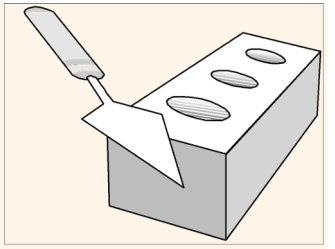
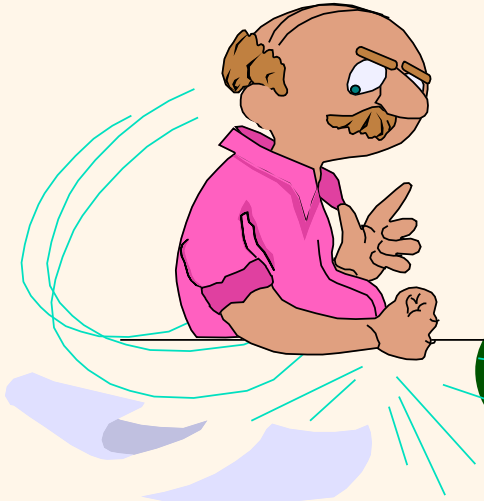


# *Introduction to Data Management*

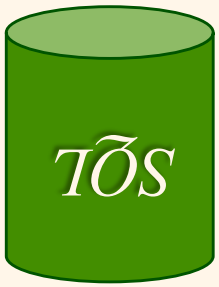


## **** The "Flipped" Edition ****

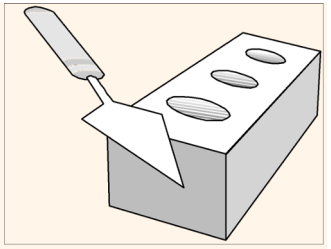
## *Lecture #5*

## *(ER → Relational Mapping)*

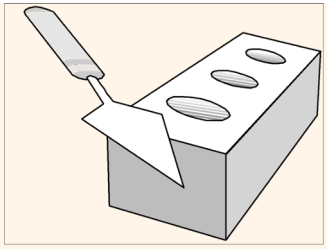
Instructor: Mike Carey
mjcarey@ics.uci.edu



# *Today's Notices*

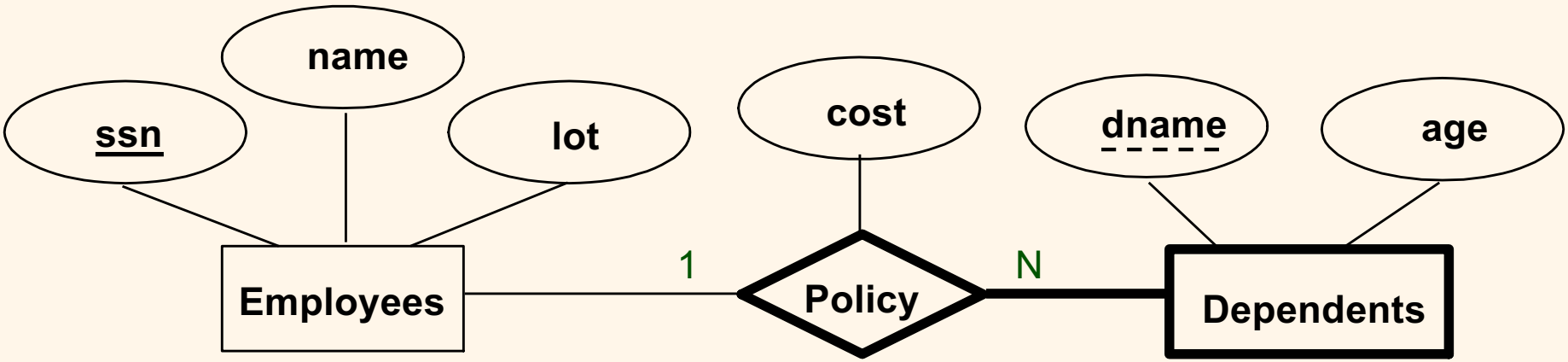



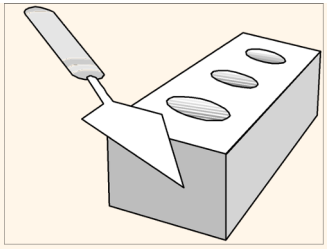
- Keep watching the wiki page:
  - http://www.ics.uci.edu/~cs122a/
- And, of course, follow the Piazza page:
  - piazza.com/uci/fall2021/cs122aeecs116/home
- HW#1 is in flight! (Keep Q's coming on Piazza...)
  - ***SWOOSH.com***
  - And, of course, avoid discussing *solutions* publicly here!
- Remember that quizzes run from Wed 3PM – Fri 3PM
  - Don't miss the free points! (You will if you fall behind!)



# *Review: Weak Entities*

- A *weak entity* can be identified (uniquely) only by considering the primary key of another (*owner*) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (1 owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.



