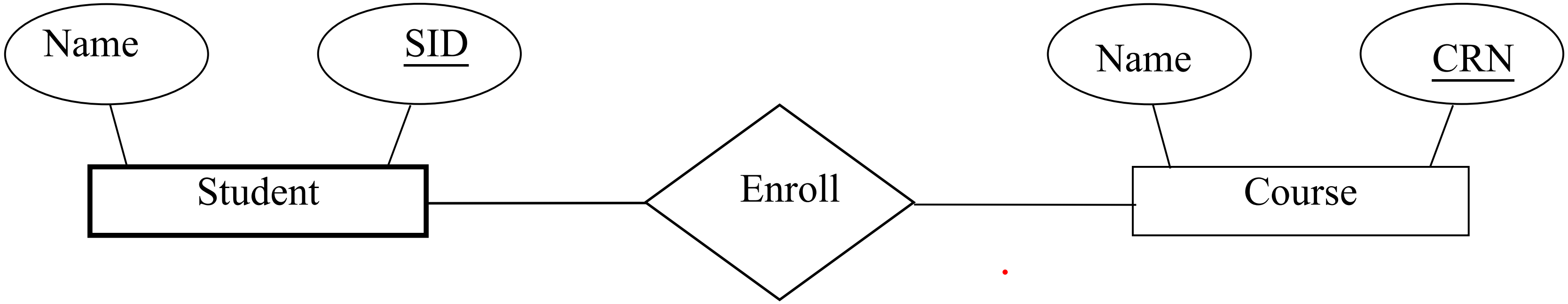
CSCI4333 Database Design & Implement

Lecture Ten – Relational Model 4

Instructor: Dr. Yifeng Gao

From Relational Schema to Table

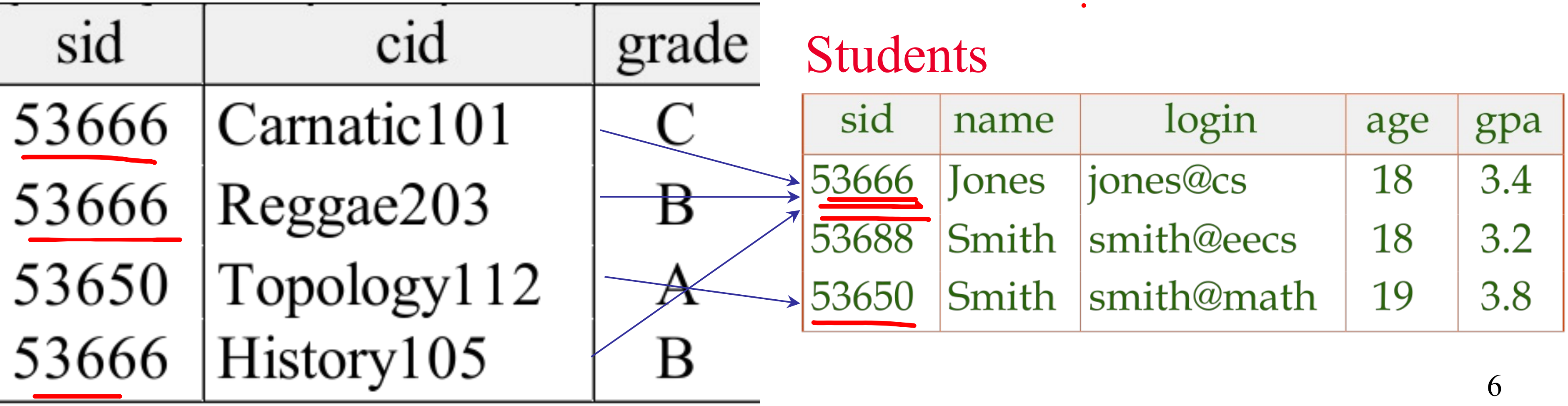


Foreign Keys, Referential Integrity

- *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.) Like a `logical pointer'.
- e.g. *sid* is a foreign key referring to Students:
  - Enrolled(*sid*: string, *crn*: string)
  - If all foreign key constraints are enforced, *referential integrity* is achieved, i.e., no dangling references.

