Logical Database Design: Mapping ER to Relational

Chapter 3, Section 3.5

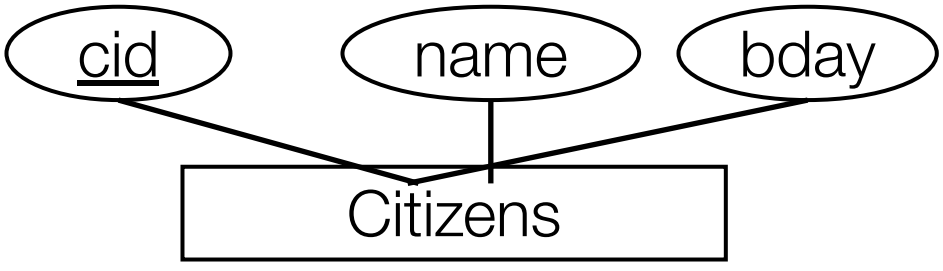
Learning Objectives

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

Recall ER Constructs

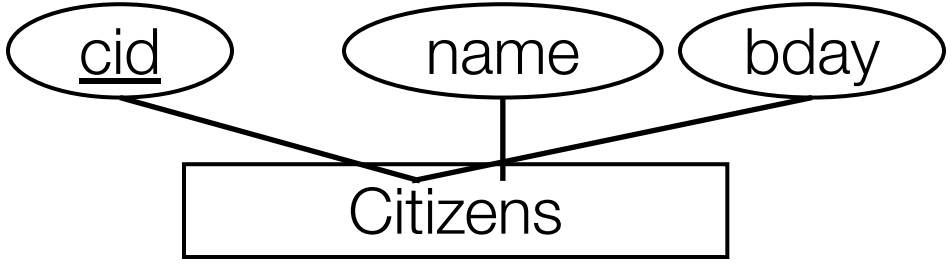
- Basic Constructs
  - Entity Sets
  - Relationship Sets
  - Attributes (of entities and relationships)
- Additional Constructs
  - ISA Hierarchies
  - Weak Entities
  - Aggregation
- Integrity Constraints
  - Key constraints and Participation constraints
  - ISA hierarchies

Entity Sets to Tables




```
CREATE TABLE Citizens
    (cid    INTEGER,
  name   CHAR(20),
  bday   DATE,
  PRIMARY KEY  (cid))
```

Question??

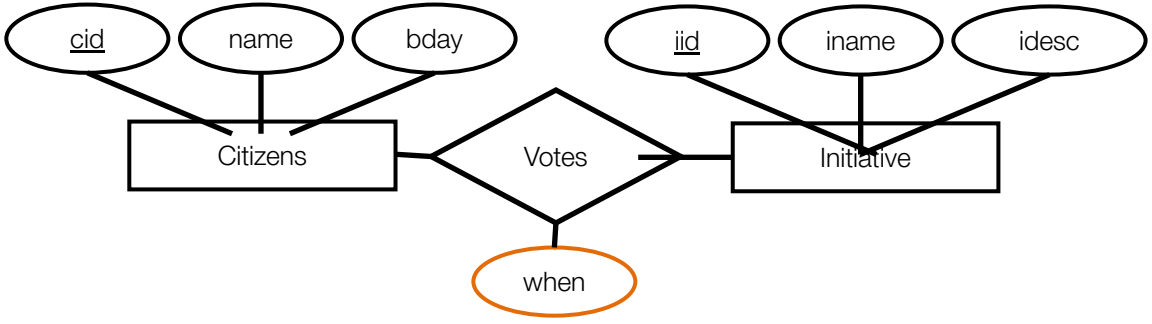



```
CREATE TABLE Citizens
      (cid    INTEGER
   name   CHAR(20),
   bday   DATE,
   PRIMARY KEY   (cid));
```

Can cid have a null value?

A. Yes
B. No
C. Depends

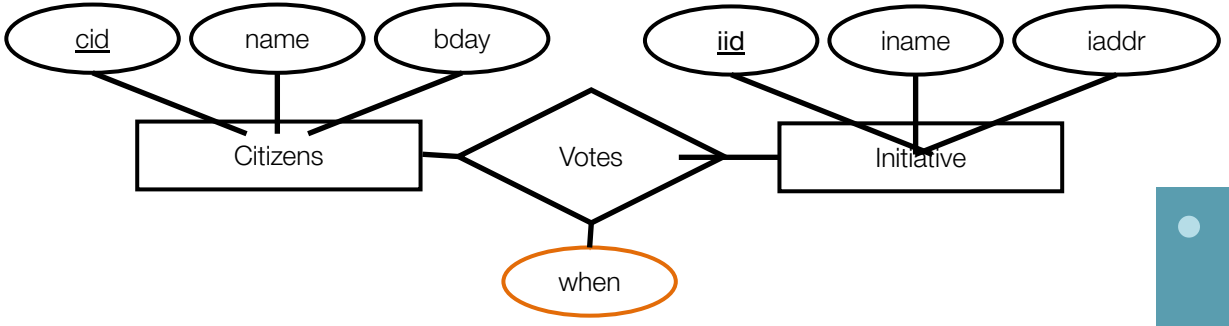
Relationship Sets to Tables



- Foreign key: keys from participating entity sets
- Descriptive attributes

```
CREATE TABLE Votes(
    cid     INTEGER,
    iid     INTEGER,
    when     DATE,
    PRIMARY KEY (cid, iid),
    FOREIGN KEY (cid)
        REFERENCES Citizens,
    FOREIGN KEY (iid)
        REFERENCES Initiative);
```

