ER – Relational Model

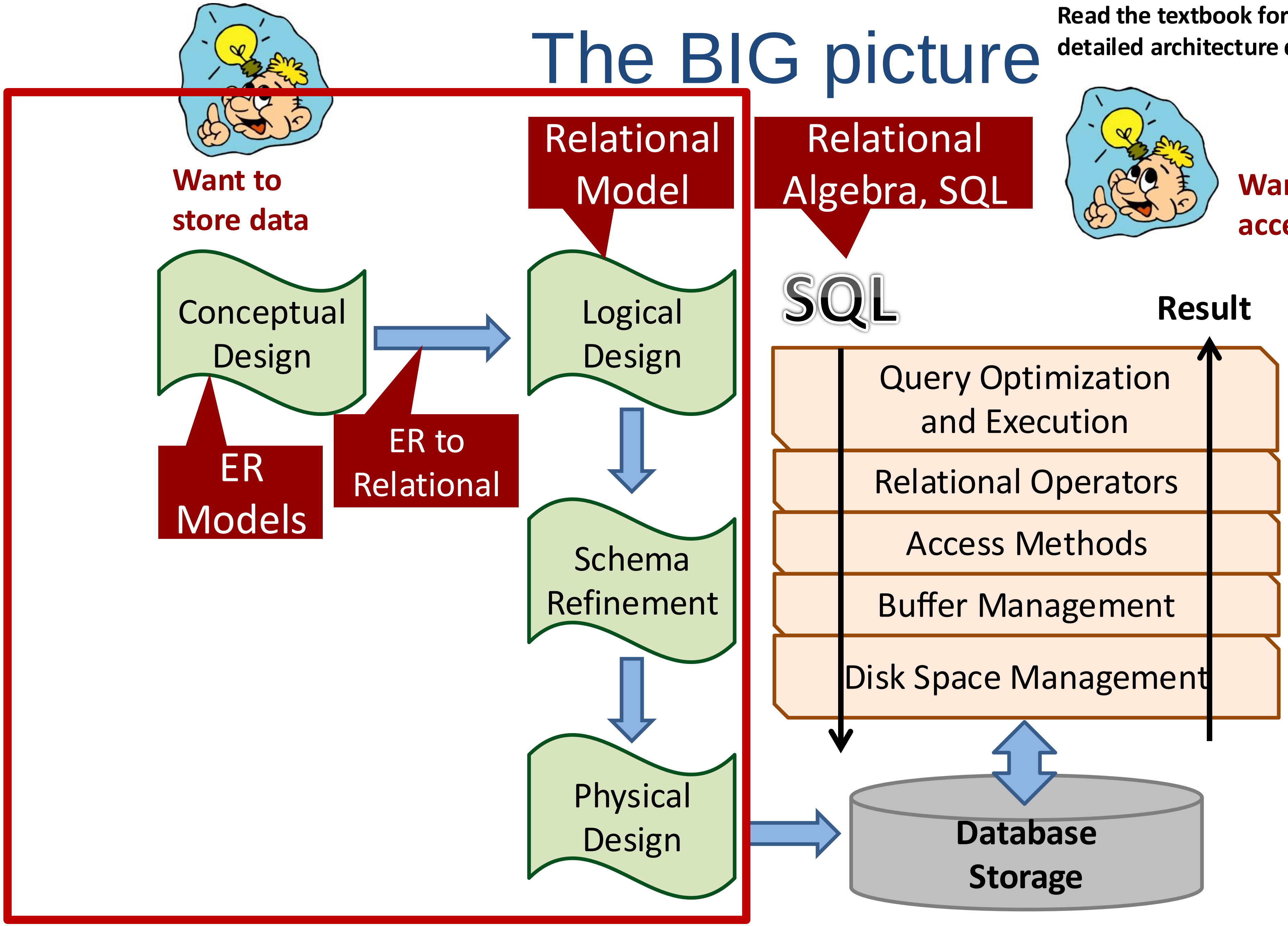
Week 3

DATA-INTENSIVE APPLICATIONS AND SYSTEMS

EPFL

The BIG picture

Read the textbook for a more detailed architecture description



ER Model

- High level, conceptual representation of how to describe the user needs and data
- Capture the practical requirements and constraints of the given use-case
- Conceptual design – entities, attributes, relations, and the constraints between them
- More complex relations with constraints:
  - Key constraints
  - Participation constraints
  - ISA hierarchy
  - Weak entities

