Database Systems
(資料庫系統)

September 20, 2004

Lecture #2

By Hao-hua Chu (朱浩華)

1

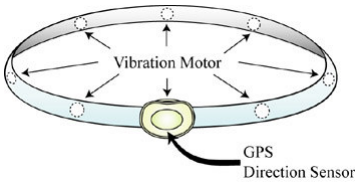
Course Administration

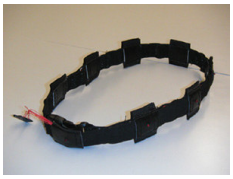
- Can everyone get the textbook?
- HW #1 is out on the course homepage
  - It is due one week from today.
- Next week reading:
  - R&G Chapters 4.1~4.2 & 5

2

Ubicomp Project of the Week: ActiveBelt (Keio University)

- Obtain directional information with tactile (touch) display.
- Why is better than visual & audio displays?
  - Free your eyes & ears
  - Won't look lost
  - Better direction





3

Chapter 2
Introduction to Database Design

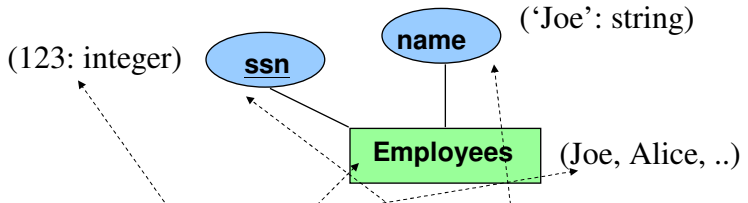
4

Database Design

- Step 1: Requirements Analysis
  - Discuss what data to store in the database.
  - Discuss what application (e.g., queries, updates, ..) needs from the database.
- Step 2: Conceptual Database Design
  - Come up with the design
  - The **Entity-Relation (ER) model** is the most popular data model.
  - The E/R model allows us to sketch the design of a database informally using pictures called *entity-relationship diagrams*.
- Step 3: Logical Database Design
  - Implement the design
  - The **relational data model** is the most popular data model.
  - Easy to map ER diagrams into logical schema (in the relational data model) -> CH 3.

5

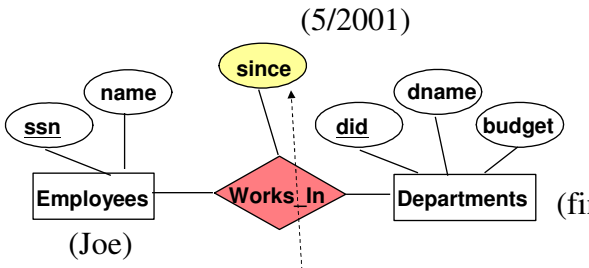
ER Model: Entity



- Proposed by Peter Chen (BS NTU EE '68) in 1976.
- **Entity**: Real-world object distinguishable from other objects (e.g., Joe).
- An entity is described by a set of **attributes**.
  - Each attribute has a **domain** of possible values.
- **Entity Set**: a collection of similar entities
- Each entity in an entity set is uniquely identified by a **key** attribute.

6

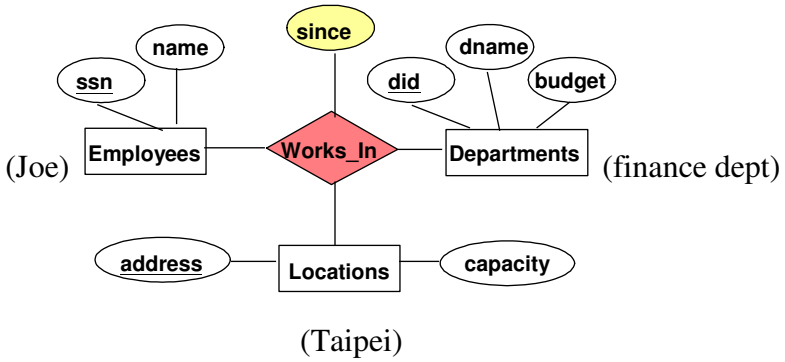
ER Model: Relationship



- **Relationship**: Association among two or more entities
  - Joe works in finance department.
- A relationship can have **description attributes**.
  - Joe has worked in finance department since 5/2001.
- **Relationship Set:** Collection of similar relationships.

