Database Systems
(資料庫系統)

10.1.2008

Lecture #3

1

Course Administration

- Please download HW #1 from course homepage
  - It is due 2 weeks from today.
- This lecture:
  - R&G Chapter 3
- Next week reading:
  - R&G Chapter 4.1 ~ 4.2, 5

2

Surface Computing (MSR)

Relational Model

Chapter 3

4

Lecture Outline

- Quick intro to Relational Model
  - Definitions: schema, instance, tuple, field, domain, etc.
  - Basic SQL Commands (Data Definition Language)
  - Integration Constraints
- Map ER Diagram to Relational Tables

5

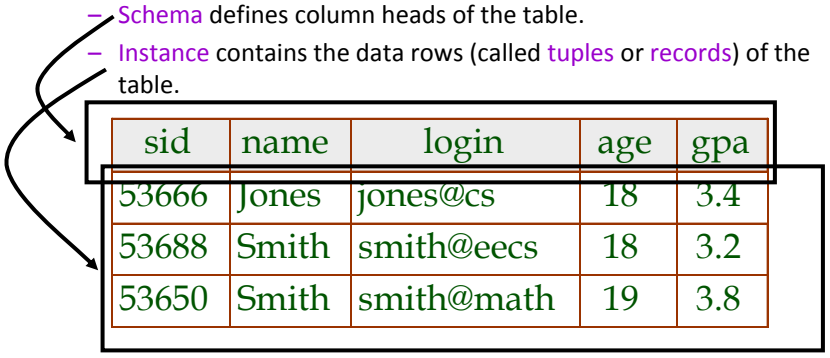
Relational Model

- Most widely used model
  - Vendors: Oracle, Microsoft, IBM (DB2), Sybase, …
- Simple
  - A relational database is a collection of relations.
  - Each relation is a table with rows and columns.
- Why do people like it?
  - Simple tabular data representation, easy to understand.
  - Ease of expressing complex query (using SQL) on the data
  - Efficient query evaluation (using query optimization)

6

Example of a Relation

- This is a "Students relation".
- A relation has two parts:
  - Schema defines column heads of the table.
  - Instance contains the data rows (called tuples or records) of the table.



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

7

Relational Schema (1)

- A relational schema specifies:
  - name of the relation: Students
  - names of fields: (sid, name, login, age, gpa)
  - domain of each field: it is type of possible values (some of built-in types in SQL are integer, real, string, and date.)
- A field can also be called an attribute or a column.

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

8

Relational Schema (2)

- We can refer to the field by
  - Field name (more common): the order of fields does not matter.
  - Position of the field (less common): the order of fields matters.
- A relational schema can be written using the following notation:
  - relation-name (field-name-1: domain-name-1, field-name-2: domain-name-2, …, field-name-n: domain-name-n)
  - Example:

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

9

Relational Instance

- A relation instance contains a set of tuples
  - A relation instance is referred to as a relation.
  - A tuple can also be called a record or a row.
  - A relation instance is a set, and it has no duplicate tuples.
  - Order of tuples is not important.

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

10

Degree and Cardinality

- Degree is the number of fields in schema (=5 in the table below)
- Cardinality is the number of tuples in relation (=3 in the table below)

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

11

Domain Constraints

- Values in the tuples' fields must satisfy the specified domain in the schema.
  - Similar to type matching in compilation
- Example of domain constraint violation:
  - Schema: Students(sid: string, name: string, login: string, age: integer, gpa: real)
  - A tuple: <sid: '50000', name: 38, login: 'dave@cs', age: 18.5, gpa: 3.3>
- There are other types of integrity constraints:
  - Key constraints, foreign key constraints, and general constraints.

12

Outline on SQL Basics

- History
- Basic commands
- Integrity constraints

13

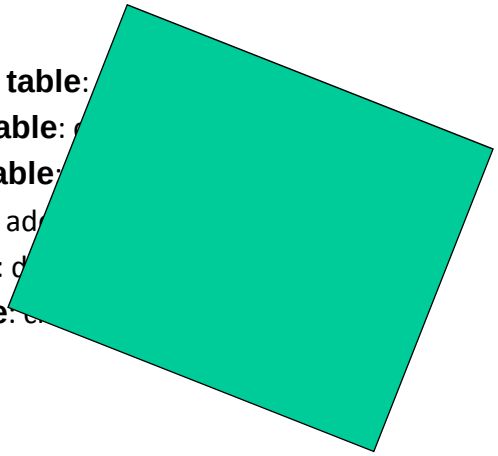
SQL History

- It is a query language for relational databases.
- Developed by IBM (system R) in the 1970s
- Need for a standard since it is used by many database vendors.
- Two standard organizations
  - ANSI (American National Standard Institutes)
  - ISO (International Organization for Standardization)
- Standards:
  - SQL-86, SQL-89, SQL-92, SQL-99 (current standard)