Impact of attribute names

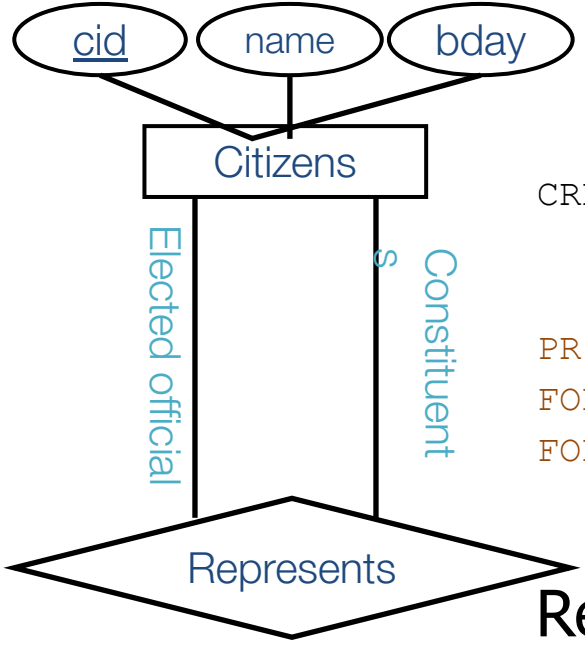


- Foreign key: keys from participating entity sets
- Descriptive attributes

```
CREATE TABLE Votes(
    citizenid     INTEGER,
    iid           INTEGER,
    when      DATE,
    PRIMARY KEY (citizenid, iid),
    FOREIGN KEY (citizenid)
        REFERENCES Citizens(cid),
    FOREIGN KEY (iid)
        REFERENCES Initiative);
```

Note that you need to specify the cid column in REFERENCES.

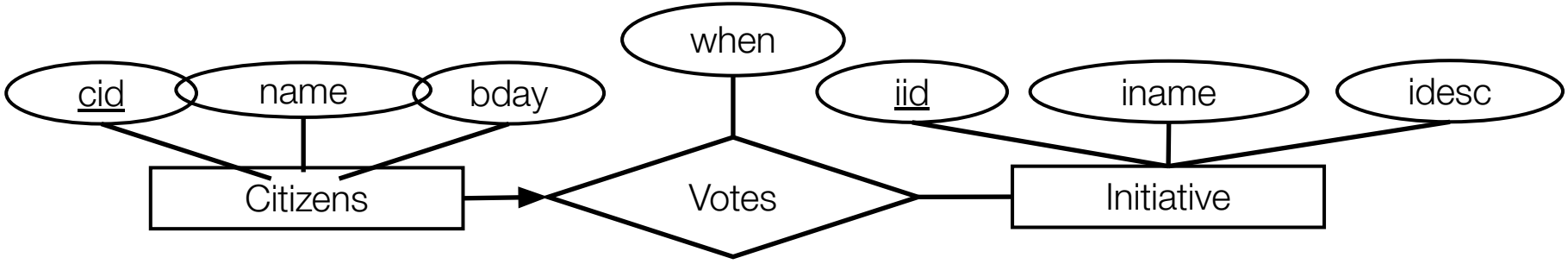
Relationship Sets to Tables




```
CREATE TABLE Represents(
    elected_cid  INTEGER,
    cons_cid     INTEGER,
PRIMARY KEY (elected_cid, cons_cid),
FOREIGN KEY (elected_cid) REFERENCES Citizens(cid),
FOREIGN KEY (cons_cid) REFERENCES Citizens(cid));
```

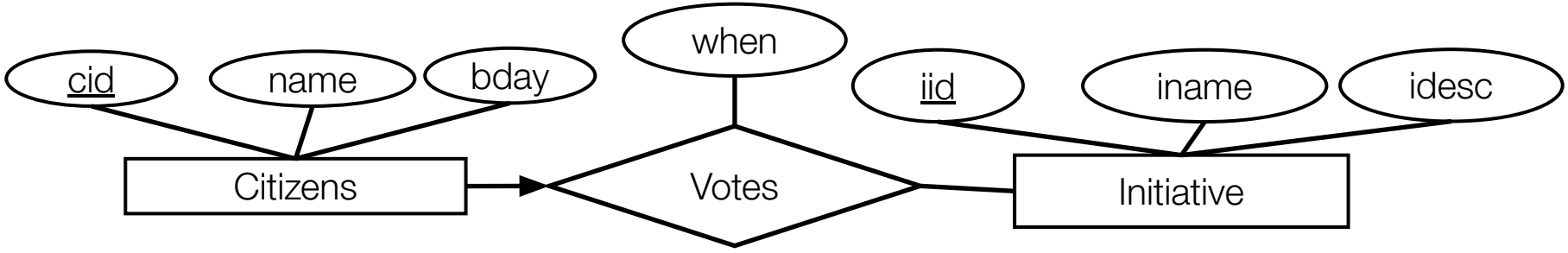
Represents is a Set of {(elected_cid, cons_cid), ….}

Key Constraints



- Notice the arrow from Citizens to Votes

Approach 1: Key Constraint

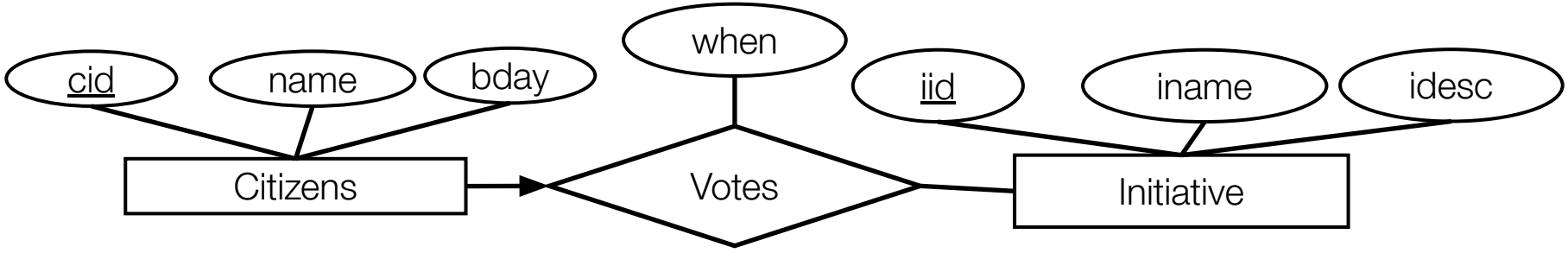


- Three Tables (Citizens, Votes, Initiative). Votes table changes:

```
CREATE TABLE  Votes
(   cid     INTEGER,
    iid     INTEGER,
    when    DATE
    -- Primary key and foreign key constraints?

 );
```