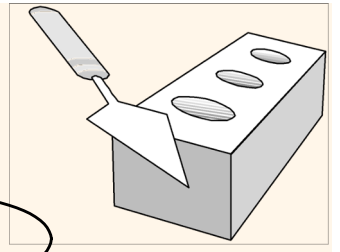


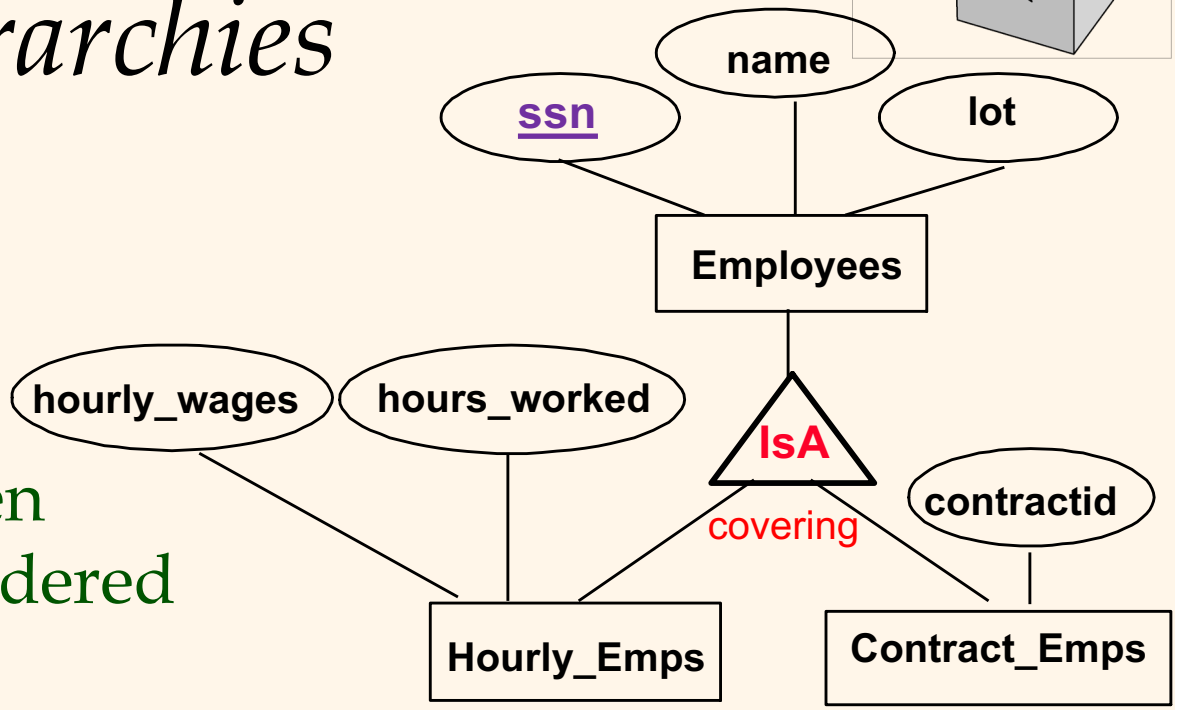
# *Translating Weak Entity Sets*

- Weak entity set and identifying relationship set are translated into a *single table*.
  - When the owner entity is deleted, all of its owned weak entities must also be deleted.

```
CREATE TABLE Dependents2 (
   dname  VARCHAR(20),
   age  INTEGER,
   cost  REAL,
   ssn  CHAR(11) NOT NULL,
   PRIMARY KEY  (dname, ssn),
   FOREIGN KEY  (ssn) REFERENCES Employees
      ON DELETE CASCADE)
```

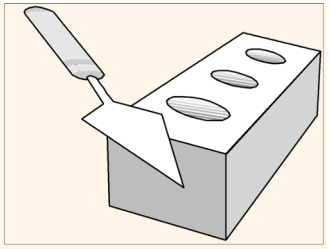
# *Review: IsA Hierarchies*






- As in C++, or other PLs, attributes are inherited.
- If we declare A **IsA** B, then every A entity is also considered to be a B entity.

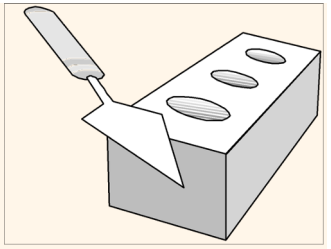
- *Overlap constraints*: Can employee Joe be an Hourly_Emps as well as a Contract_Emps entity? (***disjoint*** if not)
- *Covering constraints*: Must every Employees entity be either an Hourly_Emps or a Contract_Emps entity? (***covering*** if so)



# *From IsA Hierarchies to Relations*

- ***Most general and "clean" approach*** (recommended)**:**
  - 3 relations: Employees, Hourly_Emps, & Contract_Emps.
    - *Hourly_Emps*: Every employee recorded in Employees. For hourly emps, ***extra*** info recorded in Hourly_Emps (*hourly_wages*, *hours_worked*, *ssn*); delete Hourly_Emps tuple if its referenced Employees tuple is deleted.
    - Queries about ***all*** employees easy; those involving ***just*** Hourly_Emps require a join to access the extra attributes.
- Another alternative: Hourly_Emps & Contract_Emps.
  - ***Ex:*** *Hourly_Emps*(*ssn*, *name*, *lot*, *hourly_wages*, *hours_worked*)
  - If each employee must be in one of the two subclasses...