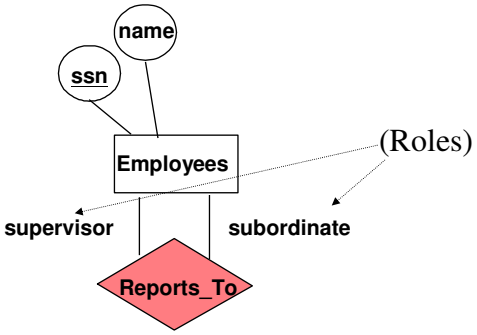
7

Ternary Relationship



8

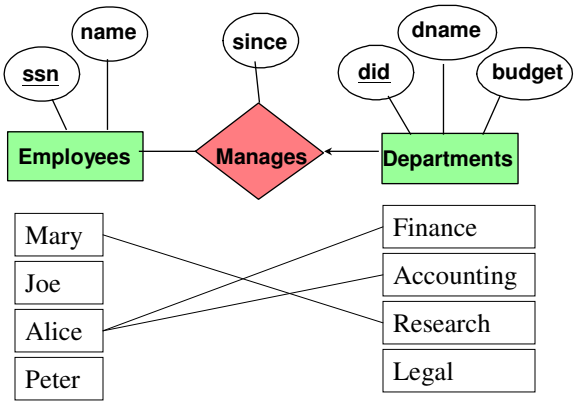
Roles in Relationship



9

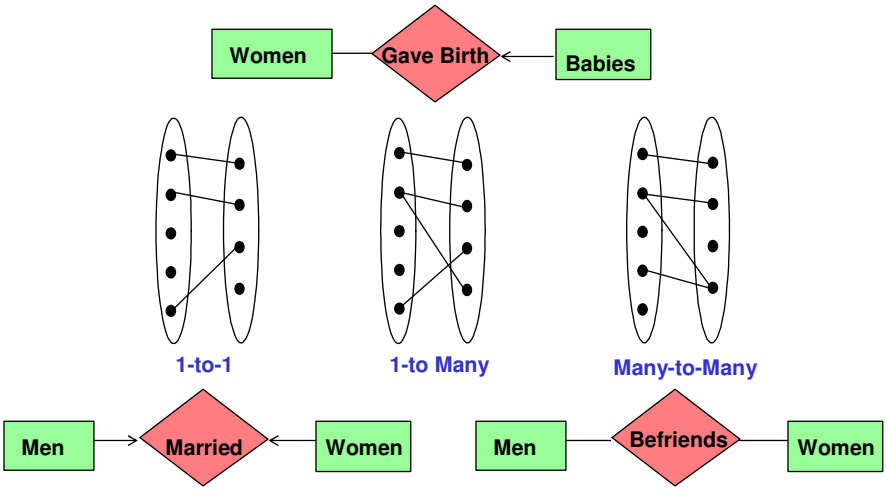
Key Constraints

- It can describe Manages relationship that each department has at most one manager.
  - One department can appear **at most once** in Manages relationship set, also called **one-to-many** relation.



