Database Management Systems

Ehsan Noei
e.noei@utoronto.ca

UNIVERSITY OF TORONTO


Requirements
ER Diagrams
Schema

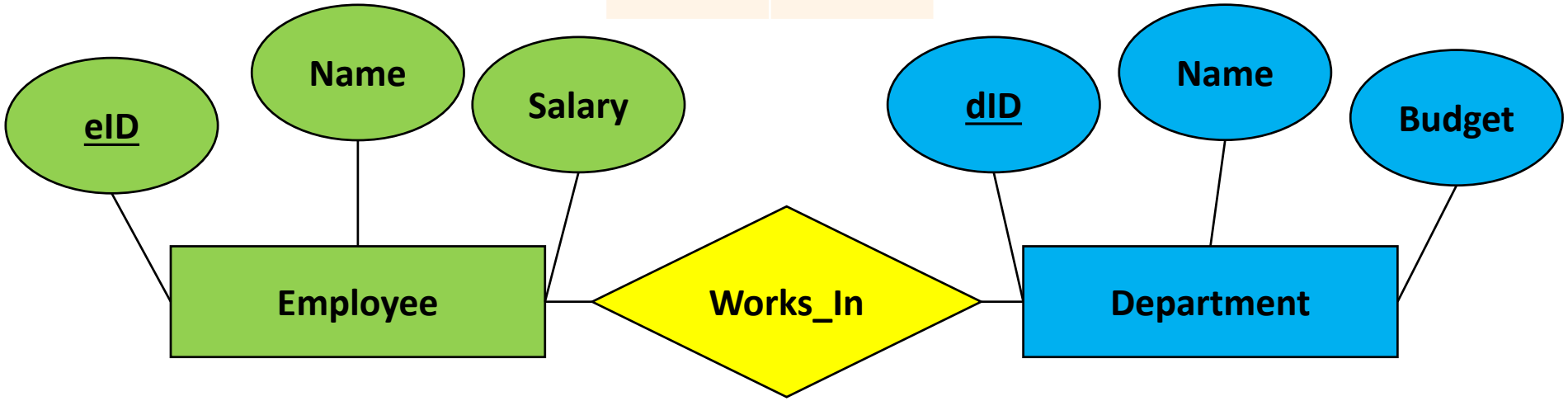
Relational Database: Definitions

- *Relation:* consists 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)

