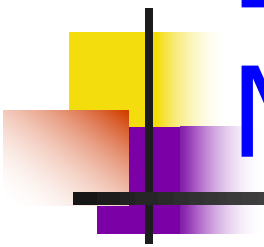
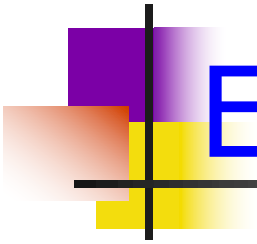


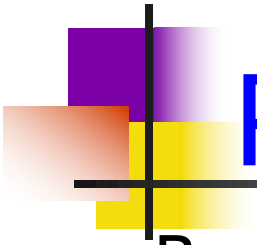
Logical Database Design: Mapping ER to Relational

Chapter 3, Section 3.5



ER Model vs. Relational Model

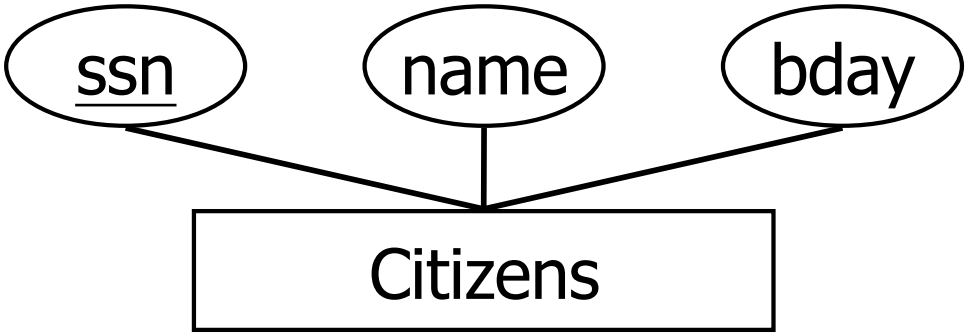
- ER Model used for conceptual design
- Relational Model implemented by modern DBMS
- Important Step: Translate ER diagram to Relational schema



Recall ER Constructs

- Basic Constructs
  - Entity Sets
  - Relationship Sets
  - Attributes (of entities and relationships)
- Additional Constructs
  - ISA Hierarchies
  - Weak Entities
  - Aggregation
- Integrity Constraints
  - Key constraints
  - Participation constraints
  - Overlap / Covering constraints for ISA hierarchies

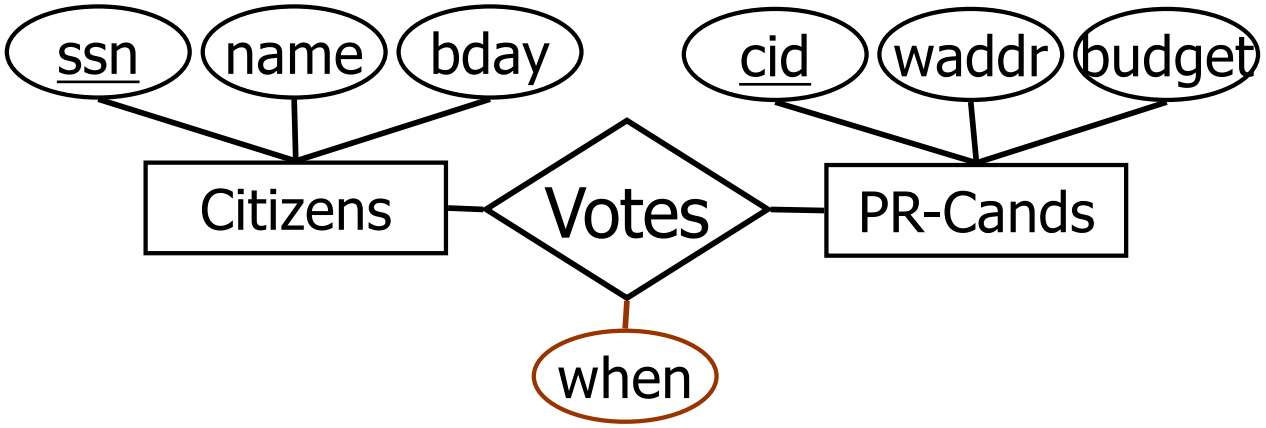
Entity Sets to Tables




```
CREATE TABLE Citizens
        (ssn   CHAR(11),
        name CHAR(20),
        bday   DATE,
        PRIMARY KEY  (ssn))
```

Can ssn have a null value?

Relationship Sets to Tables



Relationship set -> Table

Attributes:

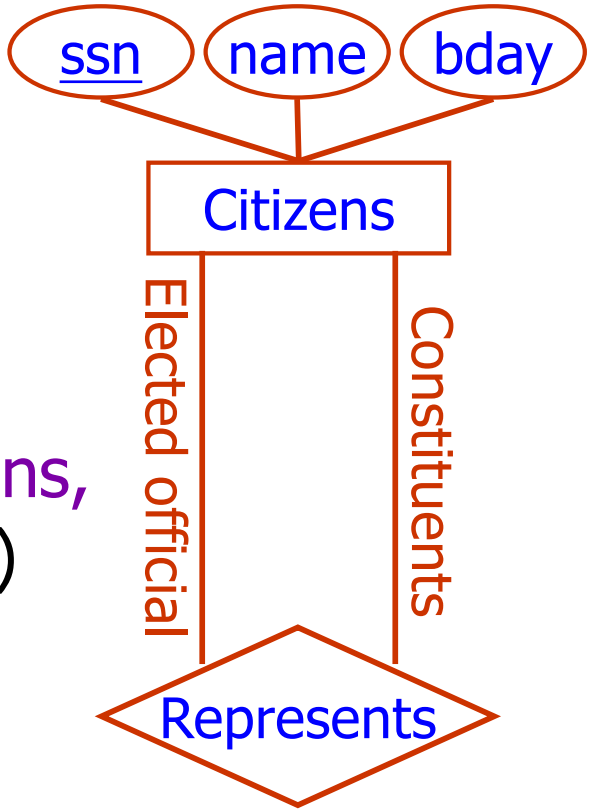
- Participating entity set primary keys
  - Foreign key
- Descriptive attributes

```
CREATE TABLE Votes(
        ssn     CHAR(11),
        cid     INTEGER,
        when    DATE,
PRIMARY KEY (ssn, cid),
FOREIGN KEY (ssn) REFERENCES Citizens,
FOREIGN KEY (cid) REFERENCES PR-Cands)
```