    *(Q: Can we **always** do this, then? A: Not w/o **redundancy**!)*

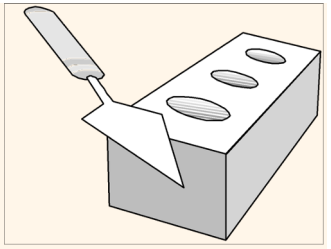


# *IsA Hierarchy Translation Options*

- I. "Delta table" approach (recommended):
  - *Emps(**ssn**, name, lot)* (All Emps partly reside here)
  - *Hourly_Emps(ssn*, wages, hrs_worked)*
  - *Contract_Emps(ssn*, contractid)*
- II. "Union of tables" approach:
  - *Emps(**ssn**, name, lot)*
  - *Hourly_Emps(ssn, name, lot, wages, hrs_worked)*
  - *Contract_Emps(ssn, name, lot, contractid)*
- III. "Mashup table" approach:
  - *Emps(kind, ssn, name, lot, wages, hrs_worked, contractid)*

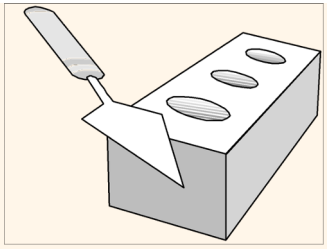
(**ssn* here is both a local PK as well as an FK referencing Emps)

**Things to consider:**
- Expected queries?
- PK/unique constraints?
- Relationships/FKs?
- Overlap constraints?
- Space/time tradeoffs?

