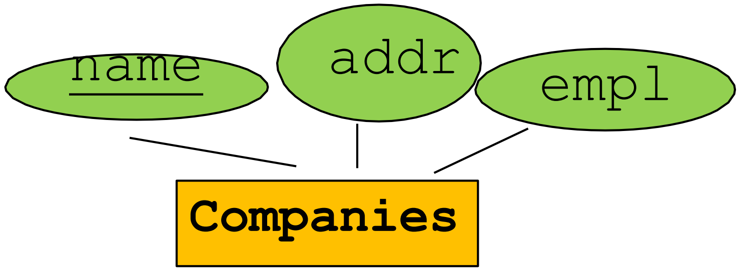
Translating an ER schema into the relational model

ER-Relational Translation

- Database design is first done using the entity-relationship model (or other semantic models such as UML)
- An ER schema must then be translated into relations
- This is a relatively straightforward process that can be automated.

Entity Sets to Relations



Companies(name, address, empl)

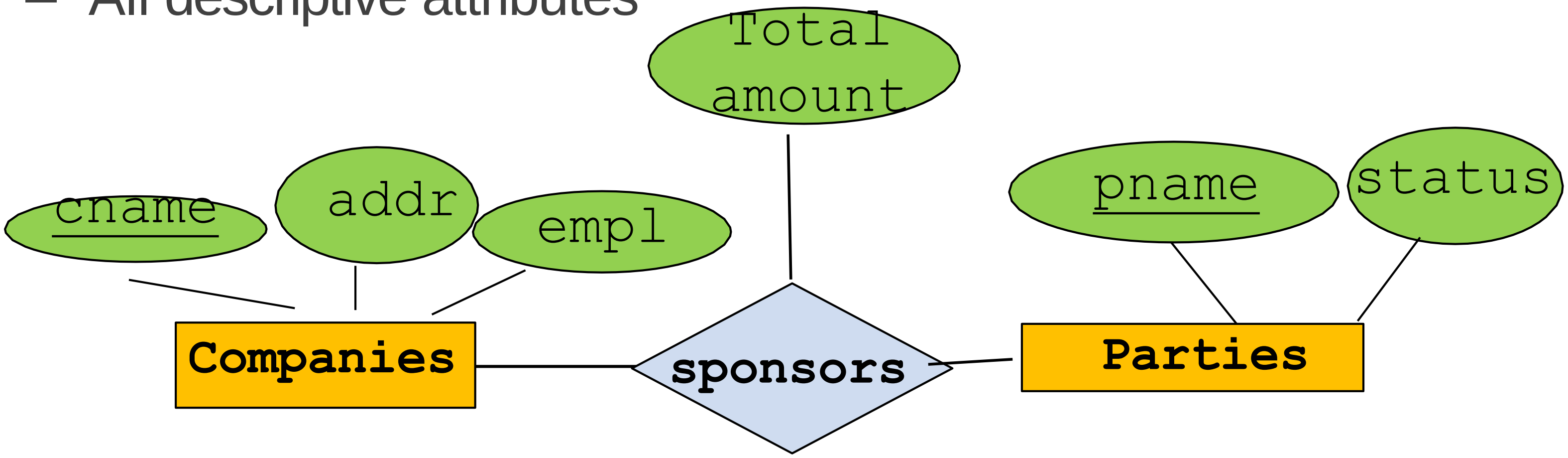
```
PostgreSQL:
CREATE TABLE Companies
     (name VARCHAR(30),
      addr VARCHAR(50),
      empl INTEGER,
      PRIMARY KEY (name))
CREATE TABLE Companies
     (name VARCHAR(30) PRIMARY KEY,
      addr VARCHAR(50),
      empl INTEGER)
```

```
DB2:
CREATE TABLE Companies
     (name VARCHAR(30) NOT NULL,
      addr VARCHAR(50),
      empl INTEGER,
      PRIMARY KEY (name))
```

name	addr	empl
BiggestEngCompanyEver	Eng. Av., H3X…	25,000
BiggestConstCommpanyEver	Constr. St. H4E…	47,000
NoNameCompany	Whatever St., …	200

Many-many Relationship Sets

- A many-to-many relationship set is **ALWAYS** translated as an individual table.
- Attributes of the table are
  - Keys for each participating entity set (as foreign keys)
    - This set of attributes forms the key for the relation
  - All descriptive attributes