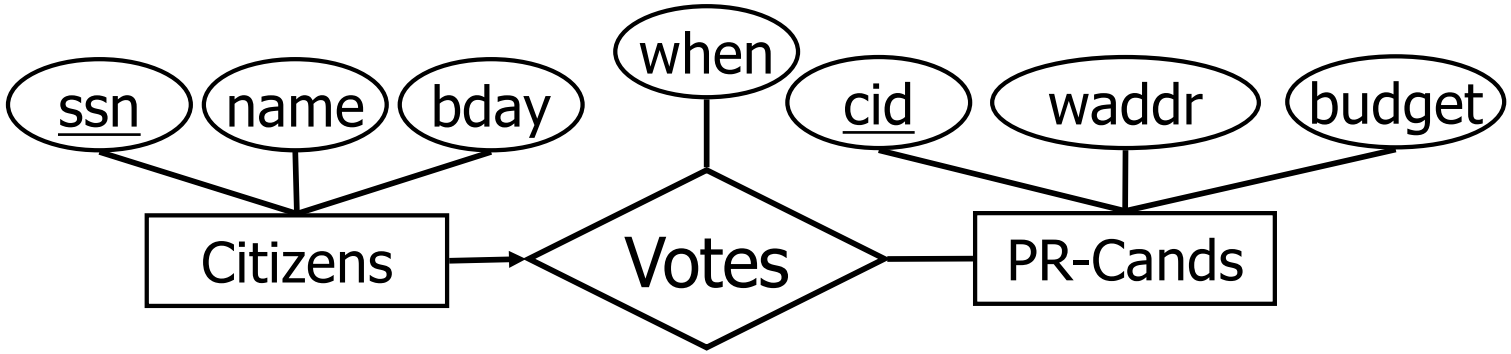
Generalizes to n-ary relationships (we will see example later)

Relationship Sets to Tables

```
CREATE TABLE Represents(
        elected_ssn  CHAR(11),
        cons_ssn  CHAR(11),
PRIMARY KEY (elected_ssn, cons_ssn),
FOREIGN KEY (elected_ssn) REFERENCES Citizens,
FOREIGN KEY (cons_ssn) REFERENCES Citizens)
```



Key Constraints

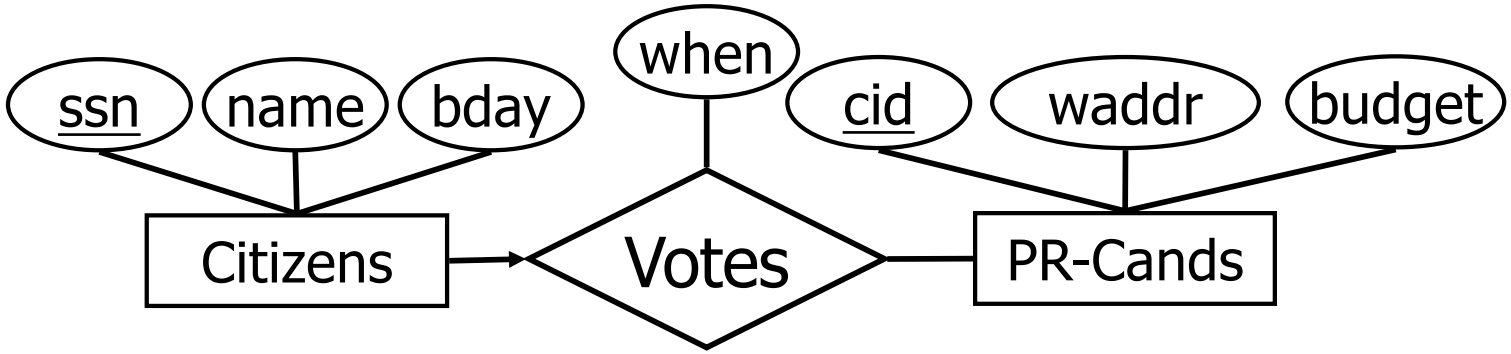


- Approach 1 – Three Tables

```
CREATE TABLE  Votes
(  ssn      CHAR(11),
   cid      INTEGER  NOT NULL,
   when  DATE,
   PRIMARY KEY (ssn),
      FOREIGN KEY (ssn) REFERENCES Citizens,
      FOREIGN KEY (cid) REFERENCES PR-Cands)
```

Same approach as before – Map each entity set and relationship set to a table

Key Constraints



- Approach 2 – Two Tables

```
CREATE TABLE  Citizen_Votes (
   ssn     CHAR(11),  name  CHAR(20),
   bday    DATE,      when  DATE,
   cid     INTEGER,
PRIMARY KEY  (ssn),
FOREIGN KEY (cid) REFERENCES PR-Cands)
```

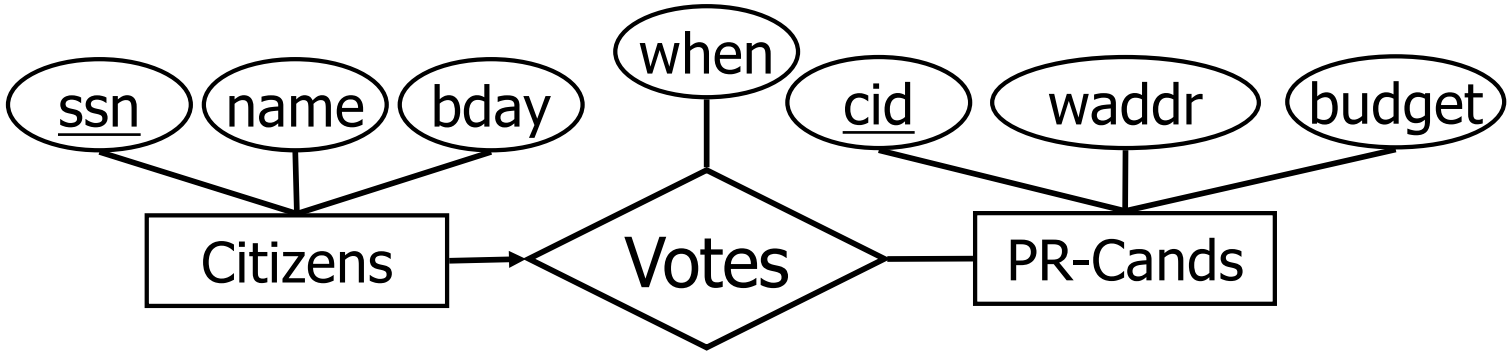
Each citizen can only vote once, so OK to fold 'Votes' relationship into 'Citizens' entity

Q: Can cid be null?