10

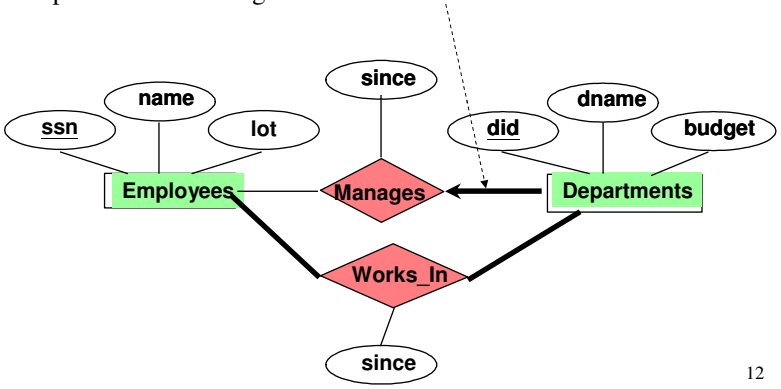
More Key Constraints



11

Participation Constraints

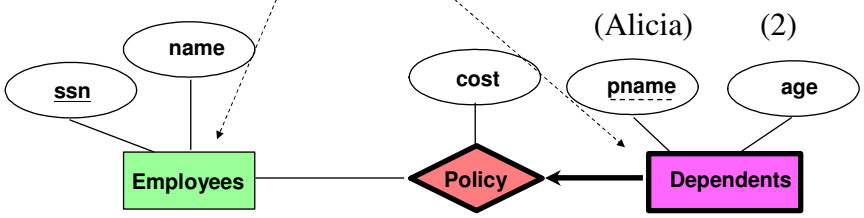
- Must every department have a manager?
  - If yes, this is a **participation constraint**: the participation of Departments in Manages is said to be **total**.



12

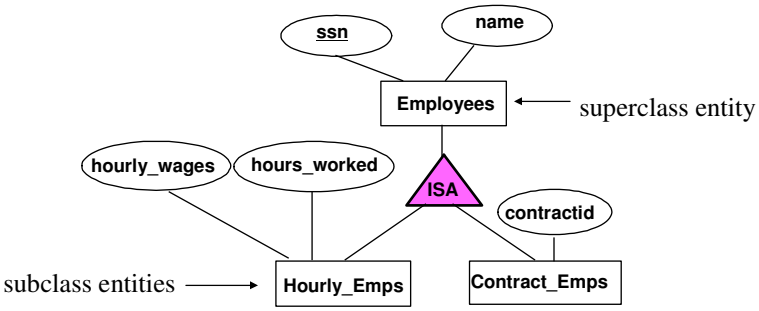
Weak Entities

- 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.



13

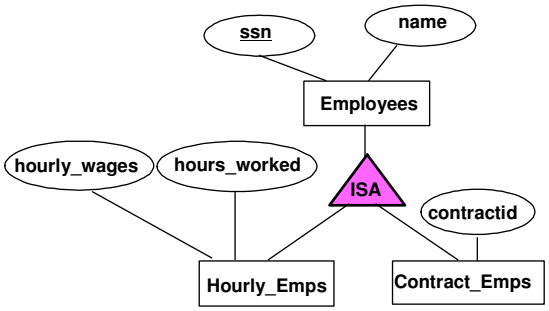
ISA (`is a') Hierarchies



- As in C++ and OO languages, attributes are inherited from superclass. If we declare A **ISA** B, every A entity is also considered to be a B entity.
- Reason for using ISA:
  - Add descriptive attributes specific (make sense) to a subclass.
  - Identify entities that participate in a relationship.

14

ISA (`is a') Constraints

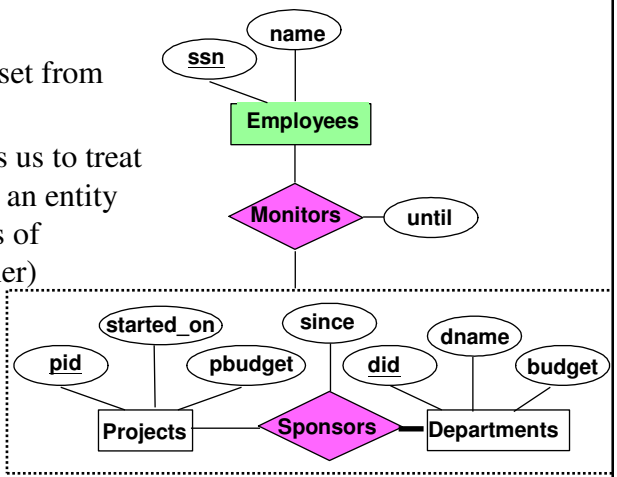


- **Overlap constraints:** Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (**Allowed/disallowed**)
- **Covering constraints**: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? **(Yes/no)**

15

Aggregation

- Create relationship set from relationship sets.
- **Aggregation** allows us to treat a relationship set as an entity set, for the purposes of participation in (other) relationships.



