CptS 451- Introduction to Database Systems

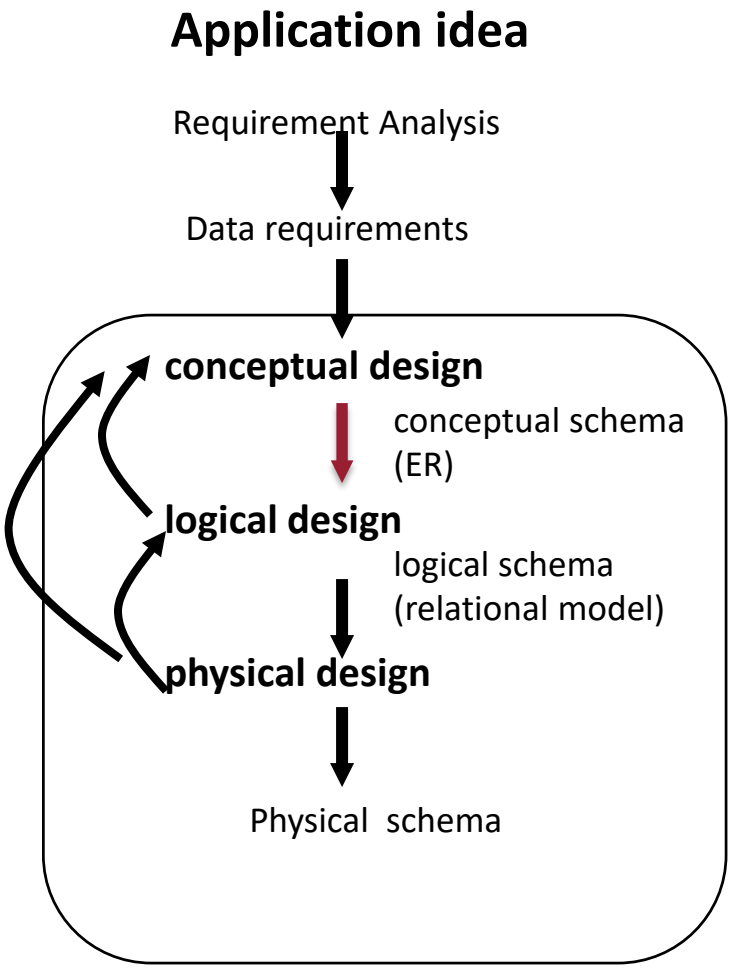
Mapping ER to Relational Model

(DMS – 3.5)

Instructor: Sakire Arslan Ay

WASHINGTON STATE UNIVERSITY
World Class. Face to Face.

Database Design Process



Database Design

Diagrams ER

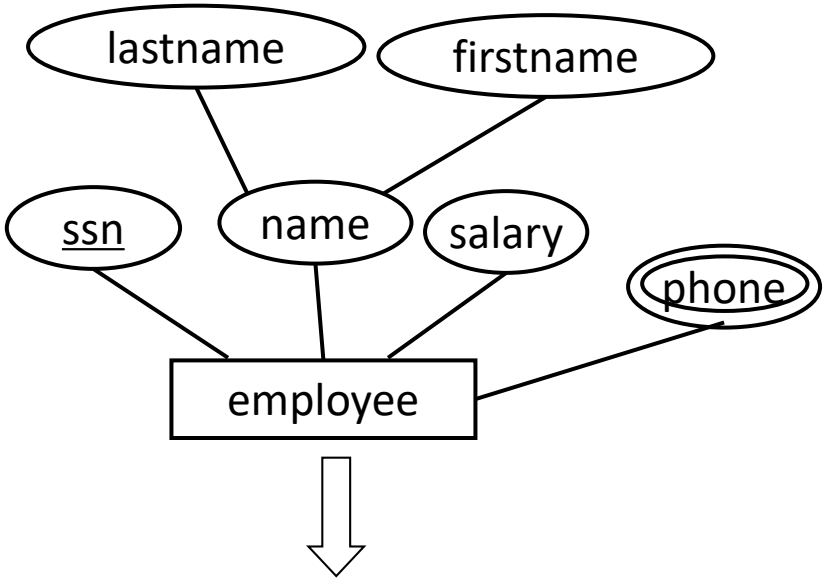
Tables;
column names: attributes
rows: tuples

Complex
file organization
and index
structures.

Today's Lecture

- **Mapping ER to Relational Model**
  - (Strong) entity sets to relations
  - ER relationship sets to relations
  - Mapping constraints
    - Key constraints
    - Combining relations
    - Participation constraints
  - Converting multi-way relationships
  - (Weak) entity sets
  - Converting aggregation
  - Converting subclass structures to relations

(Strong) Entity Sets to Relations



Relation: Employee(ssn, salary, lastname, firstname)

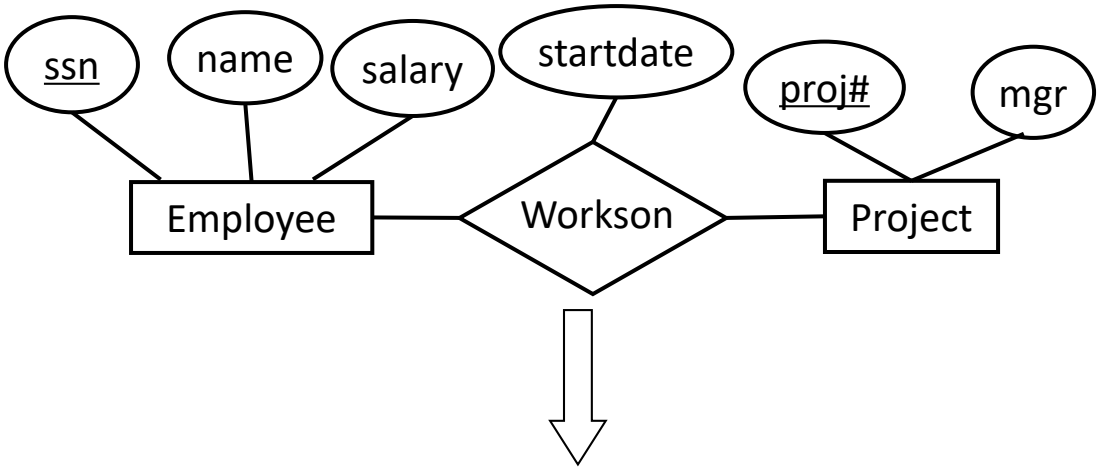
Primary Key: ssn

Relation: Employee_Phone(ssn, phone)

Primary Key: ssn, phone

ssn is also a foreign key.