Q: What if many citizens don't vote?

Q: Which approach is better?

Key Constraints



What about 1 Table?

No! This is bad design.

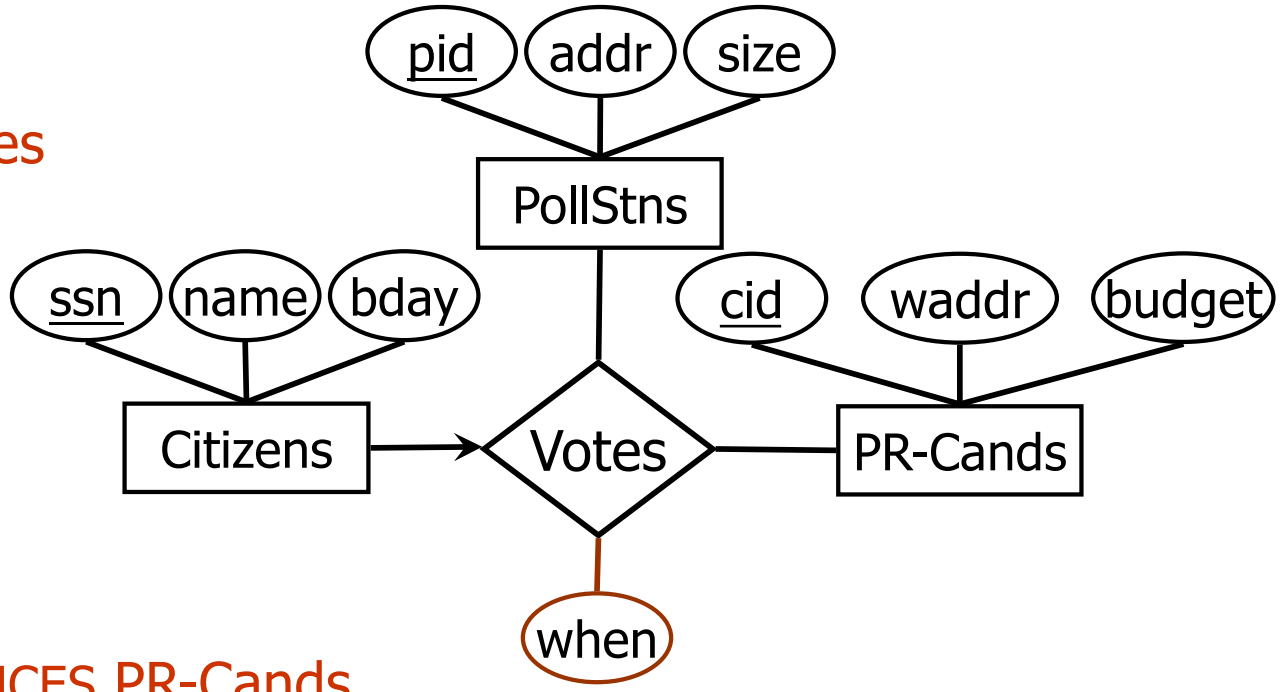
e.g., For each citizen that voted for a candidate, we would be storing candidate's information (cid, waddr, budget)

=> REDUNDANCY!

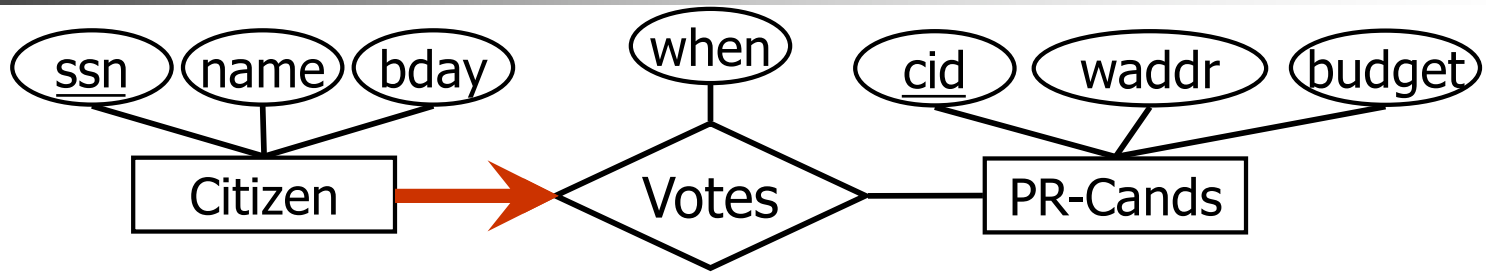
Key Constraints

Can generalize to n-ary relationships

```
CREATE TABLE Citizens_Votes
(  ssn      CHAR(11),
   name   CHAR(20),
   bday DATE,
   when   DATE,
   pid   INTEGER,
   cid    INTEGER,
PRIMARY KEY  (ssn),
FOREIGN KEY (cid) REFERENCES PR-Cands,
FOREIGN KEY (pid) REFERENCES PollStns)
```



