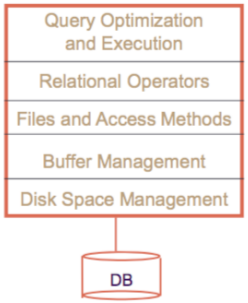
Database Definition
A database system is a computerized record-keeping system that acts as a **mediator** between the **humans** and the **physical devices** that hold them.

Database system components:
Data: Raw facts about people or objects
Software: A database management system (DBMS) is the basic software that supports the management of the database.
Hardware: Data are stored on disk. The processor and main memory are used during execution.
Users: End-users, programmers, data administrators.

Structure of DBMS



DBMS F0: Uniform Data Administration
- Access control and monitoring
- Performance tuning
- Disk space management
- Concurrency control
- Designs logical /physical schemas
- Data availability, crash recovery

DBMS F1: Reduced Application Development Time

DBMS F2: Data Independence (Important Benefit of DBMS!)
- 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.

DBMS F3: Data Integrity
- Avoid unnecessary duplication
- Ensure only correct values

DBMS F4: Data Security
- Managing access rights
- Preventing disclosure of information
- Identify intrusion attempts
- Managing risky queries (e.g. large queries)

DBMS F5: Concurrency Control
- Concurrent execution of user programs is essential for good DBMS performance. (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: e.g., check is cleared while account balance is being computed.
- DBMS ensures such problems don't arise: users can pretend they are using a single-user system.
- Key concept is *transaction*, which is an *atomic* sequence of database actions (reads/writes).
- ACID principle (**A**tomic, **C**onsistent, **I**solated, **D**urable)

DBMS F6: Crash Recovery Using the Log
- The log keeps all information about a transaction (Old value, New value, Commit or Abort)
- Ensures redo and undo capabilities

DDL: Data Definition Language
DML: Data Manipulation Language
SQL: Structured Query Language

Data Model: Collections of concepts for describing data
Schema: Description of a particular collection of data, using the given data model.

Relational Model of Data:
- Main concept: *relation*, basically a table with rows and columns.
- Every relation has a *schema*, which describes the columns, or fields.

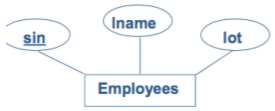
Integrity constrains (field must have something, e.g. PK) are useful to improve data integrity (accuracy and consistency of data)

Data Independence

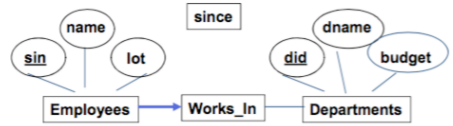
Designing Your Database (Steps)
1) Gather requirements and data
2) Create conceptual model
   a. ER (Entity Relationship) Diagram
   b. EER (Enhanced Entity Relationship) Diagram
   c. ER is often used over EER
3) Create logical Model (Relational Model
4) Implement logical models (Tables)
5) Import Data
6) Query Data (SQL) to create reports

Entity: Real-world objects. An entity is described (in DB) using a set of *attributes*.
***Entity Set*:** A collection of similar entities. E.g., all employees at a departmental store
Each entity set has a primary key (PK, UID)
Each attribute has a domain of possible values (enforcing data integrity)

Relationship: Association among two or more entities. E.g., Attishoo works in Pharmacy department.
Relationship Set: Collection of similar relationships.

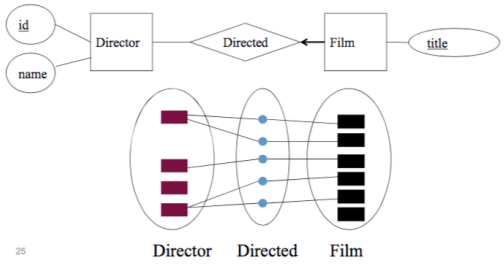


Key Constraint: When there is a key constraint, the entity can participate at most one time (E.g. An employee works in at most one department)



One-to-Many

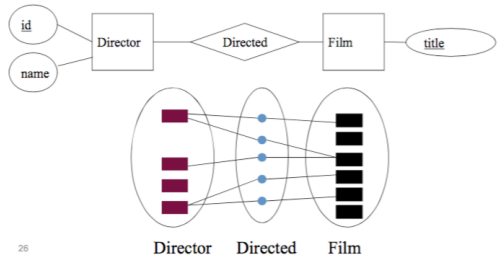
✓ A film is directed by at most one director
✓ A director can direct any number of films



25

Many-to-Many

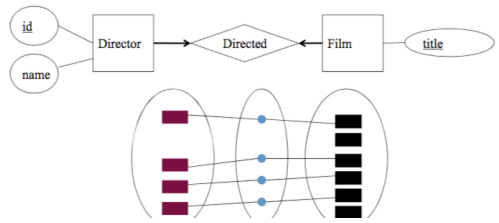
✓ A film is directed by any number of directors
✓ A director can direct any number of films



26

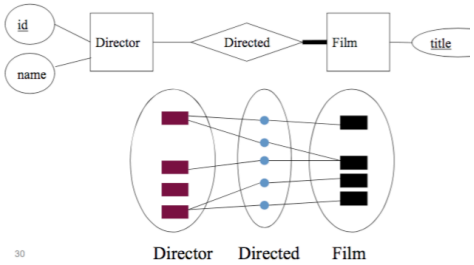
One-to-One

✓ A film is directed by at most one director
✓ A director can direct at most one film



Participation Constraints

- A film has at least one director
- A director can direct any number of films



30

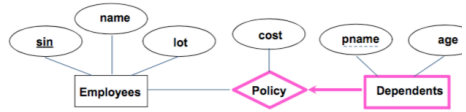
Participation Constraints in SQ

- We can capture participation constraints involv 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)
```

Weak Entities

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

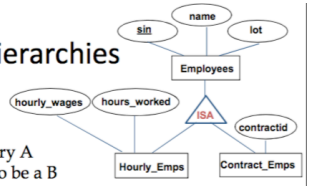


Translating Weak Entity Sets

- Weak entity set and identifying relationshi set are translated into a single table.
  - When the owner entity is deleted, all owned wea 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)
```

ISA (`is a') Hierarchies



As in Java, attributes are inherited.