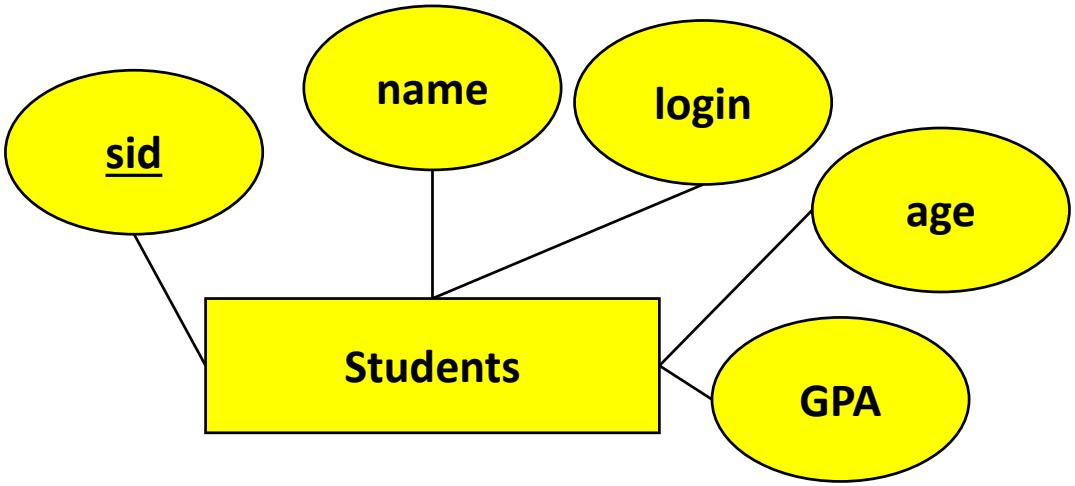
eID	Name	Salary
123	Ehsan	$10

eID	dID
123	d20

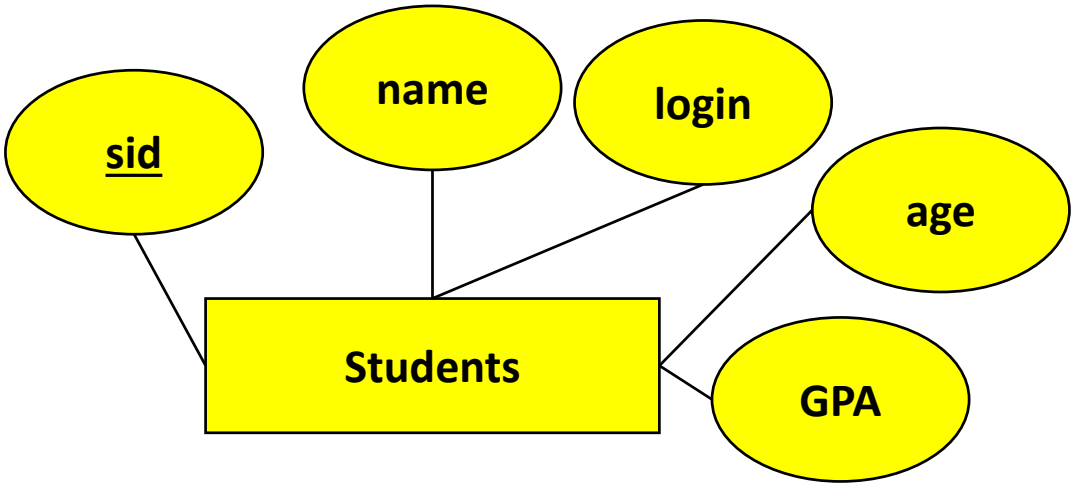
dID	Name	Budget
d20	DB	$20



Students (sid: string, *name:* string, *login:* string, *age:* integer, *gpa:* real)



CREATE TABLE Students (sid: string, *name:* string, *login:* string, *age:* integer, *gpa:* real)



SQL Query Language

- 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.
- Output of a query can be perceived as a TABLE.

SQL Query Language

SELECT < COLUMNS>

FROM <TABLE>

WHERE <CONDITION>

SQL Query Language

```
SELECT Name, Salary
        FROM Employee
        WHERE eID = '123'
```

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

SQL Query Language

```
SELECT *
        FROM Employee
        WHERE eID = '123'
```

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

SQL Query Language

```
SELECT *
        FROM Employee
        WHERE Name = 'Ehsan' AND Salary = '10$'
```

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

SQL Query Language

```
SELECT *
        FROM Employee
        WHERE Name = 'Ehsan' OR Salary = '10$'
```

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

SQL Query Language

```
SELECT *
        FROM Employee
```

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

SQL Query Language

```
SELECT E.Salary, N.Address
    FROM Employee E, Manager N
    WHERE E.Name = N.Name
```

