Information Policy

Database

C04. Relational Model

- Code: 164323-03
- Course: Information Policy
- Period: Spring 2013
- Professor: Sync Sangwon Lee, Ph. D

2013-1-WKU-IP-C04 Database / Relational Model

Contents

01. Relational Model

- Relational Model
  - "Legacy systems" in older models
    - Hierarchical model, Network model, …
  - A most widely used model
    - Vendors: IBM, Informix, Microsoft, Oracle, Sybase, etc.
  - Recent competitor: object-oriented model
    - ObjectStore, Versant, Ontos
  - A synthesis emerging: object-relational model
    - Informix Universal Server, UniSQL, O2, Oracle, DB2

01. Relational Model

- Relational Database:
  - a set of relations
- Relation: made up of 2 parts
  - Instance:
    - table with rows and columns
    - set of tuples
    - cf.
      - cardinality: the number of tuples
      - degree: the number of fields
  - Schema:
    - description of
      - relation name
      - field (or column or attribute) name
      - domain

01. Relational Model

- 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)
- Domain constraints:
  - type of field
  - constraints on values of field
- Can think of a relation as a set of rows or tuples (i.e., all rows are distinct)!

01. Relational Model

- Ex. Instance of Students Relation
  - Cardinality = 6
  - Degree = 5
  - All rows are distinct.

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Mada	mada@music	11	1.8
53832	Guldu	guldu@music	12	2.0

02. Relational Querying Languages

- 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.
  - The key: precise semantics for relational queries.
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

02. Relational Querying Languages

- The SQL Query Language
  - SQL [Sequel or S.Q.L]
    - is used as a standard language for dialogue.
    - can get or update data from data.
    - SQL is a standardized by ANSI and ISO.
    - Standards:
      - SQL-86, SQL-89, SQL-92, SQL-99, SQL-2000, …
  - Developed by IBM (system R) in the 1970s
  - Need for a standard since it is used by many vendors

02. Relational Querying Languages

- The SQL Query Language
  - To find all 18 year old students, we can write:

```
SELECT *
FROM    Students S
WHERE S.age=18
```

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Mada	mada@music	11	1.8
53832	Guldu	guldu@music	12	2.0

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

02. Relational Querying Languages

- The SQL Query Language
  - To find just names and logins, replace the first line:

```
SELECT S.name, S.login
FROM    Students S
WHERE S.age=18
```

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Mada	mada@music	11	1.8
53832	Guldu	guldu@music	12	2.0

02. Relational Querying Languages

- Querying Multiple Relations
  - What does the following query compute?

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Mada	mada@music	11	1.8
53832	Guldu	guldu@music	12	2.0

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

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

S.name	E.cid
Smith	Topology112

02. Relational Querying Languages

- Creating Relations in SQL
  - Creates the Students relation.
    - Observe that the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

```
CREATE TABLE Students (
        sid:    CHAR(20),
        name:   CHAR(20),
        login:  CHAR(10),
        age:    INTEGER,
        gpa:    REAL )
```

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Mada	mada@music	11	1.8
53832	Guldu	guldu@music	12	2.0

02. Relational Querying Languages

- Creating Relations in SQL
  - As another example,
    - the Enrolled table holds information about courses that students take.

```
CREATE TABLE Enrolled (
        sid:      CHAR(20),
        cid:      CHAR(20),
        grade:    CHAR(2))
```

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

02. Relational Querying Languages

- Modifying Relations

sid	name	login	age	gpa
53831	Mada	mada@music	11	1.7
53832	Guldu	guldu@music	12	1.9

```
UPDATE Students S
        SET      S.gpa = S.gpa – 0.1
        WHERE S.age <= 12
```

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Mada	mada@music	11	1.8
53832	Guldu	guldu@music	12	2.0

02. Relational Querying Languages

- Destroying and Altering Relations
  - Destroys the relation Students. The schema information and the tuples are deleted.
  - The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a null value in the new field.

```
DROP TABLE  Students

ALTER TABLE  Students
        ADD COLUMN firstYear: integer
        ADD COLUMN maiden-name CHAR(10)
```


02. Relational Querying Languages

- Adding/Deleting Tuples
  - Can insert a single tuple using:
  - Can delete all tuples satisfying some condition (e.g., name = Smith):

```
INSERT
        INTO Students (sid, name, login, age, gpa)
        VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)

DELETE
        FROM    Students S
        WHERE S.name = 'Smith'
```

03. Integrity Constraints

- IC: condition that must be true for any instance of the database;
  - e.g., domain constraints.
- The time for description and execution
  - When defining schema
  - When trying to update data violating IC
- A legal instance of a relation is one that satisfies all specified ICs
- If the DBMS checks ICs,
  - stored data is more faithful to real-world meaning.