Relationship Sets to Relations



ER (Strong) Relationship Sets to Relations:

1. For each entity involved in the ER relationship set, take its key attribute(s) as part of the relation schema.
2. If the ER relationship set has attributes, then add them as well.

If **many-to-many** relationship:

- the union of the primary key attributes from participating entity sets become the new relation's primary key.

Relation: Workson(ssn,proj#,startdate)

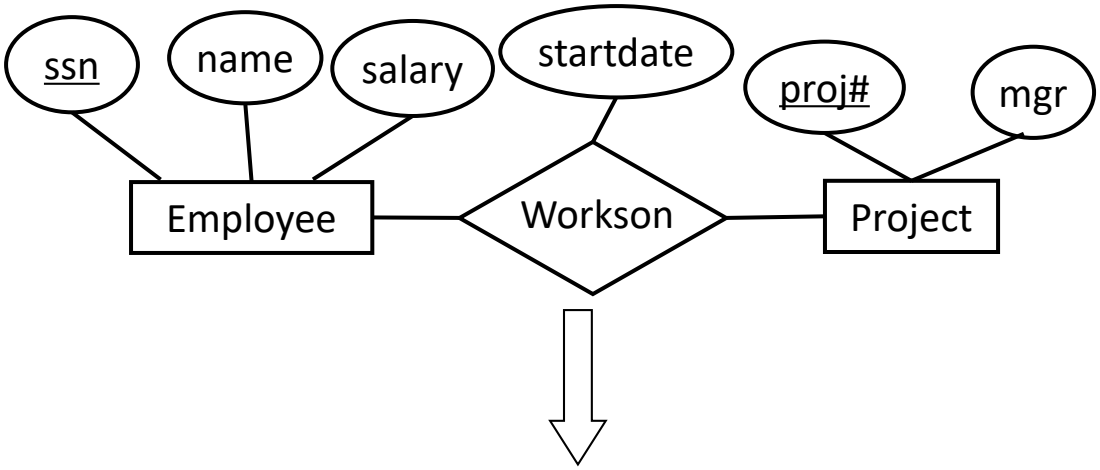
Primary Key: ssn,proj#

Foreign Keys:

Workson(proj#) REFERENCES Project(proj#)

Workson(ssn) REFERENCES Employee(ssn)

Relationship Sets to Relations



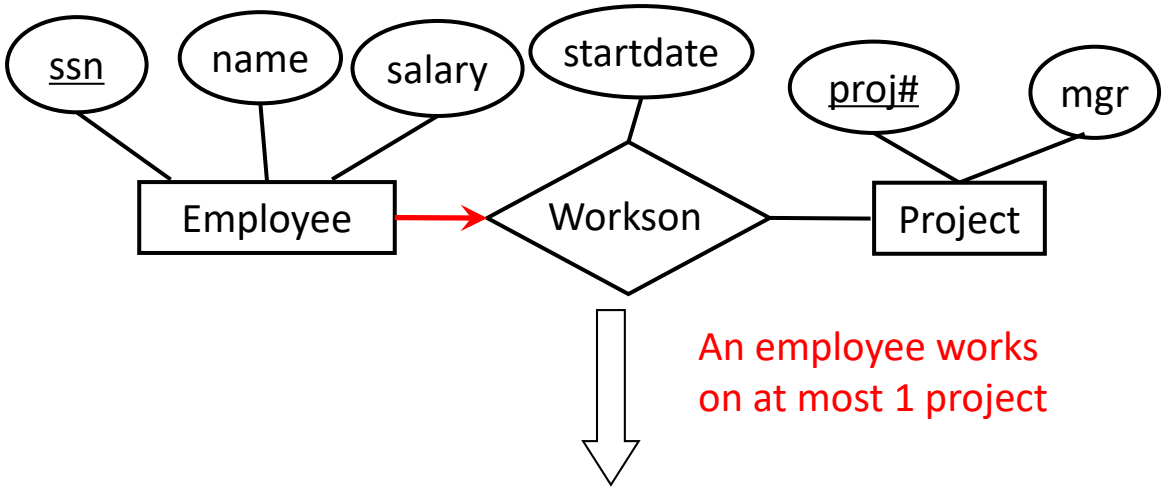

```
CREATE TABLE Workson(
    ssn         CHAR(11),
    proj#       INTEGER,
    startdate DATE,
    PRIMARY KEY (ssn,proj#),
    FOREIGN KEY (proj#) REFERENCES Project(proj#),
    FOREIGN KEY (ssn)  REFERENCES Employee(ssn)
)
```

```
CREATE TABLE Employee(
    ssn         CHAR(11),
    name        VARCHAR(30),
    salary      INTEGER,
    PRIMARY KEY (ssn)
)
```

```
CREATE TABLE Project(
    proj#       INTEGER,
    mgr         VARCHAR(30),
    PRIMARY KEY (proj#)
)
```

Relationship Sets to Relations (cont.)

Key (Multiplicity) Constraints



If **many-to-one** OR **one-to-many** relationship set:

- the primary key of the entity set on the many side of the relationship serves as the primary key.

If **one-to-one** relationship set:

- the primary key of either entity set can be chosen as primary key.