Employee

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

Manager

mID	Name	Address
666	Joe	CA
667	Steve	CA

SQL Query Language

SELECT <COLUMNS>

FROM <TABLE>

WHERE <CONDITION>

DELETE

FROM <TABLE>

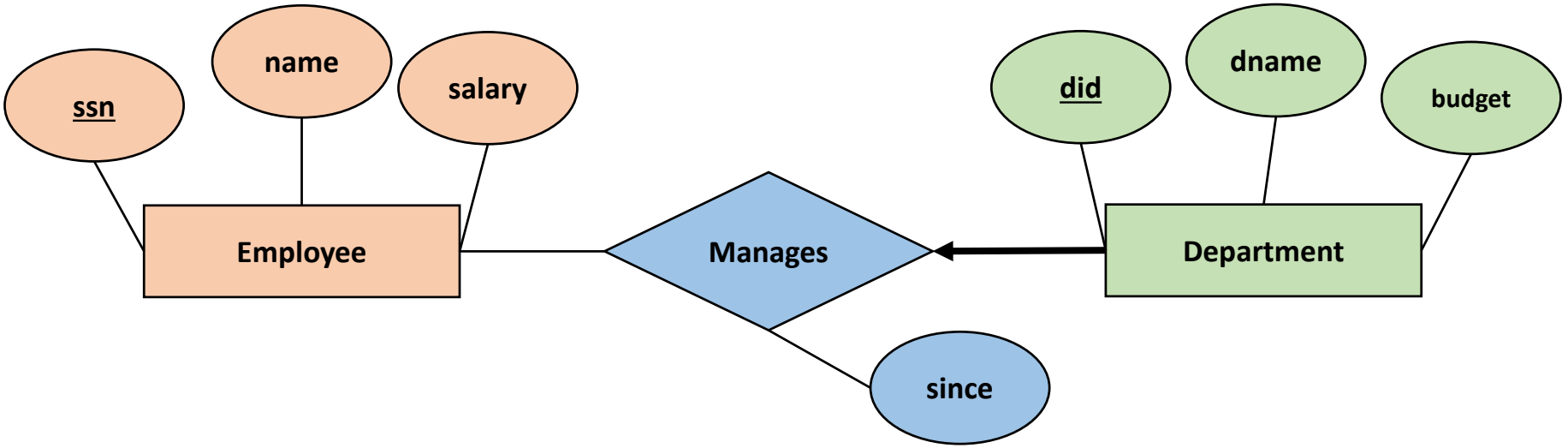
WHERE <CONDITION>

UPDATE <TABLE>

SET column1 = value1, column2 = value2, ...

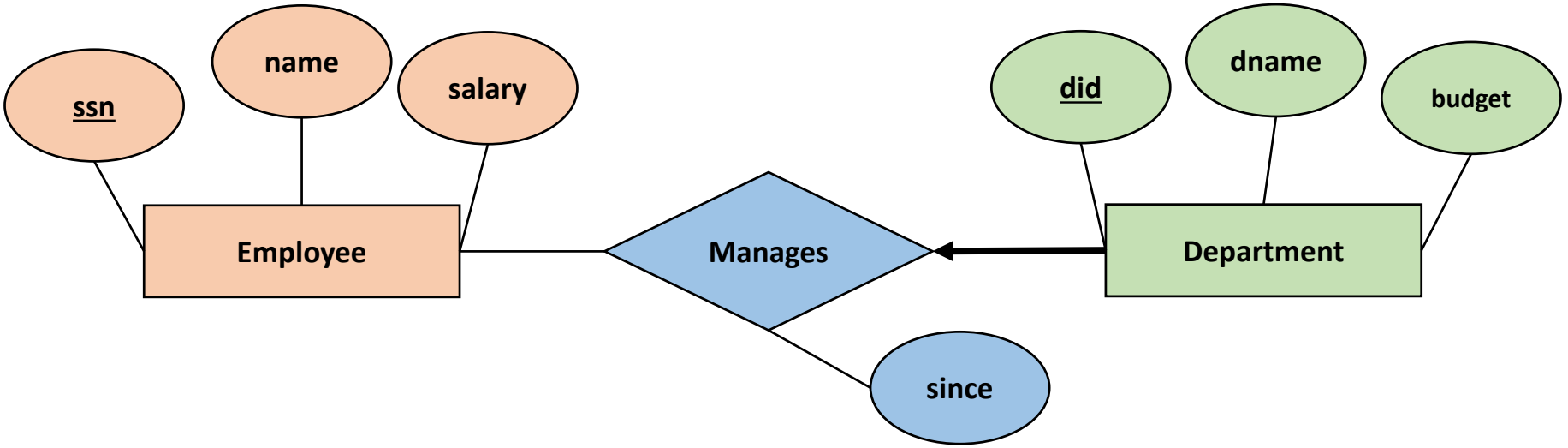
WHERE <CONDITION>

Key and Participation Constraints



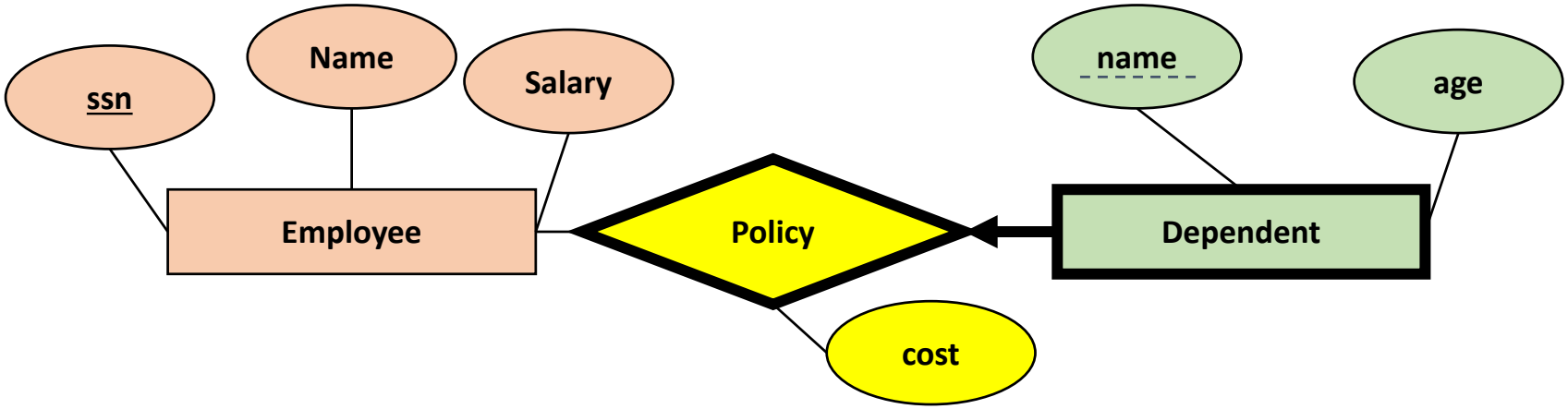
```
CREATE TABLE Manages(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES
     Employees,
  FOREIGN KEY (did) REFERENCES
     Departments)
```