16

Design Techniques

1. Avoid redundancy.
2. Don't use an entity set when an attribute will do.
3. Limit the use of weak entity sets.

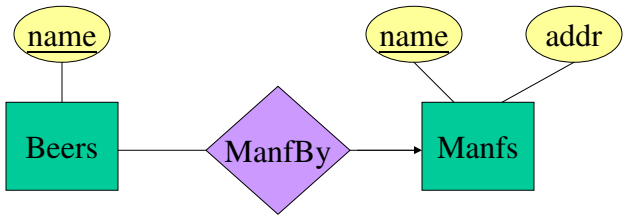
17

Avoiding Redundancy

- Redundancy occurs when we say the same thing in two different ways.
- Redundancy wastes space and (more importantly) encourages inconsistency.
  - The two instances of the same fact may become inconsistent if we change one and forget to change the other, related version.

18

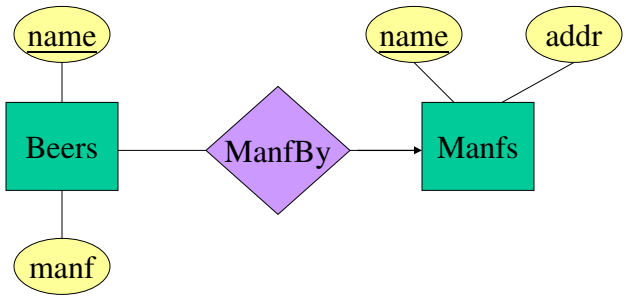
Example: Good



This design gives the address of each manufacturer exactly once.

19

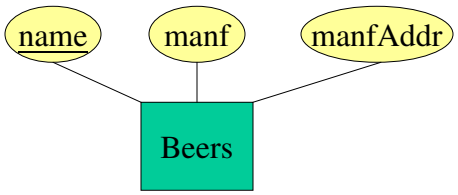
Example: Bad



This design states the manufacturer of a beer twice: as an attribute and as a related entity.

20

Example: Bad



This design repeats the manufacturer's address once for each beer; loses the address if there are temporarily no beers for a manufacturer.

21

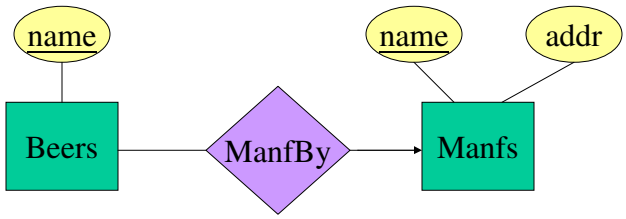
Entity Sets Versus Attributes

- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one nonkey attribute.

    or
  - It is the "many" in a many-one or many-many relationship.

22

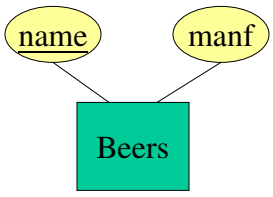
Example: Good



- *Manfs* deserves to be an entity set because of the nonkey attribute *addr*.
- *Beers* deserves to be an entity set because it is the "many" of the many-one relationship *ManfBy*.

23

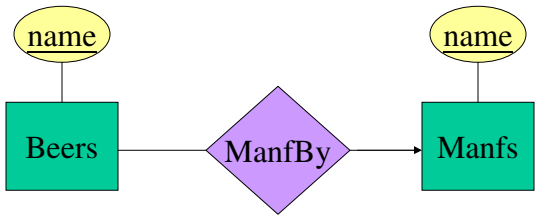
Example: Good



There is no need to make the manufacturer an entity set, because we record nothing about manufacturers besides their name.

24

Example: Bad



Since the manufacturer is nothing but a name, and is not at the "many" end of any relationship, it should not be an entity set.

25

Don't Overuse Weak Entity Sets

- Beginning database designers often doubt that anything could be a key by itself.
  - They make all entity sets weak, supported by all other entity sets to which they are linked.
- In reality, we usually create unique ID's for entity sets.
  - Examples include social-security numbers, automobile VIN's etc.

26

When Do We Need Weak Entity Sets?

- The usual reason is that there is no global authority capable of creating unique ID′s.
- Example: it is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world.