- **Revise and learn: week 2**

Relational Model

- Logical database design, based on high level conceptual design
- Translating ER model to relational model
  - Lowering the level of abstraction - concretization based on the requirements
- Schema – structural, more concrete description of relations in a database
- Integrity constraints enforce the data present/inserted follow the rules of schema
- Each attribute has assigned domain/type, and eventual value constraints
  - Null value, unique constraints
- Key constraints (minimal unique descriptor of the row – based on use-case)
- Referential integrity constraints (references to keys in other tables)
- Relational model provides **schema** – description of relations/table – maps to DBMS

- **Revise and learn: week 3**

SQL Overview (DDL, DML, Query)

- ```
  CREATE TABLE <name> ( <field> <domain>,<constraints> … )
  ```

Create the table based on schema

- ```
  INSERT INTO <name> (<field names>)
       VALUES (<field values>)
  ```

- ```
  DELETE FROM <name>
        WHERE <condition>
  ```

Populate/modify/delete the data in the table

- ```
  UPDATE <name>
     SET <field name> = <value>
   WHERE <condition>
  ```

- ```
  SELECT <fields>
    FROM <name>
   WHERE <condition>
  ```

Query the data

Revise and learn: week 3
  ```

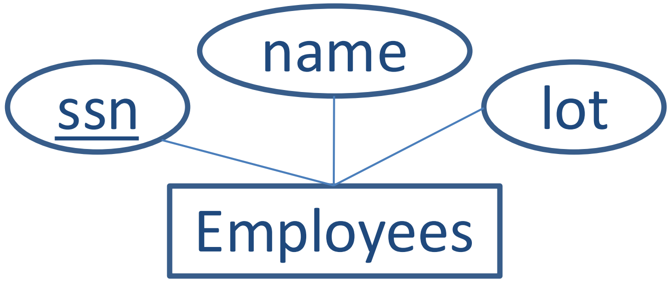
Translating ER to Relational Model

- ER model is at higher level and more expressive than relational model
- Some constraints cannot be captured directly by relational model constraints
  - In these cases more complex methods to check validity of the data are used: Check/Assert constraints, Triggers, disabling constraints until transaction ends
- Rules exist how to translate ER to Relational Model, often there is no single solution
- **The goal is to eliminate redundancy as much as possible**
  - **Also called schema normalization**

- The rules, their consequences, reasoning and limitations of some rules – in the book
- It is easier to reason and translate binary relations (observing relations 2 by 2)
  - This is why aggregates are useful, as they conceptually observe a relation as entity
- **Upcoming: an overview of most common translation rules (NON-EXHAUSTIVE LIST)**
- **OFTEN THERE IS NO SINGLE WAY TO TRANSLATE THE ER TO RELATIONAL MODEL**

Translating Entity Sets

- Create a table for every entity set
- Attributes become columns
- Specify appropriate types
- Designate Primary Key
- Specify other integrity constraints