Companies(cname,addr,empl)
Parties(pname, status)
Sponsorship(cname,pname,tamount)
cname references Companies
pname references Parties

Example Tables

Companies

cname	addr	empl
BiggestEngCompanyEver	Eng. Av., H3X…	25,000
NoNameCompany	Whatever St., …	200
…		

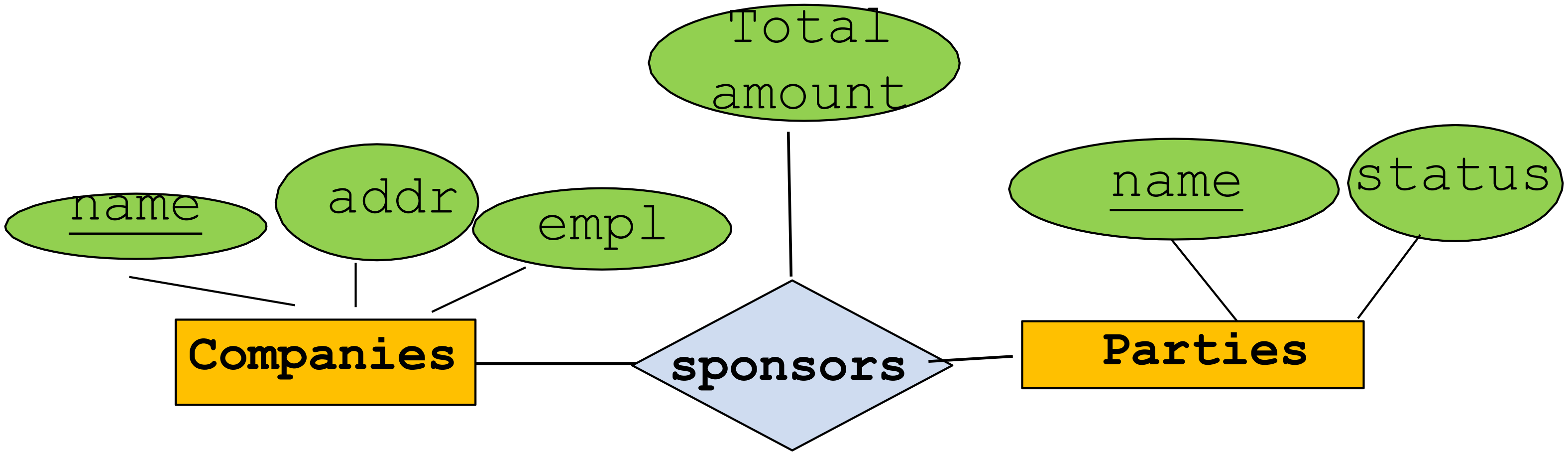
Parties

pname	status
CAQ	governing
Liberals	opposition
…	

Sponsorship

cname	pname	tamount
BiggestEngCompanyEver	Liberals	250,000
BiggestEngCompanyEver	CAQ	25,000
NoNameCompany	CAQ	50,000
…		

Many-many Relationship Sets



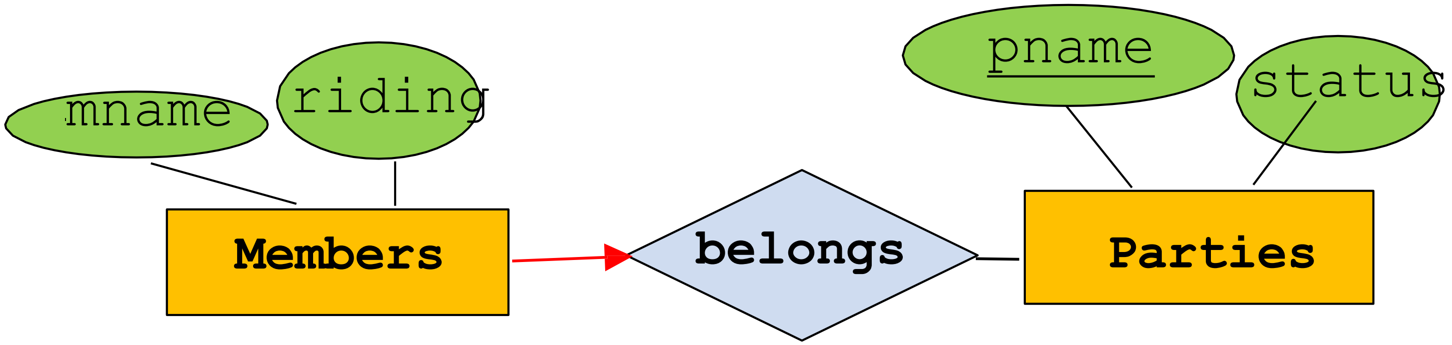
```
CREATE TABLE Companies
    (cname VARCHAR(30),
    addr VARCHAR(50),
     empl INTEGER,
     PRIMARY KEY (cname))
)
CREATE TABLE Parties
    (pname VARCHAR(20),
     status VARCHAR(10),
     PRIMARY KEY (pname))
```

