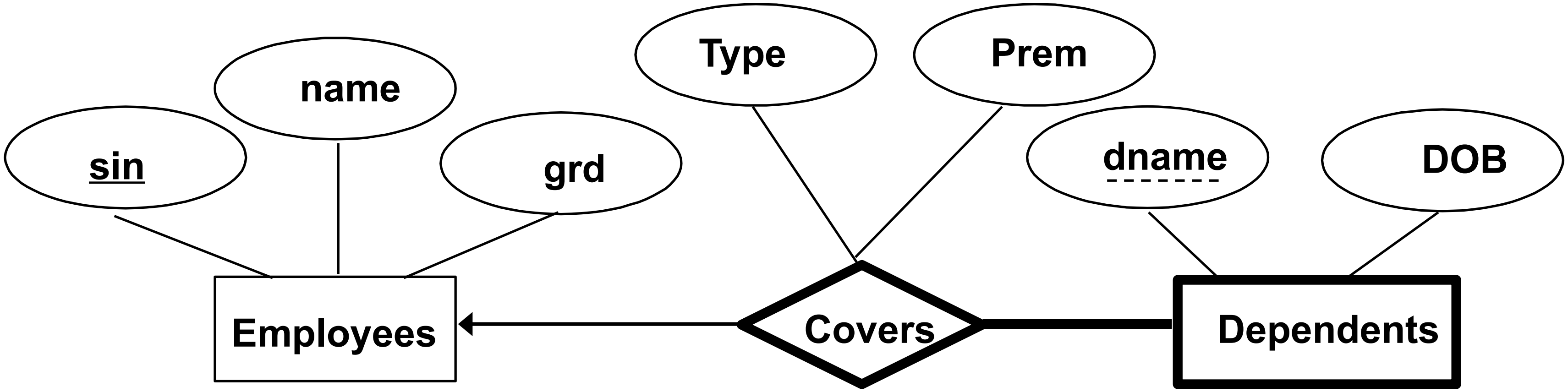
Here a department can exist without a manager.
We couldn’t insert an occurrence of this relation
without having an occurrence of a department
and employee! Once inserted, does firing the manager
create problems?



```
CREATE TABLE  WorkIn
  ( sin  CHAR(9) NOTNULL,
   did  INTEGER NOT NULL,
   DOA DATE)
```

To ensure total participation of department in WorkIn, each did value must be in at least one tuple of WorkIn: enforced by assertion

- A *weak entity* can be identified uniquely only by considering the primary key of another *strong*-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.

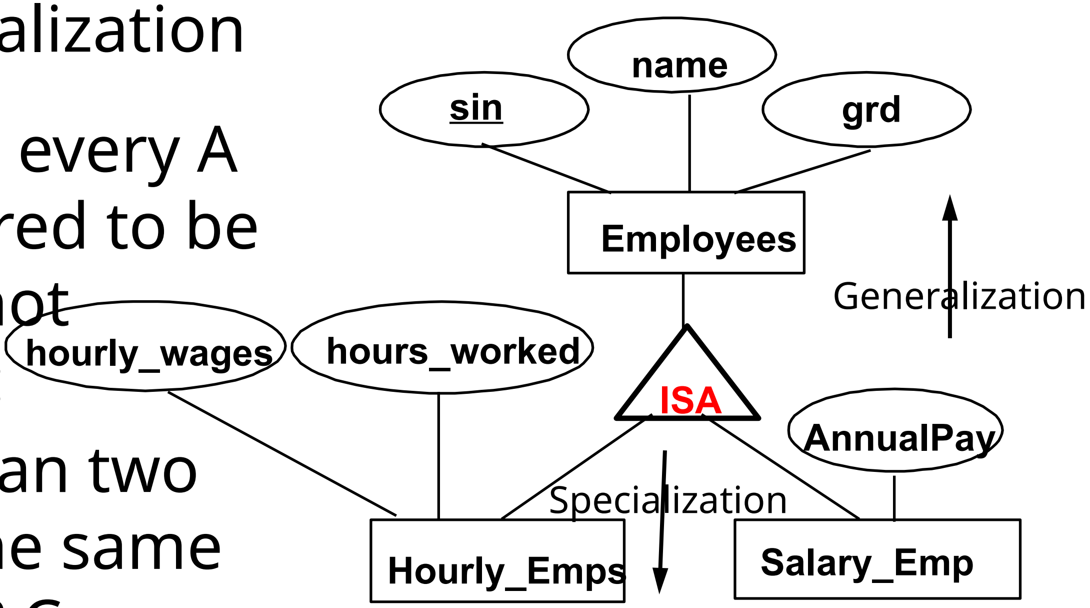


- Weak entity set and identifying relationship set are translated into a single relation.
  - Weak entity ⇒ total participation
  - When the owner entity is deleted, all owned weak entities must also be deleted.

```
CREATE TABLE  Covers
 ( dname  CHAR(20),
   DOB  DATE,
   Type INTEGER,
   Cost FLOAT,
   sin  CHAR(9) NOT NULL,
   PRIMARY KEY  (dname, sin),
   FOREIGN KEY  (sin) REFERENCES Employees,
      ON DELETE CASCADE)
```