Relation: Workson(ssn,proj#,startdate)

Primary Key: ssn

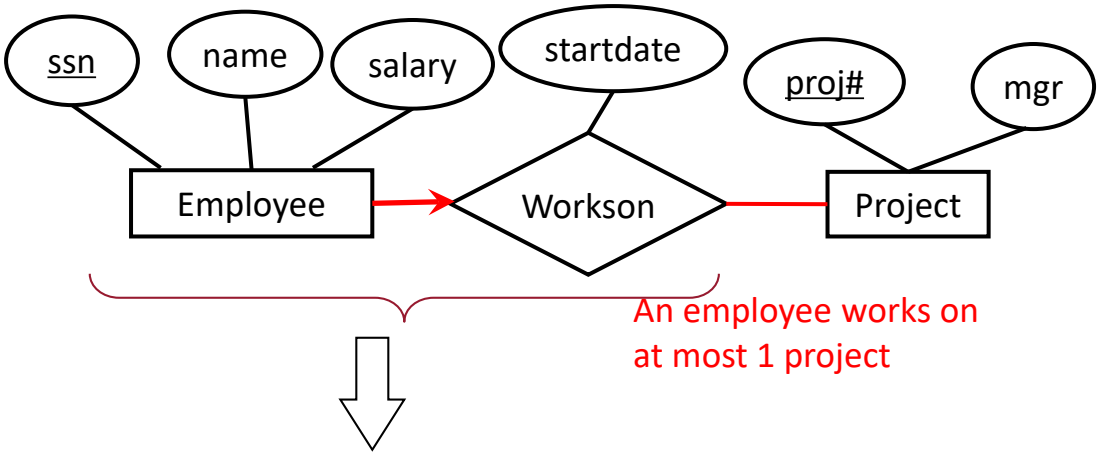
Foreign Keys:

Workson(proj#) REFERENCES Project(proj#)

Workson(ssn) REFERENCES Employee(ssn)

Relationship Sets to Relations (cont.)

Alternative Solution: Combining Relations



Relation: Employee2(ssn, name, salary, proj#, startdate)

Primary Key: ssn

Foreign Key:

Employee2(proj#) REFERENCES Project(proj#)

Notes:

In the case of partial participation, replacing a schema by an extra attribute on the "many side" schema could result in null values

- i.e., for an *employee* that is not related to any *project*, the attributes *proj#* and *startdate* will have null values

If **many-to-one** OR **one-to-many** relationship set:

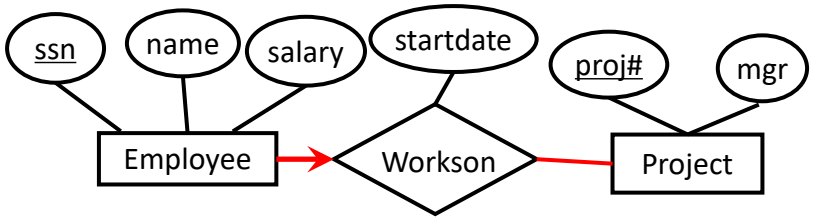
- Combine all attributes of the "many side " with:
  1. the key attributes of the "one side"
  2. attributes belonging to the relationship set.

If one-to-one relationship set:

- The extra attributes can be added to the relation on either side

Relationship Sets to Relations (cont.)

Combining Relations - Example



An employee works on at most 1 project

Relation: Employee2(ssn, name, salary, proj#, startdate)

Primary Key: ssn

Foreign Key:

Employee2(proj#) REFERENCES Project(proj#)

Table for "*employee*"

ssn	name	salary
111-11-1111	Jack	75,000
222-22-2222	Jared	70,300
333-33-3333	John	80,000
444-44-4444	Jill	70,000
555-55-5555	Jeremy	75,500

Table for "*workson*"

ssn	proj#	Startdate
111-11-1111	256	1/1/2014
222-22-2222	256	8/16/2014

Combining relation "*employee*" with relation "*workson*"

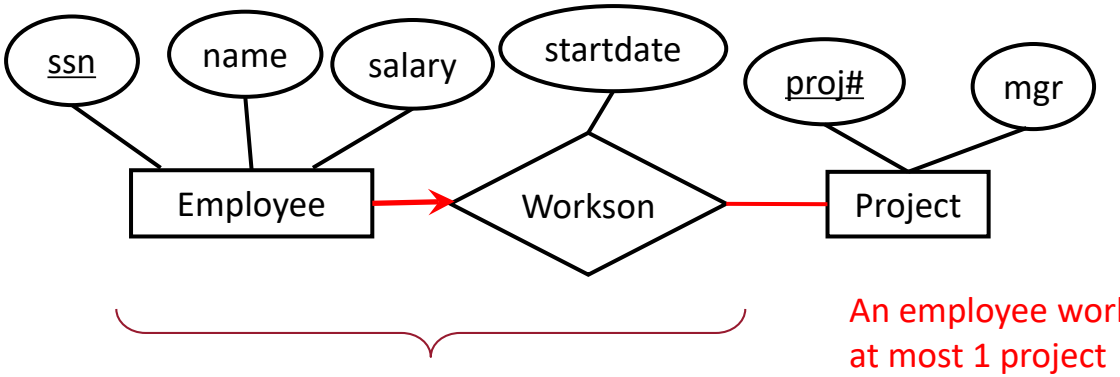
Employee2

ssn	name	salary	proj#	startdate
111-11-1111	Jack	75,000	256	1/1/2014
222-22-2222	Jared	70,300	256	8/16/2014
333-33-3333	John	80,000	NULL	NULL
444-44-4444	Jill	70,000	NULL	NULL
555-55-5555	Jeremy	75,500	NULL	NULL

Problem: NULL values

Relationship Sets to Relations (cont.)