Sponsorship(cname,pname,tamount)

```
CREATE TABLE Sponsorship
    (cname VARCHAR(30),
     pname VARCHAR(20),
     tamount INTEGER,
     PRIMARY KEY (cname,pname),
     FOREIGN KEY (cname)
    REFERENCES Companies,
     FOREIGN KEY (pname)
    REFERENCES Parties)
```

Relationships Sets with Key Constraints

- **Alternative I**: map relationship set to table
  - Many-one from entity set E1 to entity set E2: key of E1
    - i.e., key of entity-set with the key constraint is the key for the new relationship table (**mname** is now the key)
  - One-one: key of either entity set
  - Separate tables for entity sets (**Members** and **Parties**)



Members(<u>mname</u>,riding)

Parties(<u>pname</u>,status)

Membership(<u>mname</u>,pname)

```
CREATE TABLE Membership
   (mname VARCHAR(30),
   pname VARCHAR(20),
   PRIMARY KEY (mname),
   FOREIGN KEY (mname)
    REFERENCES Members,
    FOREIGN KEY (pname)
    REFERENCES Parties)
```

Example Tables

Members

mname	riding
François Legault	L'Assomption
Geneviève Guilbault	Louis-Hébert
Gabriel Nadeau-Dubois	Gouin
…	

Parties

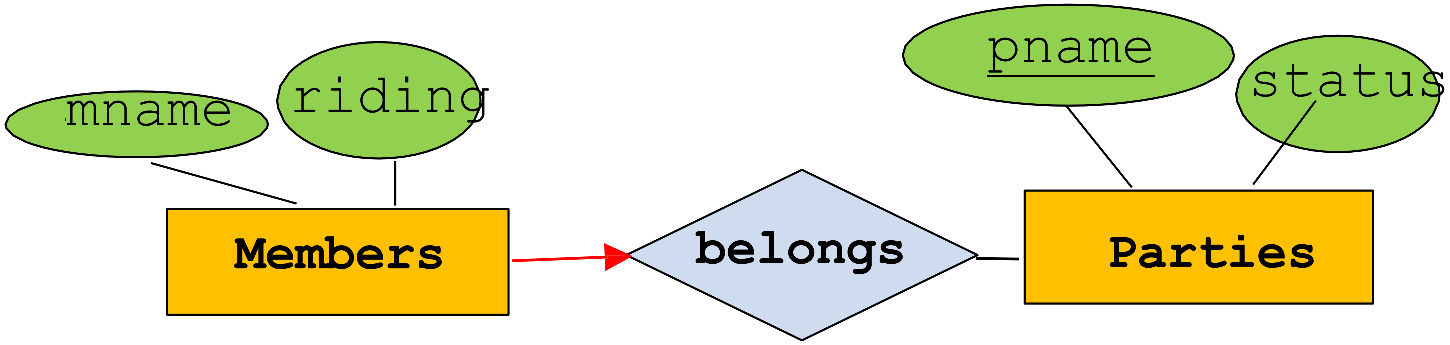
pname	status
CAQ	governing
Quebec solidaire	other
…	

Membership

mname	pname
François Legault	CAQ
Geneviève Guilbault	CAQ
Gabriel Nadeau-Dubois	Quebec solidaire
…	

Relationships Sets with Key Constraints

- **Alternative 1I**: include relationship set in table of the entity set with the key constraint
  - Possible because there is *at most* one relationship per entity
  - Not useful if many entities do not have arelationship (wasted space, many not filled values)



Members(<u>mname</u>,riding,pname)

Parties(<u>pname</u>,status)

```
CREATE TABLE Member
   (mname VARCHAR(30),
    riding VARCHAR(30),
   pname VARCHAR(20),
    PRIMARY KEY (mname),
    FOREIGN KEY (pname)
        REFERENCES Parties)
```