27

Summary

- ER model is popular for conceptual design
  - Sketch the design of a database informally using pictures
- Basic constructs in ER model: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Some additional constructs:
  - *weak entities*, *ISA hierarchies*, and *aggregation*.
- Several kinds of integrity constraints:
  - *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies.
- Design Techniques in ER model

28

Relational Model

Chapter 3

29

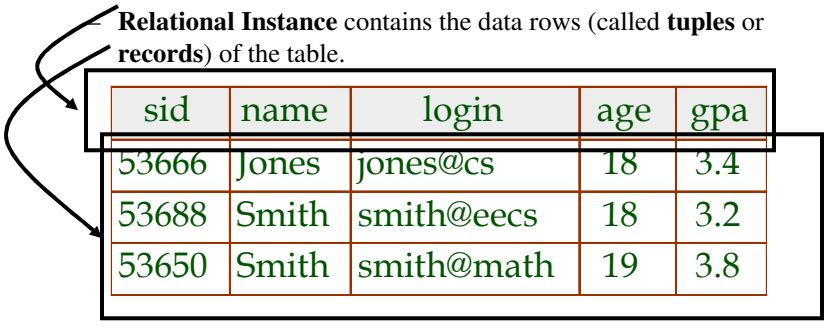
Relational Model

- It is the most widely used model.
  - Vendors: Oracle, Microsoft, IBM (DB2), Sybase, …
- It is simple and elegant.
  - A relational database is a collection of **relations.**
  - Each relation is represented as a **table** of 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)

30

Example of a Relation

- This is a "Students relation".
- A relation has two parts:
  - **Relational Schema** defines column heads of the table.
  - **Relational 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

31

Relational Schema (1)

- A relational schema specifies:
  - name of the relation: Students
  - names of **fields**: (sid, name, login, age, and 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

32

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)

33

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

34

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

35

Domain Constraints

- Values in the tuple 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**.

36

Outline on SQL Basics

- History
- Basic commands
- Integrity constraints

37

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)

38

SQL Basic Commands

- **create table**: create a table
- **drop table**: delete a table
- **alter table**: alter a field in a table
- **insert**: add a tuple
- **delete**: delete a tuple
- **update**: change field values in a tuple

39

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

40

SQL: **delete table**

drop table Students

41

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

42

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

43

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~~

44

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

45

Integrity Constraints (IC)

- **IC:** condition that must be true for **any** instance of the database; e.g., domain constraints.
  - 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.
  - DBMS should not allow illegal instances.
- DBMS checks ICs to prevent data entry errors, or command errors.

46

Types of Integrity Constraints

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

47

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).

48

Examples of Keys

- Valid keys: {sid}, {name, age} and {login}
- Invalid keys: {name} and {age}
- Valid Superkeys: {sid, gpa}, {sid, age}, and {sid, name, login, 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

49

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:

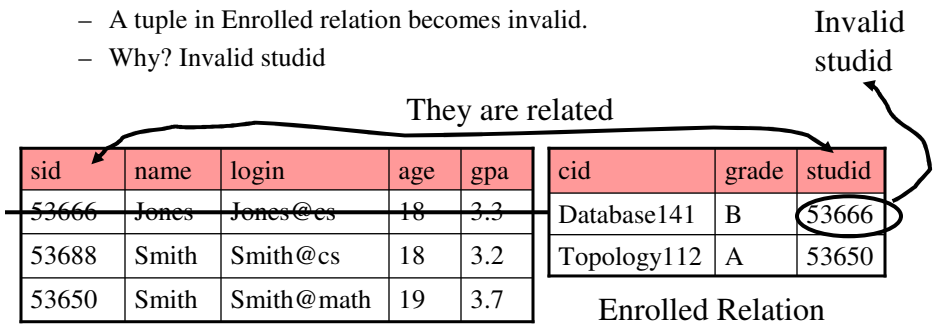
create table Students
(sid **char(20)**, name **char(20)**, login **char(10)**, age **integer**, gpa **real,**
unique (name, age),
constraint StudentsKey **primary key** (sid))

constraint name

50

Foreign Key Constraint

- Suppose some fields in one relation refer to some fields in another relation: studid in Enrolled -> sid in Students.
- Example: delete a tuple from Students
  - A tuple in Enrolled relation becomes invalid.
  - Why? Invalid studid