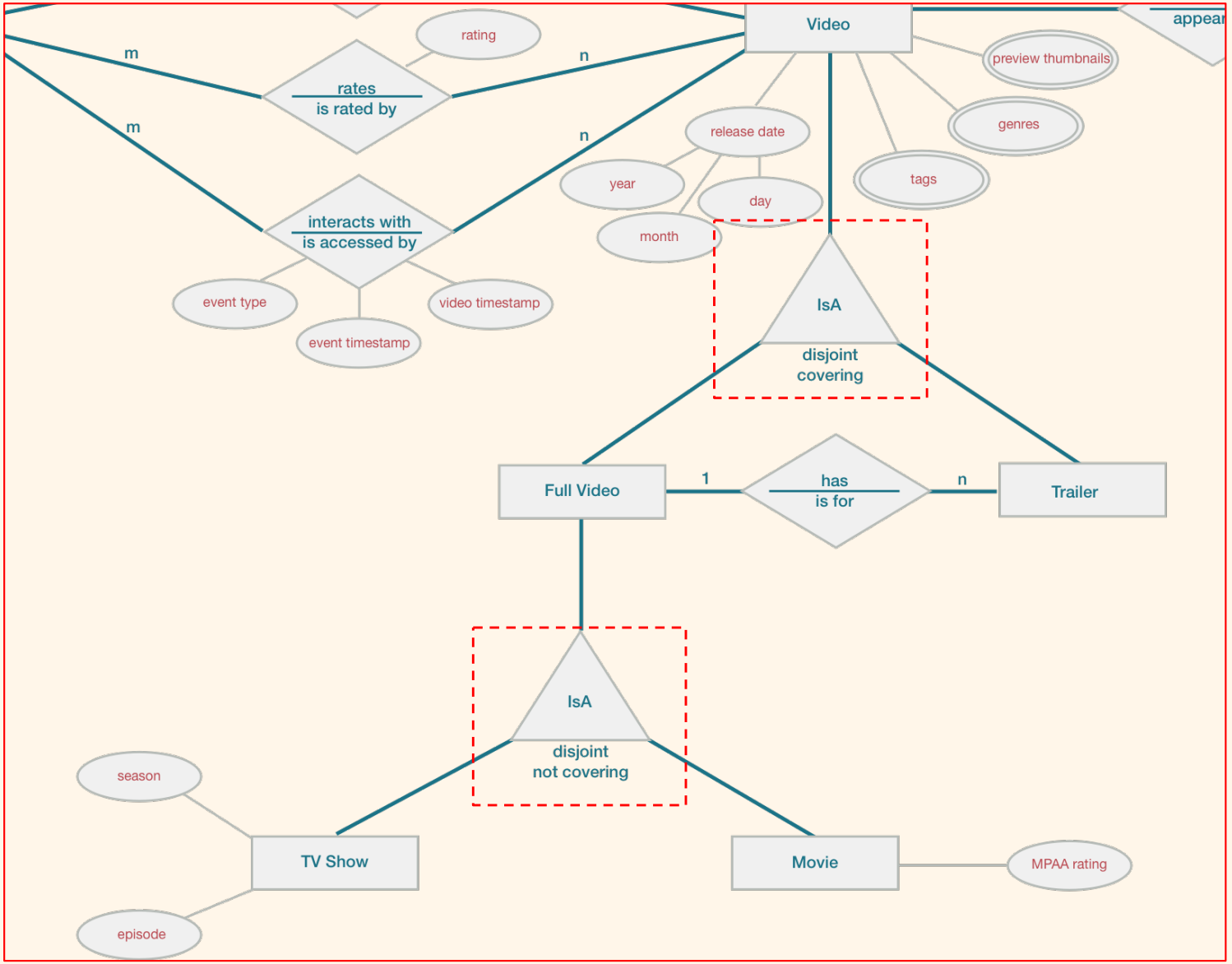


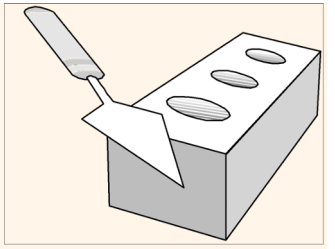
# *IsA Considerations (cont'd.)*

- Query convenience
  - *Ex:* List the names of all Emps in lot 12A
- PK enforcement
  - *Ex:* Make sure that ssn is unique for all Emps
- Relationship (FK) targets
  - *Ex:* Lawyers table REFERENCES Contract_Emps
- Handling of overlap constraints
  - *Ex:* Sally is under a contract for her hourly work
- Space and query performance tradeoffs
  - *Ex*: List all the info about hourly employee 123
  - *Ex:* What if most employees are "just plain employees"?



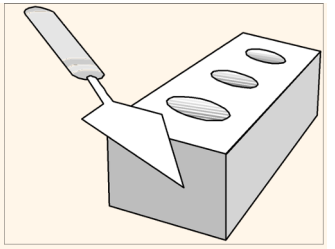
# *Logical Design for SQL (Ex. 2)*



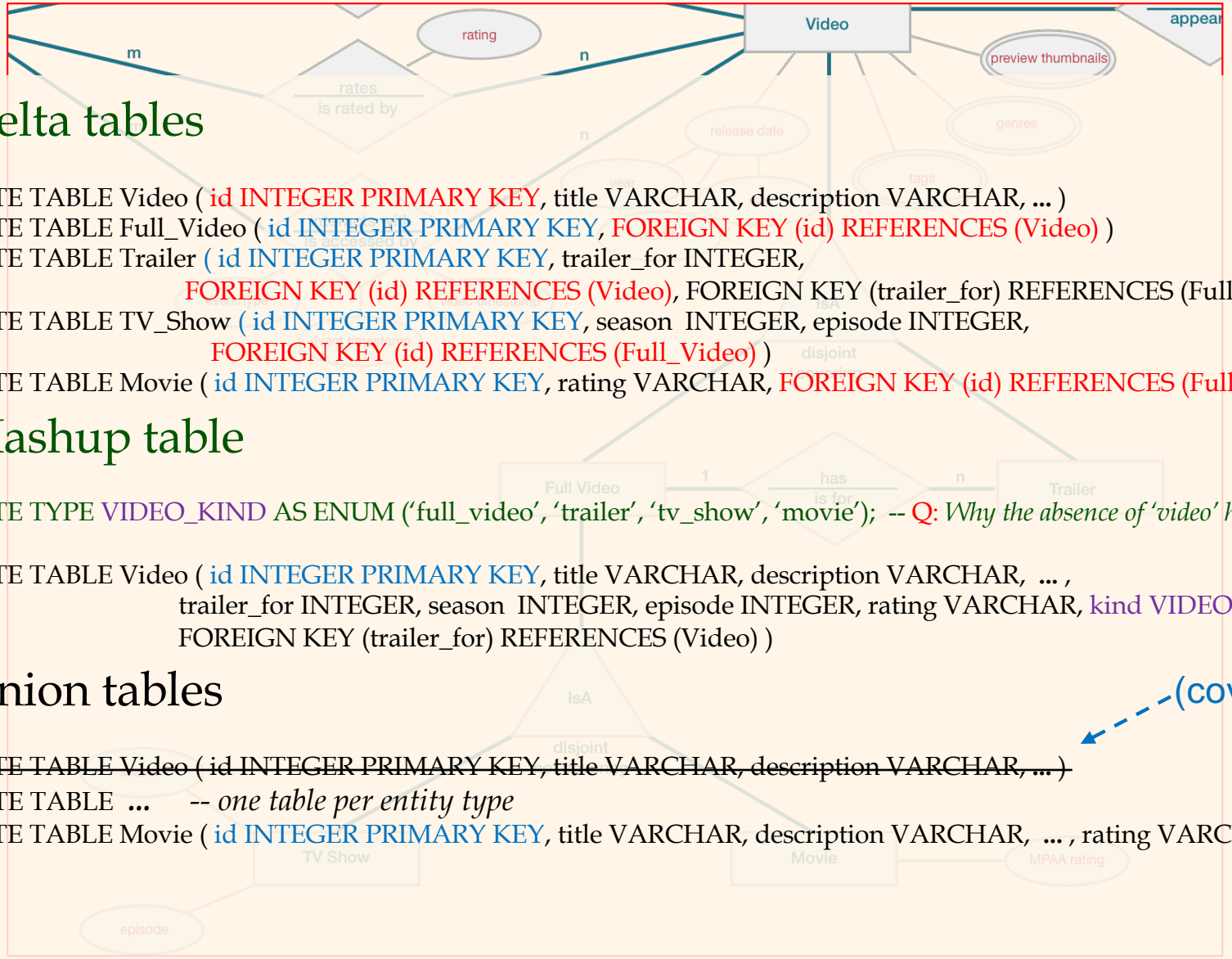


# *Logical Design for SQL (Ex. 2)*

- Again, an IsA hierarchy in an E-R schema can be "tabularized" for SQL in one of at least three ways:
  - Delta tables
    - Root table defines the PK and the top-level fields
    - "Sub-tables" have the same PK (also an FK to its parent) and the newly-appearing fields
    - Join queries are needed to materialize all instances of a non-root entity set
  - Mashup table
    - The "one table to rule them all" – has all fields from everywhere in the entity hierarchy
    - Must also tag rows with information about their particular entity type(s)
  - Union tables
    - Every table has the same PK plus *all* fields (newly-appearing and inherited) for its entity type
    - Each entity in the hierarchy therefore lives (fully and only) in a table unique to its entity type/subtype
    - Union-all queries are needed to materialize all instances of a non-leaf entity type
    - (Comes up short w.r.t. PK enforcement)