Example Tables

Members

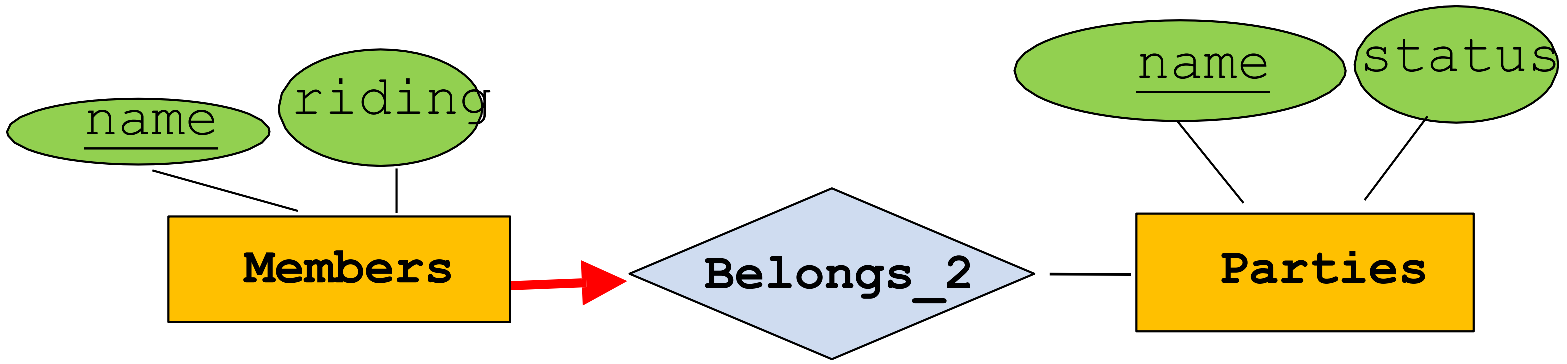
mname	riding	pname
François Legault	L'Assomption	CAQ
Geneviève Guilbault	Louis-Hébert	CAQ
Gabriel Nadeau-Dubois	Gouin	Quebec solidaire
…		

Parties

pname	status
CAQ	governing
Quebec solidaire	other
…	

Key and Participation Constraints

- Include relationship set in table of the entity set with the key constraint



Members(<u>mname</u>,riding,pname)

Parties(<u>pname</u>,status)

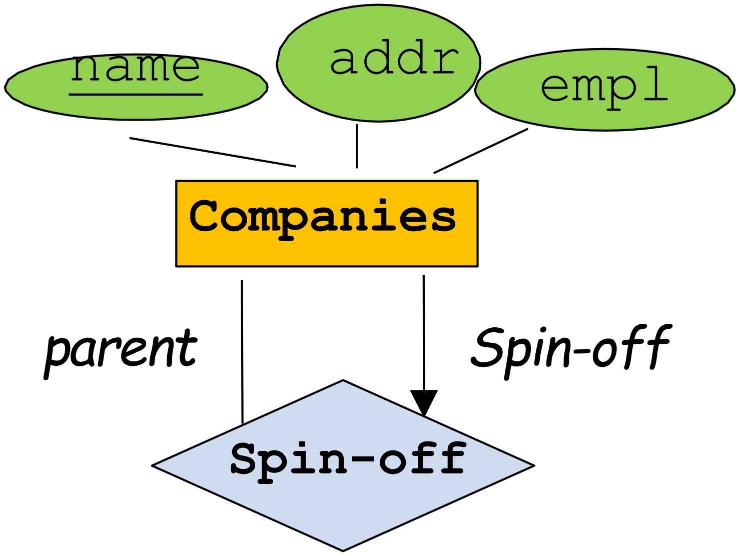
```
CREATE TABLE Member
   (mname VARCHAR(30),
    riding VARCHAR(30),
    pname VARCHAR(20) NOT NULL,
    PRIMARY KEY (mname),
    FOREIGN KEY (pname)
        REFERENCES Parties)
```

Participation Constraints

- Can usually not be reflected
- Only exception on previous slide
  - If there is a key constraint and a participation constraint

Renaming

In the case the keys of the participating entity sets have the same names we must rename attributes accordingly



- Companies(<u>name</u>, addr, empl)
- SpinOff(<u>spinoffcompany,</u> parentcompany)

```
CREATE TABLE SpinOff
   (spinoffcompany VARCHAR(30) PRIMARY
   KEY,
    parentcompany VARCHAR(30),
   FOREIGN KEY (spinoffcompany)
      REFERENCES Companies(name),
    FOREIGN KEY (parentcompany)
      REFERENCES Companies(name))
```

Renaming can also occur for foreign keys, etc.