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

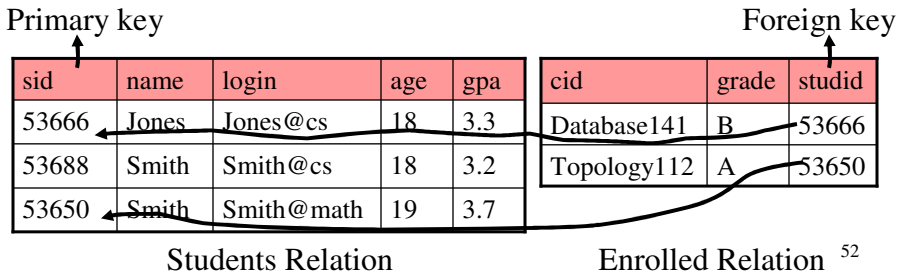
Students Relation

cid	grade	studid
Database141	B	53666
Topology112	A	53650

Enrolled Relation

Foreign Key

- 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

More on Foreign Key

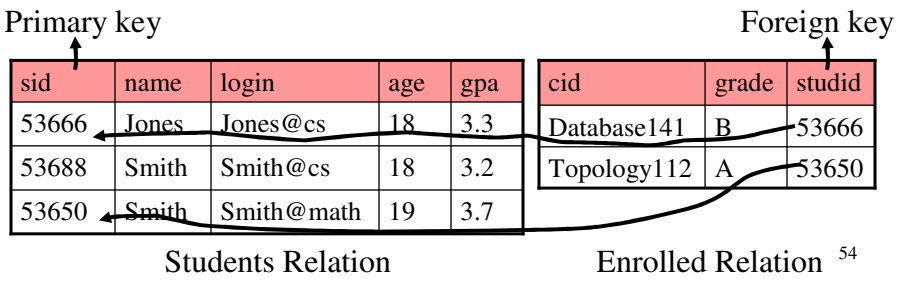
- The foreign key must refer to primary key in the referenced relation
  - Why? Foreign key is used to create links between tuples in two relations. Like a memory pointer, it must be able to uniquely identify the tuple in the referenced relation.
- The foreign key needs not be a candidate key.
  - Why? It is only used to uniquely identify a tuple in the referenced relation.
- If all foreign key constraints are enforced, **referential integrity** is achieved, i.e., no dangling pointers.
- Can you name a data model w/o referential integrity?
  - Bad links in HTML!

53

Specify Foreign Keys in SQL

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

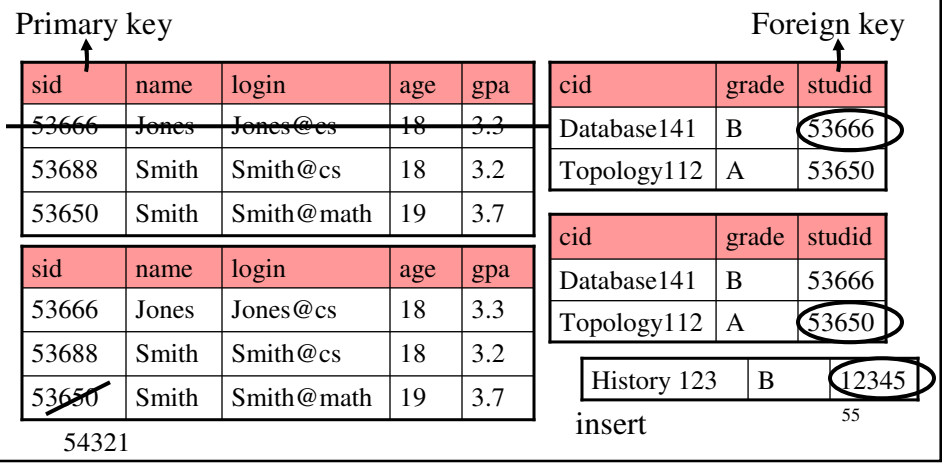
```
create table Enrolled
  (studid char(20),  cid char(20),  grade char(20),
    primary key  (studid, cid),
    foreign key (studid) references Students)
```



54

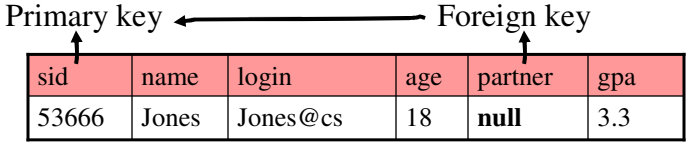
Foreign Key Constraint Violations

- They can happen in **delete**, **insert** and **update** commands.