Foreign Keys

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



Enrolled

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

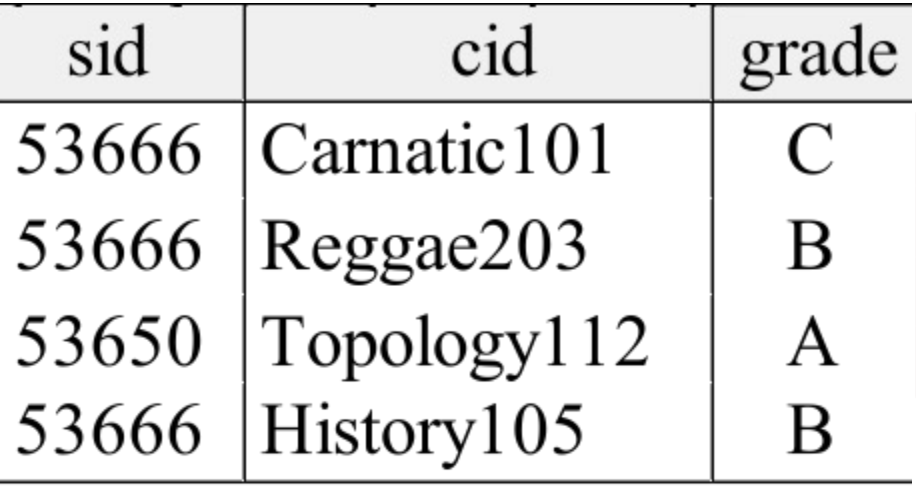
Students

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

Quick Question

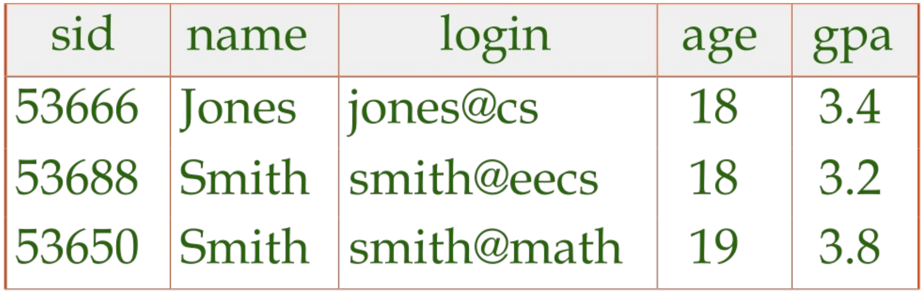
- With foreign key constraint, can I add
  - (51111, History105,A) to Enroll Table? ✗
  - (51111,John, john@cs,17,3.5) to Student Table? ✓

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'*.)

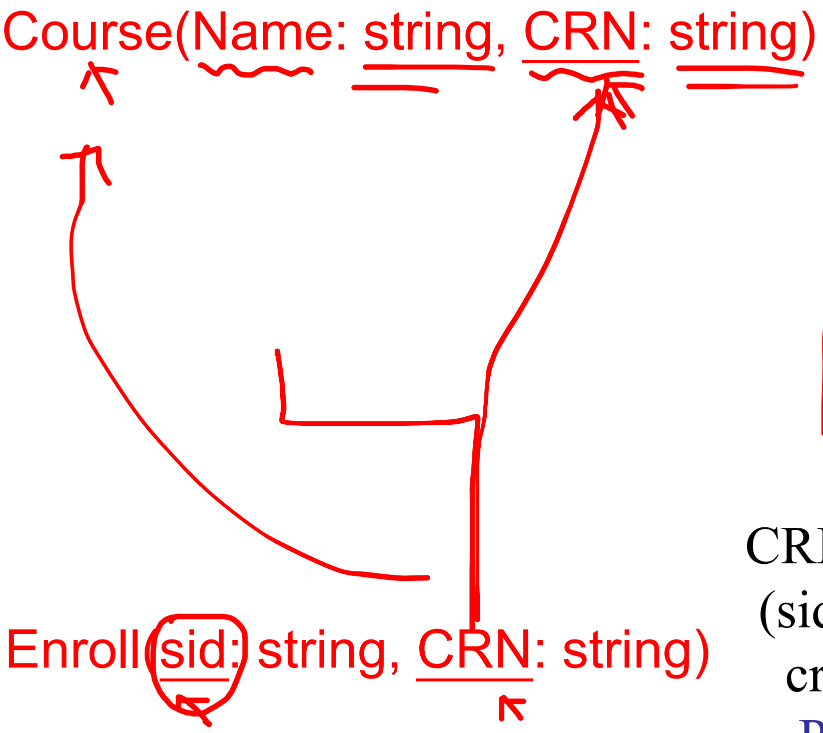
Creating Relations in SQL

Students(sid: string, name: string)