Participation Constraints

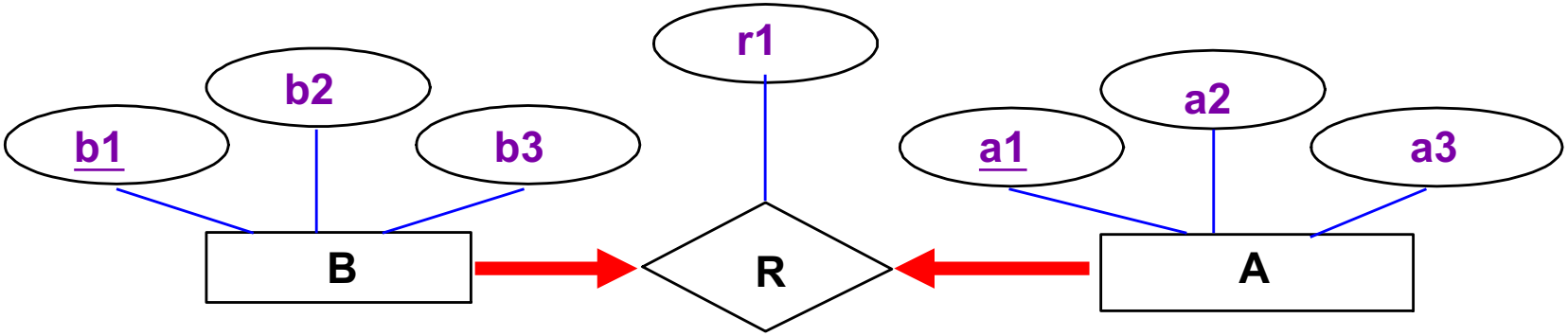


Using Approach 2

```
CREATE TABLE  Citizen_Votes(
    ssn      CHAR(11),
    name   CHAR(20),
    bday    DATE,
    when   DATE,
    cid       INTEGER NOT NULL,
PRIMARY KEY  (ssn),
FOREIGN KEY  (cid) REFERENCES PR_Cands,
       ON DELETE NO ACTION)
```

Can we enforce participation constraint using Approach 1 (three tables) ?

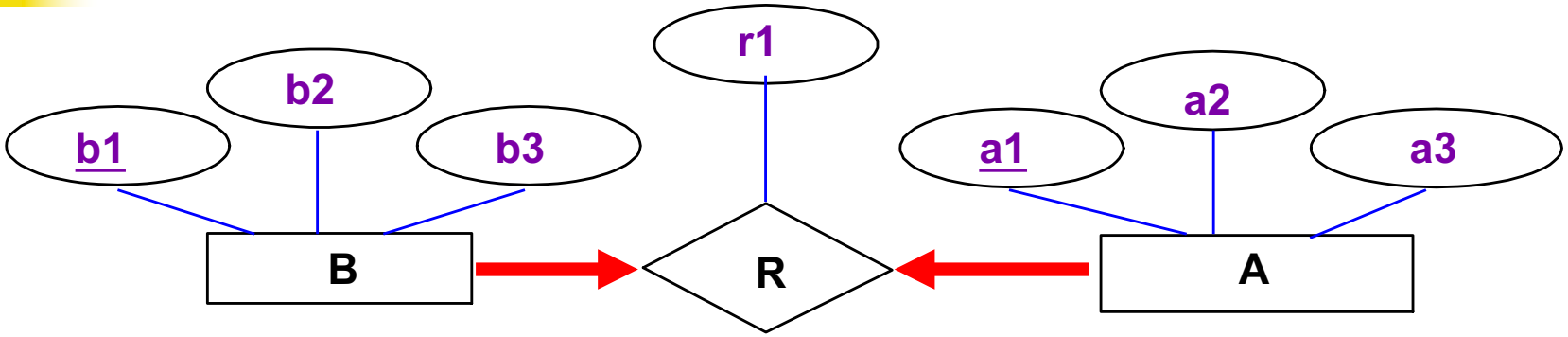
Mapping Participation Constraints




```
CREATE TABLE RAB(
        r1 Integer,
        a1 Integer,
        a2 Integer,
        a3 Integer,
        b1 Integer,
        b2 Integer,
        b3 Integer …)
```

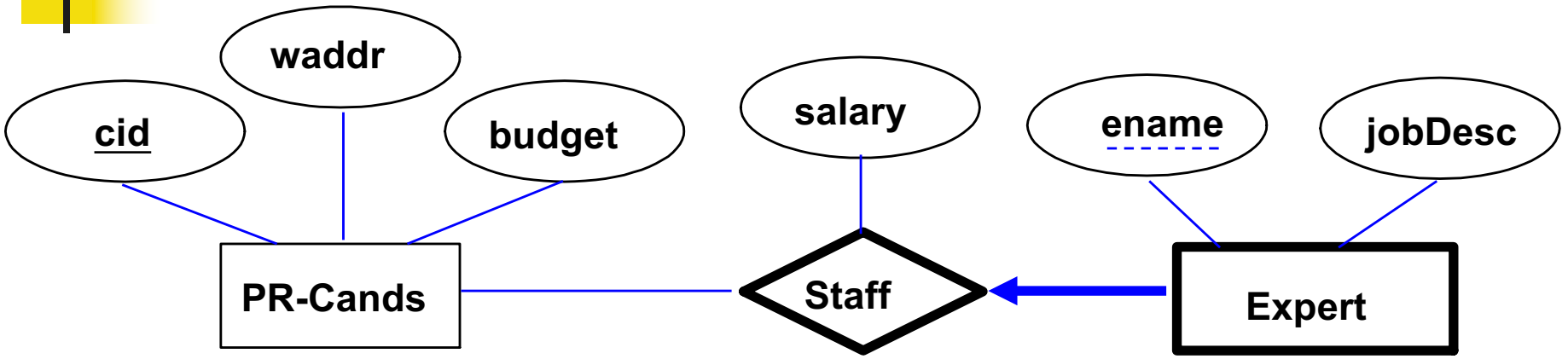
Key constraints?

Mapping Participation Constraints




```
CREATE TABLE  RAB(
        r1 Integer,
        a1 Integer,
        a2 Integer,
        a3 Integer,
        b1 Integer NOT NULL,
        b2 Integer,
        b3 Integer,
       UNIQUE (b1), PRIMARY KEY (a1))
```