2013-1-WKU-IP-C04 Database / Relational Model

03. Integrity Constraints

- Primary Key Constraints
  - A set of fields is a key for a relation if
    - no two distinct tuples can have same values in all key fields.
  - If there's one or more key for a relation, one of the keys is chosen (by DBA) to be the primary key.
    - candidate keys {sid}, {login} ➔ primary key {sid}
  - E.g.
    - The sid is a key for Students. (What about name?)
    - The set {sid, gpa} is a superkey.

03. Integrity Constraints

- Primary and Candidate Keys in SQL
  - Possibly many candidate keys (specified using UNIQUE), one of which is chosen as the primary key.
  - Given a given student and course, there is a single grade? vs. Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade.
  - Used carelessly, an IC can prevent the storage of database instances that arise in practice!

03. Integrity Constraints

- Primary and Candidate Keys in SQL

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

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

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

03. Integrity Constraints

- Primary and Candidate Keys in SQL

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Mada	mada@music	11	1.8
53832	Guldu	guldu@music	12	2.0

```
CREATE TABLE  Students (
        sid       CHAR(20)
        name      CHAR(30),
        login     CHAR(20),
        age       INTEGER,
        gpa       REAL,
        UNIQUE (name, login),
        CONSTRAINT StudentsKey PRIMARY KEY (sid) )
```


03. Integrity Constraints

- Foreign Key, Referential Integrity
  - Foreign key:
    - a set of fields for referring to tuple in other relation
    - Must corresponds to primary key of the second relation.
    - Like a ‘logical pointer’
  - E.g.
    - sid is a foreign key referring to Students:
      - Enrolled(sid: string, cid: string, grade: string)
    - If all foreign key constraints are enforced, referential integrity is achieved.
    - Only students listed in the Students relation should be allowed to enroll for courses.

03. Integrity Constraints

- Foreign Key, Referential Integrity

Enrolled

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

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

```
CREATE TABLE Enrolled (
        sid       CHAR(20),
        cid       CHAR(20),
        grade     CHAR(2),
        PRIMARY KEY (sid, cid),
        FOREIGN KEY (sid) REFERENCES Students )
```


03. Integrity Constraints

- Enforcing Referential Integrity
  - Consider Students and Enrolled;
    - sid in Enrolled is a foreign key that references Students.
  - 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'.)
  - Similar if primary key of Students tuple is updated.

03. Integrity Constraints

- Enforcing ICs
  - SQL/92 supports 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 )
```


03. Integrity Constraints

- Where do ICs come from?
  - ICs are based upon the semantics of the real-world enterprise that is being described in the database relations.
  - We can check a database instance to see if an IC is violated, but we can NEVER infer that an IC is true by looking at an instance.
    - An IC is a statement about all possible instances!
    - From example, we know name is not a key, but the assertion that sid is a key is given to us.
  - Key and foreign key ICs are the most common; more general ICs supported too.

04. View

- Not stored in the database.
- Computed as needed from a view definition
- TABLE:
  - stored & physically exist in the database.

04. View

- Creating View

Enrolled

cid	*grade*	*sid*
Carnatic 101	C	53831
Reggae203	B	53832
Topology112	A	53650
History105	B	53666

Students

sid	*name*	*login*	*age*	*gpa*
50000	Dave	Dave@cs	19	3.3
53666	Jones	Jones@cs	18	3.4
53688	Smith	Smith@ee	18	3.2
53650	Smith	Smith@math	19	3.8
53831	Madayan	Madayan@music	11	1.8
53832	Guldu	Guldu@music	12	2.0

04. View

- Creating View

```
CREATE VEIW  B-Students (name, login, course) AS
        SELECT S.sname, S.sid, E.cid
        FROM    Students S, Enrolled E
        WHERE S.sid = E.sid AND E.grade='B'
```

name	*sid*	*course*
Jones	53666	History105
Guldu	53832	Raggae203

04. View

- Creating View
  - 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
```

04. View

- Independence
  - If the schema of a stored relation is changed,
    - we can define a view with the old schema, and application that expect to see the old schema can now use this view.
- Security
  - define views that give users access to just the information which they are allowed.
- 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 of the courses they are enrolled in.