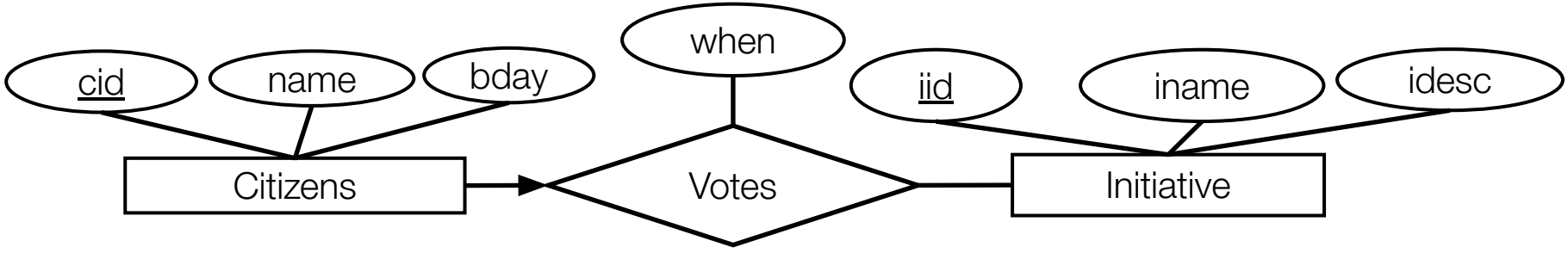
Approach 1: Key Constraint



- Three Tables (Citizens, Votes, Initiative). Votes table changes:

```
CREATE TABLE  Votes
(   cid     INTEGER NOT NULL,   -- NOT NULL is optional here
    iid     INTEGER NOT NULL,
    when    DATE,
    PRIMARY KEY (cid),  -- implies NOT NULL constraint
    FOREIGN KEY (cid) REFERENCES Citizens,
    FOREIGN KEY (iid) REFERENCES Initiative)
 );
```

Approach 2: Key Constraint

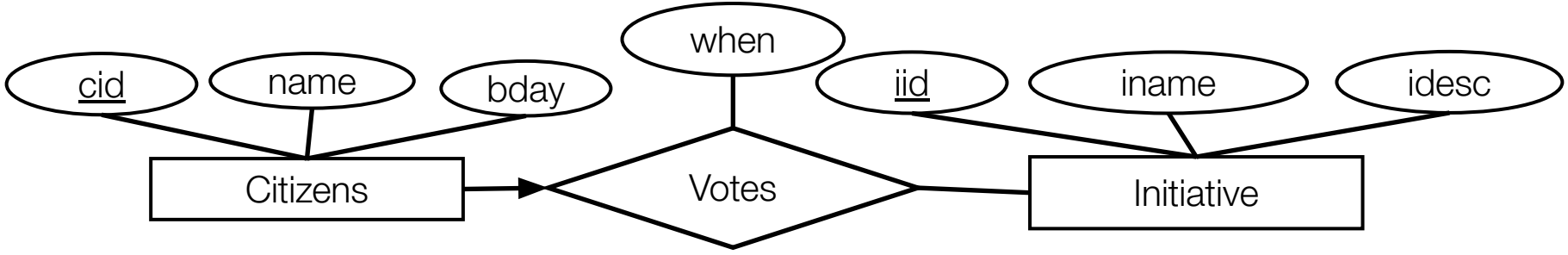


- Approach 2: Only two tables: Citizen_Votes and Initiative tables

```
CREATE TABLE  Citizens_Votes (
    cid     INTEGER,
    name    CHAR(20),
    bday    DATE,
    when    DATE,          -- can be NULL
    iid     INTEGER,       -- can be NULL
PRIMARY KEY  (cid),
FOREIGN KEY (iid) REFERENCES Initiative));
```

Each citizen can only vote once, so OK to fold 'Votes' relationship into 'Citizens' entity and use a single table for both

Question: Which approach better?



- Approach 2: Only two tables: Citizen_Votes and Initiative tables

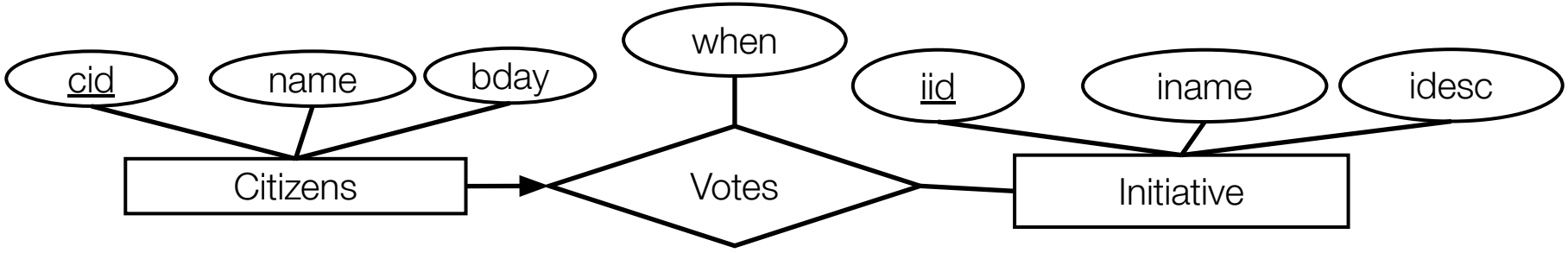
Approach 1 (separate relationship table)

```
CREATE TABLE  Votes
(  cid     INTEGER,
   iid     INTEGER NOT NULL,
   when    DATE,
   PRIMARY KEY (cid),
   FOREIGN KEY (cid) REFERENCES Citizens,
   FOREIGN KEY (iid) REFERENCES Initiative)
 );
```

Approach 2 (fold tables)