```
CREATE TABLE Employees (
  ssn CHAR(11),
  name CHAR(30),
  lot INTEGER,
  PRIMARY KEY (ssn)
);
```

Translating Relationship Sets (General Case)

- Create table for the relationship set
  - Sometimes we merge relationship with some entity (more details later)
- Add primary keys of participating entities as columns
  - Add foreign key constraints to the respective tables
- Add attributes of the relationship set as columns
- Capture as many constraints as possible
  - Some constraints may be lost (participation)
- Primary key of the relation depends on the key constraints
  - Always a subset of the primary keys of the entities

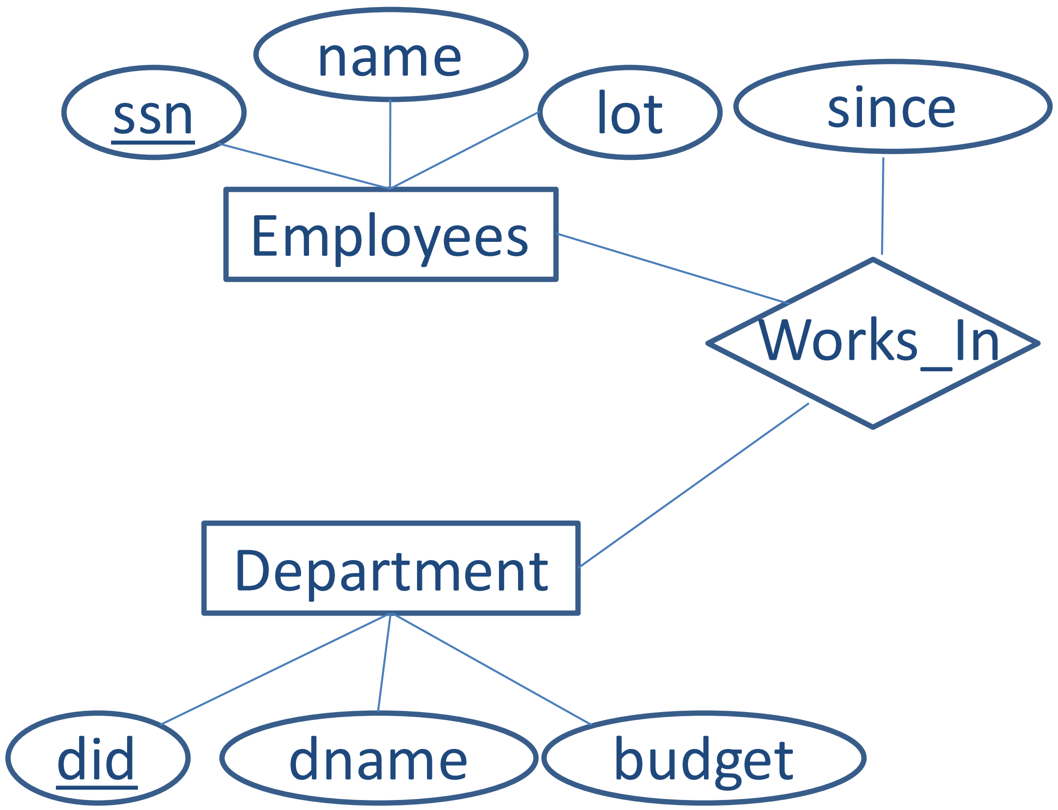
Translating N:M Key Constraints

- Possible connection between many tuples from A with B
  - (at most cartesian prod.)
  - if the tuples don’t map, they won’t be present in R

- Handle all 3 cases similarly
- Create separate table for R as described earlier

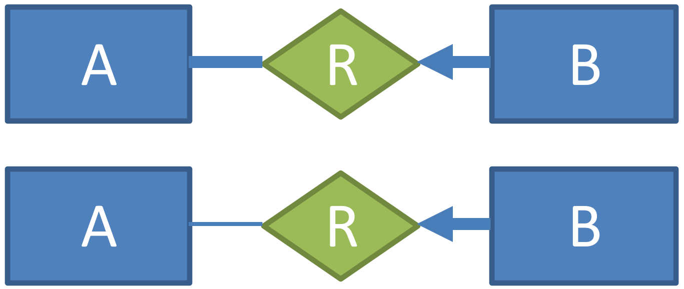
- Primary key of R is (pk(A),pk(B))
- Cannot capture participations constraints directly
  - Use assertions if necessary (expensive)

Example

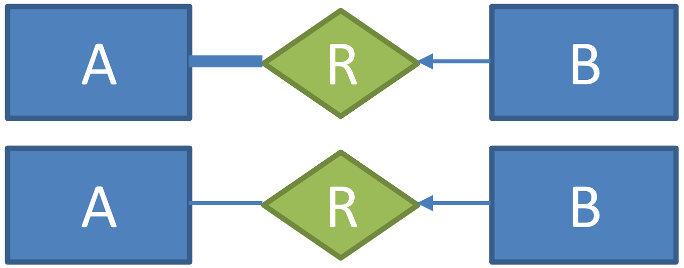



