Data Modeling

Database Systems: The Complete Book
Ch. 4.1-4.5, 7.1-7.4

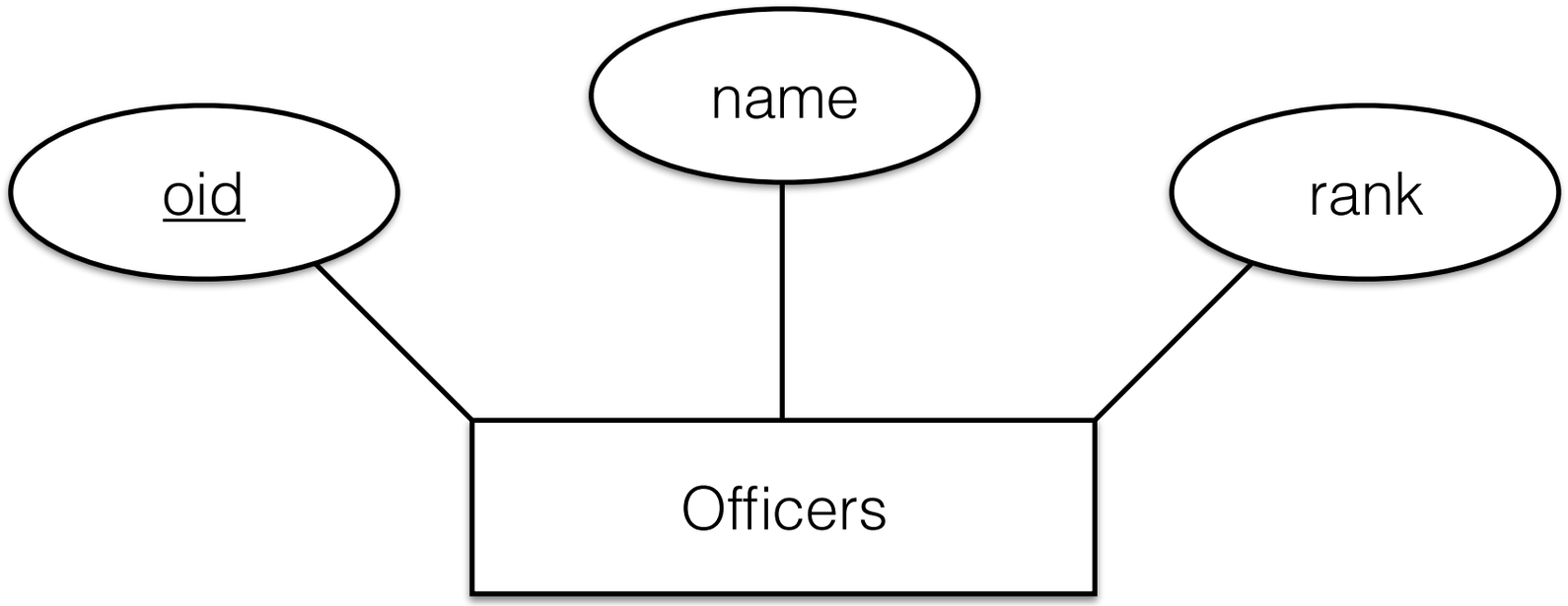
Data Modeling

- Schema: The structure of the data
  - Structured Data: Relational, XML-DTD, etc…
  - “Unstructured” Data: CSV, JSON
- But where does the schema come from?
  - Data represents concepts!
  - Model the concepts

Entity-Relation Model

- A pictorial representation of a schema
  - Enumerates all entities in the schema
  - Shows how entities are related
  - Shows what is stored for each entity
  - Shows restrictions (integrity constraints)

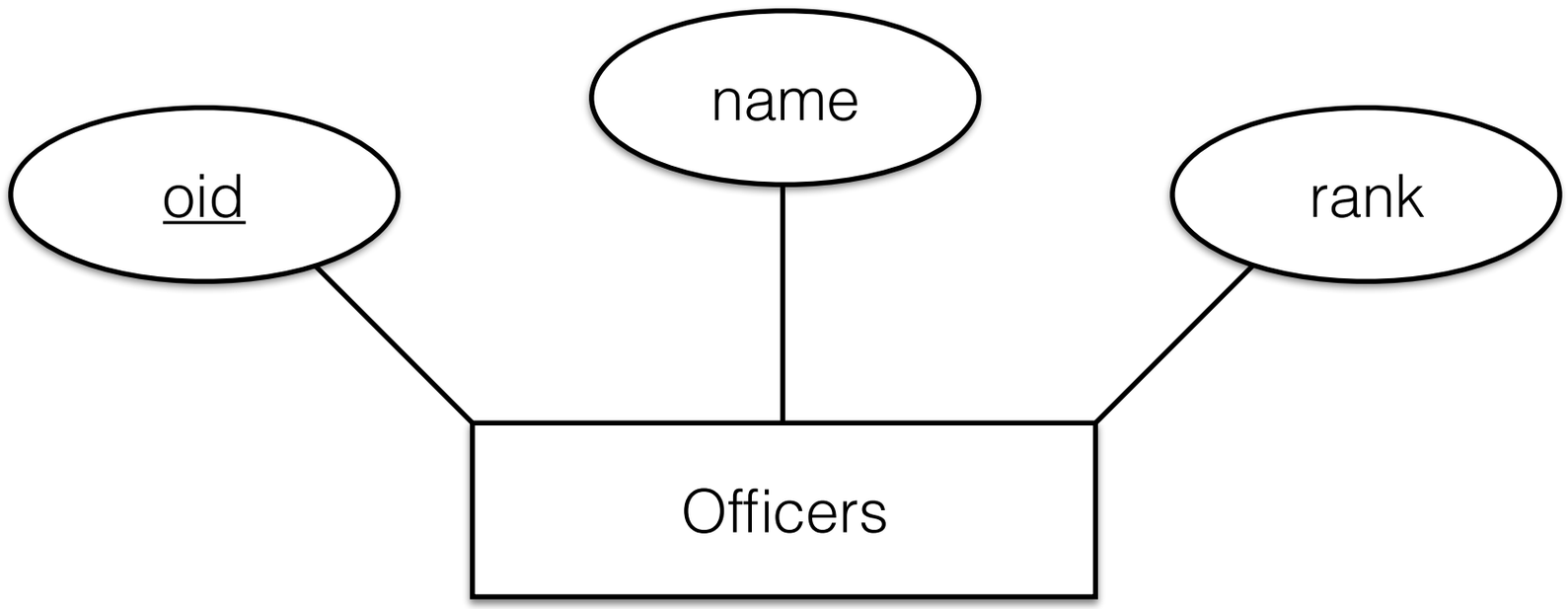
ER Model Basics



Entity: A real-world object distinguishable from other objects. *(e.g., a Starfleet Officer)*