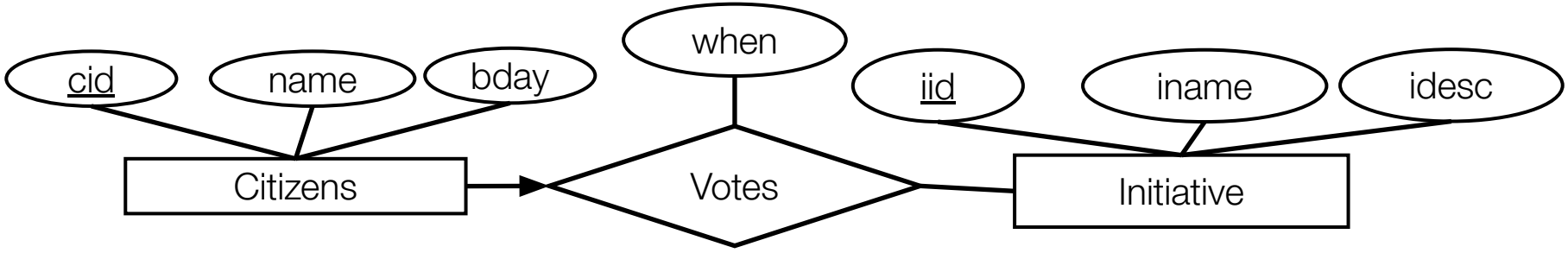
```
CREATE TABLE  Citizens_Votes (
   cid     INTEGER,
   name    CHAR(20),
   bday    DATE,
   when    DATE,
   iid     INTEGER,
   PRIMARY KEY  (cid),
   FOREIGN KEY (iid) REFERENCES Initiative)
);
```

Key Constraints



- What about folding everything into **one** table?

Key Constraints

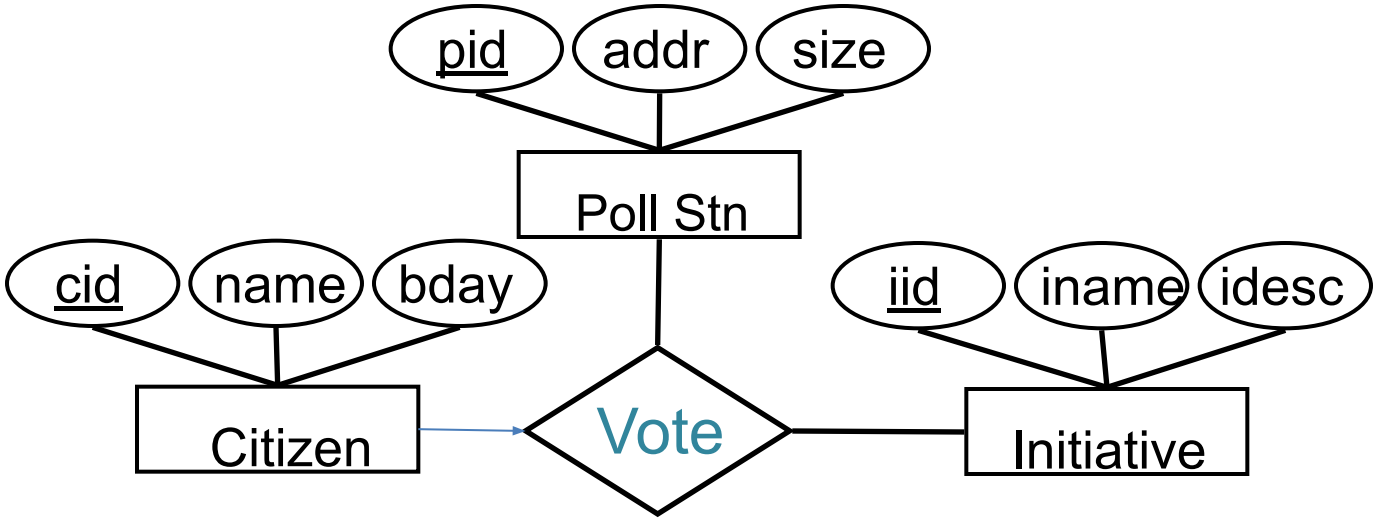


- What about folding everything into **one** table?
- No! This is bad design.

e.g., For every citizen that votes for Initiative A, we have to store iname, idesc information.

(cid, name, bday, when, iid,
iname, idesc) <= REDUNDANCY!

N-ary relationship Example

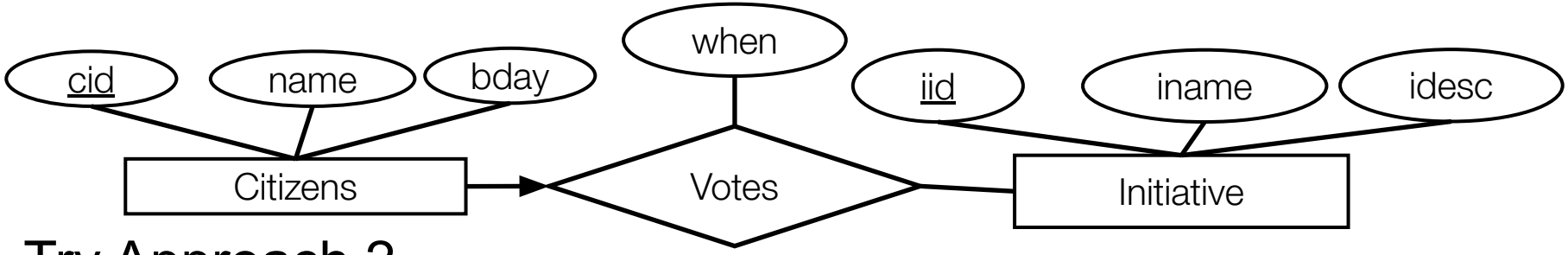


How many tables
Using Approach 1?