Generalization, Specialization

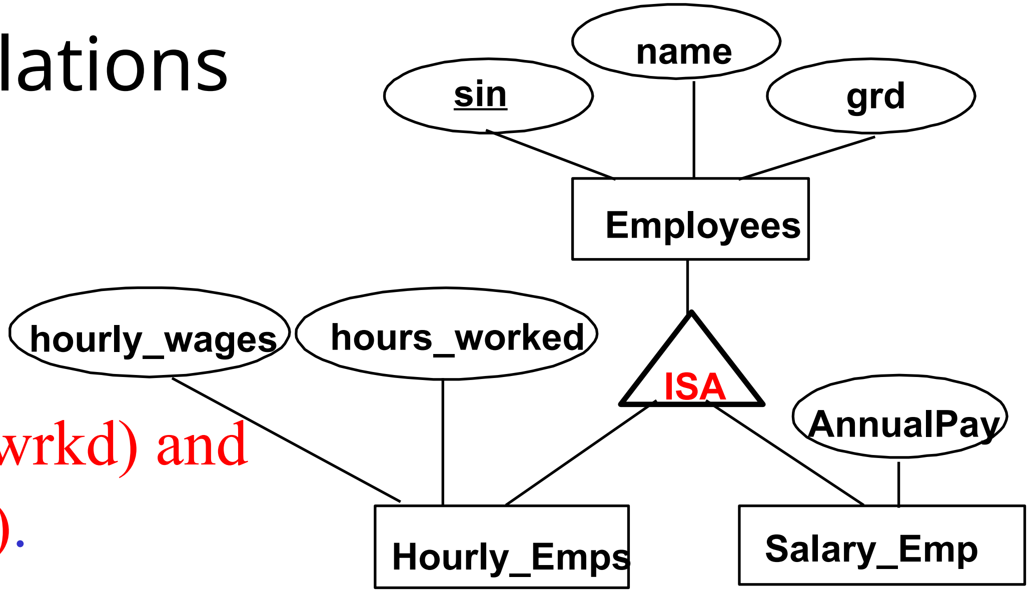
- If we declare A ISA B, every A entity is also considered to be a B entity. However, not implemented always:
- *Overlap constraints*: Can two subclasses contain the same instance of an entity? Can Carole be an Hourly_Emps as well as a Salary_Emps? *(Allowed/disallowed)*



Covering constraints: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/no)*

- Reasons for using ISA relationship:
  - To add attributes specific to a subclass.
  - To identify subset of an entity set that participate in a relationship.