An entity is described through a set of attributes

ER Model Basics

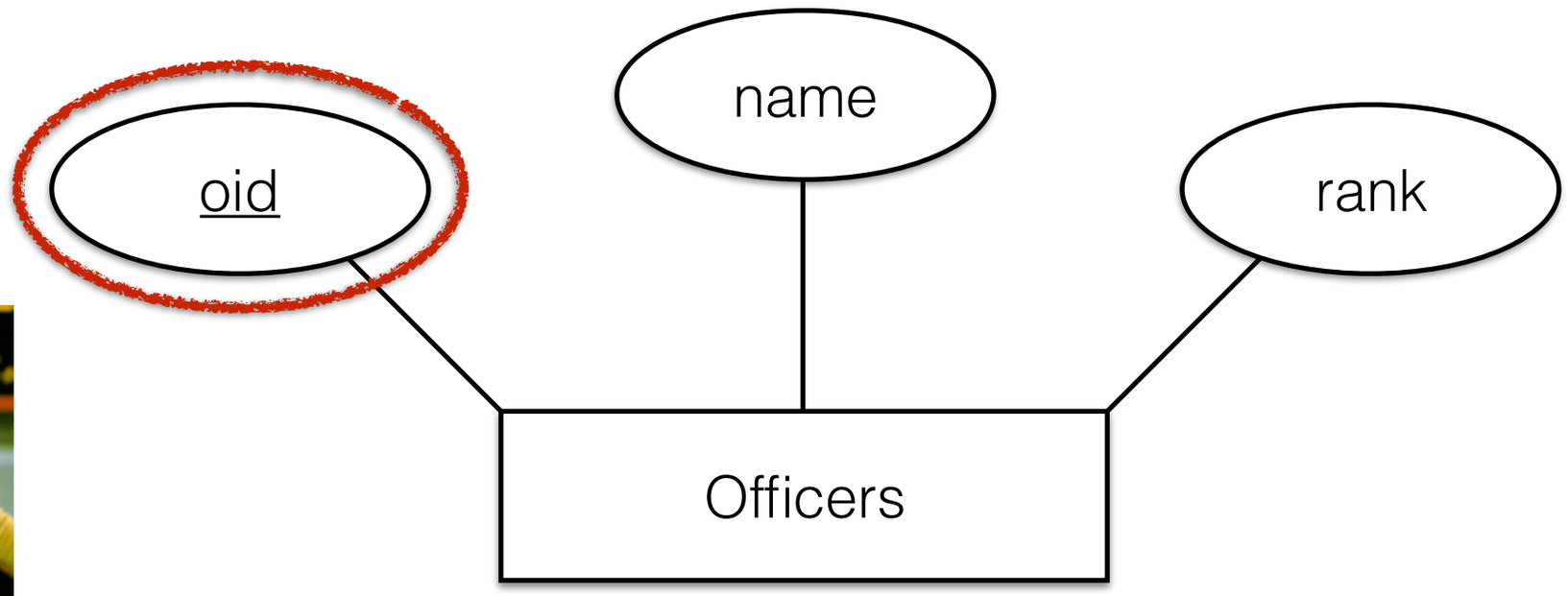


Entity Set: A collection of similar entities. *(e.g., all Officers)*

Entities in an entity set have the same set of attributes

Each attribute has a domain *(e.g., integers, strings)*

ER Model Basics

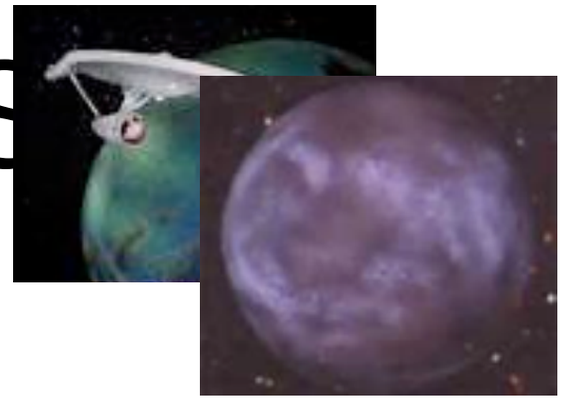


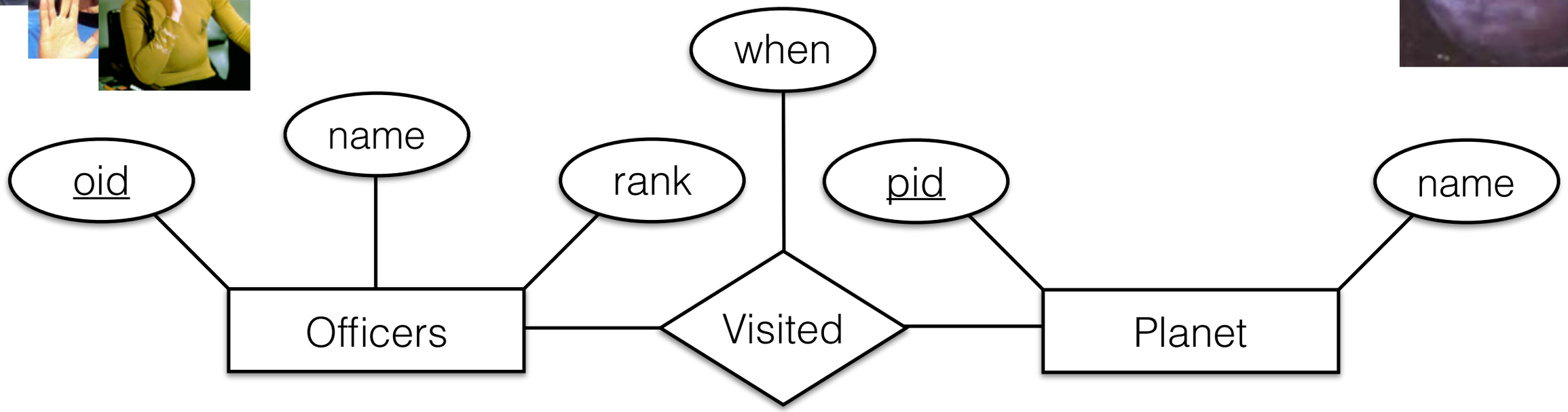
Entity sets must have a key, an attribute (or combination of attributes) guaranteed to be unique for every entity in the set.

- Officer ID for officers
- Ship ID for ships
- UBIT for UB students
- Course Code+Semester for courses

Keys are underlined in ER Diagrams

ER Model Basics

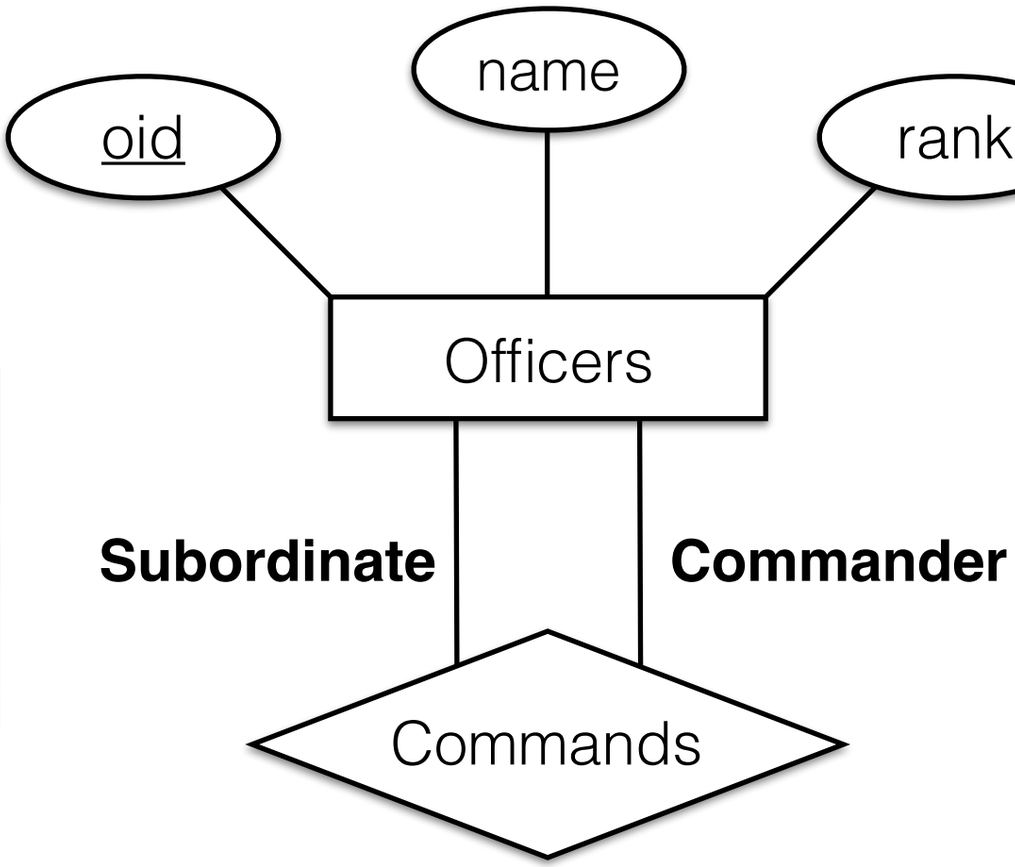




Relationship: Associations between 2 or more entities.
Relationship Set: A collection of similar relationships.
(an n-ary relationship set relates Entity sets E_1-E_n)

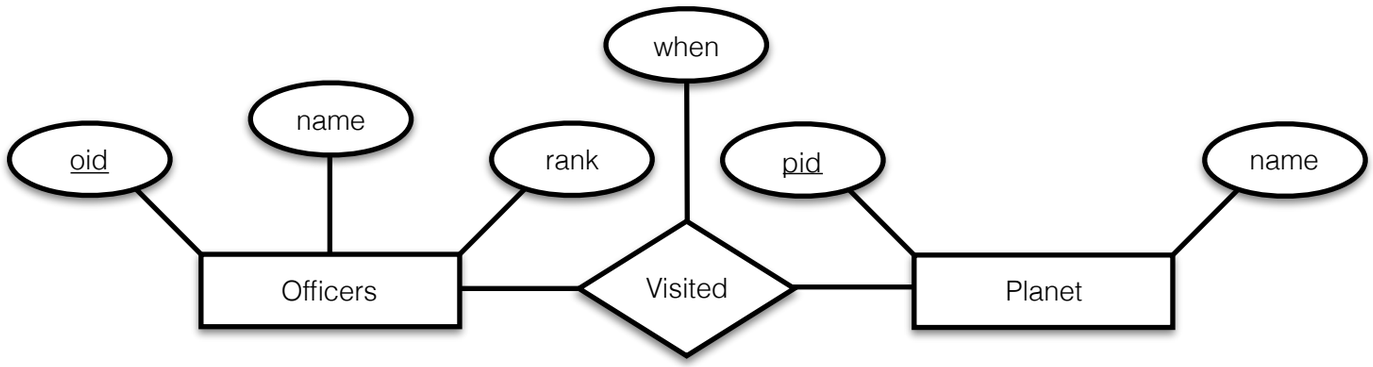
Relationships may have their own attributes.