14

SQL Basic Commands

- **create table**:
- **drop table**:
- **alter table**:
- **insert**: ad
- **delete**: d
- **update**:



15

SQL: **create table**

create table Students (sid **char(20)**, name **char(20)**, login **char(10)**, age **integer**, gpa **real**)

- Use built-in types: integer, real, char(#)
  - Similar to type definition in programming language
  - You can define your own types (describe in CH5)

sid	name	login	age	gpa

16

SQL: **delete table**

drop table Students

17

SQL: **alter table**

- Add a new field in a table

alter table Students **add** dept char[20]

sid	name	login	age	gpa	dept

- Delete a field in a table

alter table Students **drop** gpa

sid	name	login	age	dept

18

SQL: **insert**

insert into Students (sid, name, login, age, gpa)
values ('53688', 'Smith', 'smith@cs', 18, 3.2)

or you can omit the fields

insert into Students
values ('53688', 'Smith', 'smith@cs', 18, 3.2)

sid	name	login	age	gpa
53688	Smith	Smith@cs	18	3.2

19

SQL: **delete**

delete from Students **as** S **where** S.name = 'Smith'

or you can omit **as** and the tuple variable S

delete from Students **where** name = 'Smith'

sid	name	login	age	gpa
53666	Jones	Jones@cs	18	3.3
~~53688~~	~~Smith~~	~~Smith@cs~~	~~18~~	~~3.2~~
~~53650~~	~~Smith~~	~~Smith@math~~	~~19~~	~~3.7~~

20

SQL: **update**

update Students S
  set S.age = S.age + 1, S.gpa = S.gpa – 1
  where S.sid = 53688

sid	name	login	age	gpa
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	~~18~~ 19	~~3.2~~ 2.2
53650	Smith	Smith@math	19	3.7

21

Integrity Constraints (IC)

- IC: condition that must be true for any instance of the database; e.g., domain constraints.
- Why ICs?
  - Prevent data entry errors or command errors.
- ICs are specified when schema is defined, (create table)
- ICs are checked when relations are modified (add, remove, and update).
- A legal instance of a relation is one that satisfies all specified ICs.
  - Should not allow illegal instances.

22

Types of Integrity Constraints

- Domain constraint
- Key constraint
- Foreign key constraint (referential integrity)
- Other constraints

23

Key Constraint

- A key is a set of minimal fields that can uniquely identify a tuple in a relation.
  1. No two distinct tuples can have same values in all key fields, and
  2. It is minimal, (no subset of the key is another key).
  - Part 2 false? A superkey.
  - If #key >1, one of the keys is chosen (by DBA) to be the primary key.
- Why is this a constraint?
  - When a table is modified (e.g., by adding a new tuple), DBMS checks to make sure that the specified keys remain valid keys (e.g., the new tuple has a unique key value).

24

Examples of Keys

- Valid keys: {
- Invalid keys:
- Valid Superkeys:
  age, gpa)

sid	name	login	age	gpa
50000	Dave	Dave@cs	19	3.2
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	18	3.2
53650	Smith	Smith@math	19	3.7
53831	Madayan	Madayan@music	11	1.8

25

Primary and Candidate Keys

- A relation can have many possible keys, called candidate keys. But only one is chosen as the primary key.
  - DBMS may create an index on primary key to optimize tuple lookup (using primary key).
- Specify keys in SQL:

create table Students
(sid **char(20)**, name **char(20)**, login **char(10)**, age **integer**, gpa **real,**
unique (name, age),
constraint StudentsKey **primary key** (sid))

