# Data Model

Need a model for describing the structure of data and constraints, and operations on data (some sort of an *abstraction* layer)

### Summary

1. tabular representation of data
2. simple and intuitive
3. Integrity can be specified by the DBA, DBMS checks for violations
4. Powerful and natural query languages exist

## Physical Level

The problem of working with data in the physical level is that

1. it is difficult change to physical representation
2. application code becomes complex
3. impossible to implement new features rapidly

## Conceptual Level

1. Hides the details
2. Mapping from **conceptual** to **physical** schema is done by DBMS
3. **Physical Data Independence** - thus changes to the physical schema can be changed without changing applications

## External Level

1. Applications can access data through some **views**: different views of data for different categories of users (a view is computed)
2. Mapping from external to conceptual schema is done by DBMS
3. **Logical Data Independence** - Conceptual schema can be changed without changing applications

# Relational Databases

A set of relations (tables)

## Basic Idea

1. Organize data as a set of tables

### Advantages

1. simple
2. solid mathematical foundation (set theory)
3. powerful query languages
4. efficient query optimizers

## Relation

More formally, a relation is a **set** of rows (or tuples). It consists of *instance* and *schema*

### Instance

Table content with rows and columns
**Cardinality**: number of rows
**Degree / Arity**: number of fields

### Schema

Table structure with name and type of columns

# Querying Relations

1. Queries can be written intuitively, and the DBMS is responsible for efficient evaluation
2. Precise semantics for relation queries
3. Allows the optimizer to extensively re-order operations, and still ensure the answer does not change

## Examples

```
SELECT *
FROM STUDENT S
WHERE S.age = 18;
```

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

# Creating Relations in SQL

The type (**domain**) of each field is specified by user/programmer, and enforced by the DBMS

## Examples

```
CREATE TABLE Students
        (sid: CHAR(5),
        name: CHAR(10),
        login: CHAR(15),
        age: INT,
        gpa: FLOAT);
```

# Adding and Deleting Tuples

Inserting a single tuple

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES (11010, 'Andrew', 'andrew@cs', 18, 3.3);
```

Delete all tuples that satisfy a condition

```
DELETE
FROM Students
WHERE name='Bob';
```

# Integrity Constraints (ICs)

Conditions that hold for any instance of the database (domain constraints). These are **defined when schema is defined**, **checked when relations are modified**.

## Keys

1. Key attributes are used to uniquely identify an individual record
2. Used to ensure *integrity*, *efficiency*, and *proper structure of data*
3. Help uniquely *identify records*, *enforce relationships*, and *optimize db operations*

### Superkey

1. A unique combination of *one or more* attributes to identify a row

2. May contain *extraneous* attributes (non-minimal - beyond the minimum needed for uniqueness)
3. Must *guarantee uniqueness*

## Candidate Key

1. A *minimal* key that *uniquely* identifies a row
2. A table can have *multiple* candidate key
   *Every candidate key and primary key is a superkey, but not every superkey is a candidate key*

## Primary Key

A column or set of columns in a a table that uniquely identifies each row in that table. Characteristics are

- no two rows can have the same value(s) in the primary column(s)
- It **cannot** contain NULL values
- must be unique for each record
- can be a single key (simple) or combination of attributes (composite key)
- Only one primary key per table

```
CREATE TABLE customers (
        customer_id INT PRIMARY KEY,
        name VARCHAR(100)
);
```

```
CREATE TABLE enrolled (
        sid CHAR(8),
        cid CHAR(8),
        grade CHAR(2),
        PRIMARY KEY (sid, cid)
);
```

## Alternate / Secondary Key

Any candidate key that was not chosen as a the primary key

## Composite Key

A type of candidate key that consists of two or more columns

1. Used when none of the columns individually can uniquely identify a row, but together they do
2. Must be minimal

## Unique Key

A unique key or set of columns that ensures all values in a column (or set of columns) are unique across rows

1. Allows for NULL values
2. Non-NULL values must be unique

## Foreign Key

- A column or set of columns in one table that references the primary key of another table
- It establishes and enforces a relationship between two tables
  - The value in the FK column corresponds to an existing value in the PK table of the referenced table

### Participation Constraint

**Total**: All entities in the referencing table must have a reference to the referenced table
**Partial**: Some (not all) entities in the referencing table must have a reference to the referenced table