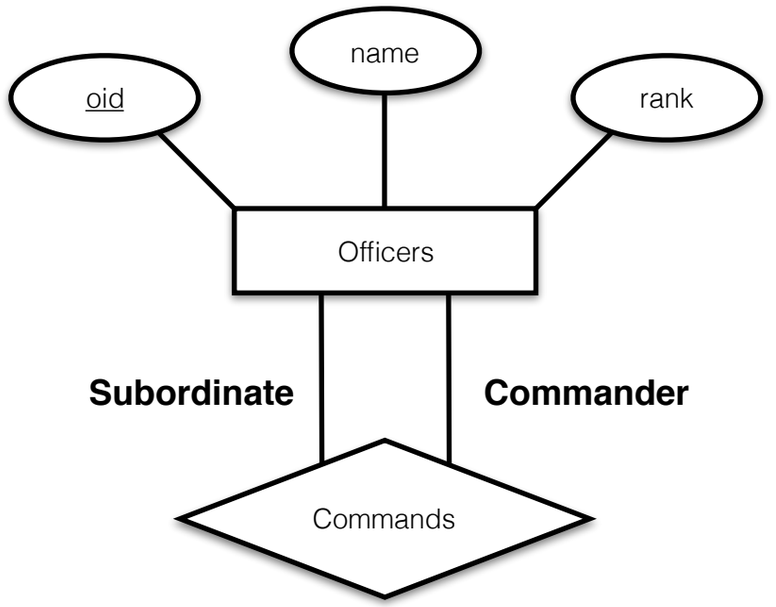
ER Model Basics

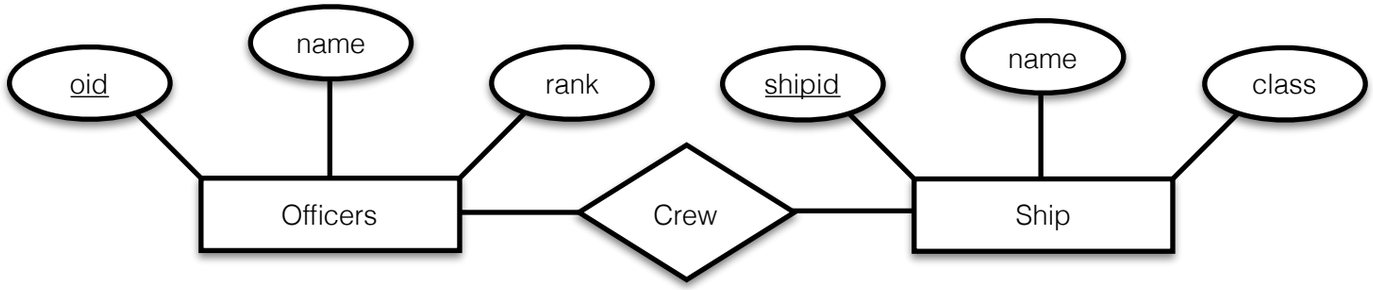


There can be relationships between entities in the same entity sets

Key Constraints



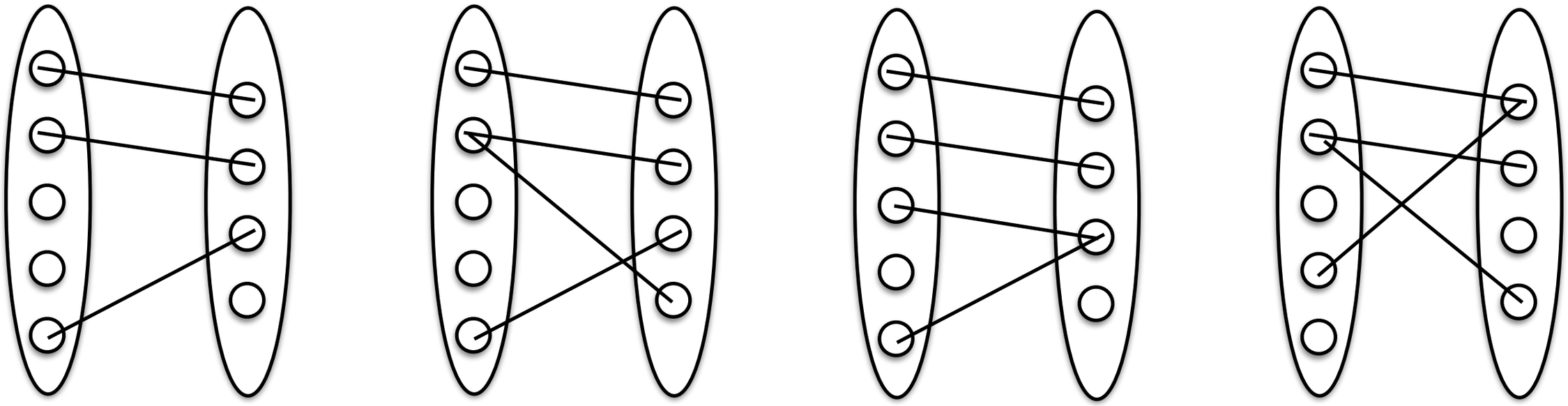




Consider these relationships

- One ship can have many crew, but each crew member has only one ship
- Each officer has one commander, but officers might have many subordinates
- Each planets may have been visited by many officers, and each officer may have visited many planets