# *Logical Design for SQL (Ex. 2)*




- Delta tables

```
CREATE TABLE Video ( id INTEGER PRIMARY KEY, title VARCHAR, description VARCHAR, ... )
CREATE TABLE Full_Video ( id INTEGER PRIMARY KEY, FOREIGN KEY (id) REFERENCES (Video) )
CREATE TABLE Trailer ( id INTEGER PRIMARY KEY, trailer_for INTEGER,
                       FOREIGN KEY (id) REFERENCES (Video), FOREIGN KEY (trailer_for) REFERENCES (Full_Video) )
CREATE TABLE TV_Show ( id INTEGER PRIMARY KEY, season  INTEGER, episode INTEGER,
                       FOREIGN KEY (id) REFERENCES (Full_Video) )
CREATE TABLE Movie ( id INTEGER PRIMARY KEY, rating VARCHAR, FOREIGN KEY (id) REFERENCES (Full_Video) )
```

- Mashup table

```
CREATE TYPE VIDEO_KIND AS ENUM ('full_video', 'trailer', 'tv_show', 'movie');  -- Q: Why the absence of 'video' here?

CREATE TABLE Video ( id INTEGER PRIMARY KEY, title VARCHAR, description VARCHAR,  ... ,
                     trailer_for INTEGER, season  INTEGER, episode INTEGER, rating VARCHAR, kind VIDEO_KIND,
                     FOREIGN KEY (trailer_for) REFERENCES (Video) )
```

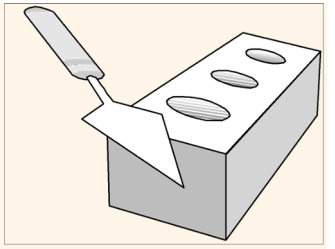
- Union tables

(covering)

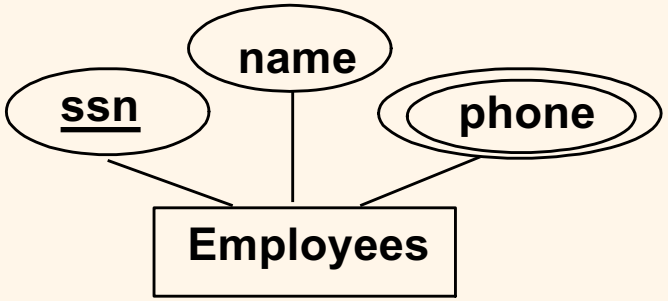
~~CREATE TABLE Video ( id INTEGER PRIMARY KEY, title VARCHAR, description VARCHAR, ... )~~

```
CREATE TABLE  ...     -- one table per entity type
CREATE TABLE Movie ( id INTEGER PRIMARY KEY, title VARCHAR, description VARCHAR,  ... , rating VARCHAR)
```

# *Mapping Advanced ER Features*

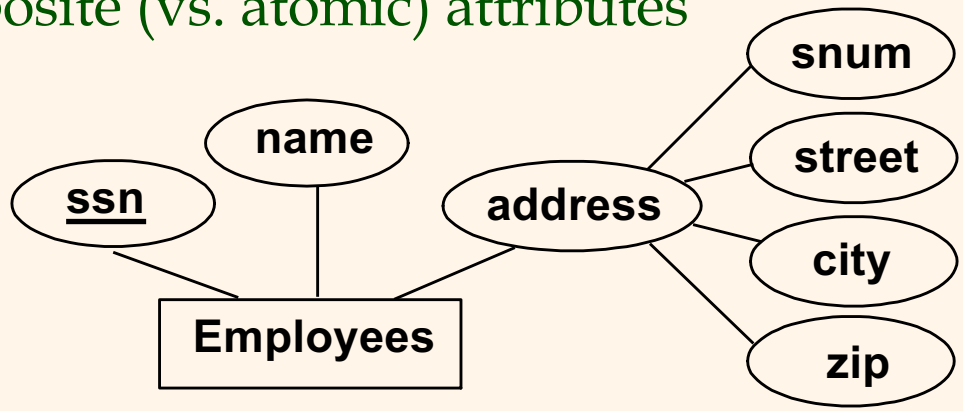


- Multi-valued (vs. single-valued) attributes



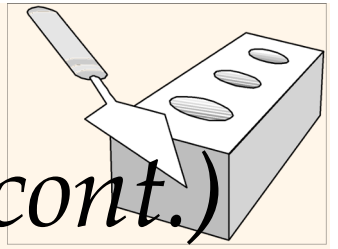

*Employees_phones(ssn, phone)*
- ssn is an FK in this table
- (ssn, phone) is its PK