```
CREATE TABLE Works_In(
    ssn CHAR(11),
    did INTEGER,
    since DATE,
    PRIMARY KEY (ssn, did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Department
);
```

Translating 1:N Key Constraints

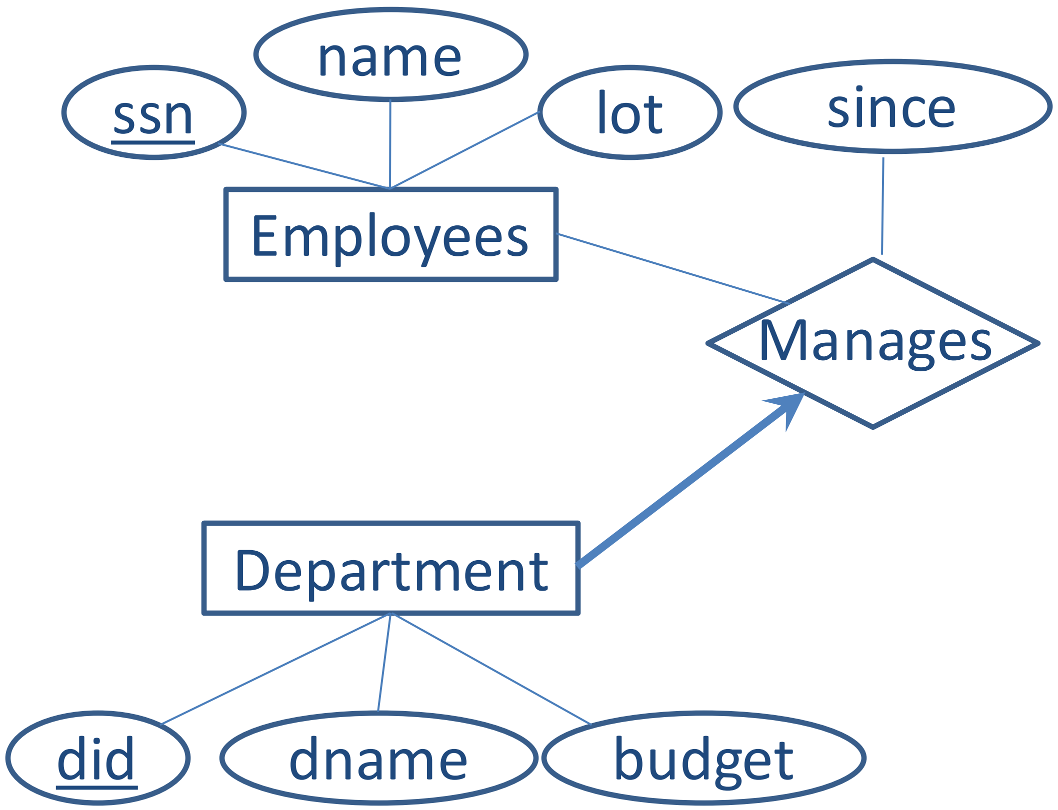


- Pk(B) uniquely identifies the relationship
  - (Pk(A), Pk(B)) not primary key because it is not minimal
- Possible to create table for R as in general case
- Another idea : merging R and B into a single table
  - Add attributes of R and primary key of A as columns in B
  - Add Foreign Key constraints to A
  - Merged RB table makes it possible to capture participation constraint on B
    - If B has total participation constraint, make pk(A) NOT NULL in B
    - Otherwise pk(A) can be NULL in B
      - Trade-off between storing NULLs in merged table or creating table for R with fewer rows



Takeaway: key and attribute migration to other tables used to enforce the key constraints
Think of reducing the redundancy, not having tuples with many NULL values

Example