Otherwise, all other translation rules apply

Examples

Companies

name	addr	empl
BiggestEngCompanyEver	Eng. Av., H3X…	25,000
BiggestConstCommpanyEver	Constr. St. H4E…	47,000
NoNameCompany	Whatever St., …	200
…		

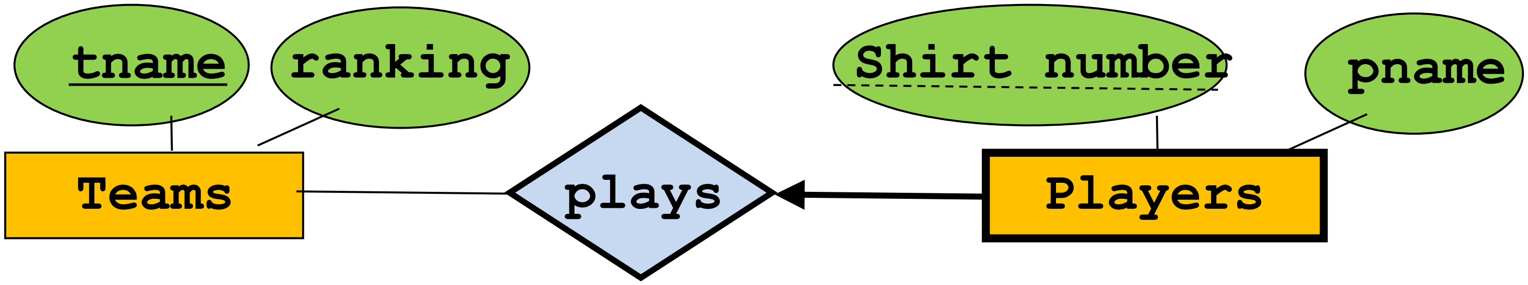
Spinoffs

spinoffcompany	parentcompany
NoNameCompany	BiggestConstCompanyEver
…	

- Alternative?

Translating Weak Entity Sets

Weak entity set and identifying relationship set are translated into a single table



- **Teams(tname,ranking)**
- **Players(tname,shirtN,pname)**

```
CREATE TABLE Players
    (tname VARCHAR(30) ,
     shirtN INT,
    pname VARCHAR(30,)
    PRIMARY KEY (tname,shirtN),
    FOREIGN KEY (tname)
      REFERENCES Teams)
```

Examples

HealthInsurance

id	amount
12345	100,000
12346	500,000
…	

Players

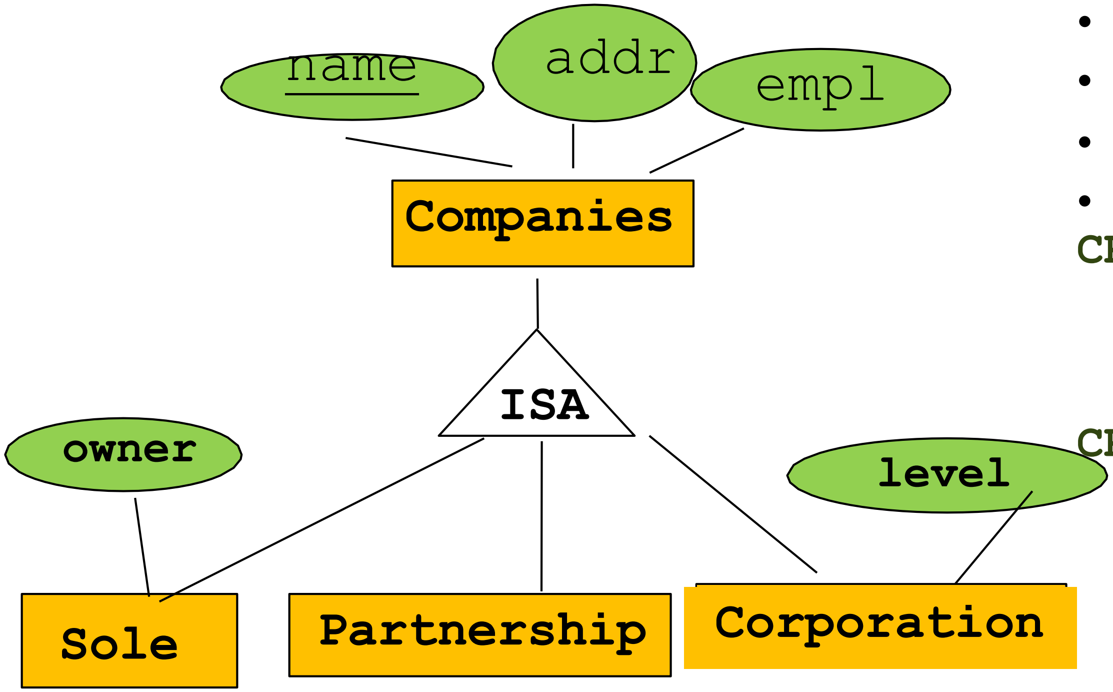
Id	FirstName	DOB
12345	Anna	2010-10-11
12345	Kim	2012-08-15
12346	Wenbo	2018-02-03
…		