- Composite (vs. atomic) attributes

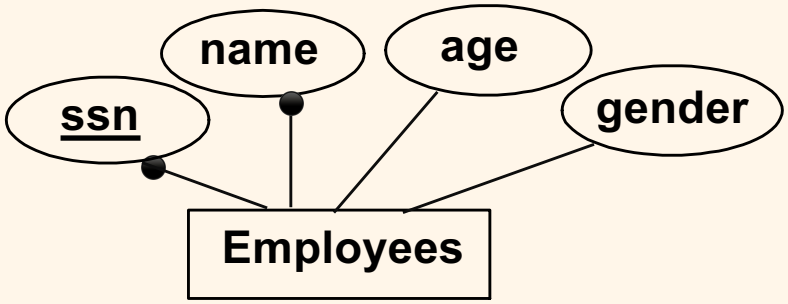



*Employees(ssn, name, address_snum, address_street, address_city, address_zip)*

# *Mapping Advanced ER Features (cont.)*



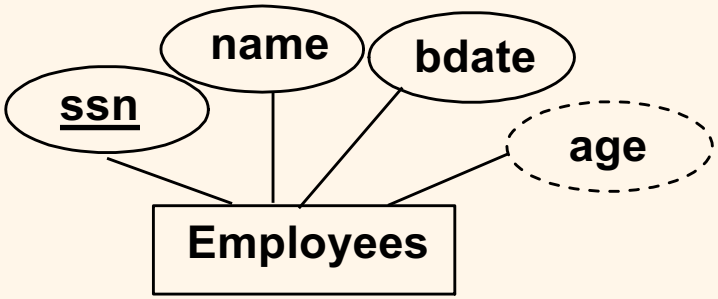
- Mandatory (vs. optional) attributes

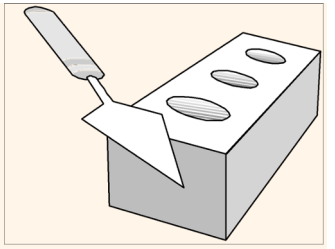



*Employees(<u>ssn</u>, name, age, gender)*

- ssn is the PK
- name VARCHAR(20) **NOT NULL**
- Note: **PRIMARY KEY → NOT NULL**

- Derived (vs. stored) attributes



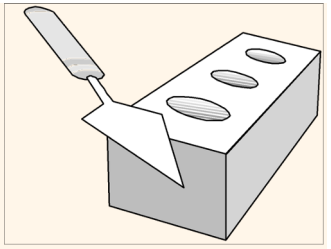



# *SQL Views (and Security)*

- A *view* is just a relation, but we store its *definition* rather than storing the (materialized) set of tuples.

```
CREATE VIEW YoungStudents (name, login)
    AS SELECT S.name, S.login
       FROM Students S
       WHERE S.age < 19;
```

- Views can be used to present needed information while hiding details of underlying table(s).
  - Given YoungStudents (but not Students), we can see (young) students *S* with only their names and logins.