constraint name

26

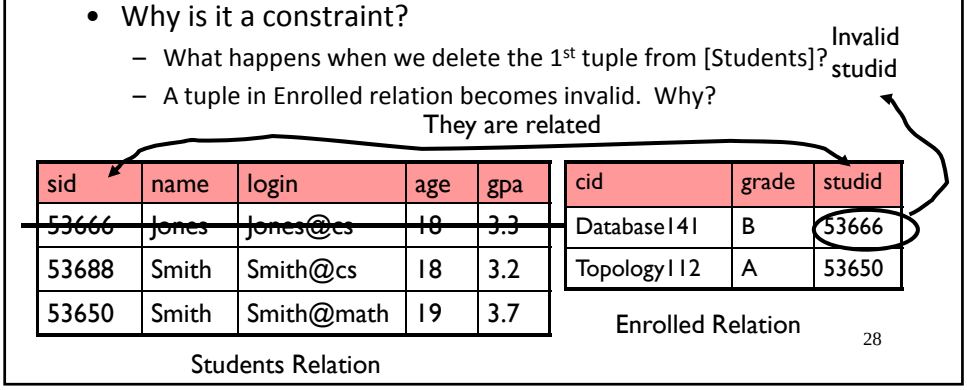
Foreign Key (Definition)

- Used when one table references another table.
- A foreign key is like a "pointer" in C, referencing a unique tuple / field in the referenced relation.
  - A foreign key constraint makes sure that there is no dangling pointer.
- Huh? Take a look at an example.

27

Foreign Key Constraint

- Specify constraint:
  - Only the Students listed in the Students relation can enroll for courses.
  - [studid] in [Enrolled] refer to some fields [sid] in [Students]
- Why is it a constraint?
  - What happens when we delete the 1st tuple from [Students]?
  - A tuple in Enrolled relation becomes invalid. Why?



sid	name	login	age	gpa
~~53666~~	~~Jones~~	~~Jones@cs~~	~~18~~	~~3.3~~
53688	Smith	Smith@cs	18	3.2
53650	Smith	Smith@math	19	3.7

Students Relation

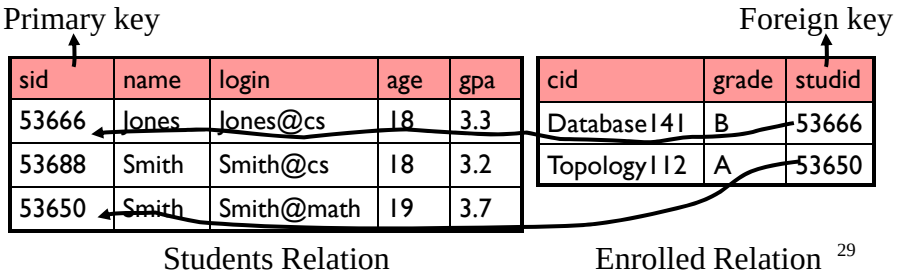
cid	grade	studid
Database141	B	53666
Topology112	A	53650

Enrolled Relation

28

Foreign Key (Definition)

- Studid is called a foreign key.
- A foreign key is like a "pointer" in C, referencing a unique tuple / field in the referenced relation.
  - A foreign key constraint makes sure that there is no dangling pointer.



sid	name	login	age	gpa
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	18	3.2
53650	Smith	Smith@math	19	3.7

Students Relation

cid	grade	studid
Database141	B	53666
Topology112	A	53650

Enrolled Relation

29

More on Foreign Key

- The foreign key must refer to primary key in the referenced relation. Why?
  - Must be able to uniquely identify the tuple in the referenced relation.
- The foreign key needs not be a candidate key. Why?
  - Only used to uniquely identify a tuple in the referenced relation.
- If all foreign key constraints are enforced, referential integrity is achieved.
  - Think of a common example w/o referential integrity?
    - Bad links in HTML

30

Specify Foreign Keys in SQL

- Constraint: only students listed in the Students relation should be allowed to enroll for courses.

```
create table Enrolled
  (studid char(20),  cid char(20),  grade char(20),
    primary key  (studid, cid),
    foreign key (studid) references Students)
```

Primary key

sid	name	login	age	gpa
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	18	3.2
53650	Smith	Smith@math	19	3.7

Students Relation

Foreign key

cid	grade	studid
Database141	B	53666
Topology112	A	53650

Enrolled Relation

31

Foreign Key Constraint Violations

- When can they happen - delete? insert? update?