Key Constraints

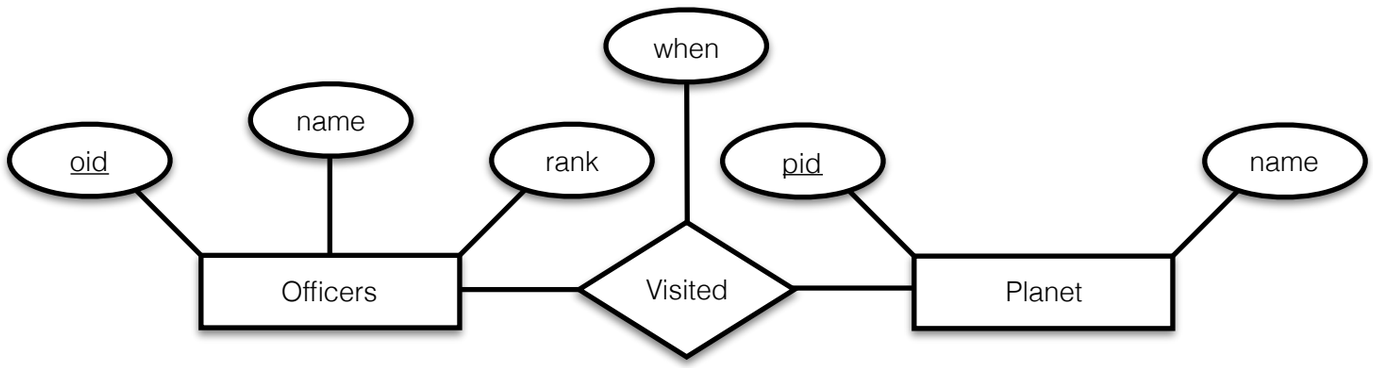


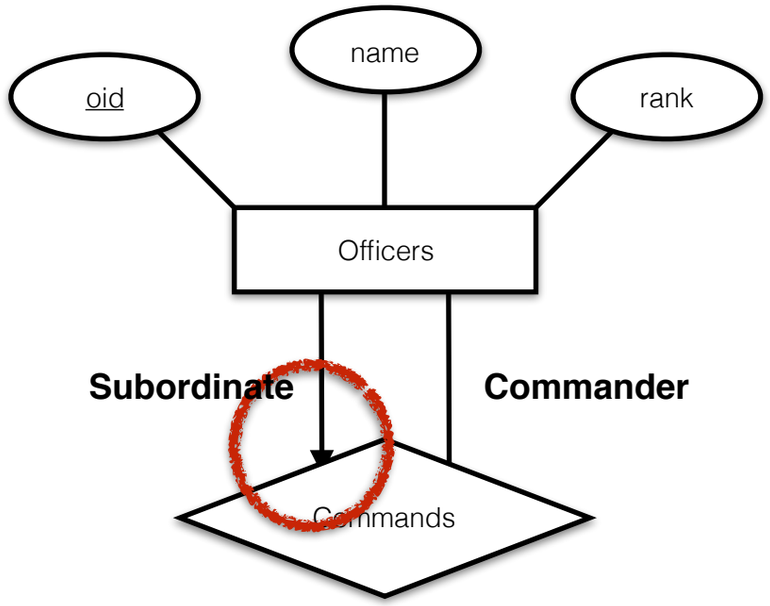
1-to-1 1-to-Many Many-to-1 Many-to-Many

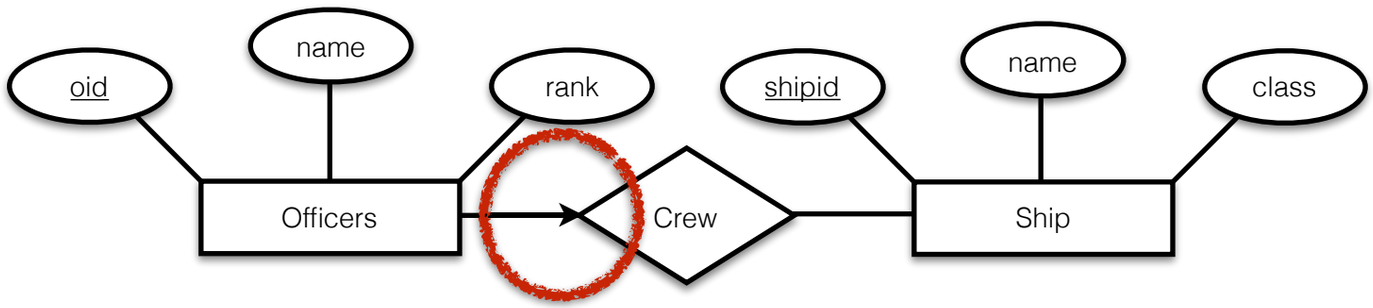
Consider these relationships

- One ship can have many crew, but each crew member has only one ship
- Each officer has one commander, but officers might have many subordinates
- Each planets may have been visited by many officers, and each officer may have visited many planets

Key Constraints



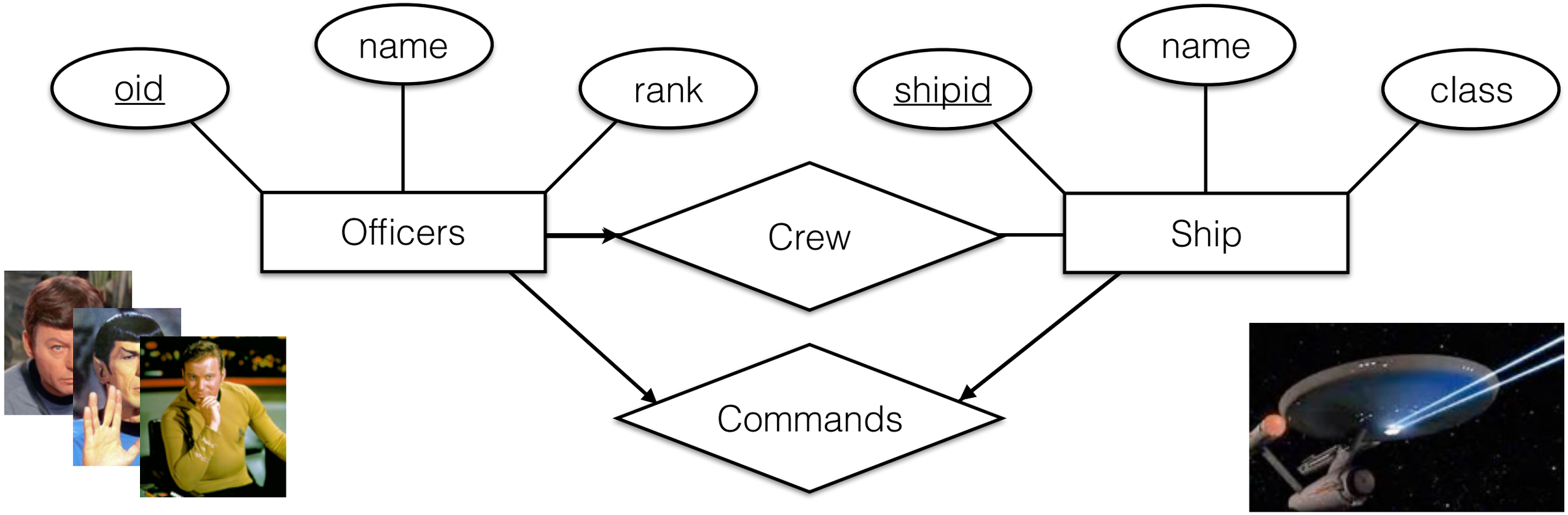




Key constraints identify entities that participate in **at most one** relationship in a relationship set

We denote key-constraints with an arrow

Participation Constraints



Every Ship must have crew, and every officer must crew a ship.
Every Ship must have a commander.

Participation constraints require participation in a relationship
(and are denoted as bold lines)

Weak Entities

A weak entity can be identified uniquely only relative to the primary key of another (owner) entity.

The weak entity must participate in a one-to-many relationship (one owner, many weak entities)

ISA (‘is a’) Hierarchies

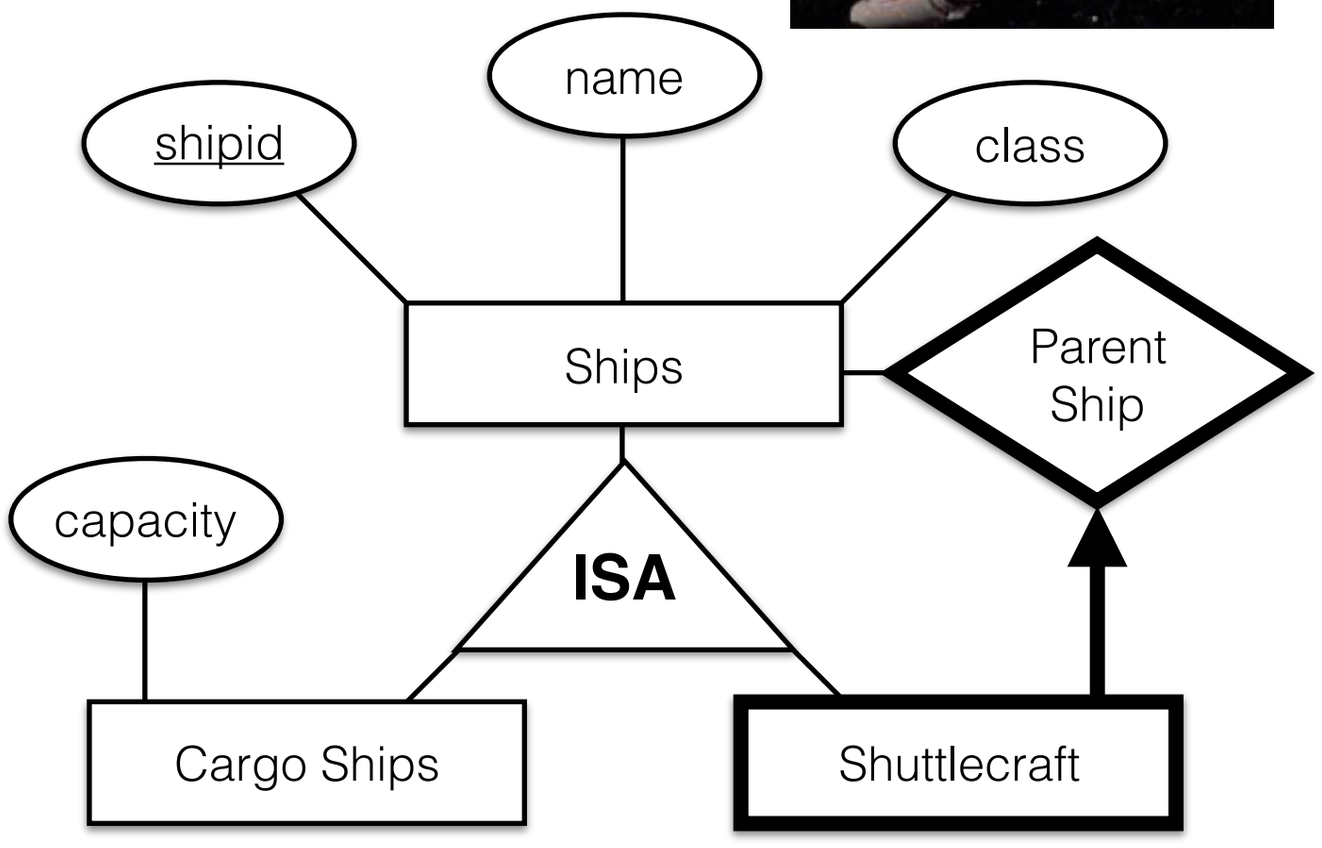
ISA Hierarchies define entity inheritance
If we declare **A ISA B**, then
every A is also considered to be a B

Overlap constraints: Can a ship be a cargo ship and a shuttlecraft?