```
CREATE TABLE Dept_Mgr(
    did INTEGER,
    dname CHAR(20),
    budget REAL,
    since DATE,
    ssn CHAR(11) NOT NULL,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees
);
```

Translating 1:1 Key Constraints

- Case 1: Both entity sets have total participation
  - Only possible when both have the same number of entities
  - Merge both entities and the relation into a single table
  - Choose either pk(A) or pk(B) as the primary key
    - Set the other one as UNIQUE

- Case 2: One entity set has total participation
  - Merge R and B as in previous slide
  - Pk(A) is foreign key in B and NOT NULL and UNIQUE
  - Mirror case handled similarly

- Case 3: both have partial participation
  - Either create new table for R or merge R with one of the entities
  - One of pk(A) or pk(B) is designated primary key
    - Other one UNIQUE
  - Trade off between storing NULLs in merged table vs new table with fewer rows

Takeaway: NOT NULL controls participation; Key selection and UNIQUE control upper bound

Translating Aggregations

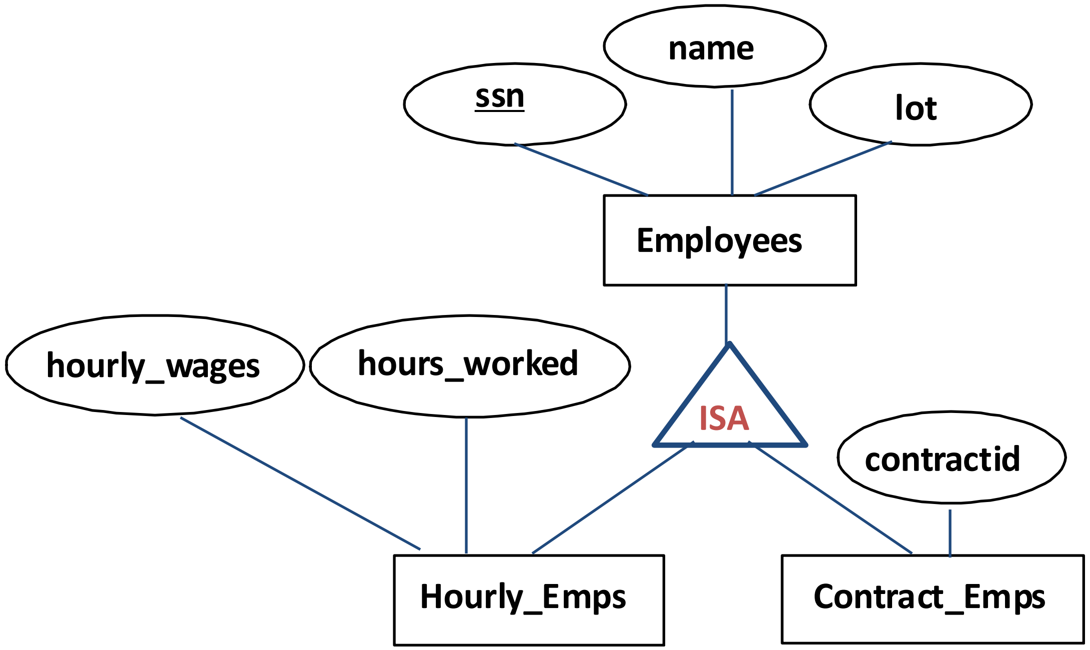
- Translate the binary relation captured by aggregation
- Then observe how the overall relation was translated (key, attributes) to a table
- Finally, using that table continue in same way, as with any other binary relation
- Sometimes possibility of merging two relationships

Translating Hierarchies

- Generally, create separate table for all entities involved
  - Add attributes of each entity to respective table
  - Also add primary key of the superclass as primary key of each of the subclass tables
  - Add foreign key constraint to superclass table
    - Any deletion of superclass must be cascaded to subclasses

- If the hierarchy is non-overlapping and covering, merge superclass entity with each subclass entity individually
  - Attributes of superclass added to each subclass
  - No table for the superclass

Example