Using Approach 2?

```
APPROACH 2:
CREATE TABLE Citizens_Votes
(  cid     INTEGER,
   name    CHAR(20),
   bday    DATE,
   pid     INTEGER,        -- NOTE: allowed to be
NULL
   iid     INTEGER,        -- NOTE: allowed to be
NULL
   PRIMARY KEY  (cid),
   FOREIGN KEY (iid) REFERENCES Initiative,
```

Participation Constraints

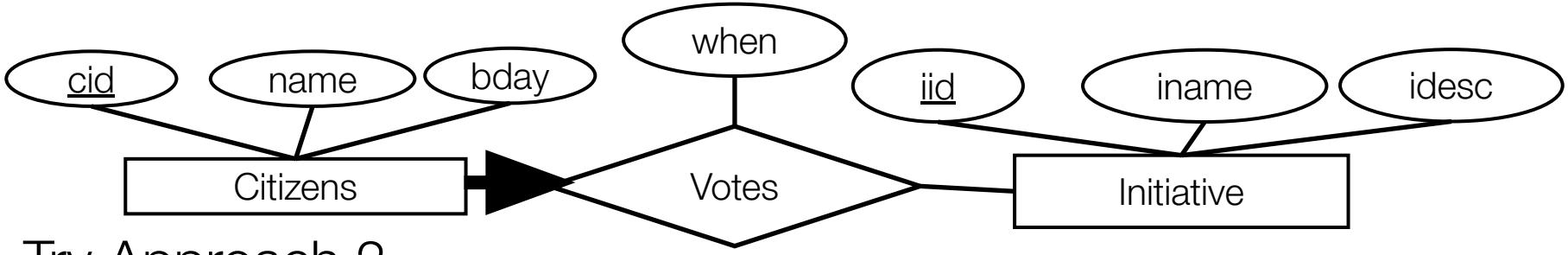


Try Approach 2

```
CREATE TABLE  Citizen_Votes(
    cid     INTEGER,
    name    CHAR(20),
    bday    DATE,
    when    DATE,
    iid     INTEGER,
   PRIMARY KEY  (cid),
   FOREIGN KEY  (iid) REFERENCES Initiative));
```

What ER constraint is missing?

Participation Constraints

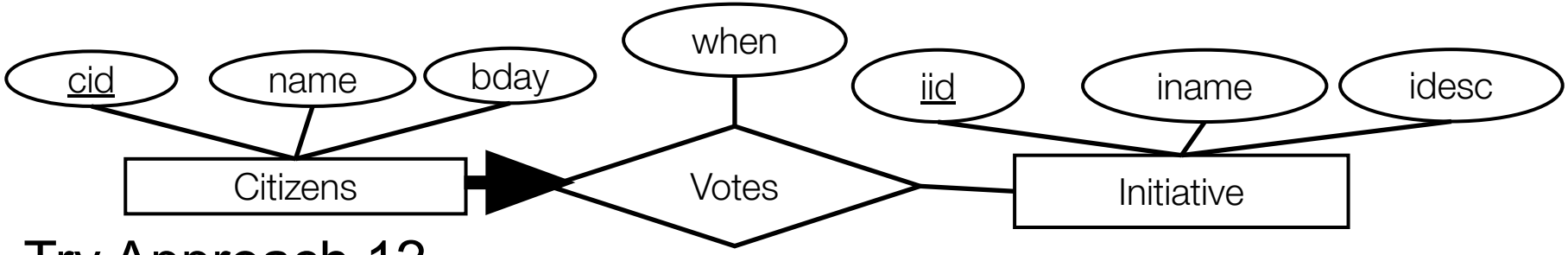


Try Approach 2

```
CREATE TABLE  Citizen_Votes(
    cid     INTEGER,
    name    CHAR(20),
    bday    DATE,
    when    DATE,
    iid     INTEGER NOT NULL,
   PRIMARY KEY  (cid),
   FOREIGN KEY  (iid) REFERENCES Initiative));
```

What ER constraint is missing?

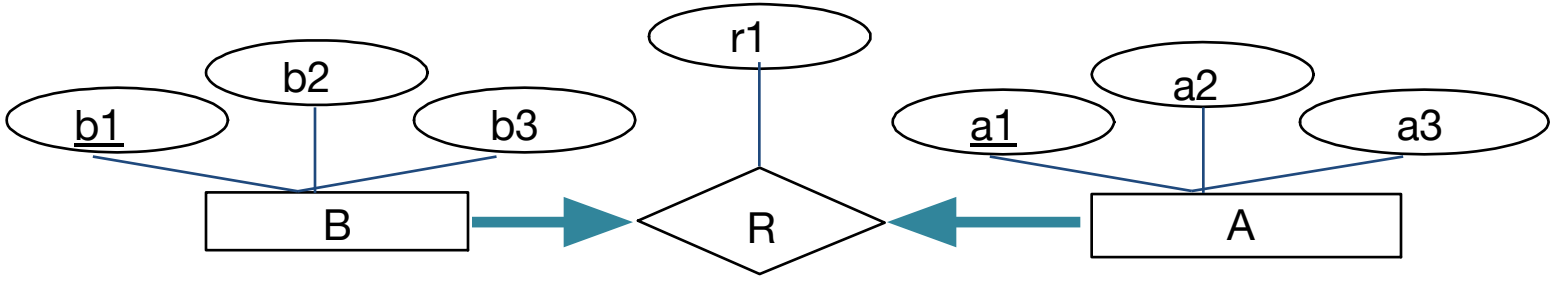
Participation Constraints



Try Approach 1?

- Citizens table, Votes table, Initiative Table
- How do you guarantee that every citizen in the Citizens table also appears in the Votes table?
- Unfortunately, this is hard to enforce in SQL. Requires multi-table constraints, which are usually not supported in DBMSes.