Covering constraints: Does every ship have to be a cargo ship or a shuttlecraft?

<u>Reasons for using ISA</u>:
Adding descriptive attributes specific to a subclass (cargo ship capacity)

Identifying entities in a specific type of relationship (shuttlecraft of a big ship)



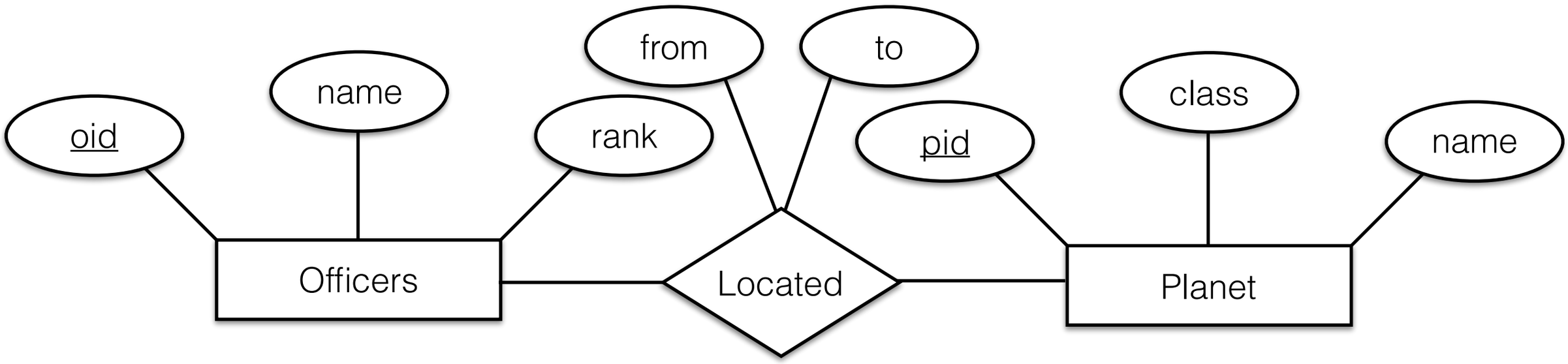
Conceptual Design in ER

- Design choices
  - Should a concept be modeled as an entity or an attribute of another entity?
  - Should a concept be modeled as an entity or a relationship between entities?
  - What kind of relationship: Binary, Ternary, N-ary?
- Constraints
  - A lot of data semantics can (and should) be captured.
  - Not all constraints are expressible in ER diagrams.

Entity vs Attribute

- Expressing the Location of an Officer
  - **Option 1**: An attribute of **Officers**
  - **Option 2**: A **Planets** entity set and a relationship set **Location**
- Which we use depends on the semantics of the data.
  - Can an **Officer** have multiple locations? (e.g., transporter accidents, time travel, etc…)
    - Attributes are single-valued, model **Planets** as entities.
  - Are the details of locations relevant to queries? (i.e., Find all officers on a Class-M planet).
    - Attributes are atomic, model **Planets** as entities.

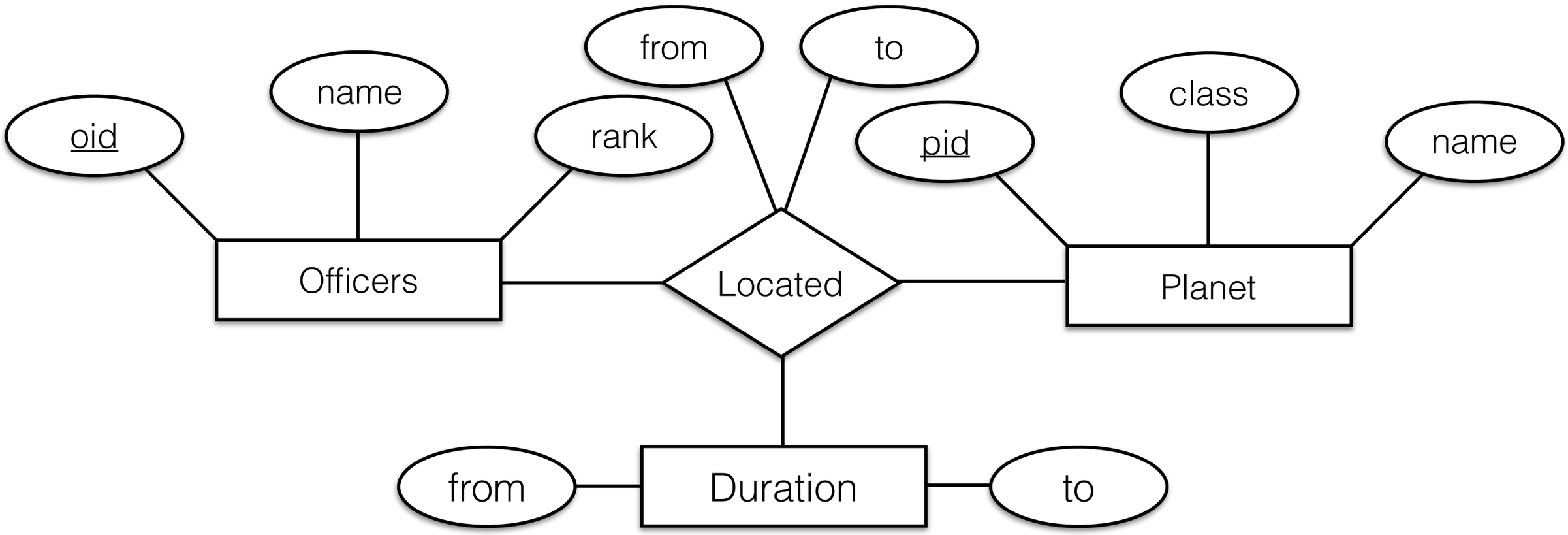
Entity vs Attribute



Problem: Can only have one location for each officer (no time ranges)

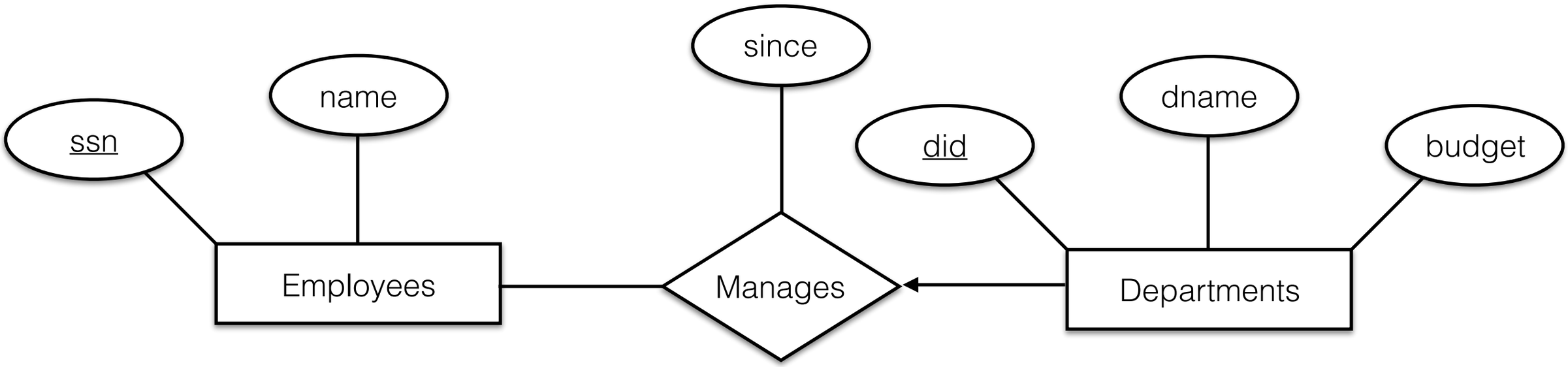
We want to encode multiple instances of the descriptive attributes of the relationship instance