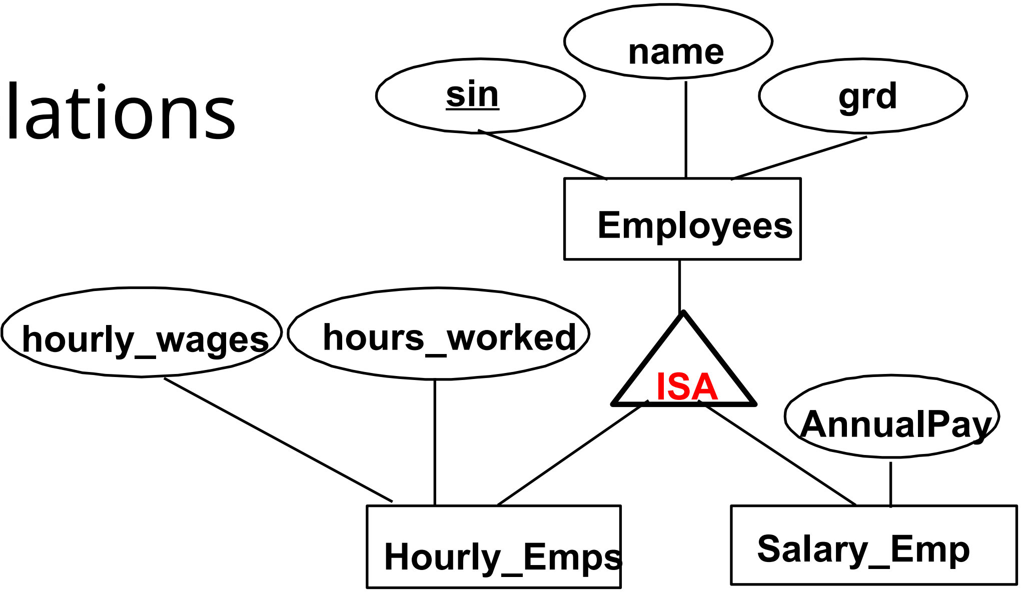
ISA relationship to Relations



Create 3 relations:
Employees(Sin, Name, Grd),
Hourly_Emps(Sin, Hwage, Hwrkd) and
Salary_Emps(Sin, AnnualPay).

- Every employee is recorded in Employees. For hourly employees, value for additional attribute are recorded in Hourly_Emps; if referenced Employees tuple is deleted, Hourly_Emps tuple must also be deleted.
- Queries involving all employees easy, those involving just Hourly_Emps require a join with Employee to get inherited attributes.

ISA relationship to Relations



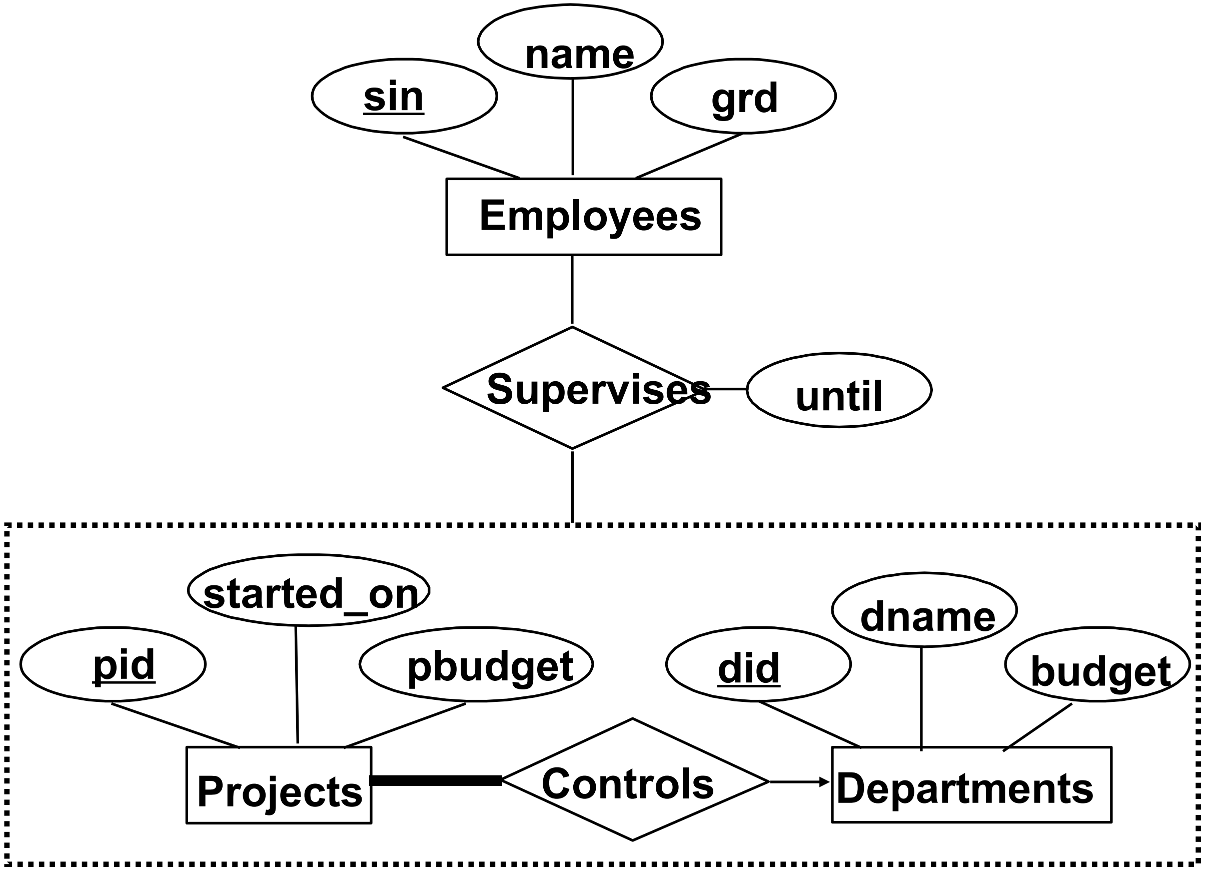
- Create two relations:
- Hourly_Emps(Sin, Name, Grd, HWrkd, Hwages) and Salary_Emps (Sin, Name, Grd, AnnualPay).