```sql
CREATE TABLE Hourly_Emps (
   ssn CHAR(11),
   hours_worked REAL,
   hourly_wages REAL,
   PRIMARY KEY (ssn),
   FOREIGN KEY (ssn) REFERENCES Employees
    ON DELETE CASCADE
);
```

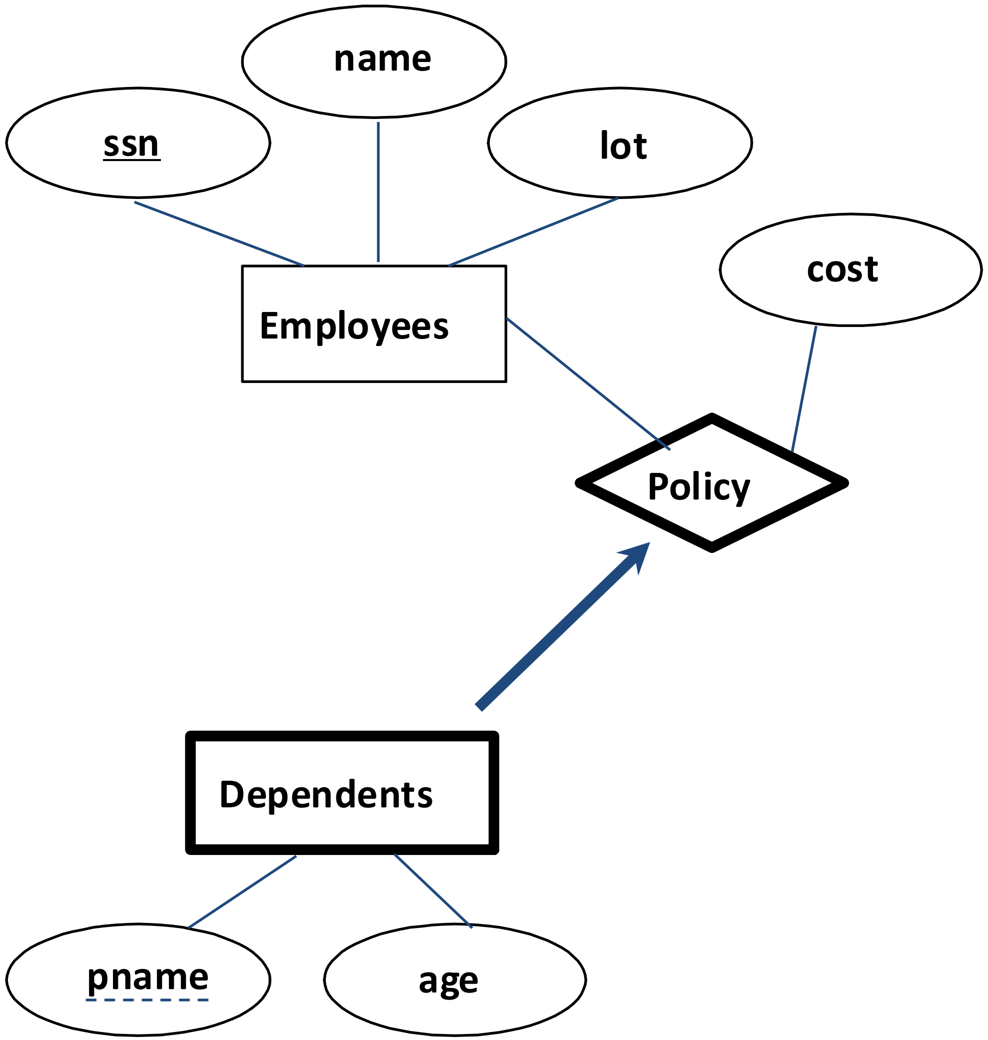
Translating Weak Entities

- Weak entities exist while the parent exist, and are uniquely identified by the parent

- Identifying relationship has key constraint as well as participation constraint on the side of the weak entity set
  - Merge R with the weak entity
  - Primary key is the combination of the owner entity pk and the weak key

- Add foreign key constraint to the table of the owner entity
  - On deletion of owner , CASCADE the delete to all children

Example




```
CREATE TABLE Dept_Policy(
    pname CHAR(20),
    age INTEGER,
    cost REAL,
    ssn CHAR(11),
    PRIMARY KEY (pname, ssn),
    FOREIGN KEY (ssn) REFERENCES Employees
     ON DELETE CASCADE
);
```

Final remarks

- Further discussion and descriptions in the book/lecture slides
- More possible cases, for example IS-A hierarchy
- Revise the book/materials/online for SQL DDL/DML
- Understand the necessity for constraints and how to express them
- Understand the required integrity constraint – is it expressible in relational model
- Keep in mind the minimality of data duplication and reducing many NULL values in table rows

- **Reminder: queries return (multi)sets, careful when you are allowed to use "=" in WHERE clause**
- **Reminder: if you are using set operations (UNION…), queries must be set compatible == same attributes (including compatible domain)**