Primary key

sid	name	login	age	gpa
~~53666~~	~~Jones~~	~~Jones@cs~~	~~18~~	~~3.3~~
53688	Smith	Smith@cs	18	3.2
53650	Smith	Smith@math	19	3.7

sid	name	login	age	gpa
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	18	3.2
~~53650~~	Smith	Smith@math	19	3.7

54321

Foreign key

cid	grade	studid
Database141	B	53666
Topology112	A	53650

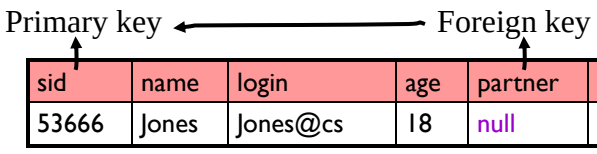
cid	grade	studid
Database141	B	53666
Topology112	A	53650
History 123	B	12345

insert

32

Self-Referral Foreign Key

- A foreign key can refer to the same relation.
- Example: each student could have a partner.
  - If a student hasn't found a partner, the value can be set to null.
  - It is ok to have null value in foreign key field.
    - But is it okay to have null value in primary key field?



sid	name	login	age	partner	gpa
53666	Jones	Jones@cs	18	null	3.3

33

General Constraints

- An example : students ages must be over 16 years old.

```
create table Students (
    sid char(20),
    name char(20),
    login char(10),
    age integer,
    gpa real,
    unique (name, age),
    constraint StudentsKey primary key (sid),
    check (age > 16)
)
```

34

More on General Constraints

- Two types
  - Table constraint: associate with a single table
  - Assertions: associate with multiple tables
    - if 1/2 of courses taken by a student has a grade = F in the Enrolled relation, the status of the student in the Students relation must be set to "In Trouble".

35

Enforcing Referential Integrity

- 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 while leaving a dangling enrolled tuple? (at least 4 possibilities)
  - Option 1: Also delete all Enrolled tuples that refer to it
  - Option 2: Disallow deletion
  - Option 3: Set studid in Enrolled tuples that refer to it to a default sid.
  - Option 4: Set studid in Enrolled tuples that refer to it to NULL.

Primary key

sid	name	login	age	gpa
53666	Jones	Jones@cs	18	3.3
53688	Smith	Smith@cs	18	3.2
53650	Smith	Smith@math	19	3.7