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?
  - Contract_Emps OVERLAP Senior_Emps
- *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity?
  - Moterboats AND Cars COVER Motor_Vehicles
  - Hourly_Emps AND Contract_Emps COVER Employee

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.

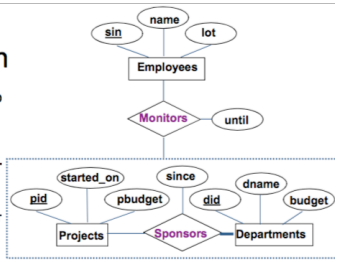
The steps

1. Identify the entities
2. Identify the relationships between the entities
3. Determine whether a relationship is 1:1, 1:M or M:N
4. Determine whether participation in a relationship is mandatory (at least one instance; full participation) or not
5. Identify Weak Entities
6. Identify ISA hierarchies and Aggregations
7. Consider possible refinements:
   a) Should a concept be modeled as an entity or an attribute?
   b) Should a concept be modeled as an entity or a relationship?
   c) Identifying relationships: Binary or ternary?

49

Aggregation

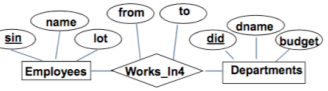
- Used when we have to model a relationship involving (entity sets and) a *relationship set*.
  - *Aggregation* allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.



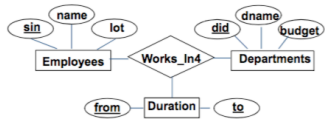
* *Aggregation*:
- Monitors is a distinct relationship, with a descriptive attribute.
- Also, can say that each sponsorship is monitored by at most one employee.

Entity vs. Attribute (Contd.)

Works_In4 does not allow an employee to work in a department for two or more periods.

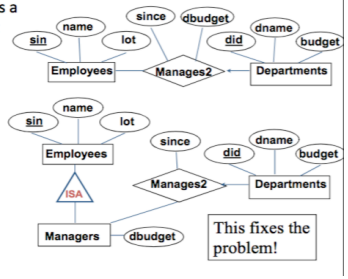


Similar to the problem of wanting to record several addresses for an employee: We want to record *several values of the descriptive attributes for each instance of this relationship.* Accomplished by introducing new entity set, Duration.



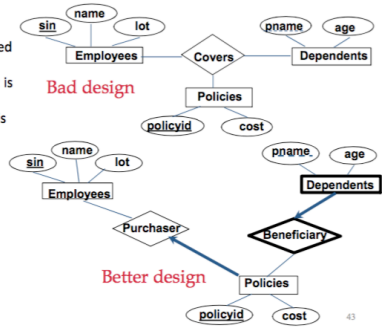
Entity vs. Relationship

What if a manager gets a discretionary budget that covers *all* managed depts?



Binary vs. Ternary Relationships

If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, first diagram is inaccurate.



43

Relational Database: Definitions

- *Relational database:* a set of *relations*
- *Relation:* 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).

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

Destroying and Altering Relations

sid	name	login	age	gpa
53666	Jones	jones@cs	18	8.4
53688	Smith	smith@eecs	18	9.2
53650	Smith	smith@math	19	8.8

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)
```

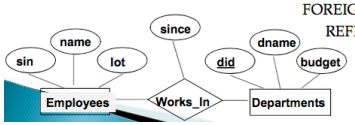
- Can delete all tuples satisfying some condition (e.g., name = Smith):

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

Superkey	An attribute or combination of attributes that uniquely identifies each entity in a table.
Candidate key	A minimal superkey that does not contain a subset of attributes that is itself a superkey.
Primary key	A candidate key selected to uniquely identify all other attribute values in a given row. No nulls allowed.
Secondary key	An attribute or combination of attributes usec strictly for data-retrieval purposes.
Foreign key	An attribute or combination of attributes in o table whose value must either – match the primary key in another table, or – be null (contain no value, i.e. empty)

Relationship Sets to Tables (for M:N)

- 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(1),
  did  INTEGER,
  since  DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn)
     REFERENCES Employees,
  FOREIGN KEY (did)
     REFERENCES Departments)
```

- Create the tables
- Enforce Integrity Constraints
  - Primary key
  - Referential integrity
    - Default is NO ACTION (*delete/update is rejecte*
    - CASCADE (also delete all tuples that refer to del tuple)
    - SET NULL / SET DEFAULT (sets foreign key value referencing tuple)

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

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

ON DELETE CASCADE/RESTRICT
ON UPDATE CASCADE/RESTRICT

```
SELECT C.Lastname, C.Firstname, C.nationality, S.capacity,
    B.capacity, B.ticket-price
  FROM Client C, Ship S, Cruise E, Cabin B
  WHERE C.guestno = E.guest AND
      E.shipname = S.shipname AND
      S.shipname = B.shipname AND
      E.cabinNumber = B.cabinNumber AND
      C.guestNo NOT IN
       (SELECT C2.guestNo
          FROM Client C2, Cruise E2, Origin O, Destination D
          WHERE C.guestNo = E.guestNo AND
          O.origin-id = E.origin-id AND
          E.destination-id = D.destination-id AND
          O.name = "Ushuaia" AND O.country = "Argentina" AND
          D.name = "Adelaide Island"
          AND D.country = "Antarctic Circle")
  ORDER BY C.Nationality;
```