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

```
CREATE TABLE Students(
        sid CHAR(20),
        name CHAR(20),
        PRIMARY KEY sid
        );
```

Creating Relations in SQL



Course(Name: string, CRN: string)

Enroll(sid: string, CRN: string)

```
CREATE TABLE Course(
        CRN CHAR(20),
        name CHAR(20),
        PRIMARY KEY CRN
        );
```

```
CREATE TABLE Enrolled
 (sid CHAR(20),
   crn CHAR(20),
    PRIMARY KEY  (sid,cid),
    FOREIGN KEY (sid) REFERENCES Students
 FOREIGN KEY (crn) REFERENCES Course
);
```

Primary and Candidate Keys in SQL

- Possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key*.
- Suppose a table only have one candidate key:

```
CREATE TABLE Enrolled
   (sid CHAR(20),
    cid  CHAR(20),
    grade CHAR(2),
    PRIMARY KEY  (sid,cid) );
```

Primary and Candidate Keys in SQL

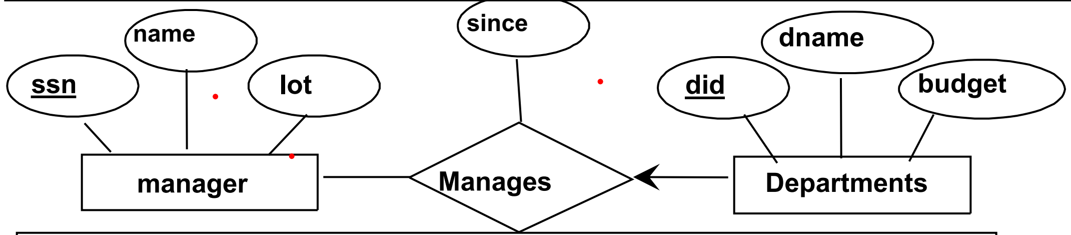
- Possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key*.
- Suppose a table only have two candidate keys:

```
CREATE TABLE Car
  (VIN CHAR(20),
   License#  CHAR(20),
   State CHAR(2),
   PRIMARY KEY  (VIN),
   UNIQUE (License#, State)
);
```

Translating ER Diagrams with Key Constraints

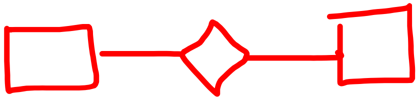
- Option 1: Map relationship to a table:
  - Note that *did* is the key now!
  - Separate tables for Employees and Departments.
- Option 2: Since each department has a unique manager, we could instead combine Manages and Departments.

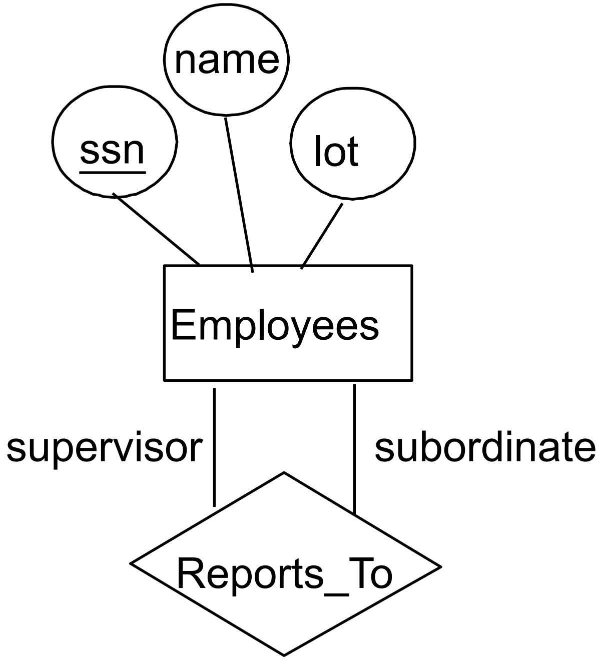
```
CREATE TABLE  Manages(
  ssn  CHAR(11),
  did  INTEGER,
  since  DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Manager,
  FOREIGN KEY (did) REFERENCES Departments);
```