Each employee must be in one of these two subclasses.

All employees require accessing Two relations

Aggregation

- *Aggregation*: models a relationship, involving entity sets and a *relationship* set, as an entity set. The aggregated entity participates in other relationships.
  - Supervise mapped to table like any other relationship set.

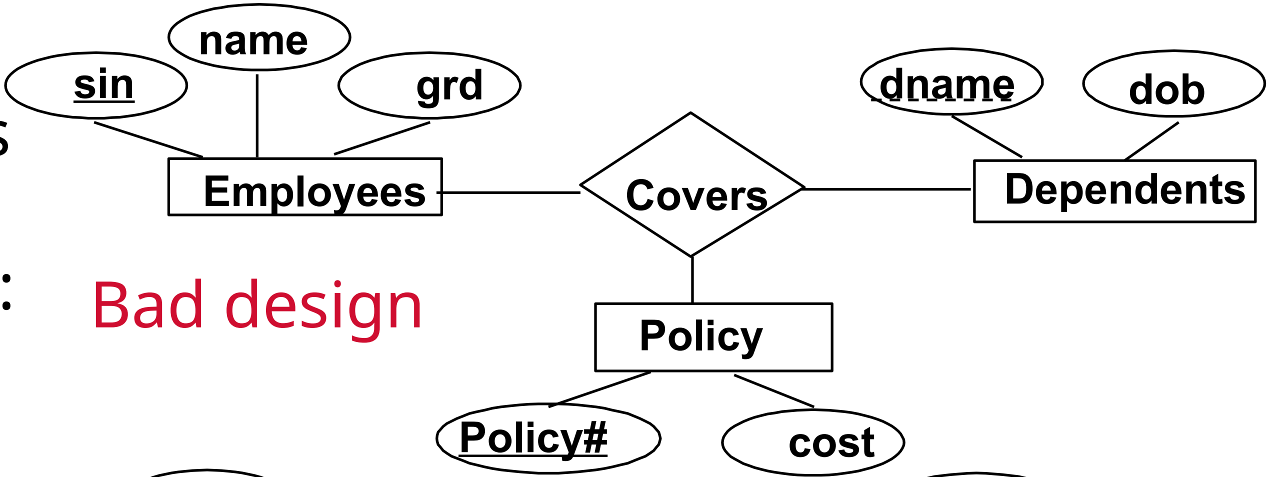


➽ *Aggregation vs. ternary relationship*:

- Supervises is a distinct relationship, with its attribute.