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



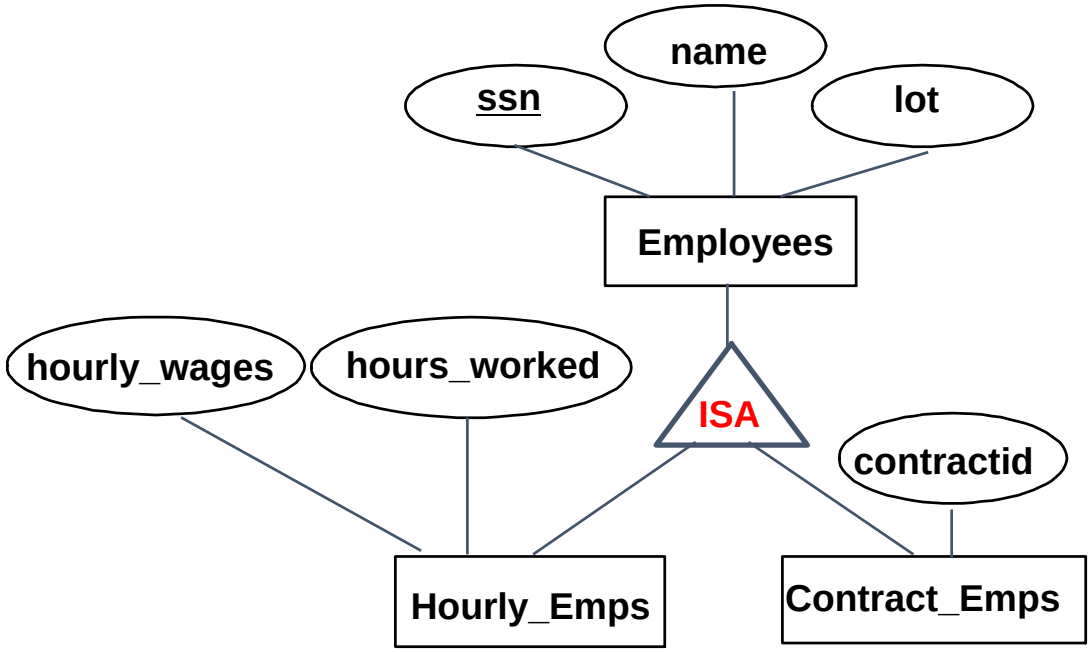
Weak Entity

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



Class Hierarchies

- Employees is specialized into subclasses.
- Hourly_Emps and Contract_Emps are generalized by Employees.