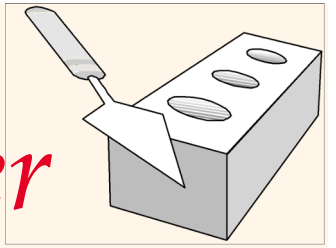


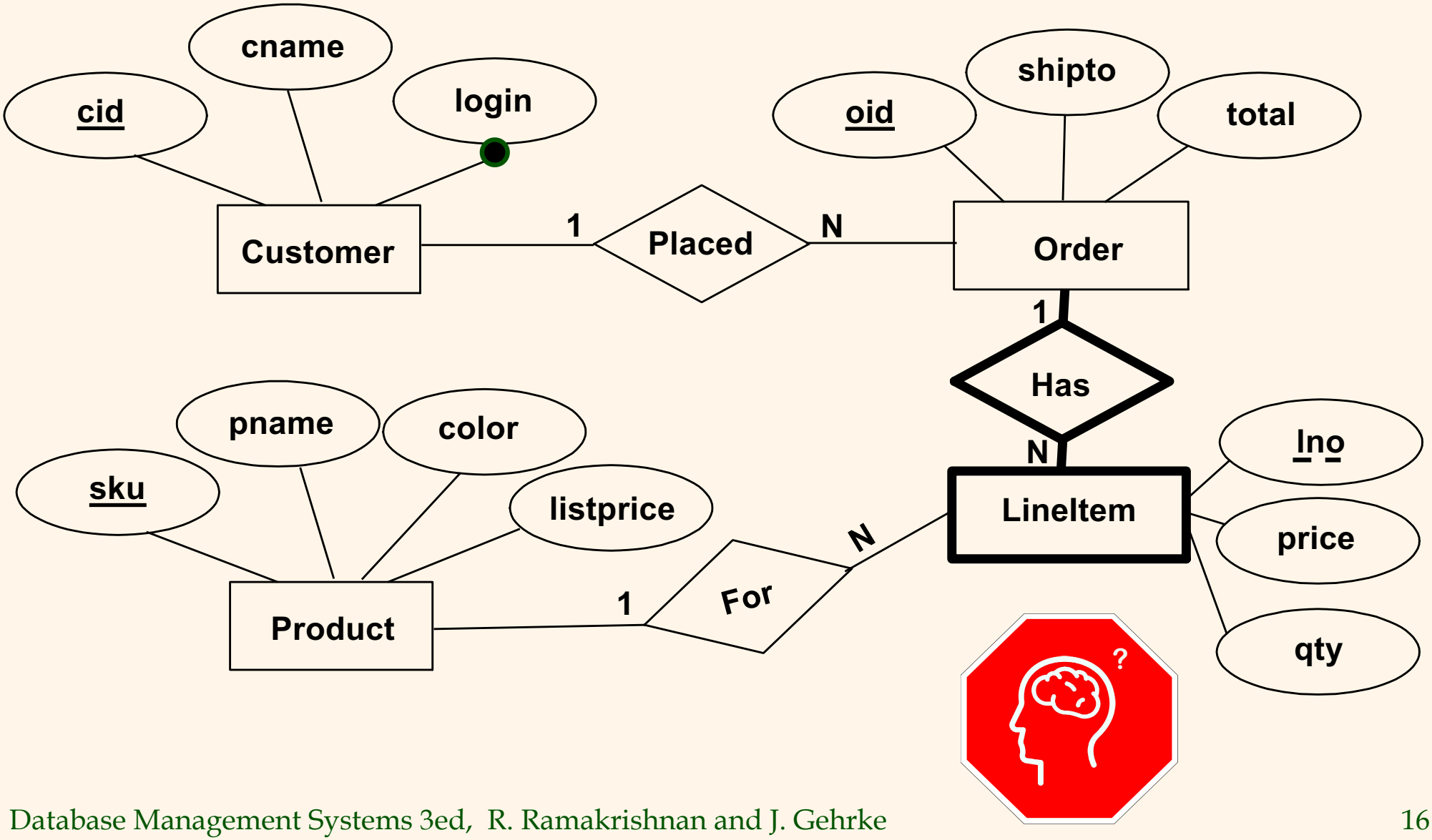
# *SQL Views (Cont'd.)*

- Other view uses in our ER translation context might include:
  - ***Derived attributes, e.g., age (vs. birthdate)***
  - Simplifying/eliminating join paths (for SQL)
  - Beautifying the "Mashup table" approach (to IsA)

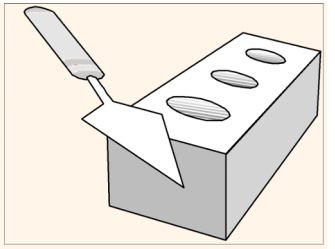
```
CREATE VIEW EmployeeView (ssn, name, bdate, age)
    AS SELECT E.ssn, E.name, E.bdate,
              date_part('year', age(E.bdate))::int
       FROM Employees E;
```

# *Review: Putting the Basics Together*





# *Review: Putting It Together (Cont'd.)*



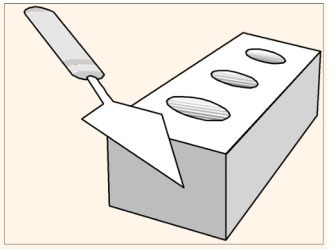
```
CREATE TABLE Customer (
   cid INTEGER,
   cname VARCHAR(50),
   login VARCHAR(20)
      NOT NULL,
   PRIMARY KEY (cid),
    UNIQUE (login))

CREATE TABLE Product (
   sku INTEGER,
   pname VARCHAR(100),
   color VARCHAR(20),
   listprice DECIMAL(8,2),
   PRIMARY KEY (sku))
```

```
CREATE TABLE Order (
   oid INTEGER,
   custid INTEGER,
   shipto VARCHAR(200),
   total DECIMAL(8,2),
   PRIMARY KEY (oid),
   FOREIGN KEY (custid) REFERENCES Customer))

CREATE TABLE LineItem (
   oid INTEGER,
   lno INTEGER,
   price DECIMAL(8,2),
   qty INTEGER,
    sku INTEGER,
   PRIMARY KEY (oid, lno),
   FOREIGN KEY (oid) REFERENCES Order
        ON DELETE CASCADE),
   FOREIGN KEY (sku) REFERENCES Product))
```

# *Review: Putting It Together (Cont'd.)*



## Customer

| cid | cname | login |
|---|---|---|
| 1 | Smith, James | jsmith@aol.com |
| 2 | White, Susan | suzie@gmail.com |
| 3 | Smith, James | js@hotmail.com |

## Product

| sku | pname | color | listprice |
|---|---|---|---|
| 123 | Frozen DVD | null | 24.95 |
| 456 | Graco Twin Stroller | green | 199.99 |
| 789 | Moen Kitchen Sink | black | 350.00 |