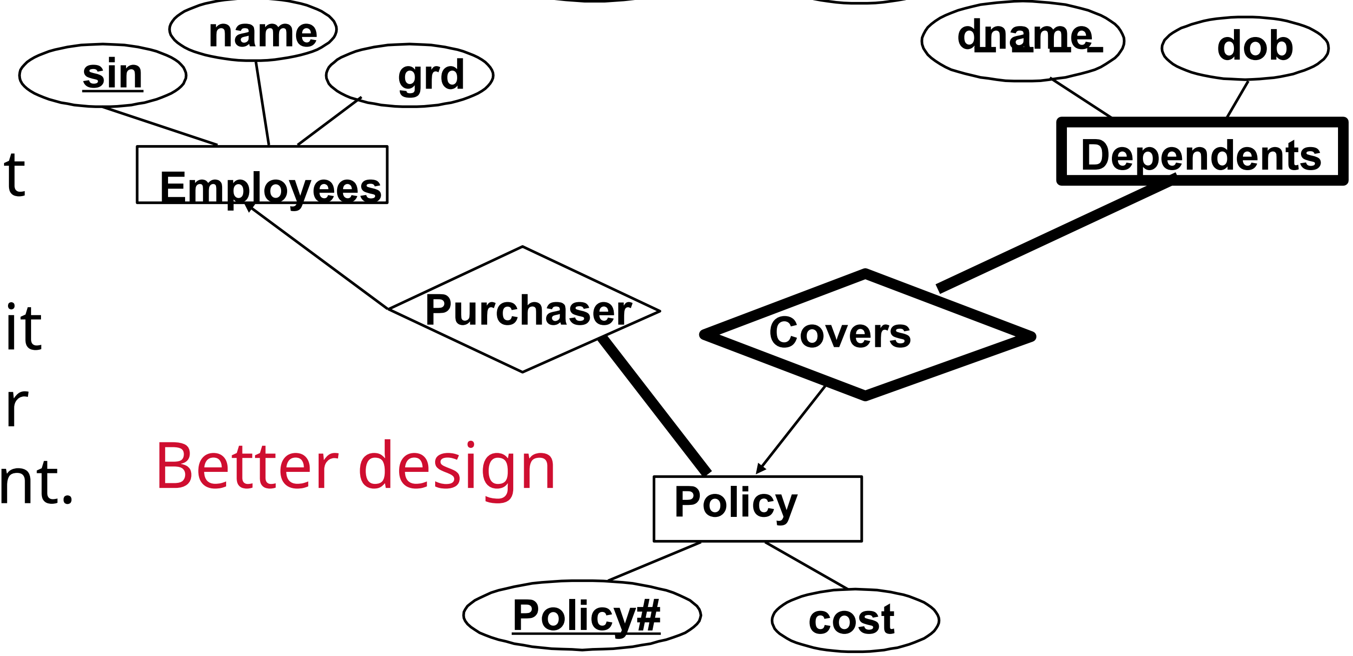
Binary vs. Ternary Relationships

- If each policy is owned by just ONE employee:



Bad design

- Key constraint on Policy requires that it can only cover one dependent.