```
CREATE TABLE enrolled (
        sid CHAR(8),
        cid CHAR(8),
        grade CHAR(2),
        PRMIARY KEY (sid, cid),
        FOREIGN KEY (sid) REFERENCES students
);
```

## Referential Integrity

Referential integrity is achieved if all foreign key constraints are enforced - **no dangling references**.

### Referential Integrity Constrains

Used to establish rules for referential keys and dependency between tables - it deals with the **referenced table** (holds the PK) and the **referencing table** (holds the FK).

### Enforcing Referential Integrity

#### Default

`NO ACTION`

- delete/update is rejected

#### Cascade

`ON DELETE CASCADE`

- declared in the **referencing table**
- if a row is deleted in the **referenced table** all associated rows in the **referencing table** will also be deleted
  `ON UPDATE CASCADE`
- declared in the **referencing table**
- if a row is updated in the **referenced table** all associated rows in the **referencing table** will also be updated

#### Null

`ON DELETE SET NULL` or `ON DELETE SET DEFAULT`

- declared in the **referencing table**
- if a row is deleted in the **referenced table** all associated rows in the **referencing table** will have their FK set to NULL
  `ON UPDATE SET NULL` or `ON UDPDATE SET DEFAULT`
- declared in the **referencing table**
- if a row is updated in the **referenced table** all associated rows in the **referencing table** will have their FK set to NULL

```
CREATE TABLE enrolled (
        sid CHAR(8),
        cid CHAR(8),
        grade CHAR(2),
        PRMIARY KEY (sid, cid),
        FOREIGN KEY (sid) REFERENCES students
                ON DELETE CASCADE
                ON UPDATE SET NULL
);
```

# Virtual Tables / Views

A view is just a relations, but we store a *definition* rather than a set of tuples. Views are virtual tables created from one or multiple tables depending on the requirement(s).

### Purpose

1. Create a subset
2. Combine data from multiple tables into one definition

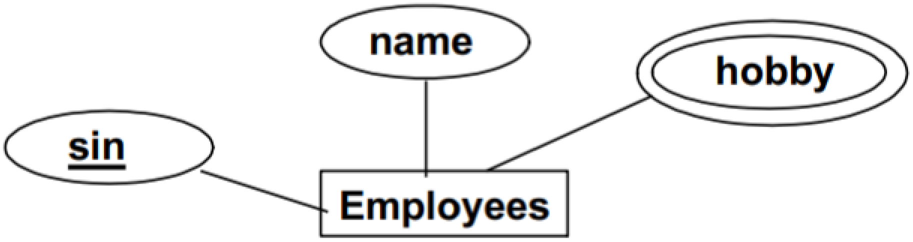
### Security

Views can be used to present necessary information (or summary) while hiding details in underlying relations.