2013-1-WKU-IP-C04 Database / Relational Model

04. View

- Updates on View
  - Updatable View
    - The views that are defined on a single base table can be updated in SQL-92.
  - Delete/Insert
    - We can delete(insert) a view row by deleting(inserting) the corresponding row from the underlying table.
    - In case of inserting, use null values in column of the underlying table that do not appear in the view.
  - An INSERT or UPDATE may change the underlying base table so that the resulting row is not in the view.

04. View

- Restrict View Updates

<A>

cname	*jyear*	*mname*
Sailing	1996	Dave
Hiking	1997	Smith
Rowing	1998	Smith

<B>

sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	Jones@cs	18	3.4
53688	Smith	Smith@ee	18	3.2
53650	Smith	Smith@math	19	3.8

04. View

- Restrict View Updates
  - If the view schema contains the primary key fields,
    - it is possible to update the view.

name	*login*	*club*	*Since*
Dave	Dave@cs	Sailing	1996
Smith	Smith@ee	Hiking	1997
Smith	Smith@ee	Rowing	1998
Smith	Smith@math	Hiking	1997
Smith	Smith@math	Rowing	1998

<C> Instance of Active & over 3.0 students

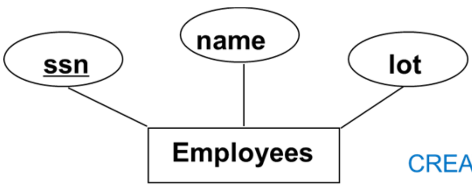
05. Destroying/Altering Tables/Views

- DROP TABLE Students RESTRICT
  - unless some view or integrity constraint refers to Students
- DROP TABLE Students CASCADE
  - any referencing views or integrity constraints are dropped as well

2013-1-WKU-IP-C04 Database / Relational Model

06. Logical DB Design: ER to Relational

- Entity sets to tables.



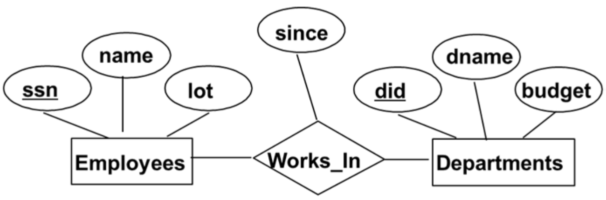

```
CREATE TABLE Employees (
        ssn     CHAR(11),
        name    CHAR(20),
        lot     INTEGER,
        PRIMARY KEY (ssn))
```

06. Logical DB Design: ER to Relational

- Relationship Sets to Tables
  - In translating a relationship set to a relation, attributes of the relation must include:
    - Keys for each participating entity set (as foreign keys).
  - This set of attributes forms a superkey for the relation.
  - All descriptive attributes.

06. Logical DB Design: ER to Relational

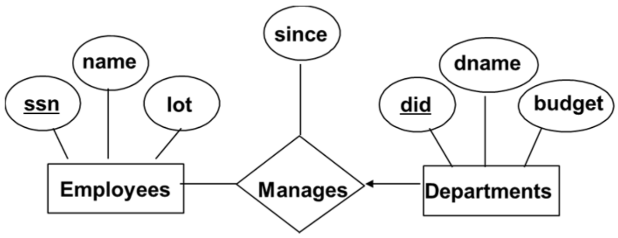
- Relationship Sets to Tables




```
CREATE TABLE Works_In (
    ssn     CHAR(1),
    did     INTEGER,
    since   DATE,
    PRIMARY KEY (ssn, did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments)
```

06. Logical DB Design: ER to Relational

- Review: Key Constraints
  - Each dept has at most one manager, according to the key constraint on Manages.



2013-1-WKU-IP-C04 Database / Relational Model

06. Logical DB Design: ER to Relational

- Translating ERD w/ Key Constraints
  - Map relationship to a table:
    - Note that did is the key now!
    - Separate tables for Employees and Departments.