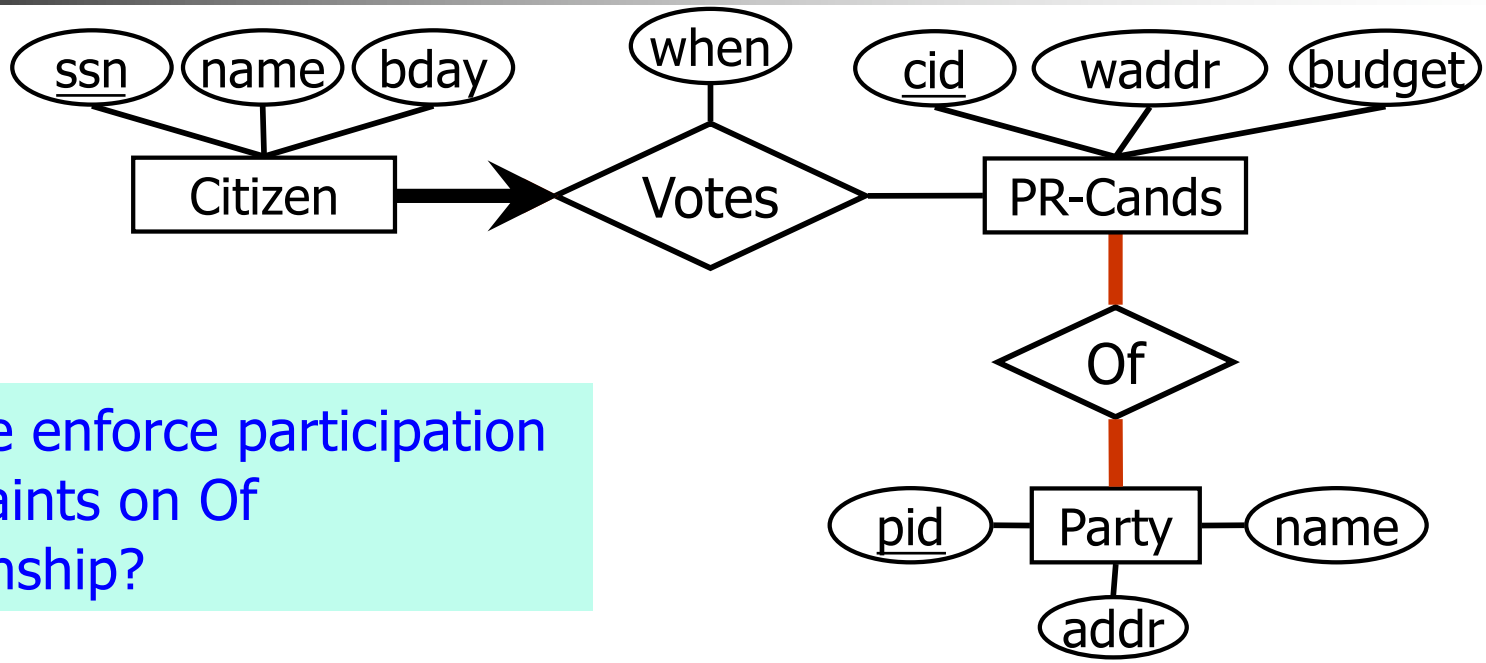
Weak Entities



- Use approach 2: Combine weak entity and owning relationship into one relation
  - Delete all weak entities when an owner entity is deleted.

```
CREATE TABLE  Expert_Staff (
   ename     CHAR(20),
   jobDesc   CHAR(40),
   salary    REAL,
   cid       INTEGER,
   PRIMARY KEY  (ename, cid),
   FOREIGN KEY  (cid) REFERENCES PR-Cands ON DELETE CASCADE)
```

Participation Constraints

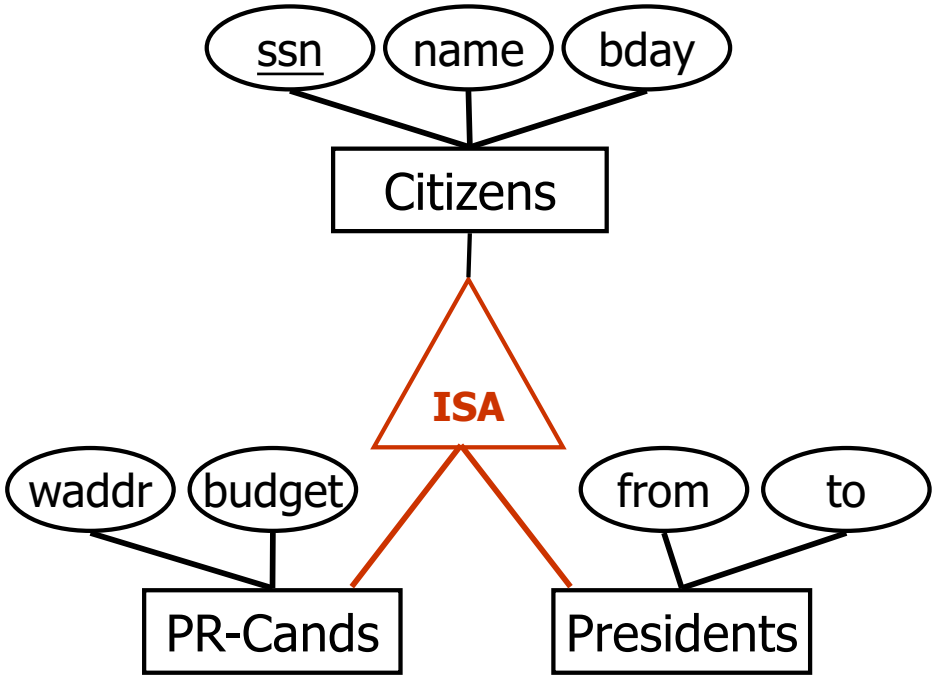


Can we enforce participation constraints on Of relationship?

Need table constraints & assertions – Later.