```
CREATE VIEW customer_orders AS
SELECT customers.customer_id, customers.name, orders.order_id
FROM customers JOIN orders ON customer.customer_id = orders.customer_id;
```

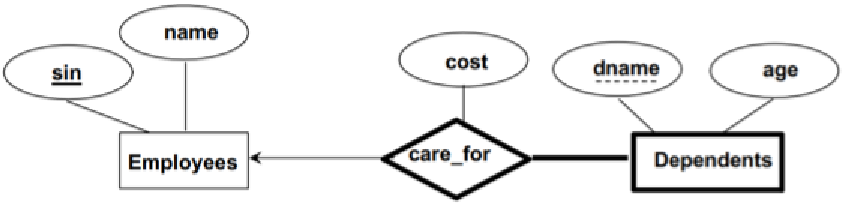
## ER - Modeling

### Fig_1



- **double circle**: set value / can hold multiple entities
    - multi/set-valued attributes are typically represented by a separate table in the relational model, with foreign keys pointing back to the original entity
- **circle**: an attribute
- **underlined**: primary key
    - multiple underlined means a composite key
- **box / rectangle**: table name

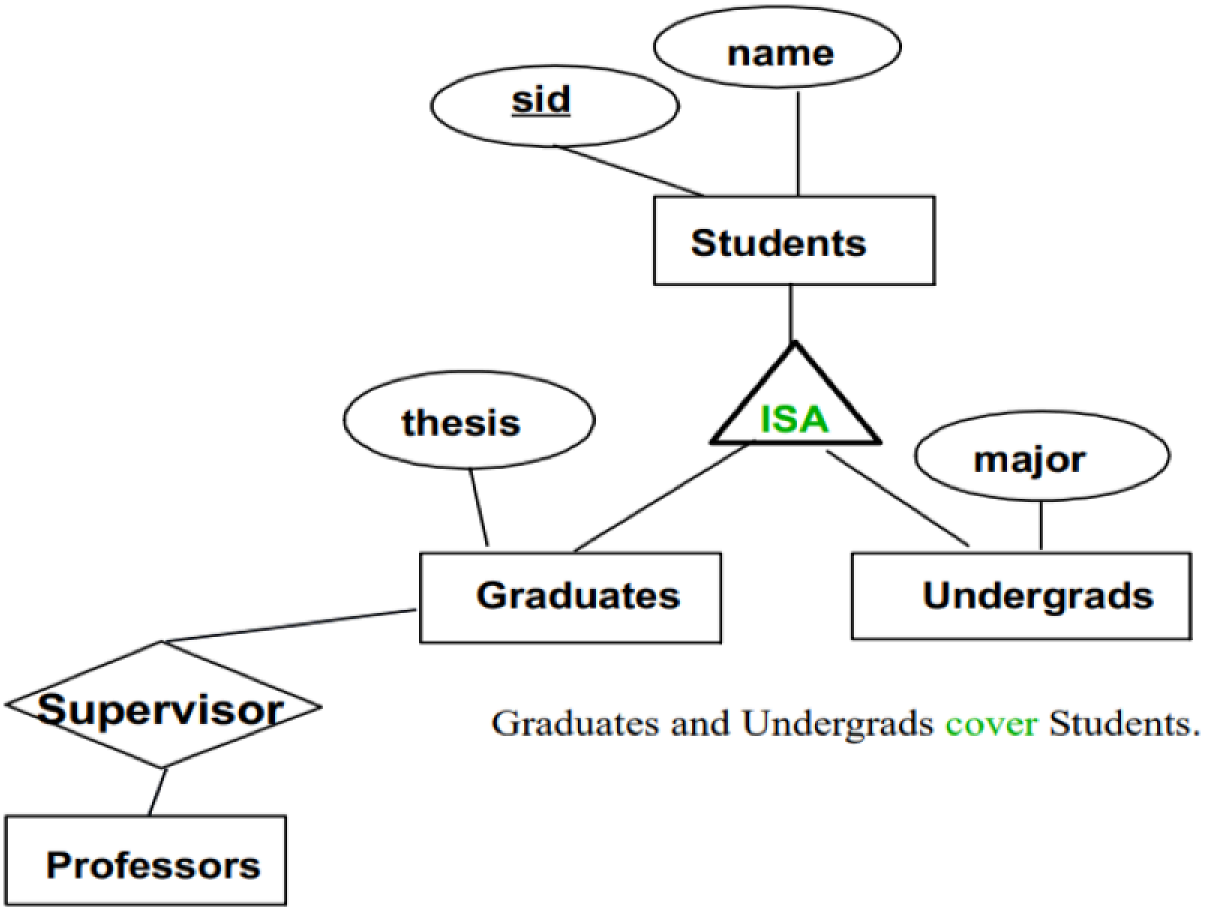
### Fig_2