Entity vs Attribute



Solution: Add a duration entity and make location a ternary relationship

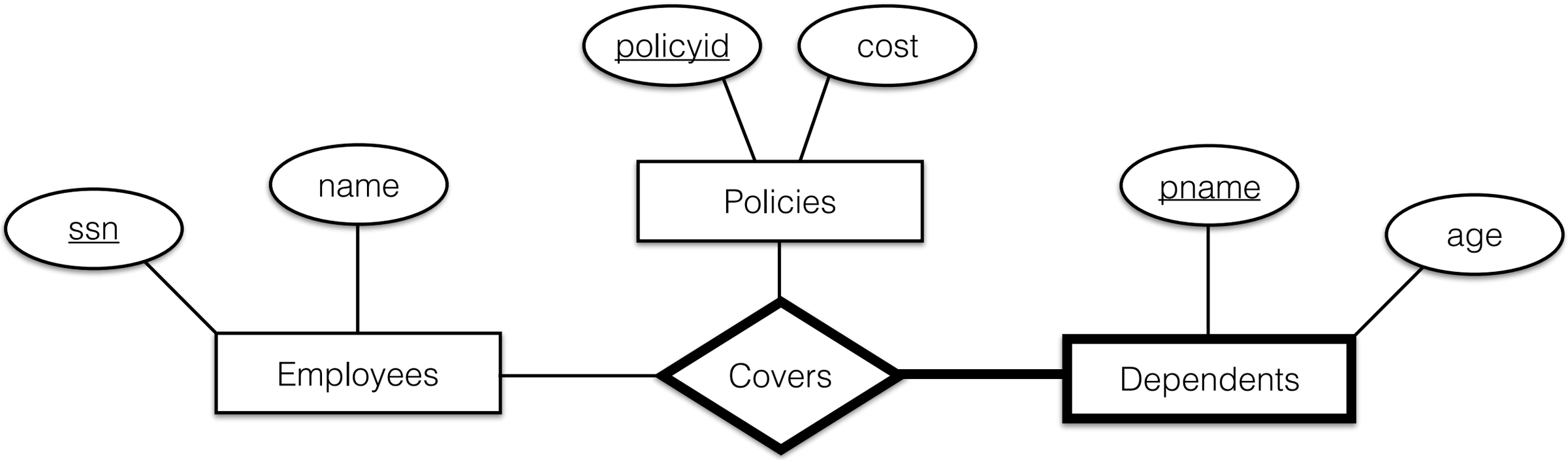
Group Work



Managers have a discretionary budget (dbudget) for each dept.

How would we modify this ER diagram if the budget were per-manager, rather than per-department

Group Work



1) What are some limitations of this ER Diagram?

2) Design an ER Diagram that addresses these issues.

Integrity Constraints

- "Correctness" Properties on Relations
  - … enforced by the DBMS.
- Typically simple uniqueness/existence properties, paralleled by ER Constraints
  - … we'll discuss more complex properties when we discuss Triggers later in the term.
- Database optimizers benefit from constraints.

Integrity Constraints

- Domain Constraints
  - Limitations on valid values of a field.
- Key Constraints
  - A field(s) that must be unique for each row.
- Foreign Key Constraints
  - A field referencing a key of another relation.
  - Can also encode participation/1-many/many-1/1-1.
- Table Constraints
  - More general constraints based on queries.

Domain Constraints

- Stronger restrictions on the contents of a field than provided by the field's type
  - e.g., 0 < Rank ≤ 5
    - Mostly present to prevent data-entry errors.

Postgres:

```
CREATE DOMAIN Rank AS REAL
   CHECK (0 < VALUE AND VALUE <= 5)
```

Oracle:

```
CREATE TABLE Officers (
   …
   Rank REAL,
   CHECK (0 < Rank AND Rank <= 5) );
```

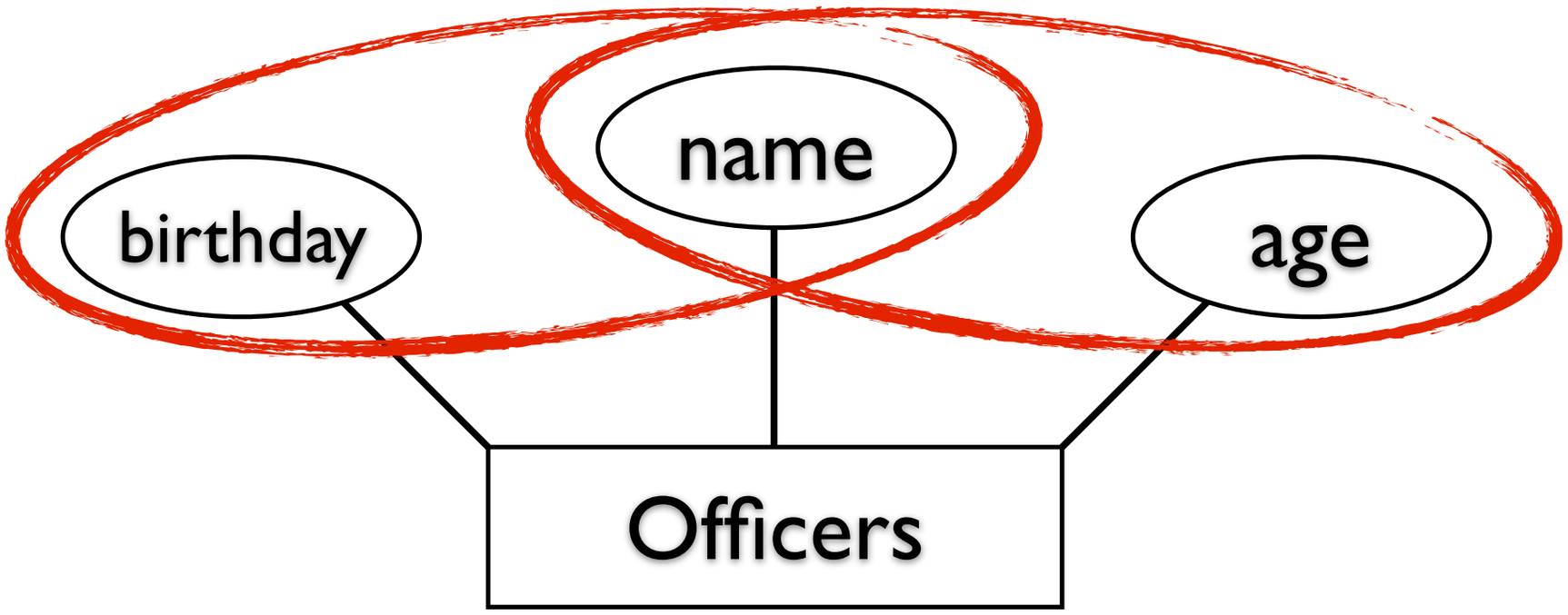
Domain Constraints

- Special domain constraint: NOT NULL
  - Field not allowed to contain NULL values.

```
CREATE TABLE Officer(
   oid INTEGER NOT NULL,
   name CHAR(50),
   birthday DATE
);
```

Key Constraints

- A set of fields that uniquely identifies a tuple in a relation.
- There can be multiple keys for a relation.