Alternative Solution: Combining Relations



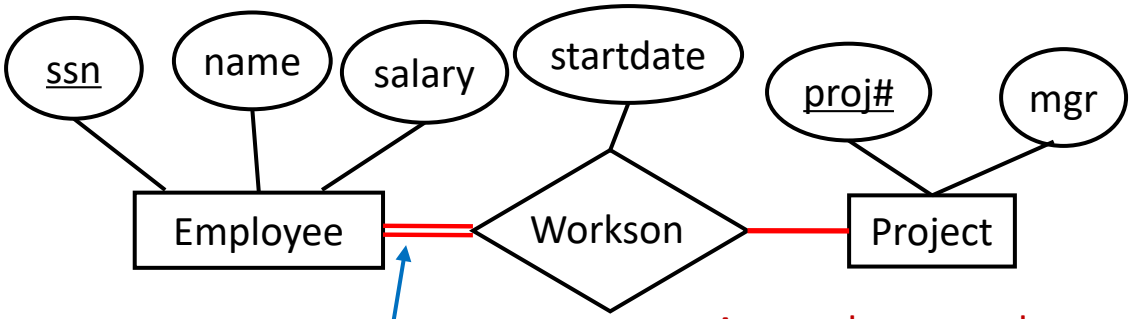

```
CREATE TABLE Project(
    proj#      INTEGER,
    mgr        VARCHAR(30),
    PRIMARY KEY (proj#)
)
```

An employee works on at most 1 project

```
CREATE TABLE Employee2(
    ssn        CHAR(11),
    name       VARCHAR(30),
    salary     INTEGER,
    proj#      INTEGER,
    startdate  DATE,
    PRIMARY KEY (ssn),
    FOREIGN KEY (proj#) REFERENCES Project(proj#)
)
```

Relationship Sets to Relations (cont.)

Participation Constraints



total participation!

An employee works on at least 1 project

Relation: Workson(ssn,proj#,startdate)
Primary Key: **ssn,proj#**
Foreign Key:
Workson(proj#) REFERENCES Project(proj#)
Workson(ssn) REFERENCES Employee(ssn)

Employee[ssn] ⊆ Workson [ssn]

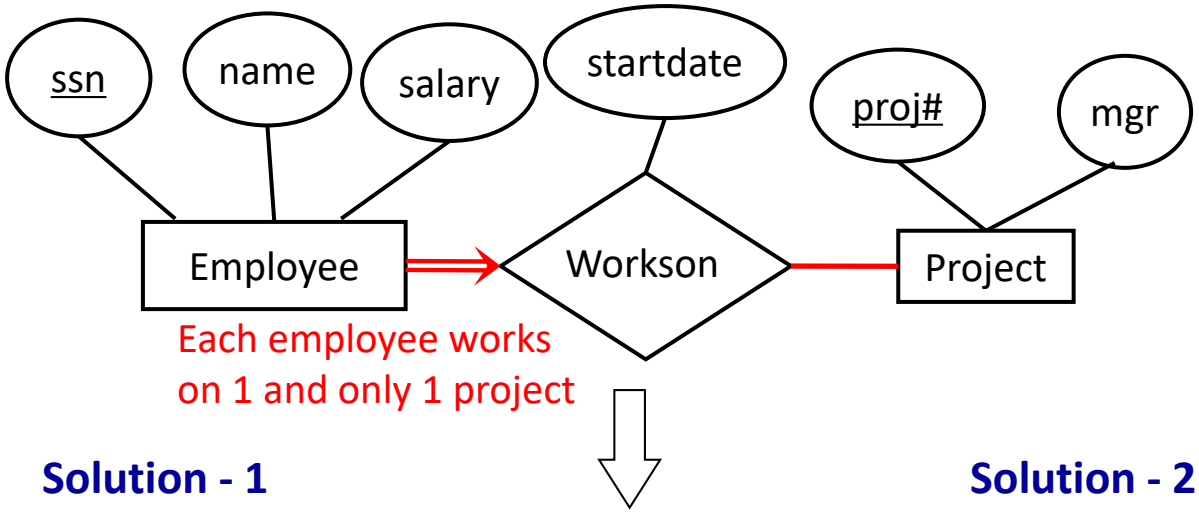
Total participation!

Won't work! Need an assertion or trigger.

```
CREATE TABLE Workson(
    ssn         CHAR(11) NOT NULL,
    proj#       INTEGER  NOT NULL,
    startdate DATE,
    PRIMARY KEY (ssn,proj#),
    FOREIGN KEY (proj#) REFERENCES
    Project(proj#),
    FOREIGN KEY (ssn)  REFERENCES
    Employee(ssn)
)
```

Relationship Sets to Relations (cont.)

Combining Constraints



Solution - 1

Add new relation:

Workson(ssn,proj#,startdate)

Key: **ssn**

Foreign Keys:

Workson(proj#) REFERENCES Project(proj#)

Workson(ssn) REFERENCES Employee(ssn)

Employee(ssn) ⊆ Workson (ssn)

Solution - 2

Update *employee* relation:

Employee2(ssn, name, salary, proj#, startdate,)

Key**: ssn**

Foreign Keys:

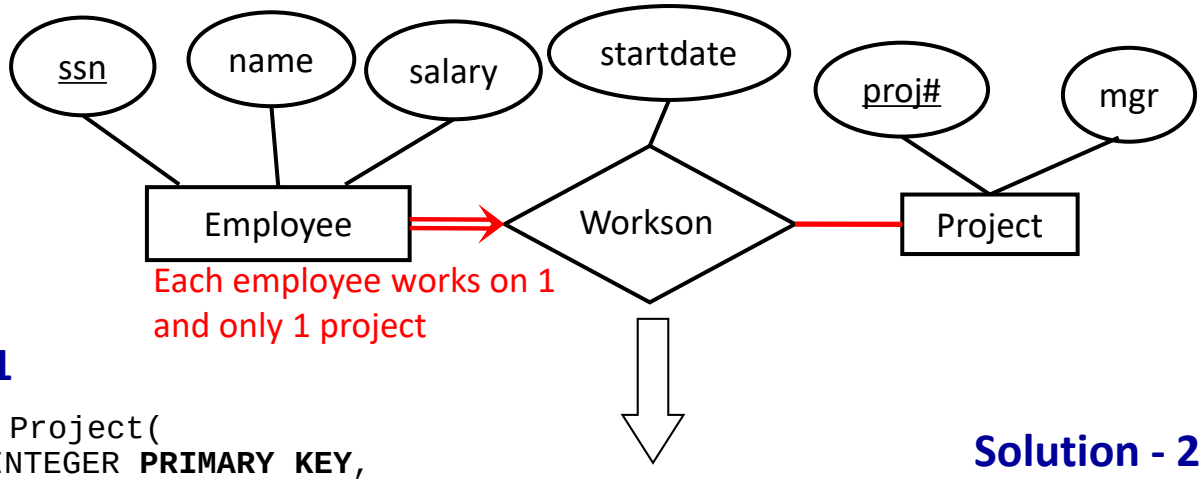
Employee2(proj#) REFERENCES Project(proj#)

Entity Identity Constraint:

proj# can't be NULL

Relationship Sets to Relations (cont.)