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.
  - It is NOT ok to have null value in primary key field.



56

General Constraints

- An example of a general constraint: say 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**))

- Two types of general constraints:
  - Table constraint: associate with a single table (above example)
  - Assertions: associate with multiple tables (if 1/3 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".)

57

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?
  - Option 1: Also delete all Enrolled tuples that refer to it.
  - Option 2: Disallow deletion of a Students tuple that is referred to.
  - Option 3: Set studid in Enrolled tuples that refer to it to a *default sid*.
  - Option 4: (In SQL, also: Set studid in Enrolled tuples that refer to it to a special value *null*, denoting *`unknown'* or *`inapplicable'*.)
- Similar if primary key of Students tuple is updated.

58

Referential Integrity in SQL

- Option 1: Default is NO ACTION (delete/update is rejected)
- Option 2: CASCADE (also delete all tuples that refer to deleted tuple)
- 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
   ON UPDATE SET DEFAULT )
```

59

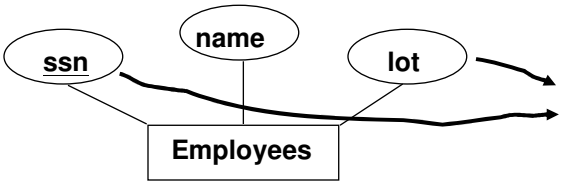
Translate ER Model to Relational Model

- An entity set to a table
- A relationship set without constraints to a table
- A relationship set with only key constraints to a table
- A relationship set with participation constraints to a table
- A weak entity set to a table
- ISA hierarchies to a table

60

Entity Sets to Tables

key attribute attributes

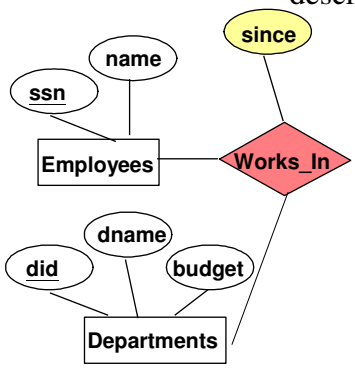



```
CREATE TABLE Employees
    (ssn CHAR(11),
    name CHAR(20),
    lot  INTEGER,
    PRIMARY KEY  (ssn))
```

61

Relationship Sets (without Constraints) to Tables

descriptive attribute