```
CREATE TABLE care_for (
        dname CHAR(20),
        age INTEGER
        cost REAL,
        sin CHAR(11) NOT NULL,
        PRIMARY KEY (dname, sin) ,
        FOREIGN KEY (sin) REFERENCES Employees ON DELETE CASCADE
);
```

- **thick-line** and **thick-square**: Weak entities
  - cannot be uniquely identified by their own attributes, rely on their strong entity's PK
  - **existence-dependent** on strong entity (identifying relationship)
  - typically identified by a **composite primary key** that includes the PK of the strong entity
  - In Fig_2, dname and sin are used as composite primary key for Dependents
- **arrow**: The arrow end of an edge is a single, while non arrow end is many
  - in Fig_2 its a *ONE EMPLOYEE* cares for *MANY DEPENDENTS*

## Fig_3

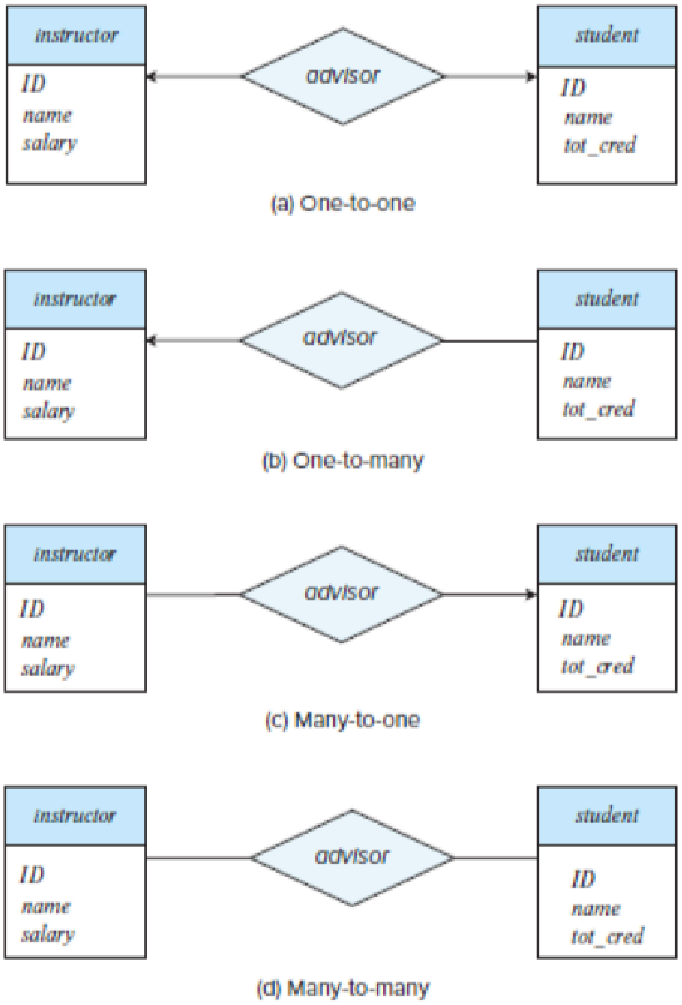


Graduates and Undergrads cover Students.