Combining Constraints



Solution - 1

```
CREATE TABLE Project(
    proj#   INTEGER PRIMARY KEY,
    mgr     VARCHAR(25),
)
CREATE TABLE Employee(
    ssn     CHAR(11) PRIMARY KEY,
    name    VARCHAR(25),
    salary  INTEGER
)
```

Represent the constraint **Employee(ssn) $\subseteq$ Workson (ssn)** as an assertion/trigger

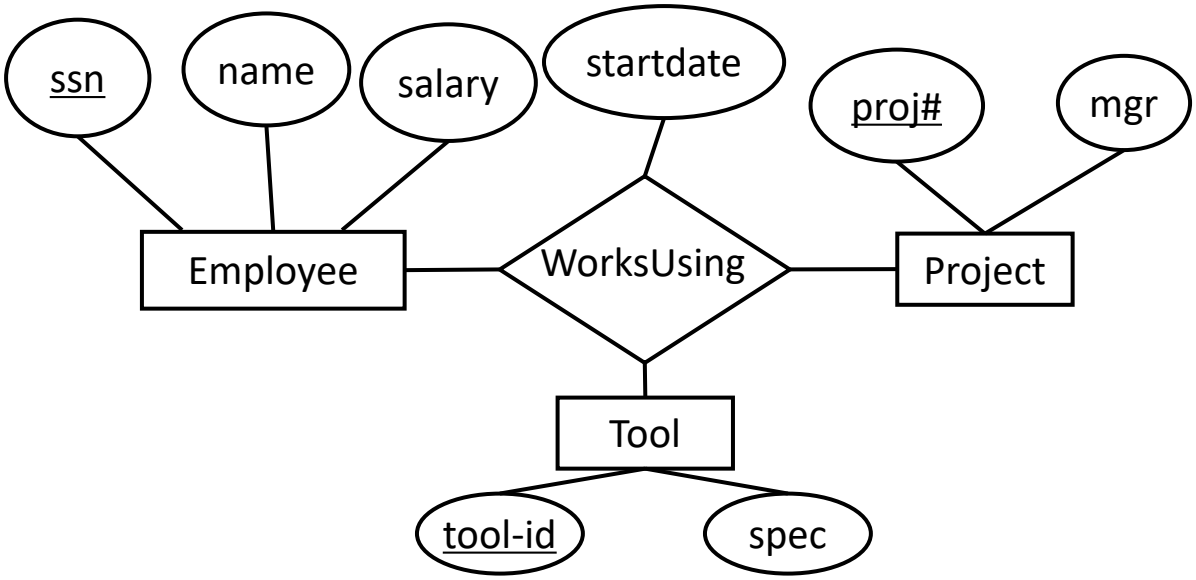
```
CREATE TABLE Workson(
    ssn         CHAR(11) PRIMARY KEY,
    proj#       INTEGER,
    startdate   DATE,
    FOREIGN KEY(ssn) REFERENCES Employee(ssn),
    FOREIGN KEY(proj#) REFERENCES Project(proj#)
)
```

Solution - 2

```
CREATE TABLE Project(
    proj# INTEGER PRIMARY KEY,
    mgr    VARCHAR(25),
)

CREATE TABLE Employee2(
    ssn     CHAR(11) PRIMARY KEY,
    name    VARCHAR(25),
    salary INTEGER,
    proj#  INTEGER NOT NULL,
    startdate DATE,
    FOREIGN KEY(proj#) REFERENCES
    Project(proj#)
)
```

Multiway Relationships



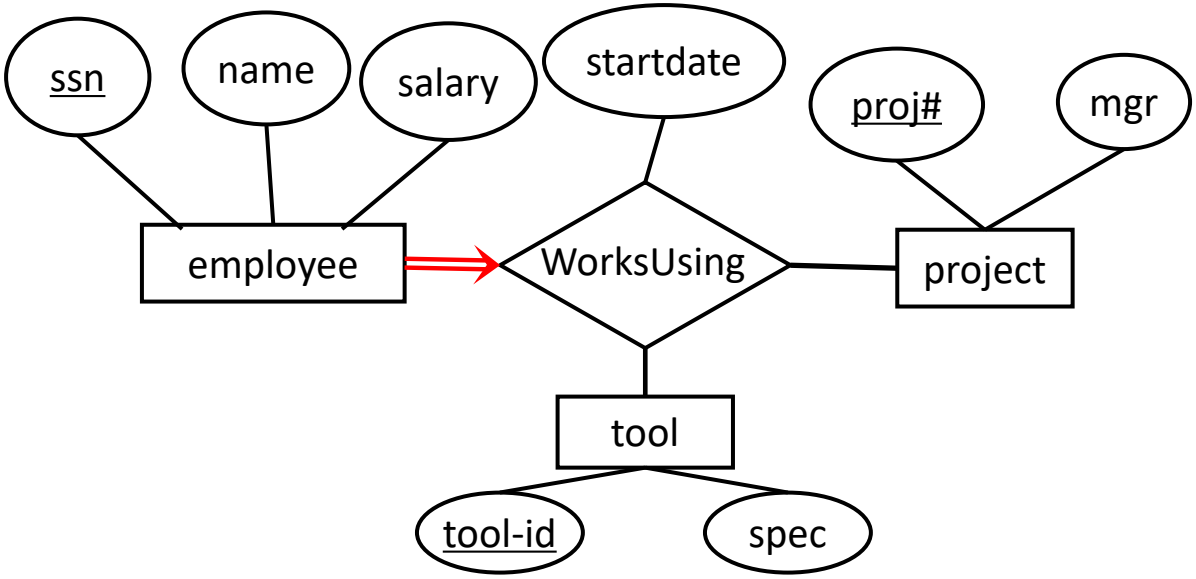
Relation: WorksUsing(ssn, proj#, tool-id, startdate)