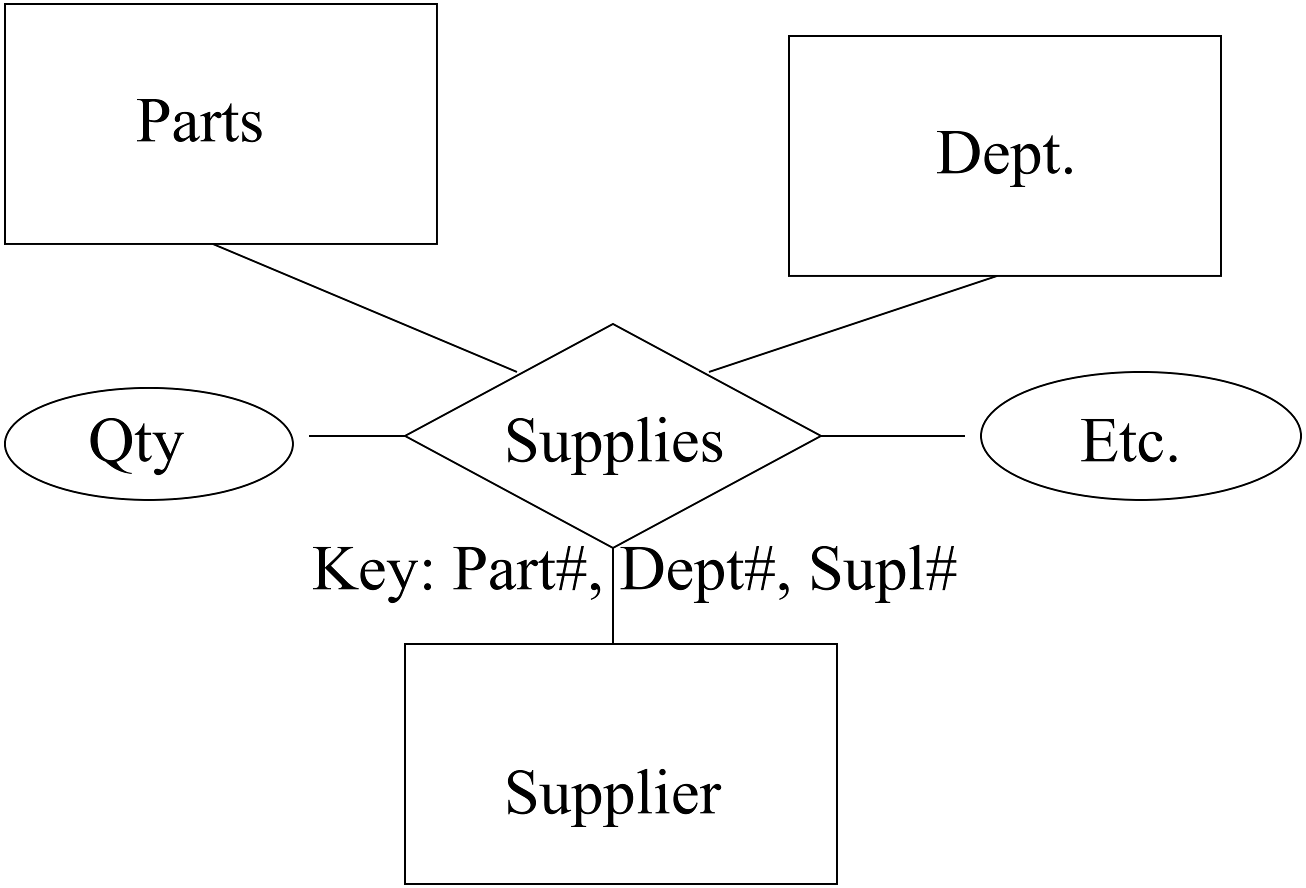
Better design

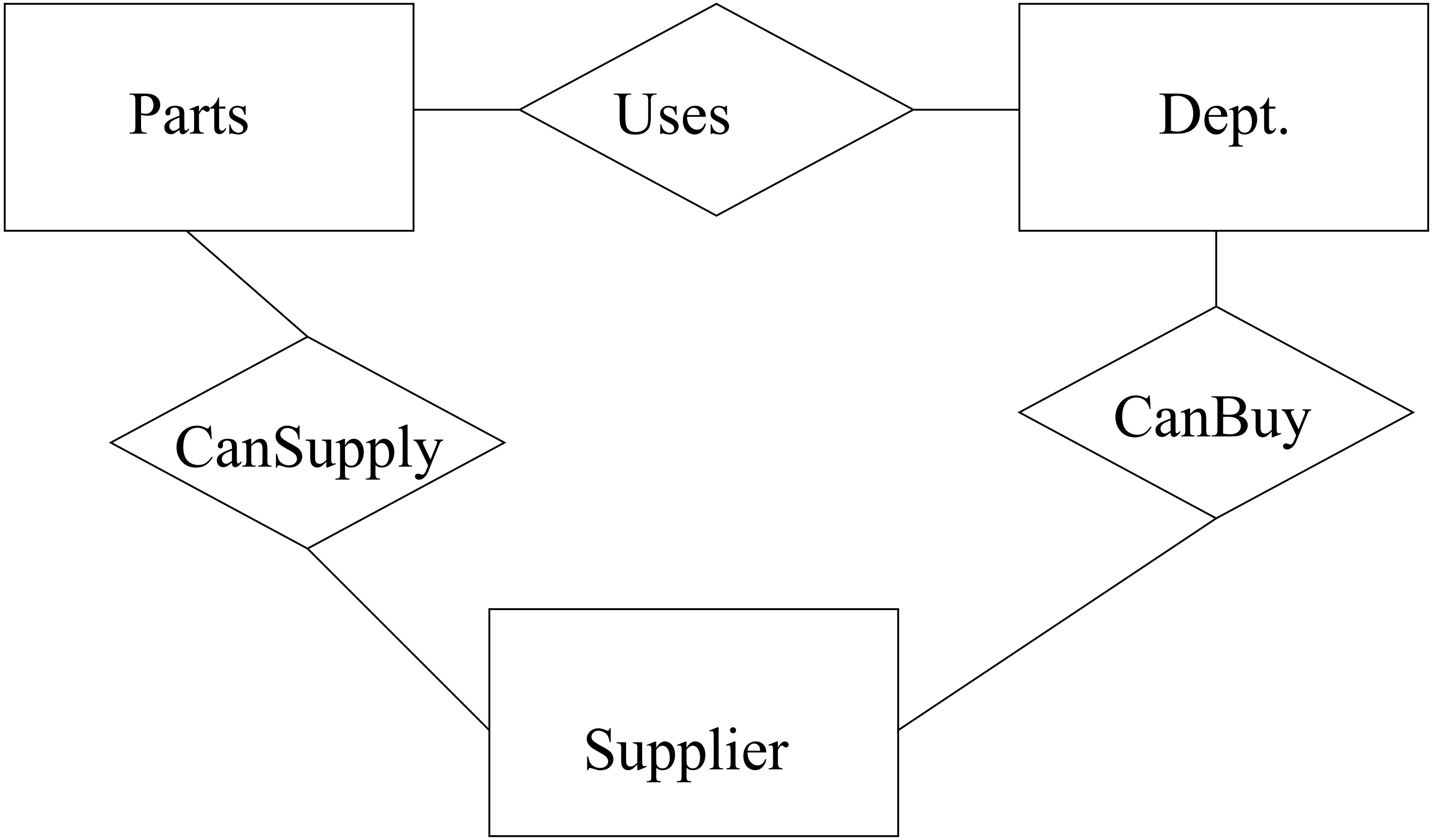
- The key constraints allow us to combine Purchaser with Policy and Covers with Dependents.
- Participation constraints lead to NOT NULL constraints.

```
CREATE TABLE  Policy (
   policy#  INTEGER,
   cost  REAL,
   sin  CHAR(9)  NOT NULL,
   PRIMARY KEY (policy#).
   FOREIGN KEY (sin) REFERENCES
                Employees,
      ON DELETE CASCADE)

CREATE TABLE Dependents (
   dname  CHAR(20),
   dob  DATE,
   policy#  INTEGER,
   PRIMARY KEY (dname, policy#).
   FOREIGN KEY (policy#)
              REFERENCES Policy,
      ON DELETE CASCADE)
```