Students Relation

Foreign key

cid	grade	studid
Database141	B	53666
Topology112	A	53650

Enrolled Relation

36

Referential Integrity in SQL

- Option 1: CASCADE (also delete all tuples that refer to deleted tuple)
- Option 2: Default is NO ACTION (delete/update is rejected)
- Options ¾: SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
  (studid CHAR(20) default "00000",
   cid CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid,cid),
   FOREIGN KEY (sid)
     REFERENCES Students
     ON DELETE CASCADE // Option 1
     ON UPDATE SET DEFAULT // Option 4
     ON UPDATE SET NULL // Option 3
)
```

37

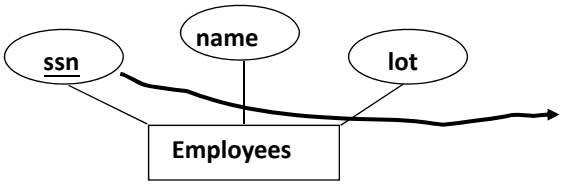
Translate ER Model to Relational Model

- An entity set to table(s)
- A relationship set without constraints to table(s)
- A relationship set with only key constraints to table(s)
- A relationship set with participation constraints to table(s)
- A weak entity set to table(s)
- ISA hierarchies to table(s)
- Aggregates to table(s)

38

Entity Sets to Tables

key attribute attributes



CREATE TABLE Employees

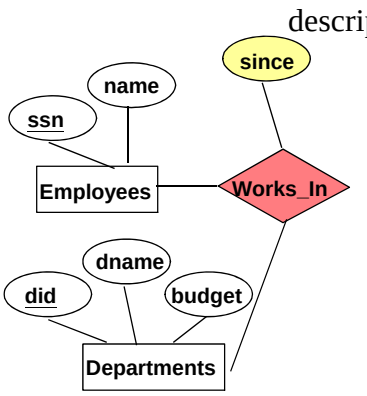
39

Translate ER Model to Relational Model

- An entity set to table(s)
- A relationship set without constraints to table(s)
- A relationship set with only key constraints to table(s)
- A relationship set with participation constraints to table(s)
- A weak entity set to table(s)
- ISA hierarchies to table(s)
- Aggregates to table(s)

40

Relationship Sets (without Constraints) to Tables



CREATE TABLE Works_In(

PRIMARY KEY
FOREIGN KEY (ssn)
REFERENCES
FOREIGN KEY (did)
REFERENCES

41

Relationship Sets to Tables

- Fields (attributes) of a table must include:
  - All descriptive attributes.
  - Keys for each participating entity set (as foreign keys).

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

42

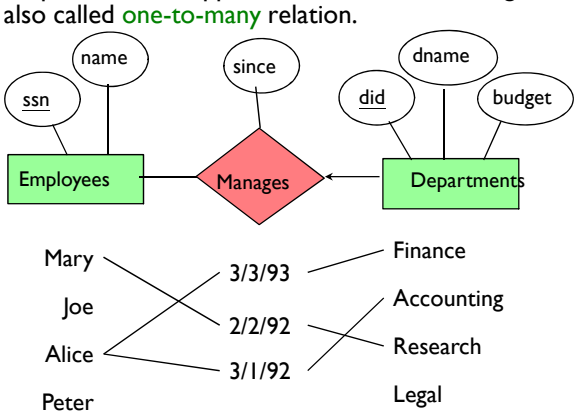
Translate ER Model to Relational Model

- An entity set to table(s)
- A relationship set without constraints to table(s)
- A relationship set with only key constraints to table(s)
- A relationship set with participation constraints to table(s)
- A weak entity set to table(s)
- ISA hierarchies to table(s)
- Aggregates to table(s)

43

Review: ER Key Constraints

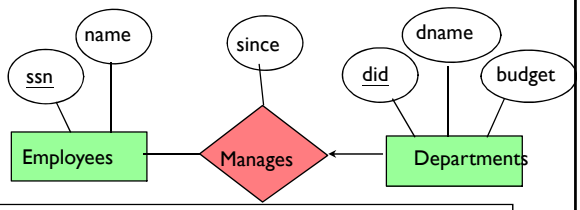
- Describe at most once (entitity) relationship
  - Manages relationship: each department has at most one manager (okay to have none).
  - One department can appear at most once in Manages relationship set, also called one-to-many relation.



44

Relationship Sets (with key Constraints) to Table

- Note that did is the key now! Why?
- Is there another way to map this?

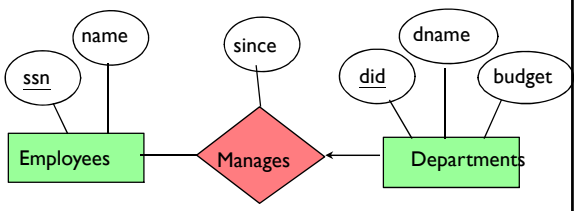



```
CREATE TABLE Manages (
  [                    ]
  PRIMARY KEY [        ]
  FOREIGN KEY (ssn) REFERENCES
  FOREIGN KEY (did) REFERENCES
```

45

Relationship Sets (with key Constraints) to Table

- Since each department has a unique manager, we could instead combine Manages and Departments.
  - Map Manages into the Departments table.