Class Hierarchies

- *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)*

Class Hierarchies

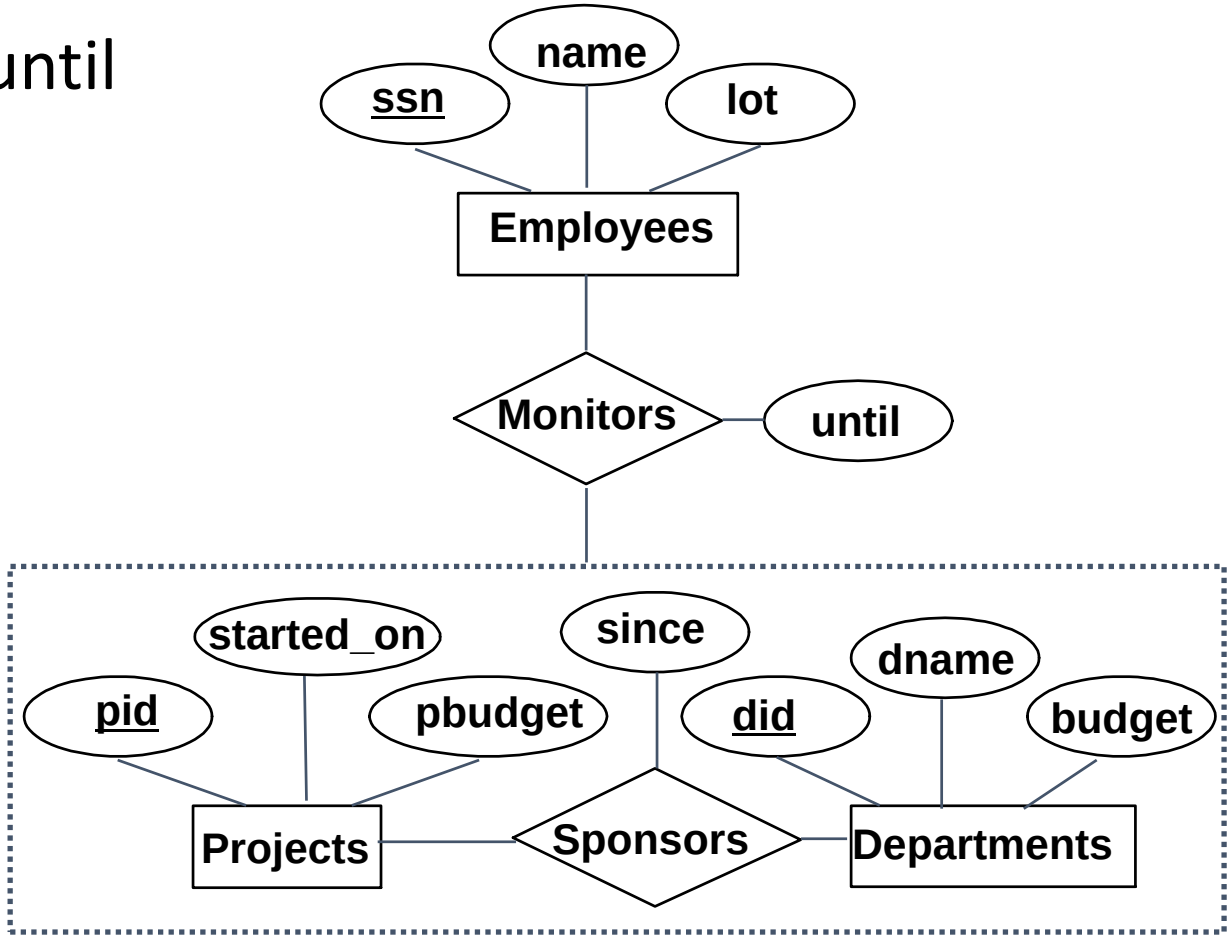
- ***General approach:***
  - 3 relations: Employees, Hourly_Emps and Contract_Emps.
    - *Hourly_Emps*: Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (*hourly_wages*, *hours_worked*, *ssn)*; must delete Hourly_Emps tuple if referenced Employees tuple is 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.

Aggregation

- Used when we have to model a relationship involving (entitity 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

- pid,did,ssn,until



Values

```
CREATE TABLE Students (
    sid int,
    name string,
    PRIMARY KEY (sid)
    )
```

Values

```
CREATE TABLE Students (
    sid int,
    name string NOT NULL,
    PRIMARY KEY (sid)
    )
```

Values

```
CREATE TABLE Students (
    sid int,
    name string DEFAULT ‘sth’,
    PRIMARY KEY (sid)
    )
```

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

- 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.

Views

- CREATE VIEW Goodstudents(sid, gpa) as select S.sid, S.gpa from Students S where S.gpa> 3.0
- How about the following:
  - INSERT into S VALUES(“100”,”JONE”,3.2)
  - INSERT into S VALUES(“101”, ”Mike”,2.8)
  - DELETE from S where S.id = “100”
  - INSERT into GS VALUES(“111”, 3.2)
  - INSERT into GS VALUES(“112”, 2.8)
  - DELETE from GS where S.id = “111”

Views and Security

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

Summary[1]

ER Model	Relational Model
Entity type	*Entity* relation
One to many or one to one relationship	Foreign key (or *relationship* relation)
Many to many relationship	*Relationship* relation and *two* foreign keys
n-ary relationship type	*Relationship* relation and *n* foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

[1] The table is adopted from Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems, Chapter 9, Sixth Edition