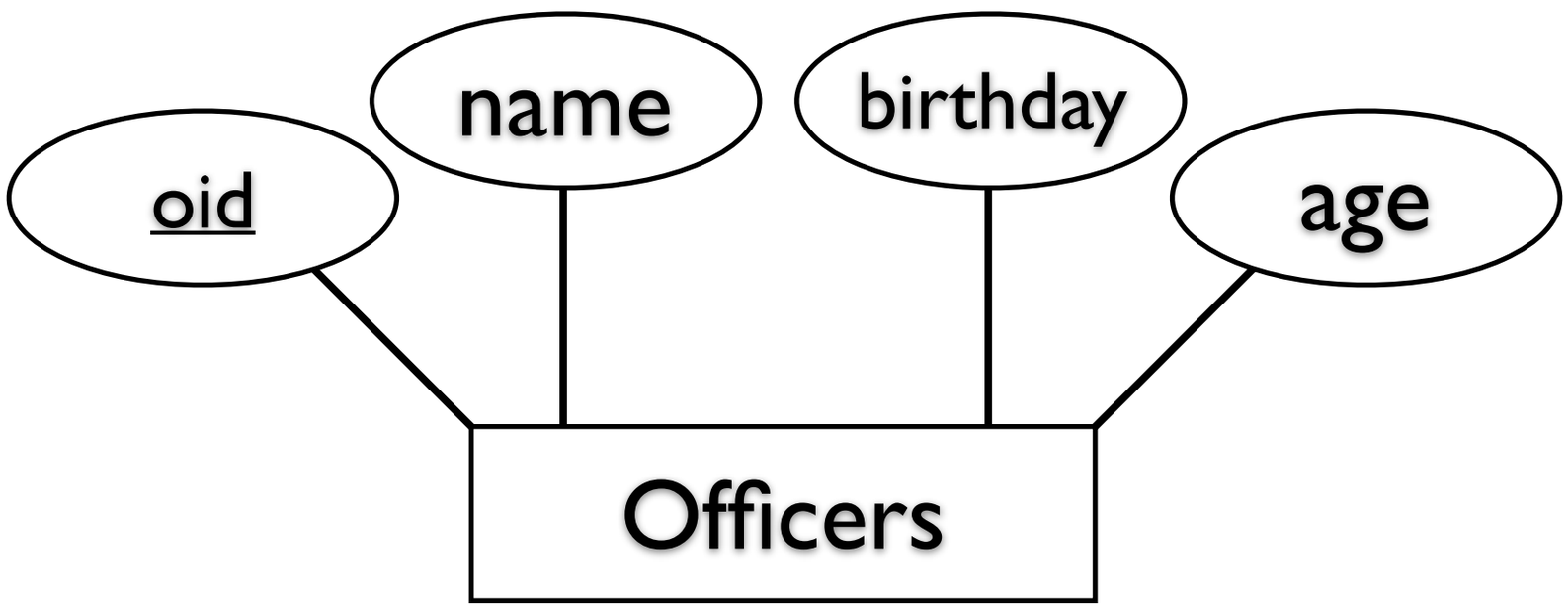


Key Constraints

- A *key* satisfies the following two properties:
  - No two distinct tuples have identical values in all the fields of a key.
    - Two officers can have the same name, or the same birthday/age, but not both name and birthday/age.
- No subset of the fields of a key has the above property.
  - Name+Age+Birthday is not a key (it is a *superkey*)
  - Name+Age is a key, and Name+Birthday is a key.

Defining Key Constraints

```
CREATE TABLE Officer(
  oid INTEGER, name CHAR(50),
  birthday DATE, age REAL,
  UNIQUE (name, age),
  CONSTRAINT OfficerDay UNIQUE (name, birthday),
  PRIMARY KEY (oid)
);
```



Defining Key Constraints

```
CREATE TABLE Officer(
   oid INTEGER, name CHAR(50),
   birthday DATE, age REAL,
   UNIQUE (name, age),
   CONSTRAINT OfficerDay UNIQUE (name, birthday),
   PRIMARY KEY (oid)
);
```

UNIQUE identifies a key constraint

Defining Key Constraints

```
CREATE TABLE Officer(
  oid INTEGER, name CHAR(50),
  birthday DATE, age REAL,
  UNIQUE (name, age),
  CONSTRAINT OfficerDay UNIQUE (name, birthday),
  PRIMARY KEY (oid)
);
```

UNIQUE identifies a key constraint

PRIMARY KEY identifies a key constraint that will commonly be used to refer to tuples in this relation.

Defining Key Constraints

```
CREATE TABLE Officer(
  oid INTEGER, name CHAR(50),
  birthday DATE, age REAL,
  UNIQUE (name, age),
  CONSTRAINT OfficerDay UNIQUE (name, birthday),
  PRIMARY KEY (oid)
);
```

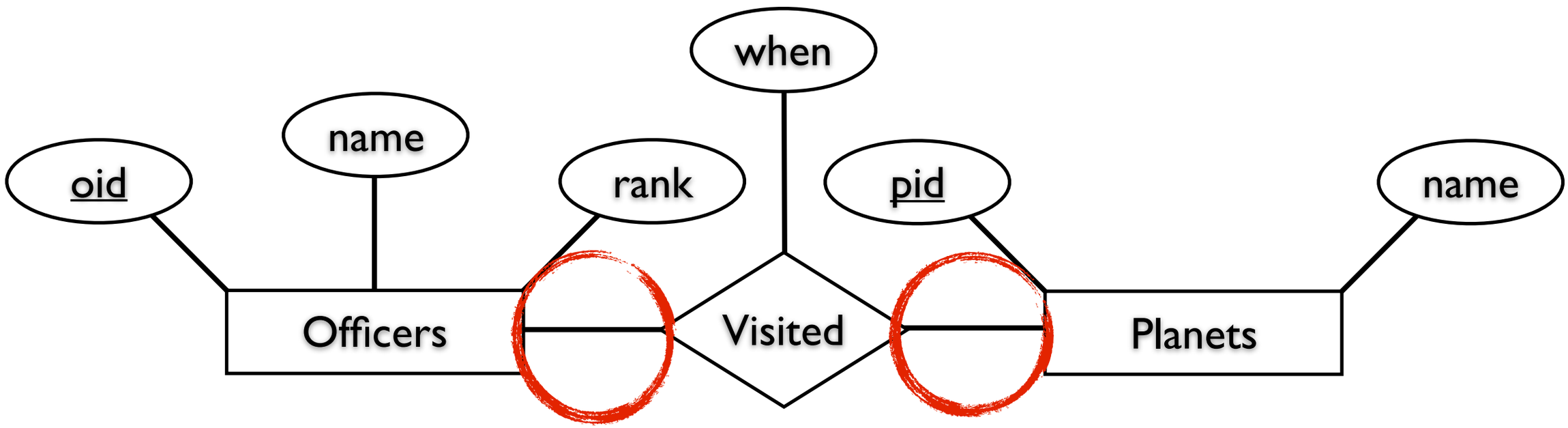
UNIQUE identifies a key constraint

PRIMARY KEY identifies a key constraint that will commonly be used to refer to tuples in this relation.

CONSTRAINT (optionally) assigns a name to any constraint.

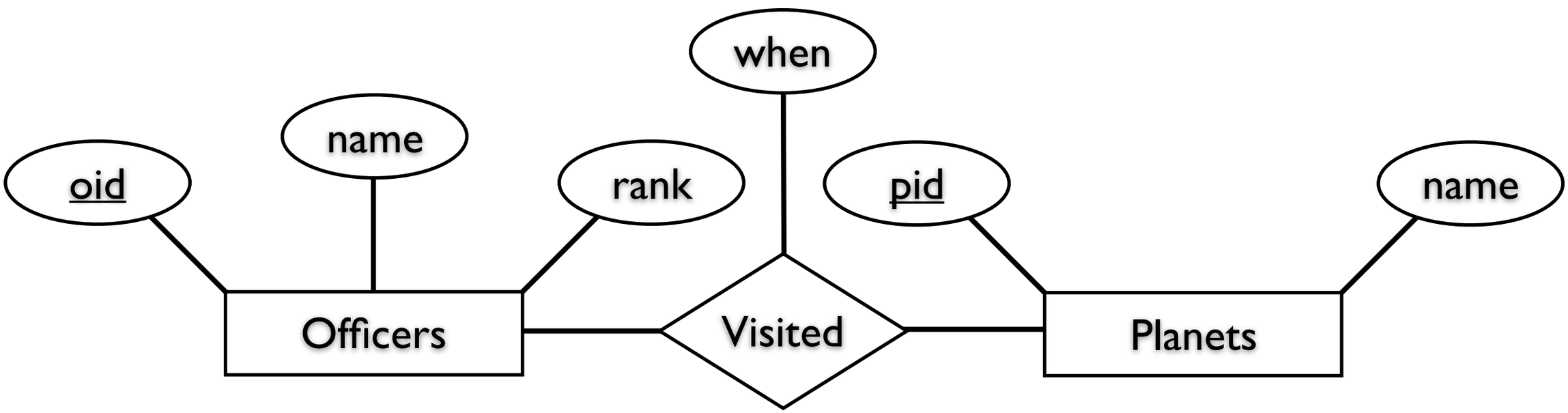
Foreign Key Constraints

- Used when a tuple in one relation needs to refer to a tuple in a different relation.
- The referenced tuple must exist.