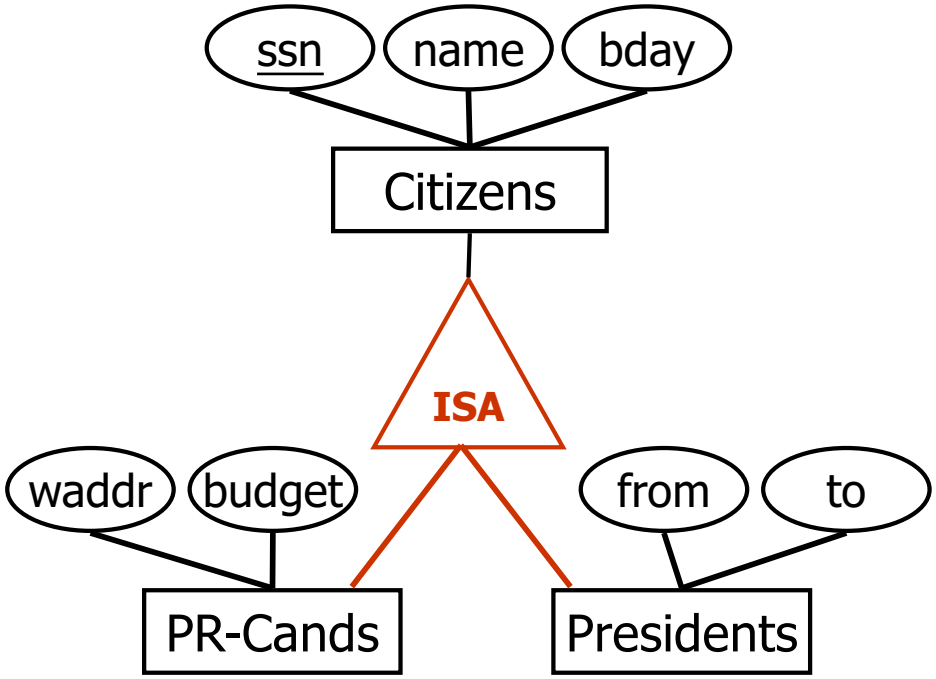
ISA Hierarchies – General Approach

- Three relations:
  - Citizens (ssn, name, bday)
  - PR-Cands (ssn, waddr, budget)
  - Presidents (ssn, from, to)
- Queries
  - Involving all citizens => Easy
  - Involving just PR-Cands, need to join PR-cands with Citizens to get some attributes

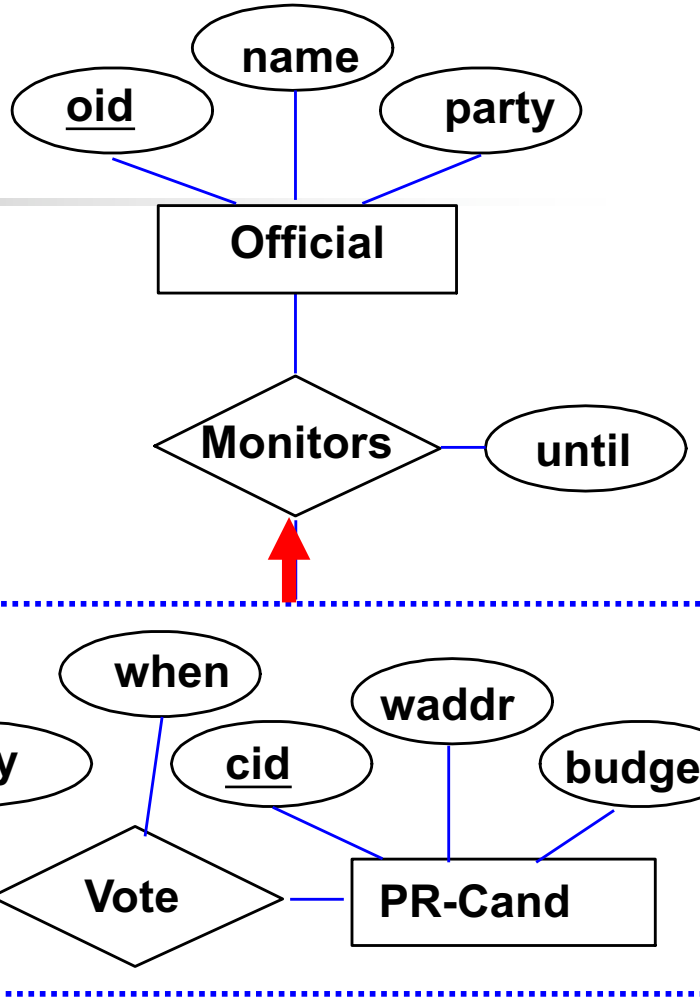


ISA Hierarchies – Alternative

- Two relations:
  - PR-Cands (ssn, name, bday, waddr, budget)
  - Presidents (ssn, name, bday, from, to)
- Problems
  - What if citizen is both?
    - Redundancy
  - What if citizen is neither?
    - Use General approach



Aggregation



- Translation similar to relationship sets
- Monitors primary key
  - (ssn, cid, oid)

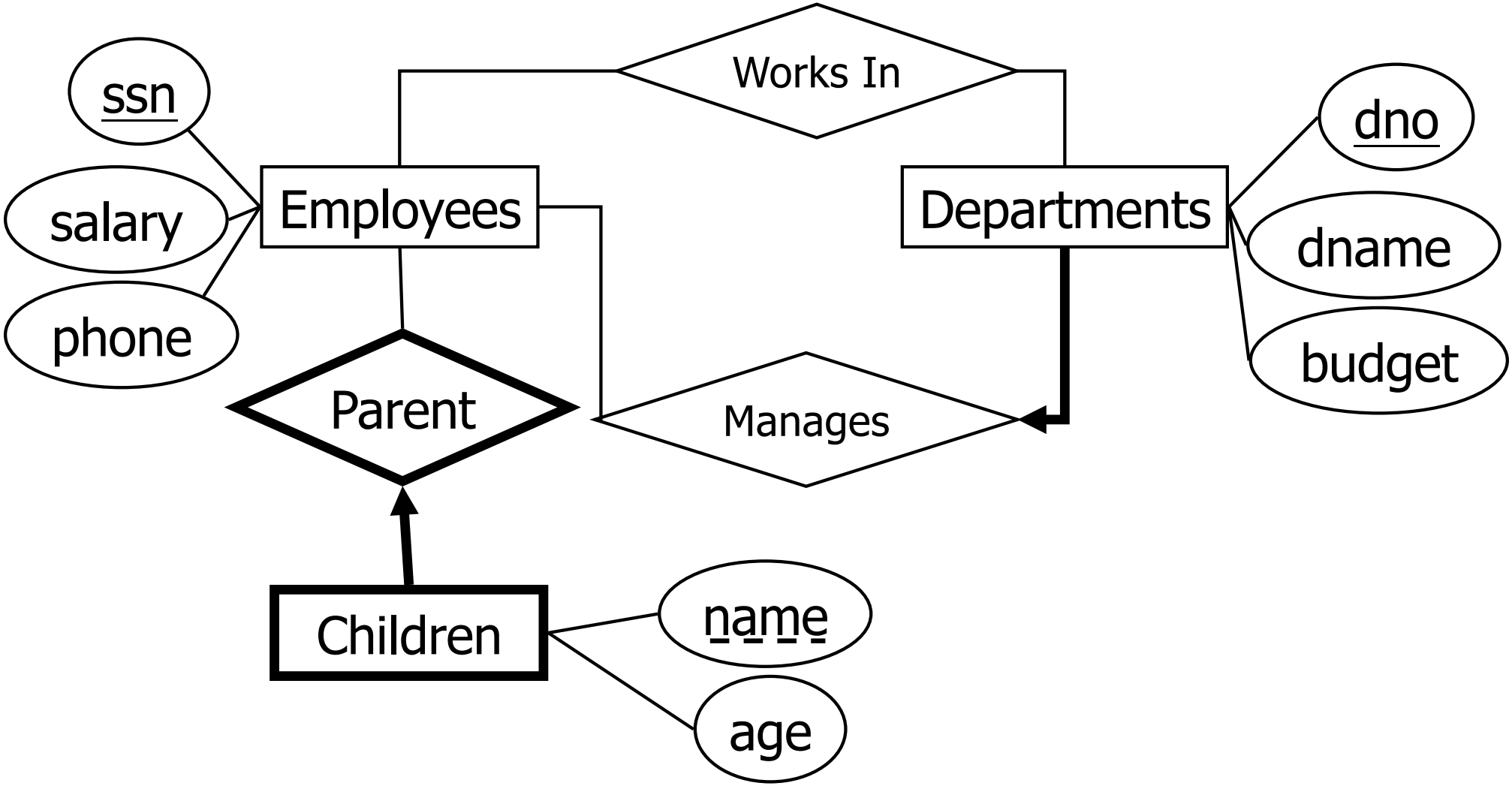
- What if every Vote must have have exactly one monitor?