Translating ISA Hierarchies

General Approach: distribute information among relations

Relation of superclass stores the general attributes and defines key

Relations of subclasses have key of superclass and addit. attributes



- **Companies(<u>name</u>, addr, empl)**
- **Sole(<u>name</u>, owner)**
- **Partnership(<u>name</u>)**
- **Corporation(<u>name</u>, level)**

```
CREATE TABLE Companies
     (name VARCHAR(30) PRIMARY KEY,
      addr VARCHAR(50),
      empl INTEGER)
CREATE TABLE Corporation
   (name VARCHAR(30) PRIMARY KEY,
    level VARCHAR(20),
    FOREIGN KEY (name)
       REFERENCES Companies)
```

Contrast:

- E/R: sub-entity sets do NOT have primary key attribute (that would be redundant)
- Relational: sub-tables have primary key attribute which represents a reference to the parent table
  - That's not redundant: it's the encoding of the ISA symbol!

Examples

Companies

name	addr	empl
BiggestEngCompanyEver	Eng. Av., H3X…	25,000
BiggestConstCompanyEver	Constr. St. H4E…	47,000
NoNameCompany	Whatever St., …	200
…		

Corporation

name	level
BiggestEngCompanyEver	large
…	

Sole

name	owner
NoNameCompany	Bugs Bunny
…	

Overlapping/disjoint ?

Covering / non-covering ?

Translating ISA Hierarchies (contd.)

- Object-oriented approach:
  - Sub-classes have all attributes;
  - if an entity is in a sub-class it does not appear in the super-class relation;
  - No relation for superclass if covering
    - **Companies(name, addr, empl)**
    - **Corporation(name,addr,empl,level)**
    - **Sole(name, addr, empl, owner)**
    - **Partnership(name,addr,empl)**

Companies

name	addr	empl
BiggestConstCompanyEver	Constr. St. H4E…	47,000
…		

Corporation

name	addr	empl	level
BiggestEngCompanyEver	Eng. Av., H3X…	25,000	large
…			

Sole

name	addr	empl	owner
NoNameCompany	Whatever st.	200	Bugs Bunny
…			

Object-oriented

Pro/Contra:

+ A query asking for all information about Corporations (name, addr, empl, level) only has to ran through one table.

-A Query wanting the names of all companies has to read all four tables

- Overlapping sub entity sets => undesired redundancy

- **Companies(name, addr, empl)**
- **Corporation(name,addr,empl,level)**
- **Sole(name, addr, empl, owner)**
- **Partnership(name,addr,empl)**

Translating ISA Hierarchies (contd.)

- Last Alternative: one big relation
  - Create only one relation for the root entity set with all attributes found anywhere in its network of subclasses.
  - Put NULL in attributes not relevant to a given entity

Companies(name,addr,empl,owner,level)

name	addr	empl	level	owner
BiggestEngCompanyEver	Eng. Av., H3X…	25,000	large	NULL
BiggestConstCompanyEver	Constr. St. H4E…	47,000	NULL	NULL
NoNameCompany	Whatever St., …	200	NULL	Bugs B.
…				

One Big relation

Pro/Contra:

+ All information in one big table; never need to join information from several tables

- Lot's of possible NULL values
- If a sub-class has a relationship set, with another entity set we cannot enforce that only the tuples of that subclass can have relationships in that relationship set.