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

Relationship with Roles

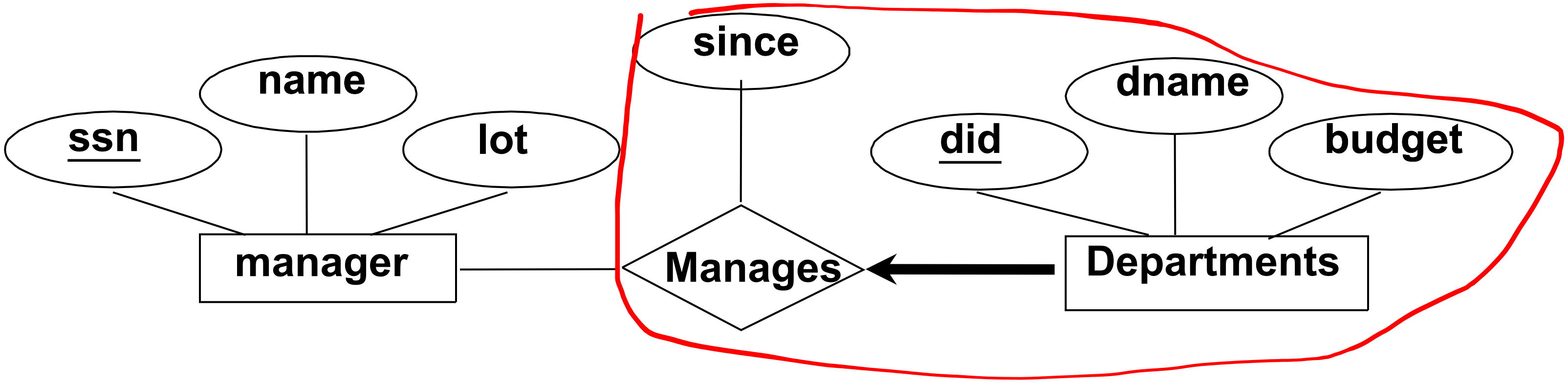




```
CREATE TABLE  Reports_To(
    supervisor_ssn  CHAR(11),
    subordinate_ssn  CHAR(11),
    PRIMARY KEY  (supervisor_ssn, subordinate_ssn),
    FOREIGN KEY (supervisor_ssn) REFERENCES Employees(ssn),
    FOREIGN KEY (subordinate_ssn) REFERENCES Employees(ssn));
```

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!)



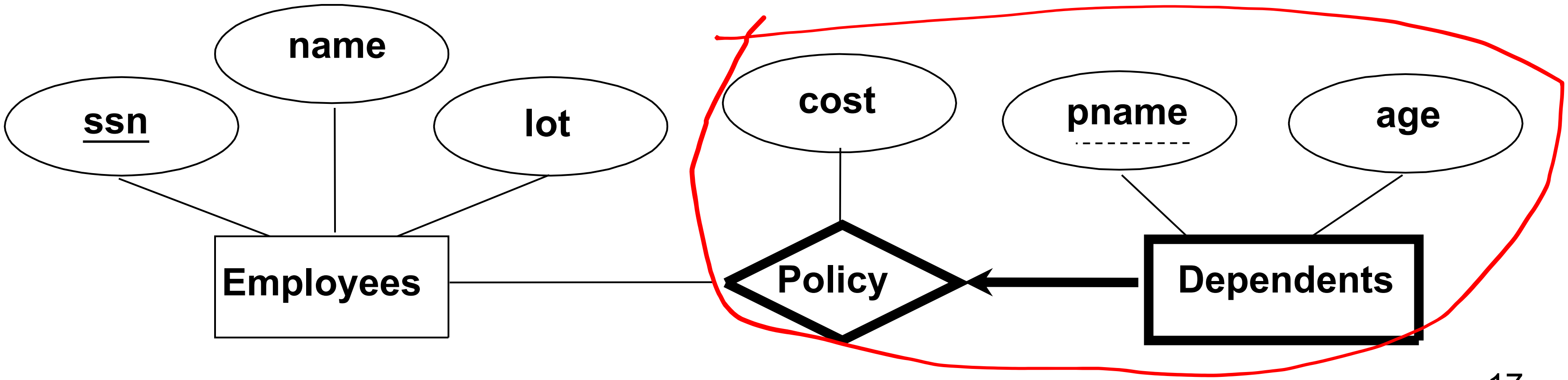
Participation Constraints in SQL

- 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);
```

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)
```

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 )
```

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