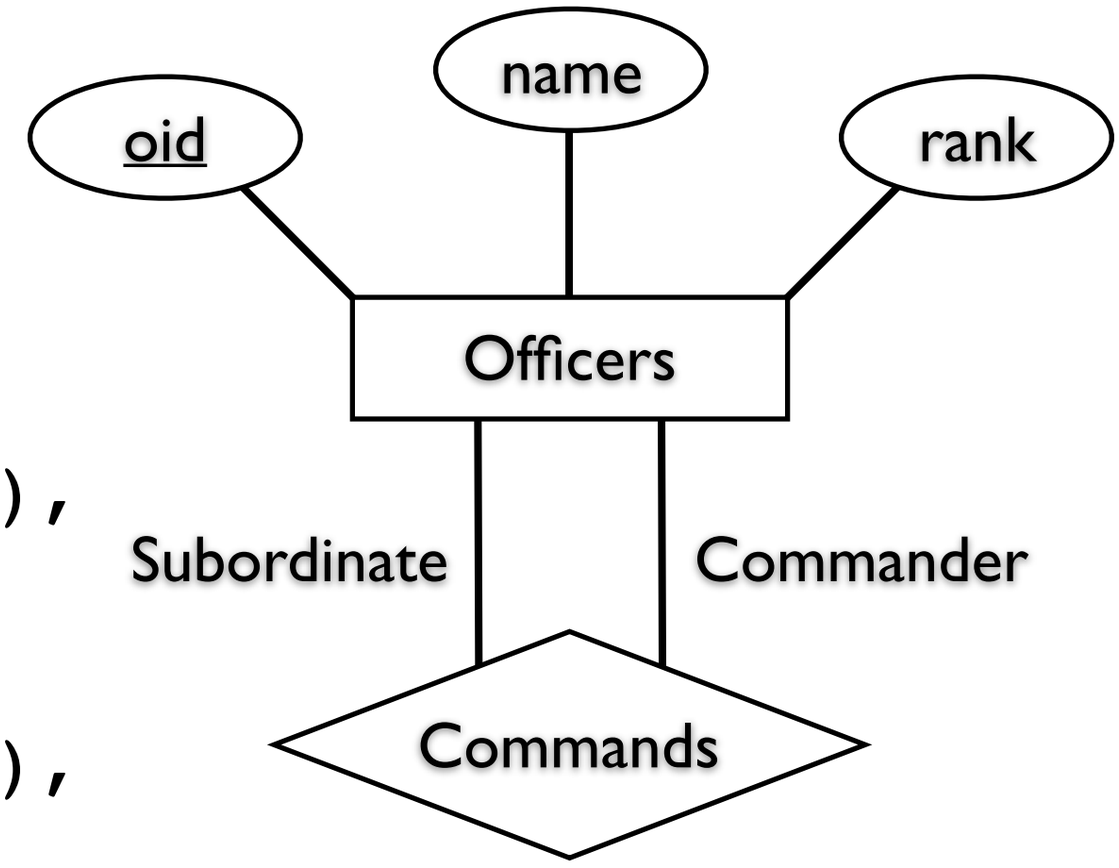
Foreign Key Constraints

```
CREATE TABLE Visited(
  oid INTEGER, pid INTEGER, when DATE,
  PRIMARY KEY (oid, pid),
  FOREIGN KEY (oid) REFERENCES Officers,
  FOREIGN KEY (pid) REFERENCES Planets
);
```



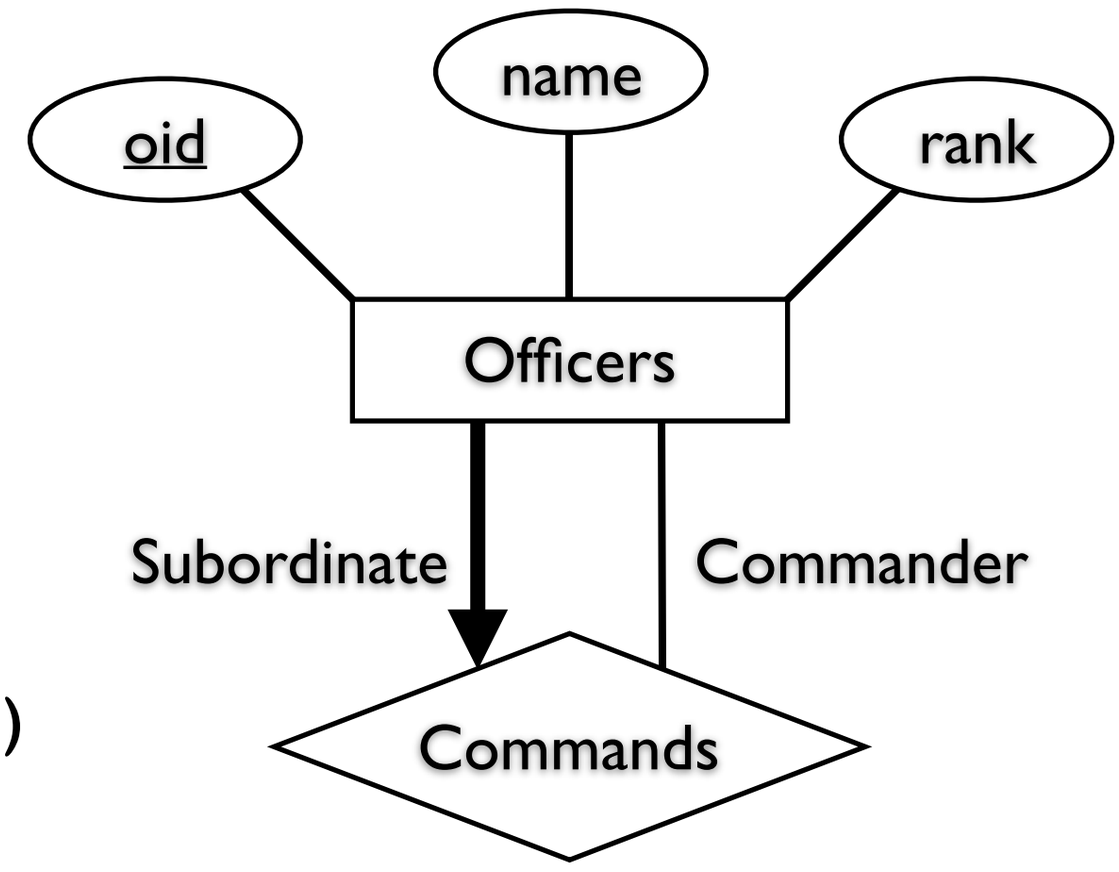
Foreign Key Constraints

```
CREATE TABLE Commands (
   Subordinate INTEGER,
   Commander INTEGER,

   PRIMARY KEY
     (Subordinate, Commander),

   FOREIGN KEY (Subordinate)
     REFERENCES Officers(oid),
   FOREIGN KEY (Commander)
     REFERENCES Officers(oid)
);
```



Foreign Key Constraints

```
CREATE TABLE Officers (
   …
   Commander INTEGER,
   …
   FOREIGN KEY (Commander)
      REFERENCES Officers(oid)
);
```



What about the Fleet Admiral (no commander)?
How do we insert the first tuple into Officers?

Enforcing Constraints

- Basic Enforcement
  - Reject Inserts/Deletions/Updates that introduce constraint violations.
- Insertions: Domain, Key, FK Constraints
- Updates: Domain, Key, FK Constraints
- Deletions: Only FK Constraints

Referential Integrity Enforcement

- Foreign Key Constraints are complex
  - DBMSes will attempt to rectify violations rather than reject the violating update.

- How should we react to an inserted tuple that references a nonexistent foreign key?
- How should we react to a referenced tuple being deleted?
- How should we react to a referenced tuple being updated?

Referential Integrity Enforcement

How should we react to an inserted tuple that references a nonexistent foreign tuple?

REJECT

Referential Integrity Enforcement

How should we react to a referenced tuple being deleted? (Delete Planet)

1. Delete all referencing tuples (Visited)
2. Disallow the deletion until there are no referencing tuples
3. Replace the referencing foreign key by some default value (or NULL).

Referential Integrity Enforcement

How should we react to a referenced tuple being updated? (Planet.pid changes)

1. Update all referencing tuples (change Visited.pid)
2. Disallow the update until there are no referencing tuples
3. Replace the referencing foreign key by some default value (or NULL).

Referential Integrity Enforcement

```
CREATE TABLE Visited(
  oid INTEGER, pid INTEGER, when DATE,
  PRIMARY KEY (oid, pid),
  …
  FOREIGN KEY (pid) REFERENCES Planets
     ON DELETE CASCADE
     ON UPDATE NO ACTION
);
```

`CASCADE`	Delete or Update Reference
`NO ACTION`	Reject Deletion or Update
`SET DEFAULT v` `SET NULL`	Replace Reference with *v* or *NULL*

Constraint Validation

- A *Transaction* is a batch of DBMS Operations
- `SET CONSTRAINT [name] IMMEDIATE;`
  - Perform constraint checking immediately after an insert/update/delete.
- `SET CONSTRAINT [name] DEFERRED;`
  - Perform constraint checking at the end of a transaction (commit time).

Table Constraints

```
CREATE TABLE Officer(
   oid INTEGER,
   name CHAR(50),
   ship CHAR(5)
   PRIMARY KEY (oid)
   FOREIGN KEY (ship) REFERENCES Ships(sid)
   CHECK ( ‘Enterprise’ <> (SELECT Name
                            FROM Ship S
                            WHERE S.sid = Officer.ship))
);
```

CHECK clause can contain any conditional expression

If the conditional evaluates to false, the command is rejected

Multi-Table Constraints

Keep the number of Planets and Space Stations Over 100

```
CREATE TABLE SpaceStations (
CREATE ASSERTION SaveTheFederation
  CHECK ( 100 > (SELECT COUNT(*) FROM Planets)
                +(SELECT COUNT(*) FROM SpaceStations))
);
```

ASSERTION defines a CHECK that is not associated with any specific table.