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

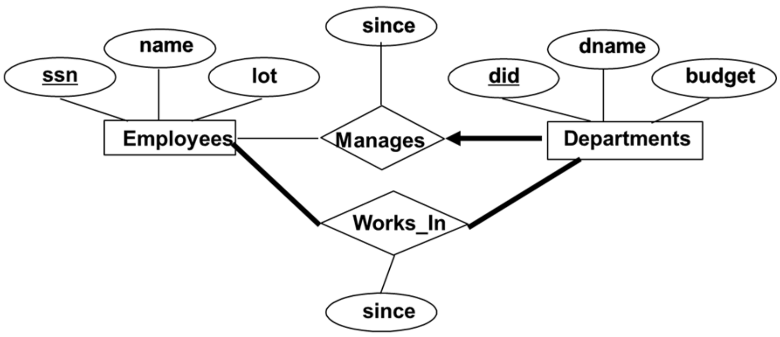
06. Logical DB Design: ER to Relational

- Translating ERD w/ Key Constraints
  - Since each department has a unique manager, we could instead combine Manages and 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 )
```


06. Logical DB Design: ER to Relational

- 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!)



06. Logical DB Design: ER to Relational

- 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
                   ON DELETE NO ACTION)
```


2013-1-WKU-IP-C04 Database / Relational Model

06. Logical DB Design: ER to Relational

- Participation Constraints in SQL
  - How can we show total participation, Works_In?
    - Use Assertion!

```
CREATE ASSERTION empMustWork
CHECK (
        (
        SELECT COUNT (E.ssn)
        FROM    Employees E
        WHERE E.ssn NOT IN (
                                SELECT distinct W.ssn
                                FROM    Works_In W
                                )
) = 0
)
```

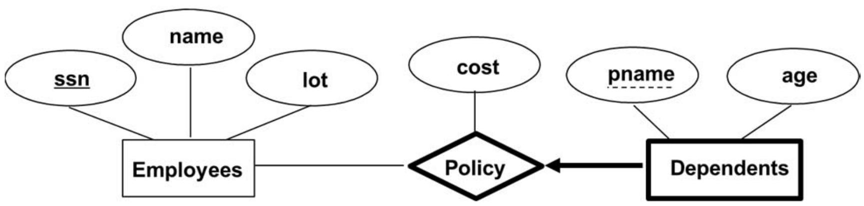
06. Logical DB Design: ER to Relational

- Review: 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.
  - 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.

06. Logical DB Design: ER to Relational

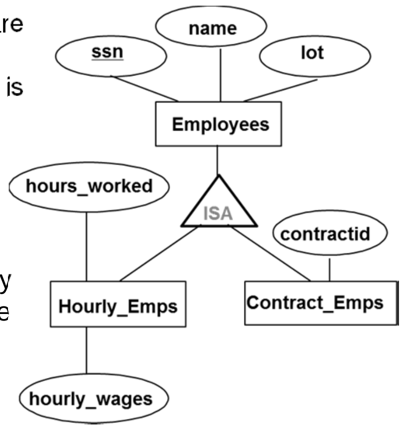
- Translating Weak Entity Sets

```
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)
```



06. Logical DB Design: ER to Relational

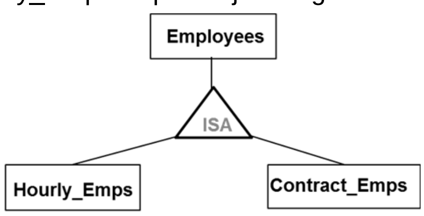
- Review: ISA Hierarchies
  - As in C++, or other PLs, attributes are inherited.
  - If we declare A ISA B, every A entity is also considered to be a B entity.
    - Overlap constraints: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (Allowed/disallowed)
    - Covering constraints: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (Yes/no)



2013-1-WKU-IP-C04 Database / Relational Model

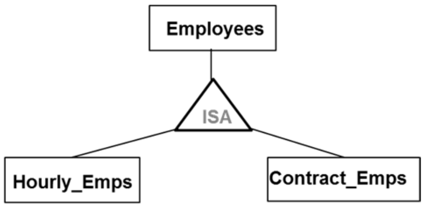
06. Logical DB Design: ER to Relational

- Translating ISA Hierarchies to Relations
  - 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.