Key: **ssn, proj#, tool-id**

Foreign Keys:

WorksUsing(proj#) REFERENCES Project(proj#)
WorksUsing (ssn) REFERENCES Employee(ssn)
WorksUsing (tool-id) REFERENCES Tool(tool-id)

Multiway Relationships



Relation: WorksUsing(ssn, proj#, tool-id, startdate)

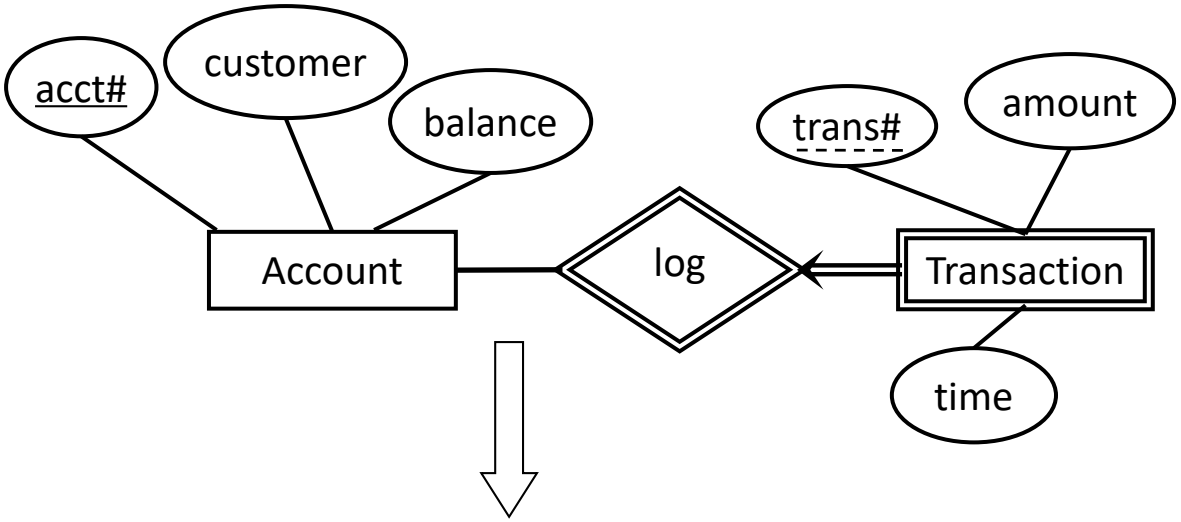
Key: **ssn**

Foreign Keys:

WorksUsing(proj#) REFERENCES Project(proj#)
WorksUsing(ssn) REFERENCES Employee(ssn)
WorksUsing(tool-id) REFERENCES Tool(tool-id)

Employee(ssn) $\subseteq$ WorksUsing (ssn) ← Total participation!

Weak Entity Sets



ER **Weak** Entity Sets to Relations:

Include the following in the schema:

1. All attributes of the weak entity
2. All key attributes of the owner entity (or entities)

```
CREATE TABLE Account(
     acct#     INTEGER PRIMARY KEY,
     customer  VARCHAR(30),
     balance   FLOAT
)

CREATE TABLE Transaction(
     acct#     INTEGER,
     trans#    BIGINT,
     amount    FLOAT,
     time      DATETIME,
     PRIMARY KEY(acct#,trans#),
    FOREIGN KEY(acct#) REFERENCES
     Account(acct#)
)
```

Relations:

Account(acct#,customer,balance)

Key: acct#

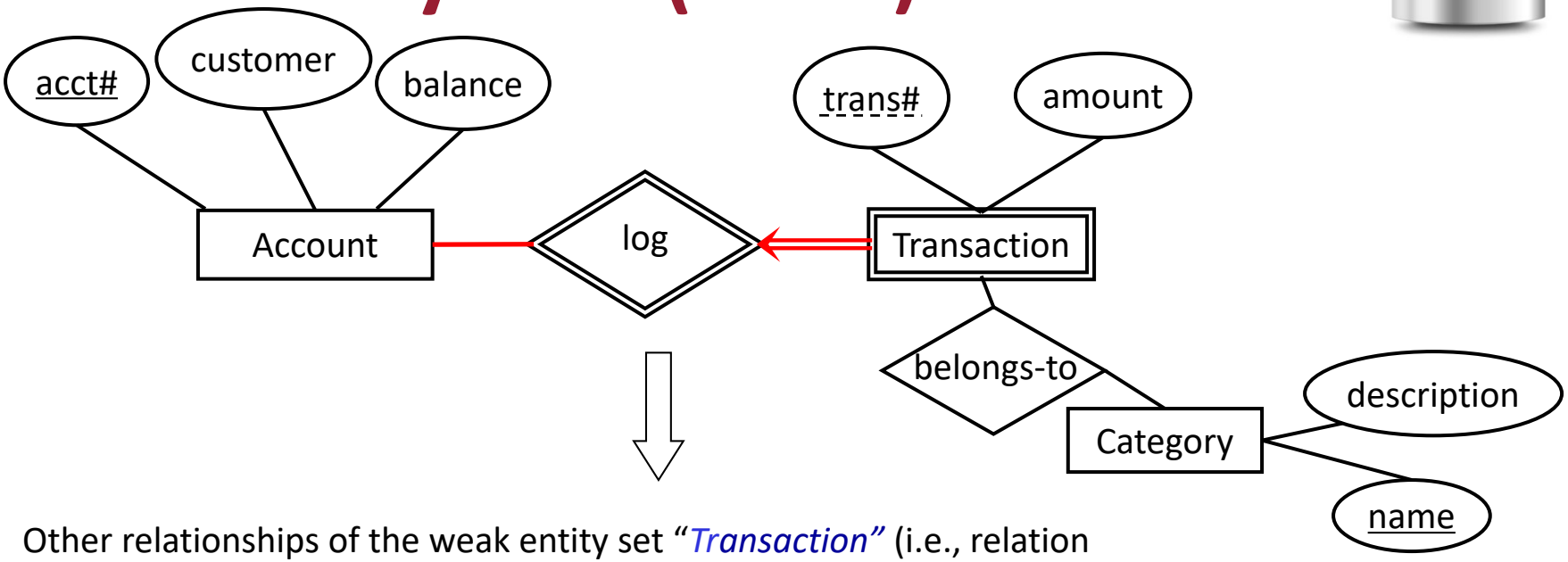
Transaction(acct#,trans#,amount, time)