```
CREATE TABLE Dept_Mgr(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  ssn CHAR(11),   // can be null -> at most one
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees)
```

46

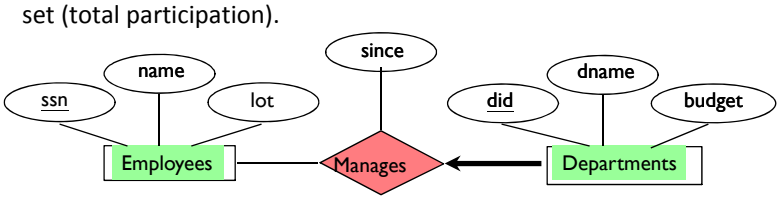
Translate ER Model to Relational Model

- An entity set to table(s)
- A relationship set without constraints to table(s)
- A relationship set with only key constraints to table(s)
- A relationship set with participation constraints to table(s)
- A weak entity set to table(s)
- ISA hierarchies to table(s)
- Aggregates to table(s)

47

Review: Participation Constraints

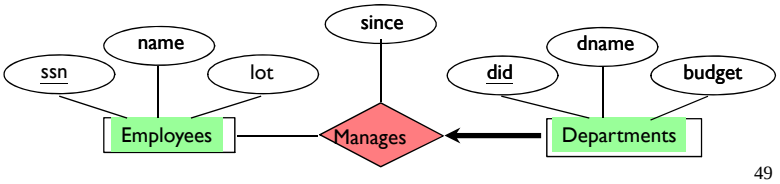
- Describe all (entitity) participation relationship
  - Must every department have a manager?
    - If yes, this is a participation constraint
  - All Departments entities must participate in the Manages relationship set (total participation).



48

Participation Constraints to Table

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



49

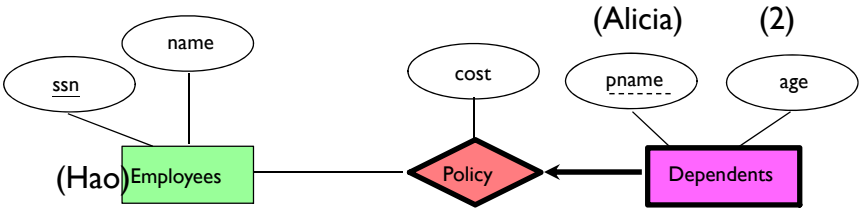
Translate ER Model to Relational Model

- An entity set to table(s)
- A relationship set without constraints to table(s)
- A relationship set with only key constraints to table(s)
- A relationship set with participation constraints to table(s)
- A weak entity set to table(s)
- ISA hierarchies to table(s)
- Aggregates to table(s)

50

Review: Weak Entities

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

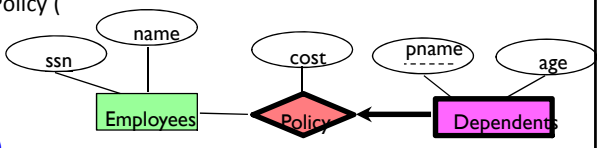


51

Weak Entity Sets to Table

- 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  Dependent_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)
```



52

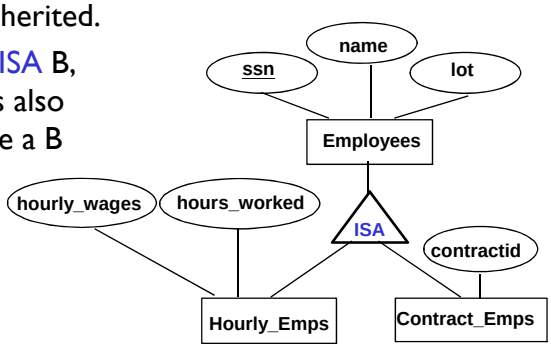
Translate ER Model to Relational Model

- An entity set to table(s)
- A relationship set without constraints to table(s)
- A relationship set with only key constraints to table(s)
- A relationship set with participation constraints to table(s)
- A weak entity set to table(s)
- ISA hierarchies to table(s)
- Aggregates to table(s)

53

Review: ISA Hierarchies

- As in C++, or other PLs, attributes are inherited.
- If we declare A ISA B, every A entity is also considered to be a B entity.



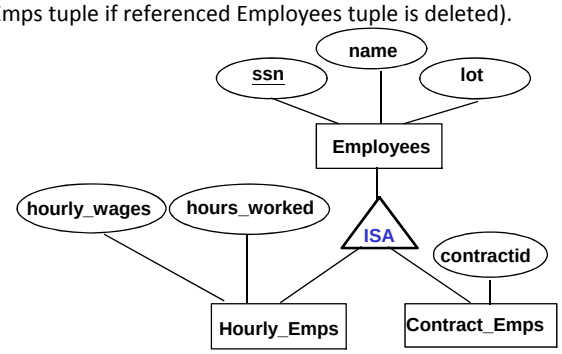
54

ISA Hierarchies to Tables

- General approach:
  - 3 tables: 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).

```
CREATE TABLE employees (
   ssn  CHAR(11),
   name  CHAR(20),
   lot  INTEGER,
  PRIMARY KEY  (ssn))

CREATE TABLE hourly_emps (
  hourly_wages  INTEGER,
  hours_worked  INTEGER,
  ssn  CHAR(11),
  PRIMARY KEY (ssn)
  FOREIGN KEY  (ssn) REFERNECES employees ON DELETE
CASCADE)
```