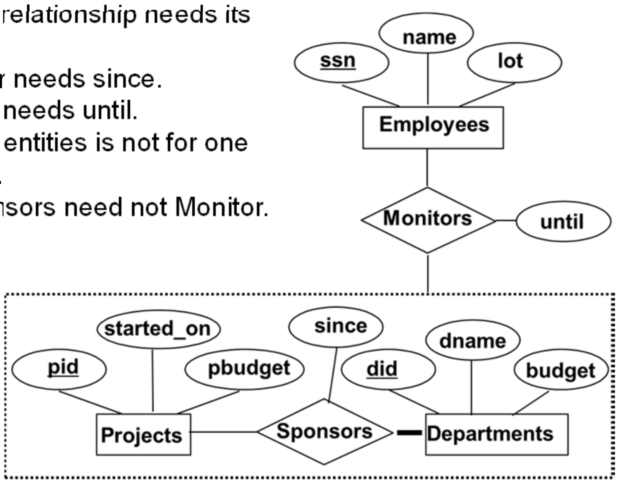
06. Logical DB Design: ER to Relational

- Translating ISA Hierarchies to Relations
  - 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.



06. Logical DB Design: ER to Relational

- Review: Aggregation
  - Differs from ternary relations.
    - When each relationship needs its attributes.
      - Sponsor needs since.
      - Monitor needs until.
    - When three entities is not for one relationship.
      - All Sponsors need not Monitor.



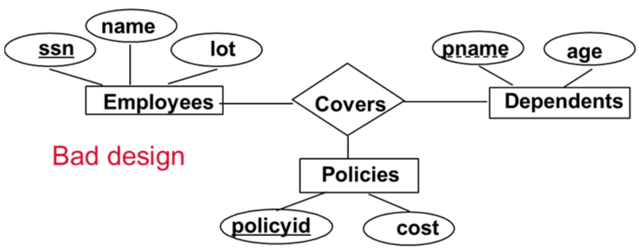
06. Logical DB Design: ER to Relational

- Review: Aggregation
  - Creating Tables

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

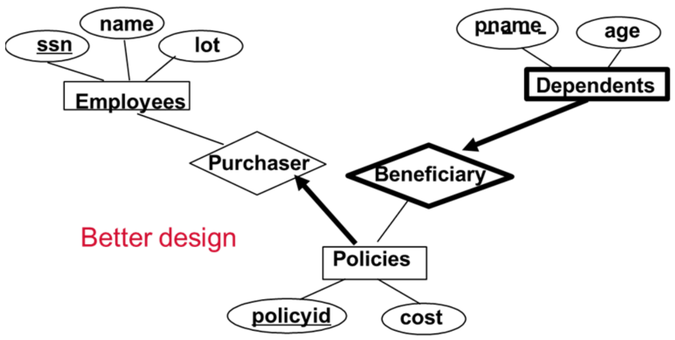

06. Logical DB Design: ER to Relational

- Review: Binary vs. Ternary Relationships
  - If each policy is owned by just 1 employee:
    - Key constraint on Policies would mean policy can only cover 1 dependent!



06. Logical DB Design: ER to Relational

- Review: Binary vs. Ternary Relationships
  - What are the additional constraints in the 2nd diagram?



06. Logical DB Design: ER to Relational

- Review: Binary vs. Ternary Relationships
  - The key constraints allow us to combine Purchaser with Policies and Beneficiary with Dependents.
  - Participation constraints lead to NOT NULL constraints.
  - What if Policies is a weak entity set?

06. Logical DB Design: ER to Relational

- Review: Binary vs. Ternary Relationships

```
CREATE TABLE Policies (
        policyid  INTEGER,
        cost      REAL,
        ssn       CHAR(11) NOT NULL,
        PRIMARY KEY (policyid),
        FOREIGN KEY (ssn) REFERENCES Employees
                  ON DELETE CASCADE )

CREATE TABLE Dependents (
        pname     CHAR(20),
        age       INTEGER,
        policyid  INTEGER,
        PRIMARY KEY (pname, policyid),
        FOREIGN KEY (policyid) REFERENCES Policies
                  ON DELETE CASCADE)
```


06. Tips

- Relational Model
  - 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