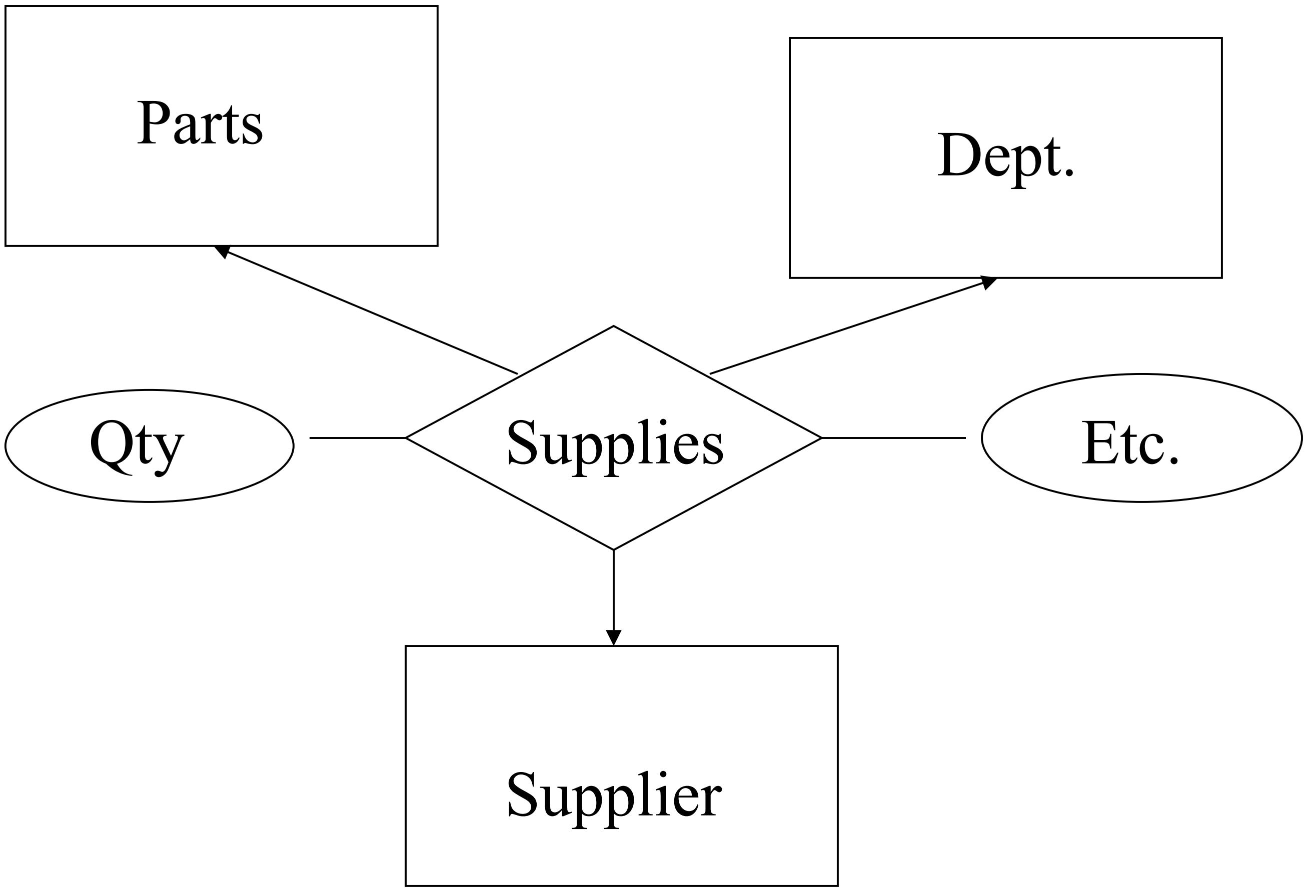
An example where a ternary relation is better than a number of binary relations is the following:

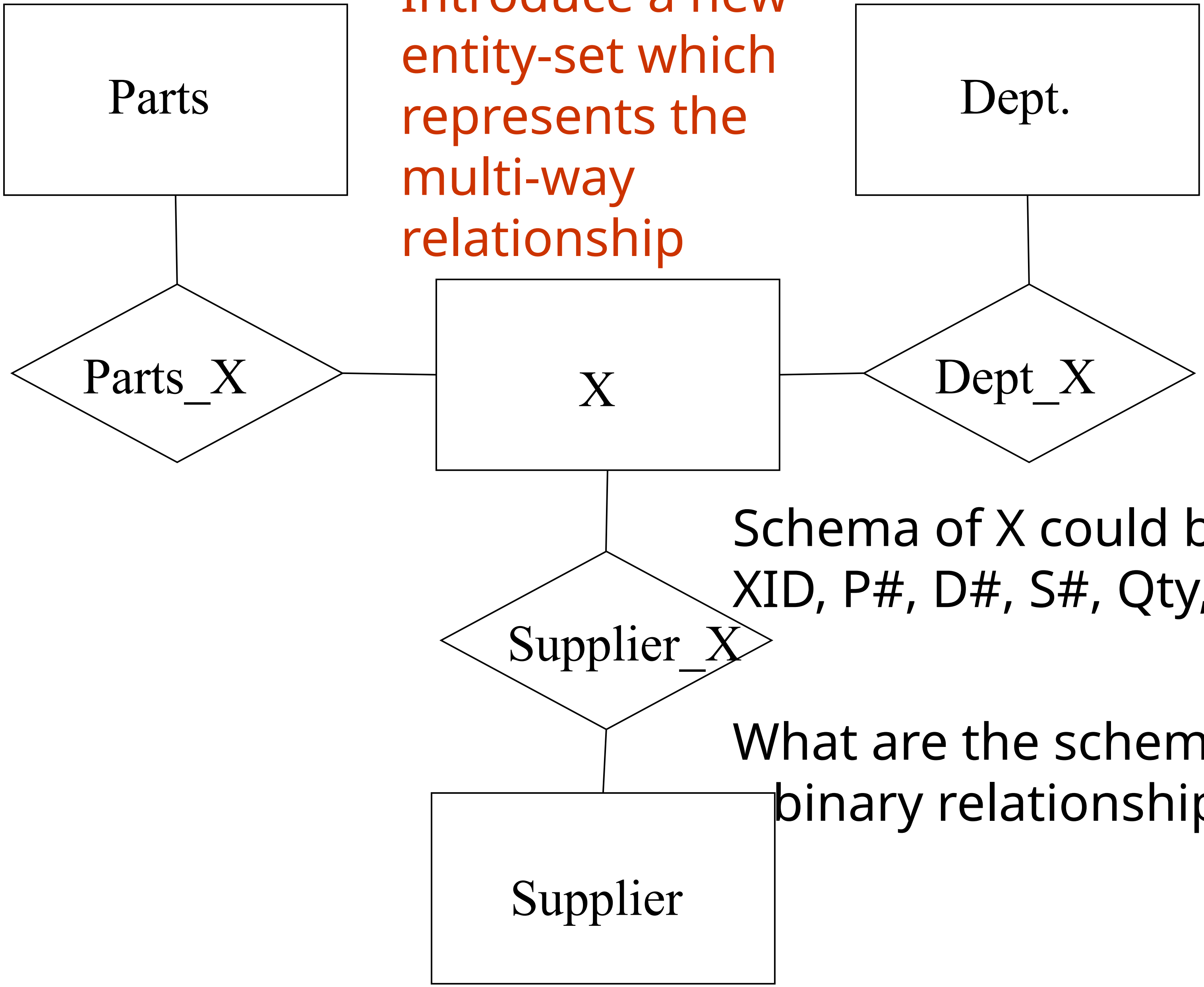



Parts
Uses
Dept.
CanSupply
CanBuy
Supplier

- Supplies relates entity sets Parts, Departments and Suppliers, and has descriptive attributes price, quantity etc.
- A number of binary relationships may not convey the semantics
- Supplier ``CanSupply'' Part, Dept. ``Uses'' Part, and Dept. ``Can Buy'' from Supplier does not imply that Dept has a PO to buy Part from S.
  - How do we record the following: which part, quantity price?

A department can order only one part from a supplier?





Introduce a new entity-set which represents the multi-way relationship

Schema of X could be XID, P#, D#, S#, Qty, …

What are the schema of the binary relationship?

Constraints Beyond the ER Model

- Functional dependencies:
  - *A department can order only one part from a given supplier.*
    - □ Can’t express this in ternary Supplies relationship.
  - Normalization refines ER design by considering FDs.
- Inclusion dependencies:
  - Special case: Foreign keys (ER model can express these).
  - *e.g., At least 1 person must report to each manager.* (Set of *sin* values in Manages must be subset of *supervisor_ssn* values in Reports_To.)
- General constraints:
  - *e.g., Manager’s discretionary budget less than 10% of the combined budget of all departments he or she manages.*