-Might be hard by looking at atuple to know which subclass(es) it belongs to as attributes might be NULL despite the fact that the tuple belongs to a subclass

Translating Aggregation

- Translate first inner, then outer relationship set
- Handle constraints as usual

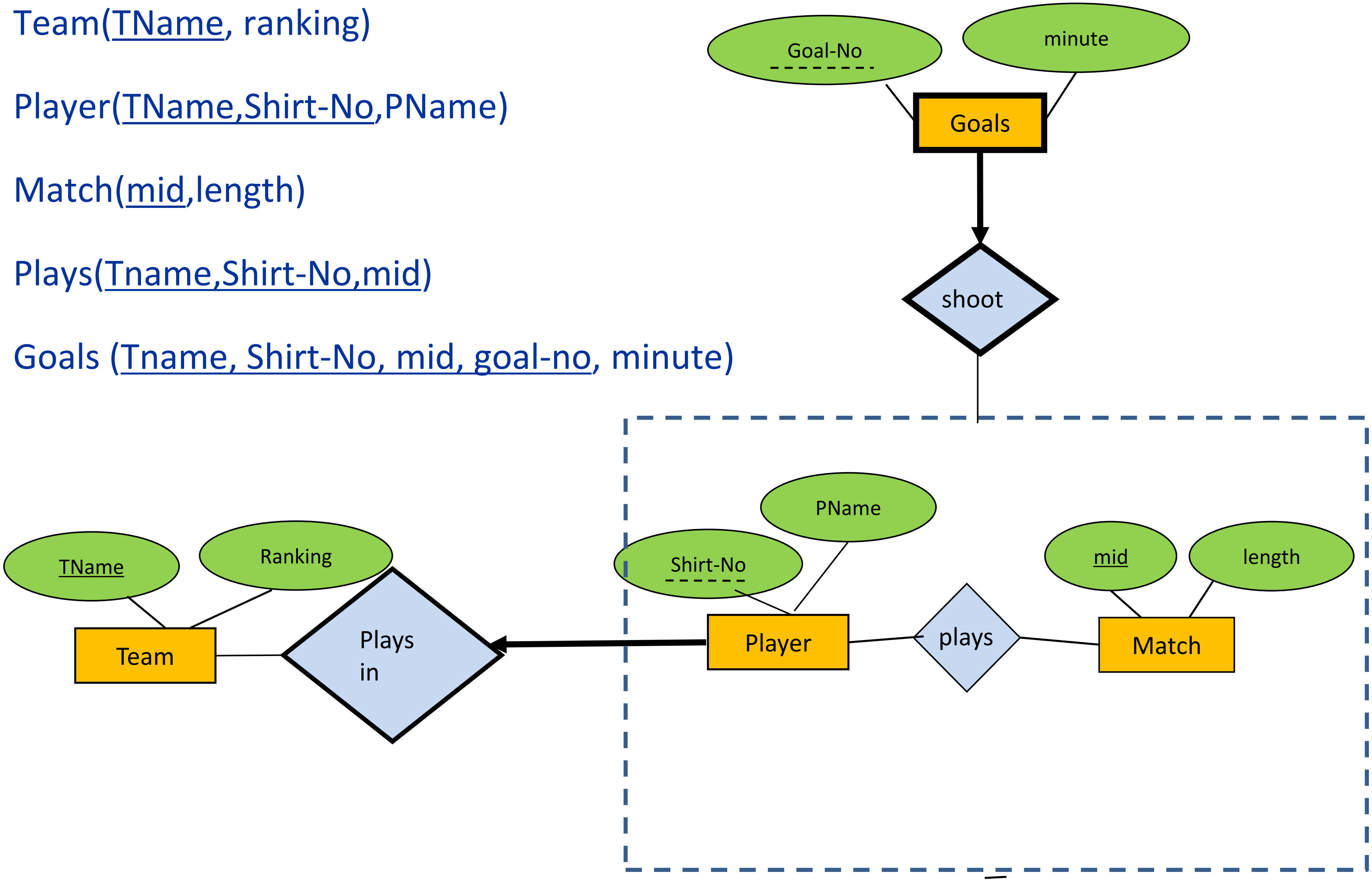
Team(TName, ranking)

Player(TName,Shirt-No,PName)

Match(mid,length)

Plays(Tname,Shirt-No,mid)