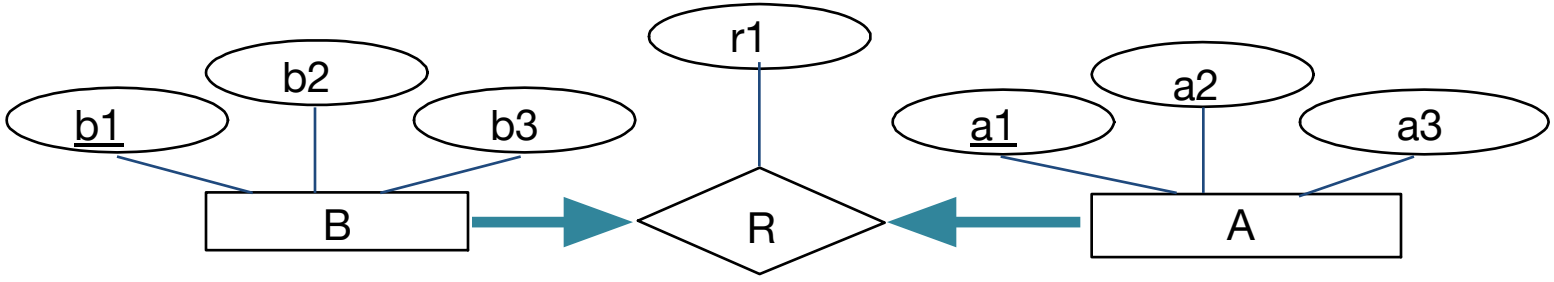
Mapping Participation Constraints (1-to-1)




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

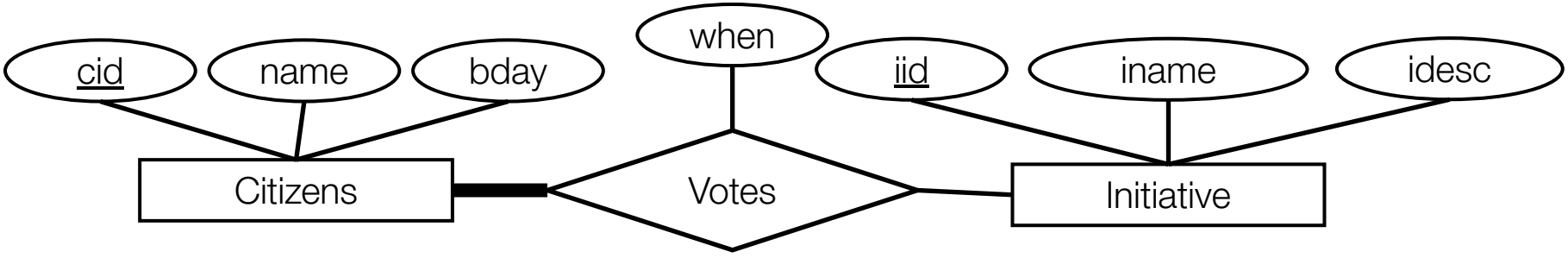
Key constraints?

Mapping Participation Constraints (1-to-1)




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

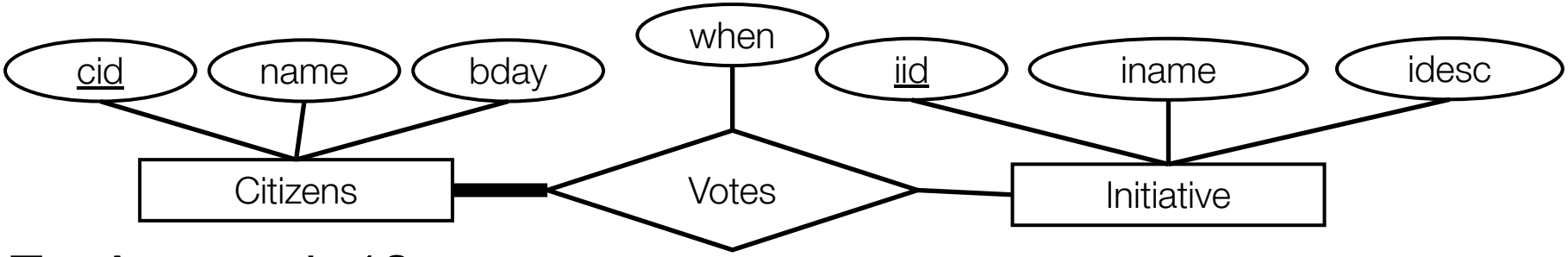
Participation Constraint Only



Try Approach 2?

- Doesn't work since a Citizen can vote multiple times.

Participation Constraint Only



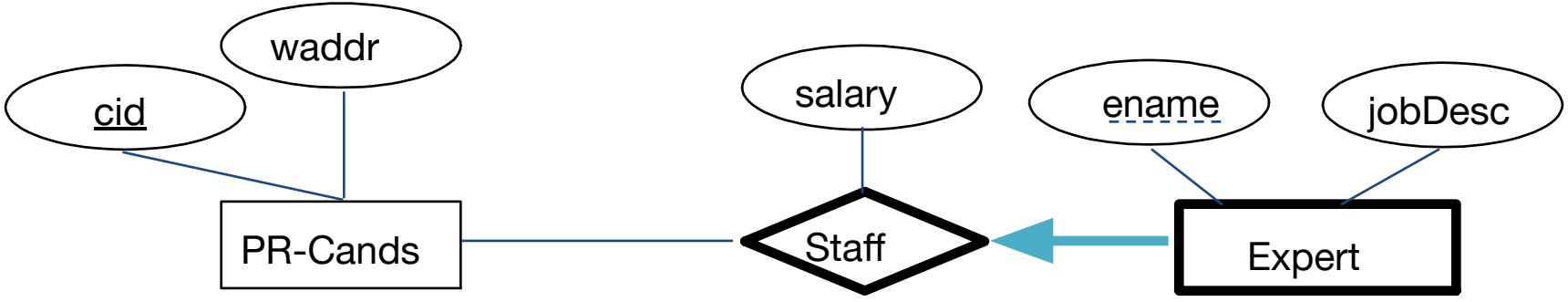
Try Approach 1?

- Citizens table, Votes table, Initiative Table
- How to guarantee that every Citizen appears in the Votes table?
- Unfortunately, this is hard to enforce in SQL. Requires multi-table constraints, which are usually not supported in DBMSes.

- **Typical Practical Solution:** Ignore participation-only constraints – don't bother enforcing them in the database.