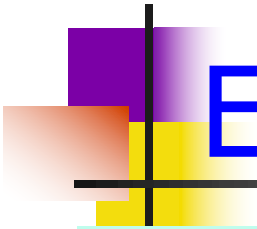
Fold Vote and Monitors into 1 table

Exercise – Part 1

- A company database needs to store information about
  - employees (identified by ssn, with *salary* and *phone* attributes),
  - departments (identified by dno, with *dname* and *budget* attributes), and
  - children of employees (with *name* and *age* attributes).
- Employees work in (zero or more) departments;
- Each department is managed by exactly one employee
- A child must be identified uniquely by name when the parent (who is an employee; assume only one parent works for the company) is known.
- We are not interested in information about a child once the parent leaves the company.
- Draw an ER diagram that captures this information

ER Diagram (One Solution)





Exercise – Part 2

- Write SQL Statements to create the corresponding relations, and to capture as many of the constraints as possible.

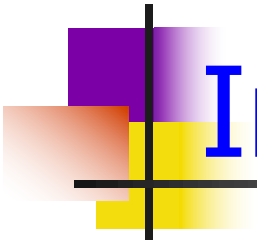
SQL DDL (One Solution)

```
CREATE TABLE employees (
ssn INTEGER,
salary REAL,
phone CHAR(10),
PRIMARY KEY(ssn))
```

```
CREATE TABLE works (
ssn INTEGER,
dno INTEGER,
PRIMARY KEY (ssn, dno),
FOREIGN KEY (ssn)
   REFERENCES employees,
FOREIGN KEY (dno)
   REFERENCES departments)
```

```
CREATE TABLE departments (
dno INTEGER,
dname CHAR(20),
budget real,
manager INTEGER NOT NULL,
PRIMARY KEY (dno),
FOREIGN KEY (manager)
          REFERENCES employees)
```

```
CREATE TABLE children (
name CHAR(20),
age REAL,
parent INTEGER NOT NULL,
PRIMARY KEY(name, parent),
FOREIGN KEY(parent)
   REFERENCES employees
   ON DELETE CASCADE)
```



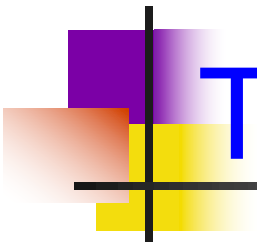
Integrity Constraints

- Describes conditions that must be satisfied by every legal instance
- Types of integrity constraints
  - Domain constraints
  - Primary key constraints
  - Foreign key constraints
  - **General constraints**

Table Constraints (5.7 in book)

- More general than key constraints
- Can use a query to express constraint
  - Constraints checked each time table updated
  - CHECK constraint always true for empty relation

```
CREATE TABLE Sailors
        ( sid INTEGER,
        sname CHAR(10),
        rating INTEGER,
        age REAL,
        PRIMARY KEY (sid),
        CHECK ( rating >= 1
                AND rating <= 10)
         );
```



Try it out in sqlplus or sqlite

```
CREATE TABLE   Sailors
        ( sid  INTEGER,
        sname  CHAR(10),
        rating  INTEGER,
        age  REAL,
        PRIMARY KEY  (sid),
        CHECK (rating >= 1 AND rating <= 10));

INSERT INTO Sailors VALUES (1, 's1', 11, 25);
```

Do you get a constraint violation error?

More general CHECK

- Interlake boats cannot be reserved. **Note: these are not supported in Oracle or SQLite**

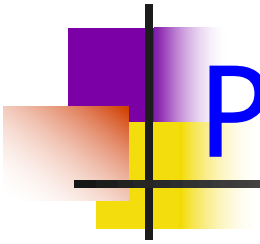
```
CREATE TABLE Reserves
        ( sname CHAR(10),
        bid INTEGER,
        day DATE,
        PRIMARY KEY (bid,day),
        CONSTRAINT noInterlakeRes
        CHECK (`Interlake' NOT IN
                ( SELECT B.bname
                FROM Boats B
                WHERE B.bid=bid)))
```

Constraints Over Multiple Relations

- For general constraint over multiple tables, SQL standard provides assertions.

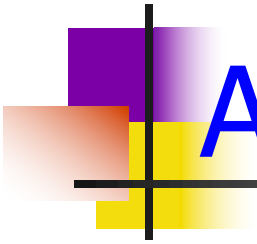
Number of boats plus number of sailors is < 100

```
CREATE ASSERTION smallClub
CHECK
((SELECT COUNT (S.sid) FROM Sailors S) +
  (SELECT COUNT (B.bid) FROM Boats B) < 100)
```



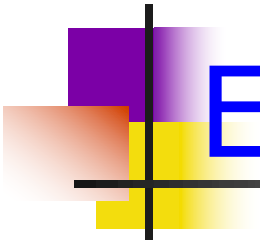
Practical Considerations

- CHECK with subqueries and ASSERTIONS
  - Part of SQL standard (since 1992?)
  - But, major databases do not support them
    - Main concern: performance.
- Instead, most major database systems require you to use *triggers* to achieve the same goals. Triggers are procedural.