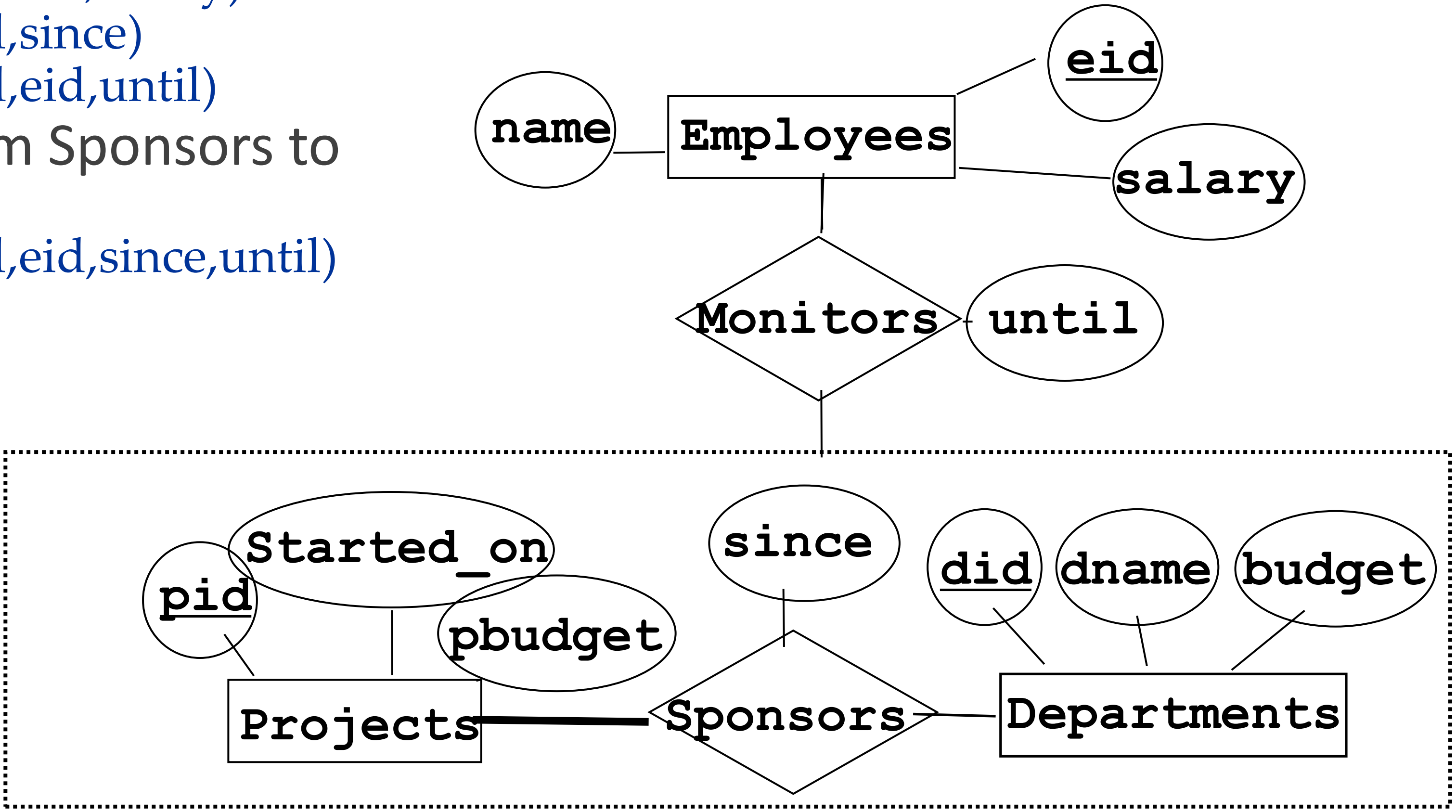
Goals (Tname, Shirt-No, mid, goal-no, minute)



Translating Aggregation – Another example

- Translate first inner, then outer relationship set
- Handle constraints as usual
- No key constraints
  - Projects(pid,started_on,pbudget)
  - Departments(did,dname,budget)
  - Employees(eid,name,salary)
  - Sponsors(pid,did,since)
  - Monitors(pid,did,eid,until)
- Key constraint from Sponsors to Employees
  - Sponsors(pid,did,eid,since,until)
  - No Monitors

- Key constraint from Projects to Departments
  - Projects(pid, started_on, pbudget, did, since)
  - No Sponsors
  - Monitors(pid, eid, until)


eid
name
Employees
salary
Monitors
until
Started_on
since
pid
did
dname
budget
pbudget
Projects
Sponsors
Departments