Use Approach 1

Weak Entities

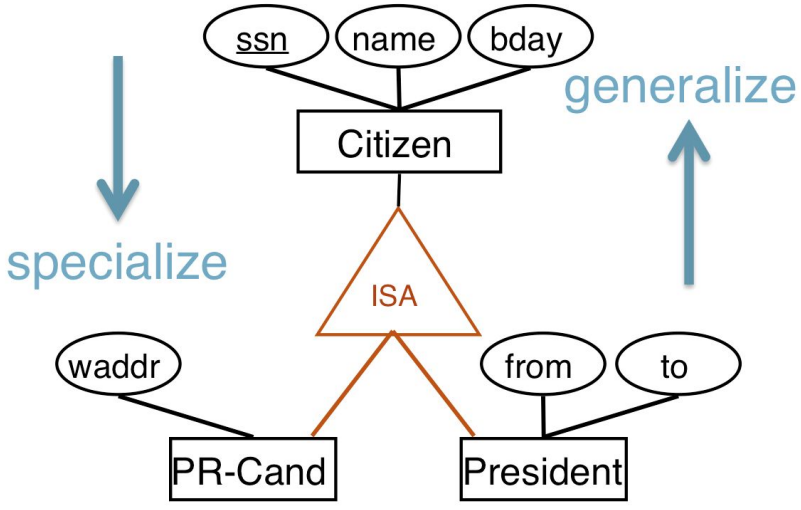


- Use Approach 2: Combine weak entity and owning relationship into one relation
  - Delete all weak entities when an owner entity is deleted.

```
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)
```

ISA Hierarchies: General Approach

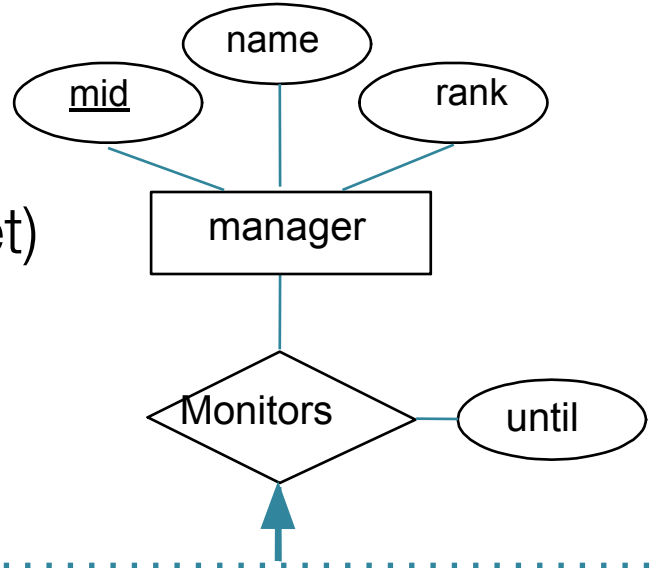
- Three relations:
  - Citizens(ssn, name, bday)
  - PR-Cands(ssn, waddr, budget)
  - President(ssn, from, to)
  - Add foreign key constraints



- Queries:
  - Involving all citizens => Easy
  - Involving just PR-Cands => may need to join PR-cands with Citizens to get the needed attributes

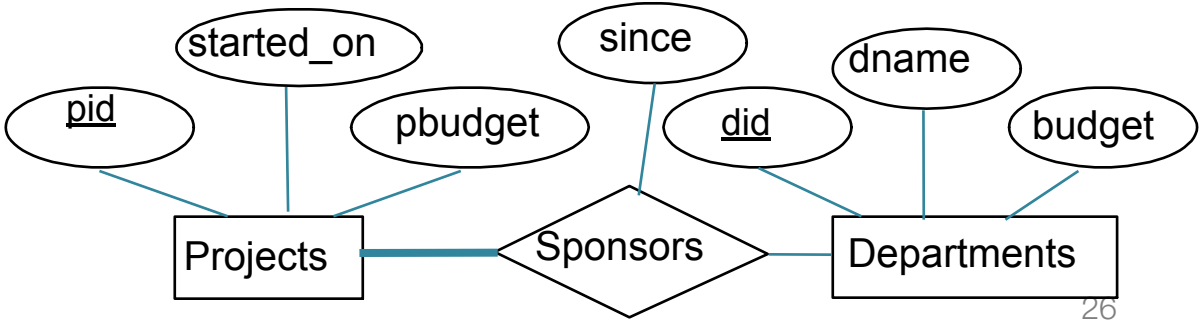
Aggregation – E.g., Monitors

- Tables needed (Approach 1):
  - Projects (pid, started, pbudget)
  - Departments (did, dname, budget)
  - Sponsors (pid, did, since)
  - Manager(mid, name, rank)
  - Monitors(pid, did, mid, until)



Exercise:

Try Approach 2

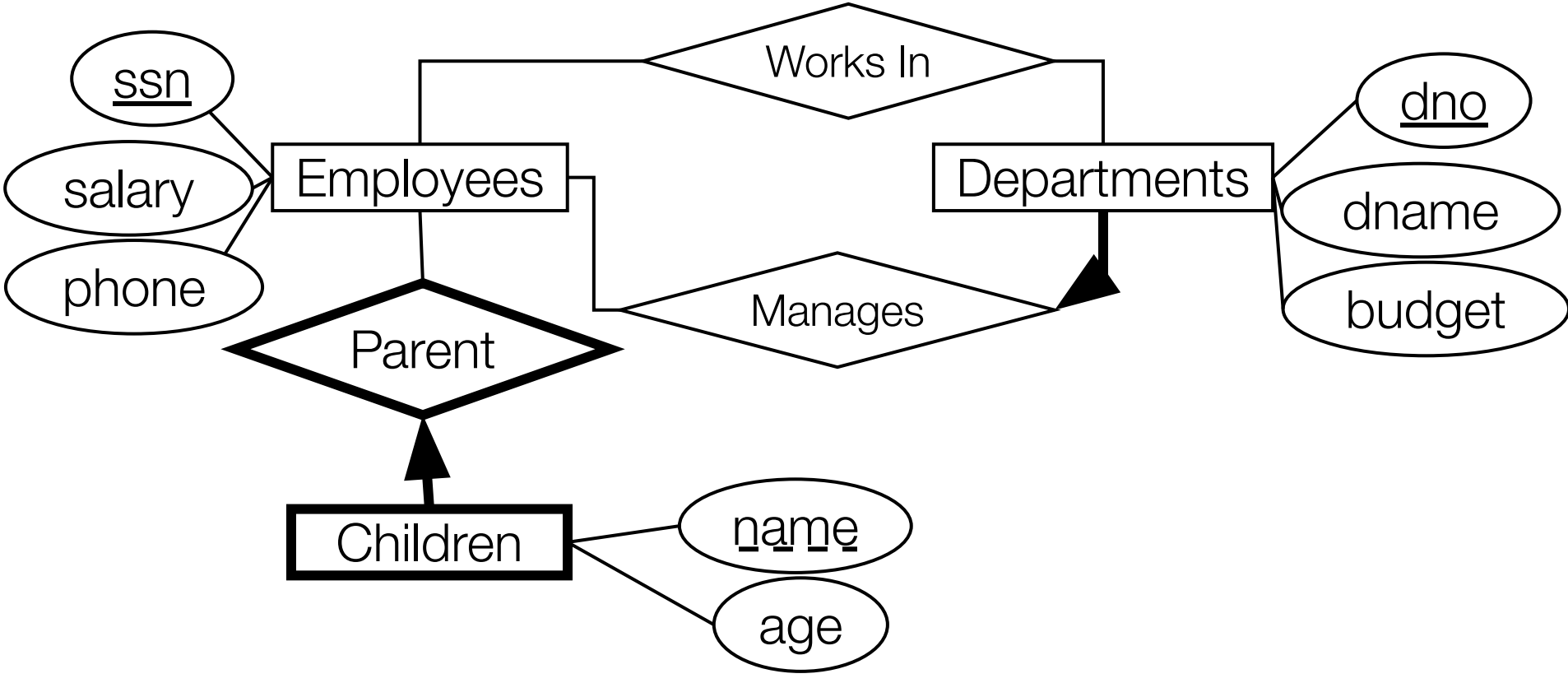


On your own

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)



On your own

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(ssn))
```

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

Integrity Constraints

- Describe 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

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

```
CREATE TABLE Athlete
(aid INTEGER PRIMARY KEY,
 name CHAR(30),
 age INTEGER,
 country CHAR(20),
 sport CHAR(20)),
 CHECK  ( age >= 18 AND age <= 80 ));
```

Try it out in sqlplus or sqlite or duckDB

```
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?

```
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);
```

```
> Error: CHECK constraint failed: Sailors
```

More general CHECKs

- SQL standard allows cross-table CHECK constraints
- But, they are not supported in most systems – expensive to enforce.
- Thus, we will not worry about them.

Active Databases & Triggers

Trigger: Procedure that starts automatically if specified changes occur to the DBMS

- Three parts:
  - Event (activates the trigger)
  - Condition (test that is run when the trigger is activated)
  - 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 – Log entries

```
CREATE TABLE Employee (
Empno NUMBER PRIMARY KEY,
Ename VARCHAR2(10),
Job VARCHAR2(9),
Mgr NUMBER(4),
Hiredate DATE,
Sal NUMBER(7,2));
```

```
CREATE TABLE Emp_log (
 Emp_id NUMBER,
 Log_date DATE,
New_salary NUMBER,
Log_entry VARCHAR2(20));
```

```
CREATE OR REPLACE TRIGGER Log_salary_increase AFTER UPDATE
ON Employee FOR EACH ROW WHEN (new.Sal > 1000)
BEGIN
        INSERT INTO Emp_log (Emp_id, Log_date, New_salary, Log_entry)
            VALUES   (:new.Empno, SYSDATE, :new.SAL, 'NEW SAL');
END;
```

This is a row-level trigger (triggered per row update). If new salary above 1000, log entry is added.

E.g. by executing this SQL:

```
UPDATE Emp_tab SET Sal = Sal + 1000.0 WHERE
Job = 'Professor';
```

Example – Log entries

```
CREATE TABLE Employee (
Empno NUMBER PRIMARY KEY,
Ename VARCHAR2(10),
Job VARCHAR2(9),
Mgr NUMBER(4),
Hiredate DATE,
Sal NUMBER(7,2));
```

```
CREATE TABLE Emp_log (
Log_date DATE,
Log_entry VARCHAR2(20));
```

```
CREATE OR REPLACE TRIGGER Log_emp_update
     AFTER UPDATE ON Employee
BEGIN
    INSERT INTO Emp_log (Log_date, Log_entry)
       VALUES (SYSDATE, 'Employee Table CHANGED');
END;
```

This is a statement-level trigger. The trigger will fire once per update statement.

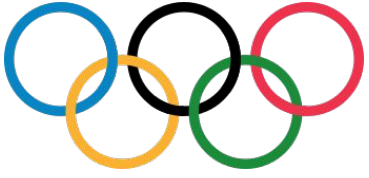
Triggers

- When condition is checked, success/failure can be used to trigger arbitrary actions.
- Used for many things:
  - Check complex actions (such as credit limit in a shopping application)
  - Generate logs for auditing and security checks.

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,
     FOREIGN KEY (oid) REFERENCES Olympics
     );
```

CASCADE Using Triggers

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

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

- Try out the file athlete_trigger_cascade.sql in sqlplus
- Also try it out in sqlite to see if it supports triggers the same way
- Try removing a row from athlete. Does it cascade to Compete via the trigger?
- Try dropping the trigger:
  - `DROP TRIGGER triggername;`
- What happens now on deleting a row from Athlete?

Triggers: Pitfalls and Pain

- Triggers can be recursive!
  - Chain of triggers can be hard to predict, which makes triggers difficult to understand and debug
- Errors with “mutating” table
  - A table that is currently being modified by an UPDATE, DELETE, or INSERT statement, or a table that might be updated by the effects of a DELETE CASCADE constraint
  - The session that issued the triggering statement cannot query or modify a mutating table
  - Used to prevent a trigger from seeing **inconsistent data**