Key: acct#, trans#

Foreign Key: Transaction(acct#) REFERENCES Account(acct#)

No relation for the identifying relationship set "Log."

Weak Entity Sets (cont.)



- Other relationships of the weak entity set "*Transaction*" (i.e., relation *belongs-to*) should have the complete key as its key

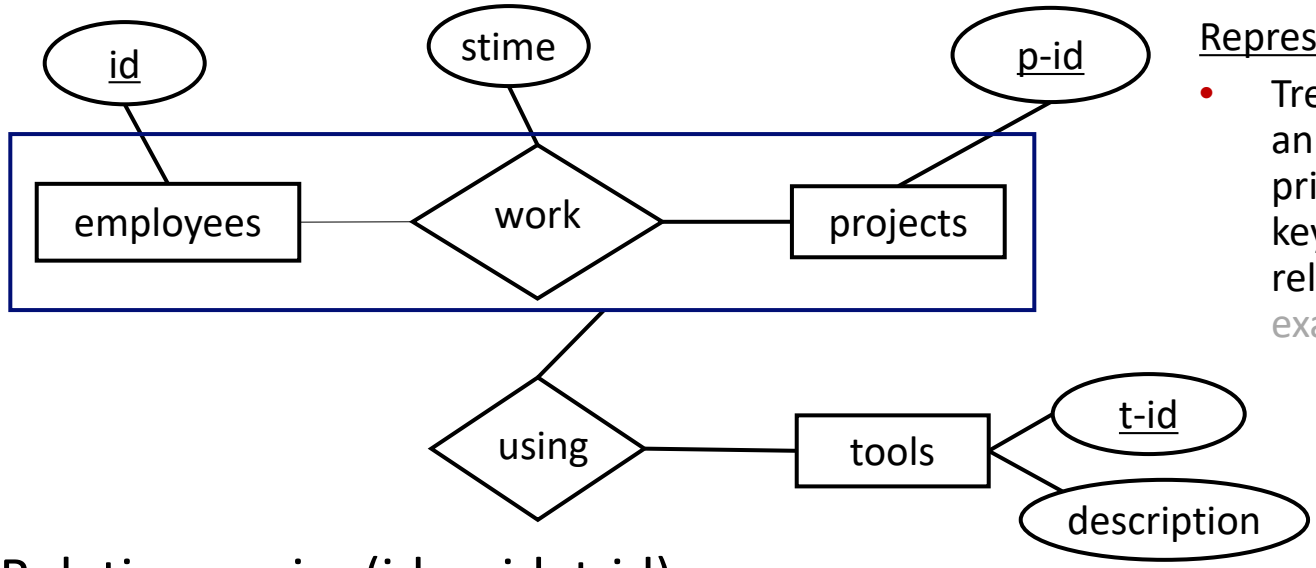
Relations:

Category (name, description)
Key: name

BelongsTo (acct#, trans#, name)
Key: acct#, trans#, name
Foreign Keys:
BelongsTo(acct#,trans#) REFERENCES Transaction(acct#,trans#)
BelongsTo(name) REFERENCES Category(name)

Other relations are "Account" and "Transaction" (see previous slide)

Aggregation



Representing Aggregation:

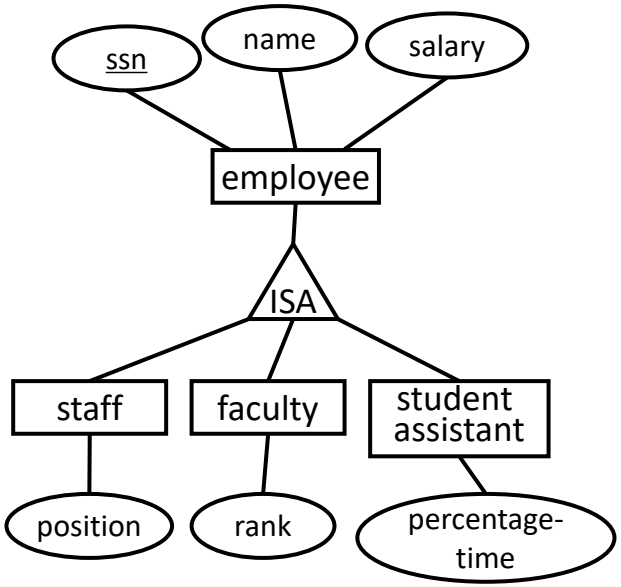
- Treat the aggregation like an entity set, whose primary key is the primary key of the aggregate relation (e.g. "*work*" in the example)

Relation: using(id, p-id, t-id)
Key: id, pid, tid
Foreign Keys:
using(id, p-id) REFERENCES work(id,p-id)
using(t-id) REFERENCES tools(t-id)

What other relations are there in the overall schema?
employees, projects, tools, and work (in addition to using)