```
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)
```

62

Relationship Sets to Tables

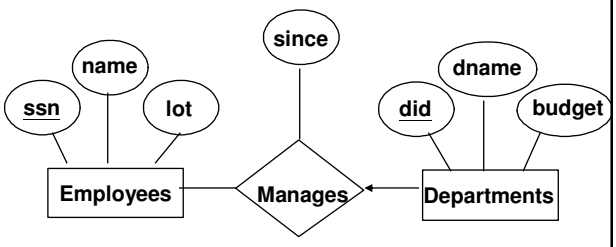
- Fields (attributes) of a table must include:
  - Keys for each participating entity set (as **foreign keys**).
    - This set of attributes forms a **superkey** for the relation.
  - All descriptive attributes.

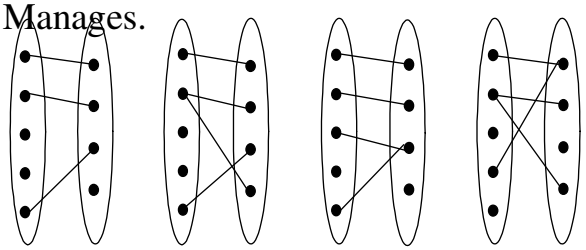
```
CREATE TABLE Works_In(
  ssn  CHAR(11),
  did  INTEGER,
  since  DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn)
     REFERENCES Employees,
  FOREIGN KEY (did)
     REFERENCES Departments)
```

63

Review: Key Constraints

- Each dept has at most one manager, according to the *key constraint* on Manages.





Translation to relational model?

64

Relationship Sets (with key Constraints) to Tables

- Map a relationship set to a table:
  - Note that did is the key now! Why?
  - Since each department has a unique manager, we could instead combine Manages and Departments.
- Second approach:
  - Map Manages into the Departments table.

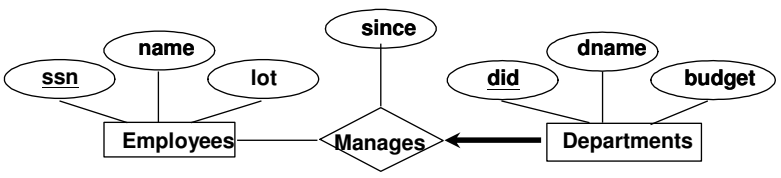
```
CREATE TABLE Manages(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  FOREIGN KEY (did) REFERENCES Departments)
```

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

65

Review: Participation Constraints

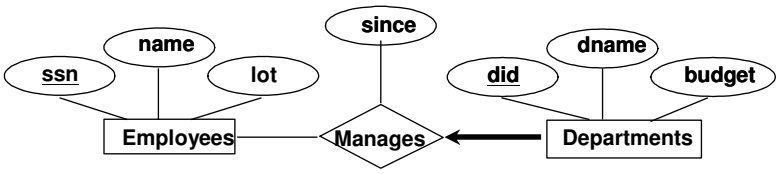
- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).
    - Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)



66

Participation Constraints in SQL

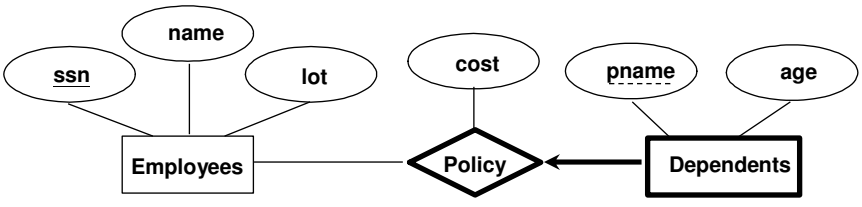
```
CREATE TABLE  Dept_Mgr(
  did  INTEGER;
  dname  CHAR(20),
  budget  REAL,
  ssn  CHAR(11) NOT NULL,  // must have one!
  since  DATE,
  PRIMARY KEY  (did),
  FOREIGN KEY  (ssn) REFERENCES Employees)
```



67

Review: Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - One-to-many relationship set (1 owner, many weak entities).
  - Total participation in this *identifying* relationship set.



68

Translating Weak Entity Sets

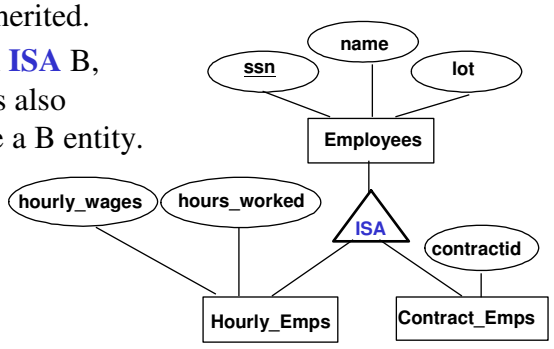
- Weak entity set and identifying relationship set are translated into a single table.
  - When the owner entity is deleted, all owned weak entities must also be deleted.

```
CREATE TABLE  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)
```

69

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.



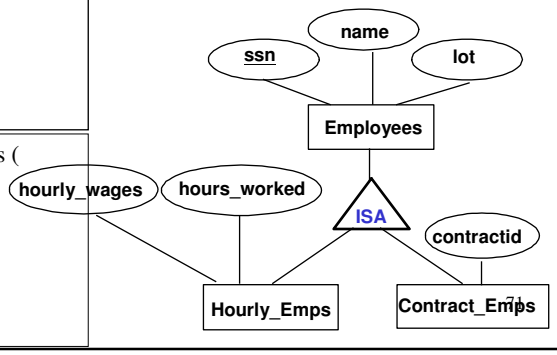
70

Translating 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)
```



71

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.

72

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.

73

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
- Rules to translate ER to relational model
- Define views for security

74