```
CREATE TABLE Undergrads (
        sid CHAR(8) NOT NULL
        major CHAR(12),
        PRIMARY KEY (sid),
        FOREIGN KEY (sid) REFERENCES Students ON DELETE
);
```

- **ISA Triangle**: Subclass-superclass relationship
  - ISA hierarchy
  - *Superclass*: more general type and contains attributes common to all
  - *Subclass*: specific entity type and inherits all the attributes in the superclass and have additional attributes of each own

### Fig_4



(a) One-to-one

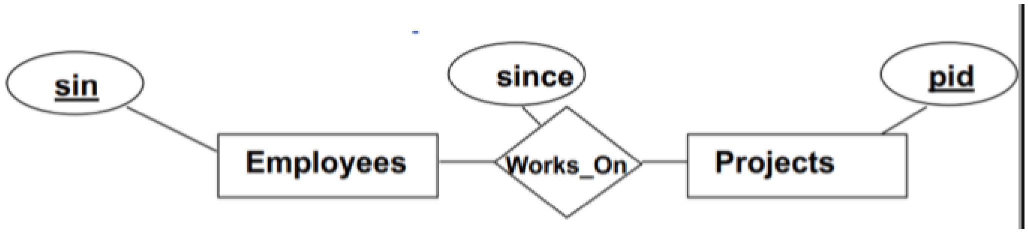
(b) One-to-many

(c) Many-to-one

(d) Many-to-many

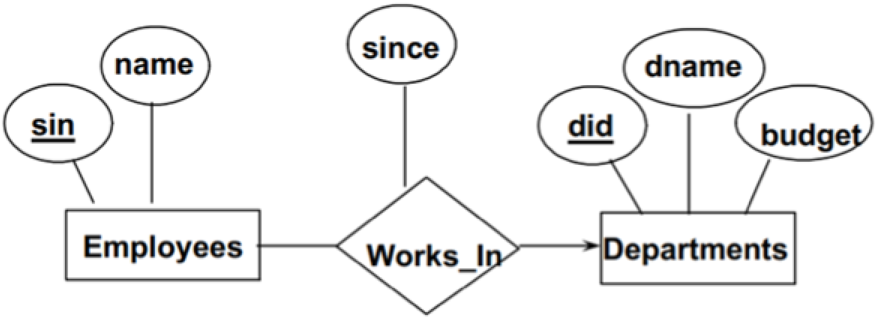
- **Triangle**: means relationships

### Fig_5



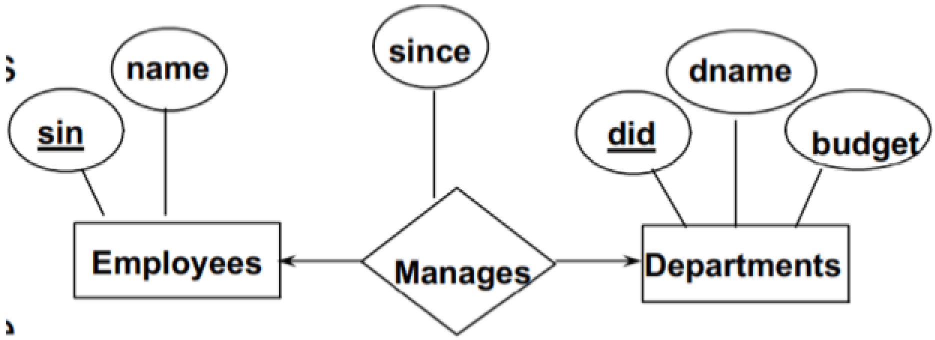
```
CREATE TABLE Works_ON(
        sin CHAR(11),
        pid INTEGER,
        since DATE,
        PRIMARY KEY (sin, pid),
        FOREIGN KEY (sin) REFERENCES Employees,
        FOREIGN KEY (pid) REFERENCES Projects
);
```

### Fig_6



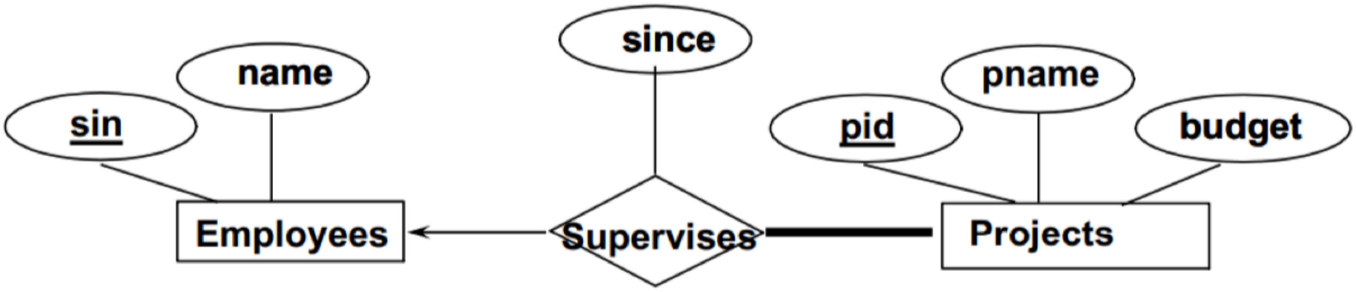
```
CREATE TABLE Works_In(
        sin CHAR(11),
        did CHAR (3),
        since DATE,
        PRIMARY KEY(sin),
        FOREIGN KEY (did) REFERENCES Departments
);
```

Fig_7



```
CREATE TABLE Manages(
        sin CHAR(11) UNIQUE,
        did CHAR (3),
        since DATE,
        PRIMARY KEY(did),
        FOREIGN KEY (sin) REFERENCES Departments
);
```

Fig_8



```
CREATE TABLE Supervises(
        pid INTEGER,
        sin CHAR(11),
        since DATE,
        pname CHAR(20),
        budget REAL
        PRIMARY KEY (PID) NOT NULL,
        FOREIGN KEY (SIN) REFERENCES Employees ON DELETE NO ACTION
);
```

- **thick-line**: A total participation constraint
    - this means that *all* projects *must* have an employee that supervises it