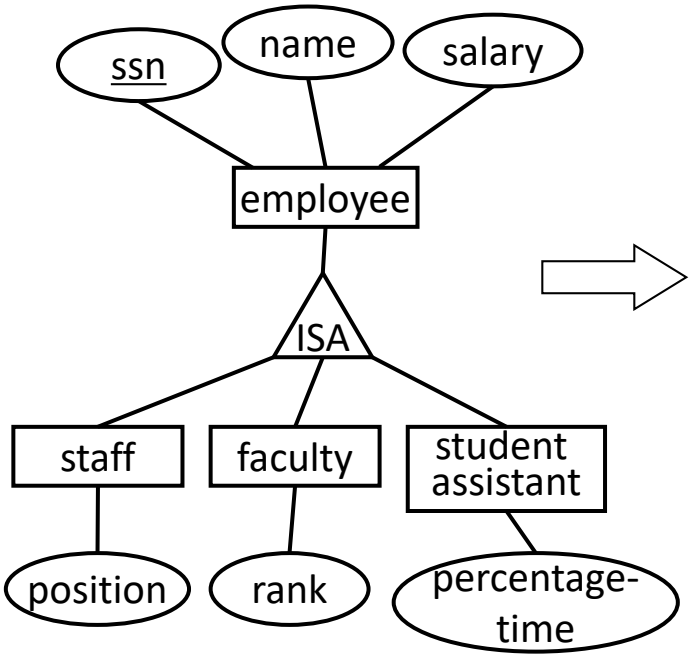
Subclass/Superclass Structures to Relations

- Several approaches are available:
  1. ER Approach – partial participation
  2. Object Oriented Approach – total participation and disjoint



Subclass/Superclass Structures to Relations

1. ER Approach



Relations:

employee(ssn, name, salary)
staff(ssn, position)
faculty(ssn, rank)
studentassistant(ssn, percentage-time)

Key:

ssn for all relations

Foreign Keys:

staff(ssn) REFERENCES **employee**(ssn)
faculty(ssn) REFERENCES **employee**(ssn)
studentassistant(ssn) REFERENCES **employee**(ssn)

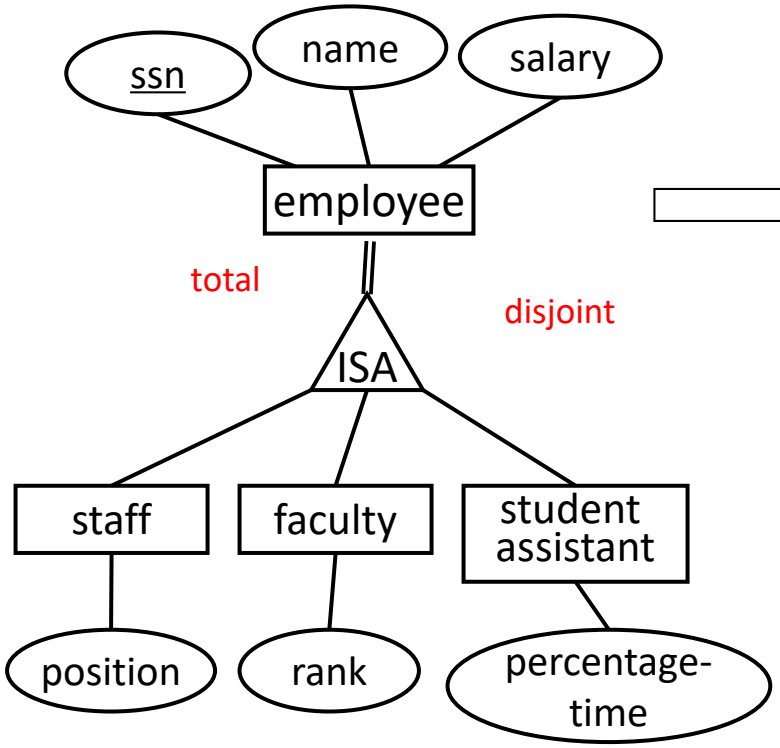
Note: cannot represent a total constraint

ER Approach:

- **Create tables for all entity sets, and treat specialized entity subsets like weak entity sets (without discriminators)**

Subclass/Superclass Structures - Disjoint & Total Participation

2. O-O Approach



Relations:

staff (ssn, name, salary,position)

faculty (ssn, name, salary, rank)

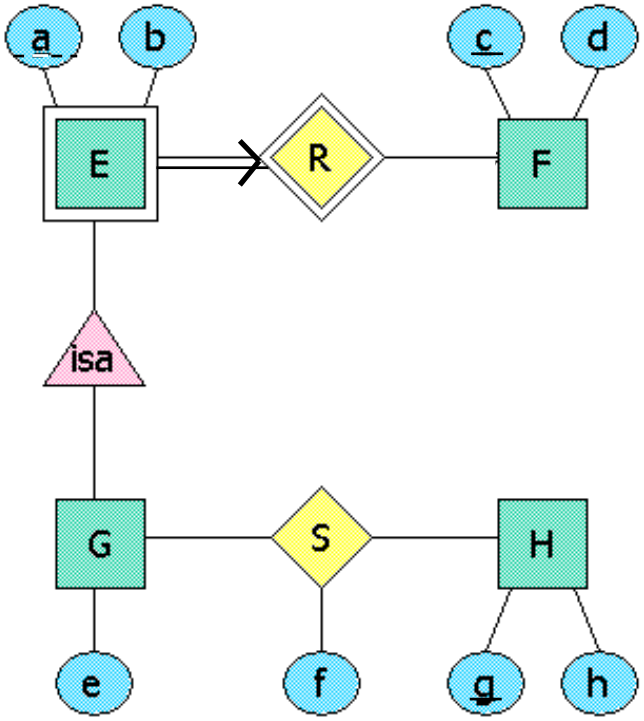
studentassistant (ssn, name, salary, percentage_time)

Key: ssn for all the relations

Requires a union to construct all employees

- Cannot use the design if it is a partial constraint: cannot represent employees who are not staff, faculty, or student assistants!
- Not a good design if it is an overlap constraint: if staff could also be a student assistant, then redundancy arises

ER to Relational Mapping –Example1



Translate the above E/R diagram to relations.

ER to Relational Mapping –Example2