Active Databases & Triggers

- **Trigger:** procedure that starts automatically if specified changes occur to the DBMS
- Three parts:
  - Event (activates the trigger)
  - Condition (tests whether the trigger should run)
  - Action (what happens if the trigger runs)
    - Before and After Triggers
- Trigger Execution
  - Row-level Triggers: Once per modified row
  - Statement-level Triggers: Once per SQL statement



Example

- Student table with columns: sid, name, age.
- Event: Insert new row into Students
- Condition: New student's age >= 18
- Event: Update a counter variables that tracks university's total enrollment

Oracle Trigger Basic Syntax

CREATE [OR REPLACE] TRIGGER <trigger_name>
{BEFORE | AFTER} {INSERT | DELETE | UPDATE} ON <table_name>
[REFERENCING [NEW AS <new_row>] [OLD AS <old_row>]]
[FOR EACH ROW [WHEN (<condition>)]]
<trigger_body>

In Oracle, <trigger_body> is a PL/SQL block

Oracle Triggers
Not quite the same
As SQL Standard

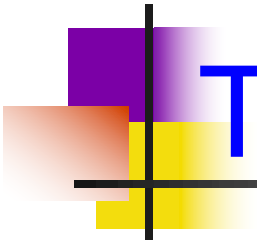
Triggers – Use for Auditing

```
CREATE TABLE Score(uniqname CHAR(10), grade NUMBER);
CREATE TABLE ScoreLog(uniqname CHAR(10) , gradeprev NUMBER, gradenew NUMBER);


CREATE OR REPLACE TRIGGER trig1 AFTER UPDATE ON Score
FOR EACH ROW
     WHEN (NEW.grade <> OLD.grade)
    BEGIN
         INSERT INTO ScoreLog VALUES(:NEW.uniqname, :OLD.grade, :NEW.grade);
     END;
.
RUN

INSERT INTO Score VALUES('s1', 23);
INSERT INTO Score VALUES('s2', 50);
UPDATE Score Set grade = 100 WHERE uniqname = 's1';
SELECT * FROM Score;
SELECT * FROM ScoreLog;
```

Try it Out!



To see compilation errors

- Use the following:

  SHOW ERRORS TRIGGER <triggername>

Triggers – Use for error-checks

```
CREATE OR REPLACE TRIGGER trig2 BEFORE INSERT OR UPDATE ON Score
FOR EACH ROW
     WHEN (NEW.grade < 0 OR NEW.grade > 100)

    BEGIN
         RAISE_APPLICATION_ERROR(-20001, 'Bad grade');
     END;
.
RUN
```

- Aborts an INSERT or UPDATE command if grade is not in range [0,100]. Try doing:

```
INSERT INTO Score VALUES('s3', -10);
```

Triggers – Fix inserted data

```
CREATE OR REPLACE TRIGGER trig3 BEFORE INSERT OR UPDATE ON Score
FOR EACH ROW
     WHEN (NEW.grade > 100)
    BEGIN
         :NEW.grade := 100;
     END;
.
RUN
```

```
INSERT INTO Score VALUES('s4', 200);
```

What value is inserted into Score?

Simulate MySQL's AUTO_INCREMENT

```
CREATE TABLE Person
(
id INT NOT NULL AUTO_INCREMENT,
last VARCHAR(24) NOT NULL,
first VARCHAR(24)
)

INSERT INTO Person VALUES('john', 'doe');
```

AUTO_INCREMENT is NOT available in Oracle.

See Discussion Slides on how to use triggers in Oracle to accomplish equivalent this.

Oracle Trigger Example Contd.

- First trigger executed *before* the activating statement, second executes *after* the activating statement.
- In combination with:
  - "FOR EACH ROW" - execute once per modified record
  - (default) - execute once per activating statement.
- Activating statements can be:
  - "INSERT"
  - "DELETE"
  - "UPDATE"

Recall CASCADE constraints

```
CREATE TABLE Athlete
(aid INTEGER PRIMARY KEY,
 name CHAR(30),
 country CHAR(20),
 sport CHAR(20));
```

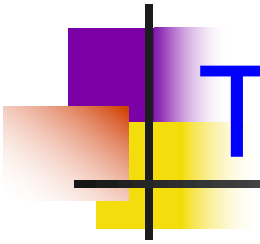
```
CREATE TABLE Olympics
(oid INTEGER PRIMARY KEY,
 year INTEGER,
 city CHAR(20));
```

```
CREATE TABLE Compete
  (aid INTEGER,  oid INTEGER,
   PRIMARY KEY  (aid, oid),
   FOREIGN KEY (aid)
        REFERENCES Athlete
        ON DELETE CASCADE
  );
```

CASCADE Using Triggers

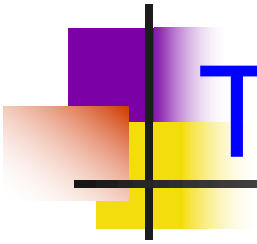
```
CREATE TABLE Compete
   (aid INTEGER,  oid INTEGER,
    PRIMARY KEY  (aid, oid),
    FOREIGN KEY (aid)
          REFERENCES Athlete );

CREATE OR REPLACE TRIGGER cascade_on_delete
AFTER  DELETE ON Athlete
FOR EACH ROW
BEGIN
   DELETE FROM Compete
  WHERE Compete.aid = :OLD.aid;
END;
/
```



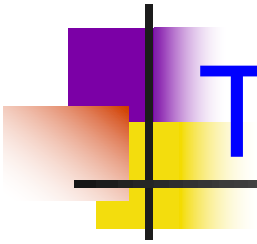
Trying out triggers

- Play with triggers in SqlLite and Oracle.
- To drop the trigger:
  - DROP TRIGGER *triggername;*



Triggers: Pitfalls and Pain

- Triggers can triggers other triggers
  - Update to table A causes update to table B
  - Update to table B causes update to table C
- Even cycles can occur
- Debugging difficult
- You may see an error with “mutating” table
  - Typically, due to a SELECT on a table in the middle of being modified inside a trigger



Triggers vs. Constraints

- Both used to maintain consistency
- Constraints are easier to understand than triggers
  - Always prefer them when you have a choice
- Triggers are more powerful but harder to debug