55

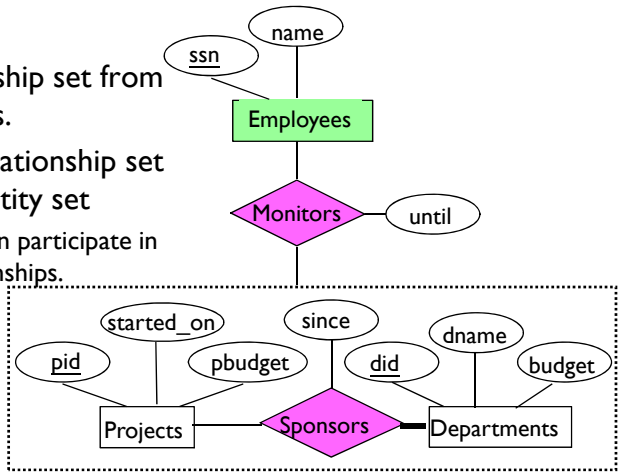
Translate ER Model to Relational Model

- An entity set to table(s)
- A relationship set without constraints to table(s)
- A relationship set with only key constraints to table(s)
- A relationship set with participation constraints to table(s)
- A weak entity set to table(s)
- ISA hierarchies to table(s)
- Aggregates to table(s)

56

Review: Aggregation

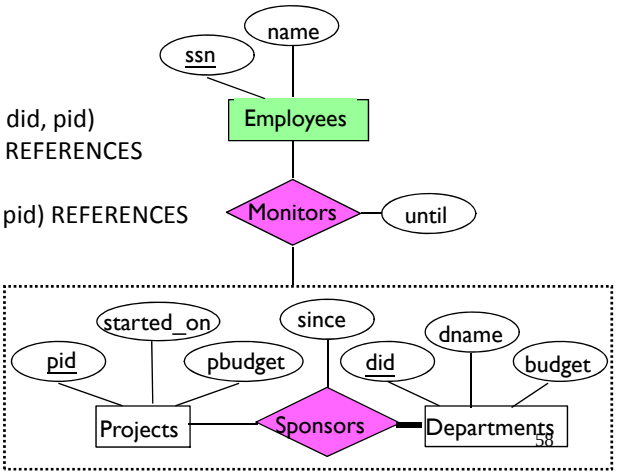
- Create relationship set from relationship sets.
- Aggregation: relationship set turns into an entity set
  - So that they can participate in (other) relationships.



57

Aggregation to Tables

```
CREATE TABLE monitors (
  ssn      CHAR(11),
  until    DATE,
  did      INTEGER,
  pid      INTEGER,
  PRIMARY KEY  (ssn, did, pid)
  FOREIGN KEY  (ssn) REFERENCES
     Employees
  FOREIGN KEY  (did, pid) REFERENCES
     Sponsors
)
```



58

Views

- A view is just a relation, but we only store its definition, rather than its tuples/rows in database.

```
CREATE  VIEW  StudentsInHistory105(name, sid)
    AS  SELECT   S.name, S.sid
    FROM  Students S, Enrolled E
    WHERE  S.sid = E.studid and E.cid='History105'
```

- 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.

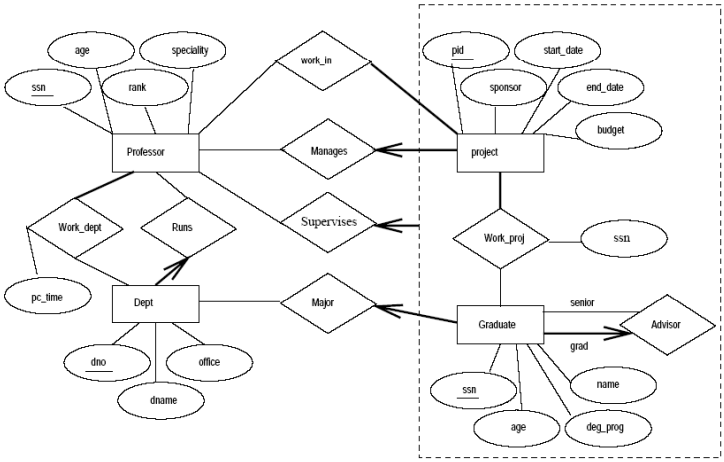
59

Views and Security

- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
- Example: a student can use the view StudentsInHistory105 to find out his/her classmates.
- But a student cannot find out the gpa of his/her classmates.

60

Can you translate this ER into Tables?



61

Summary

- Relational model is about tabular representation of data.
  - Simple and intuitive, currently the most widely used.
- Integrity constraints
  - Domain constraints, key constraints, foreign key constraints, general constraints
- Basic SQL commands
  - Create, update, and delete tables
  - Insert and delete tuples
- Translate ER to relational model
- Define views for security

62