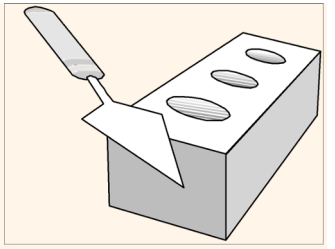
## Order

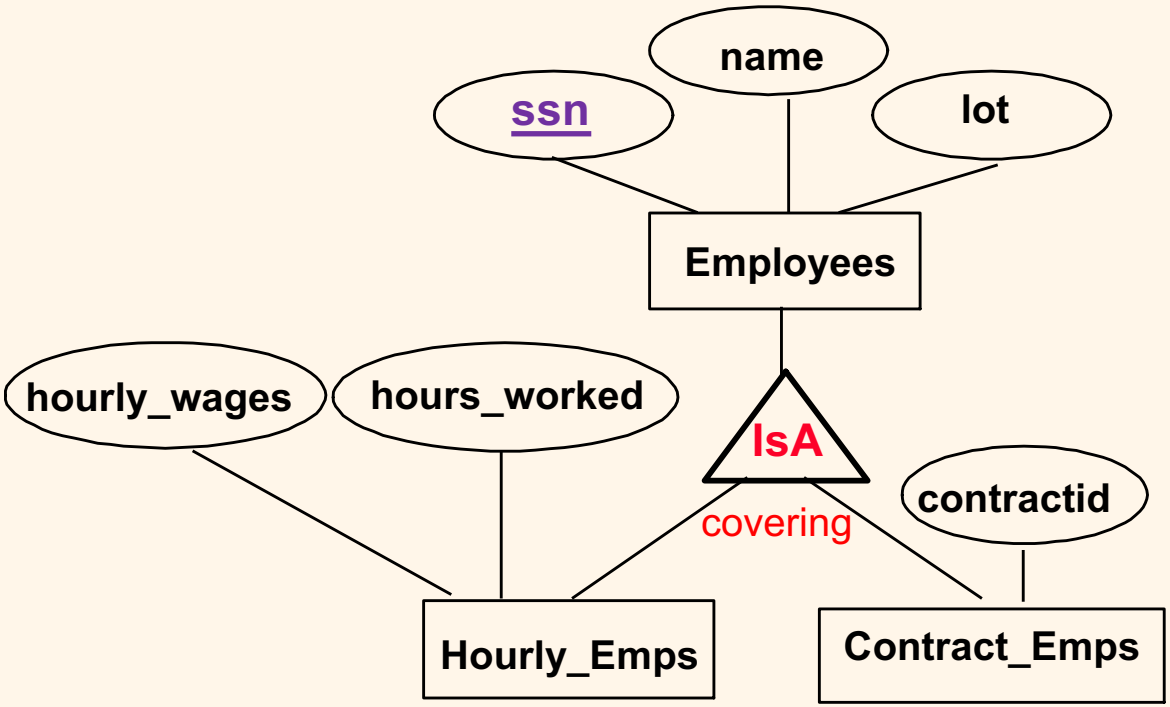
| oid | custid | shipto | total |
|---|---|---|---|
| 1 | 3 | J. Smith, 1 Main St., USA | 199.95 |
| 2 | 1 | Mrs. Smith, 3 State St., USA | 300.00 |

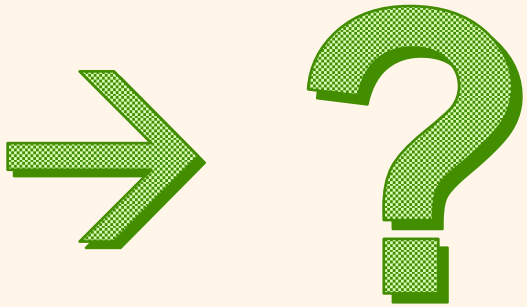
## LineItem

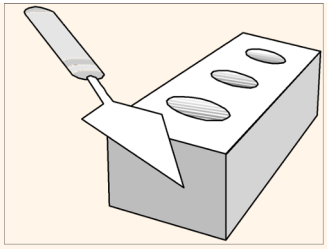
| oid | lno | price | qty | item |
|---|---|---|---|---|
| 1 | 1 | 169.95 | 1 | 456 |
| 1 | 2 | 15.00 | 2 | 123 |
| 2 | 1 | 300.00 | 1 | 789 |

# *Wait! Clarifying IsA Mappings...*

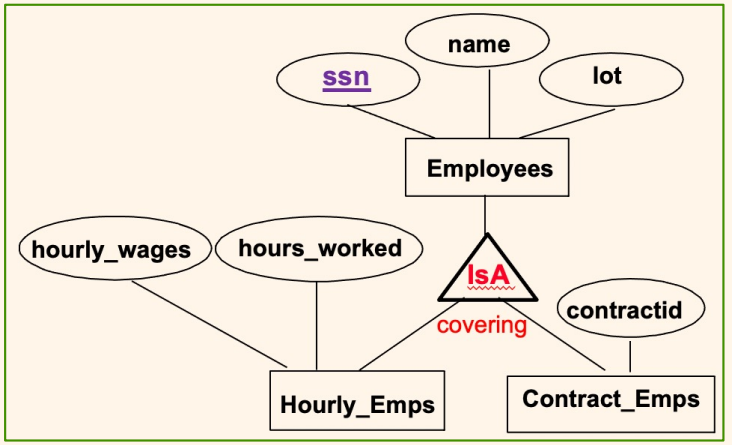






# *Delta Table Approach*




## Employees

| ssn | name | lot |
|---|---|---|
| ~~1~~ | ~~Joe~~ | ~~12~~ |
| 2 | Shelly | 16 |
| 3 | Arvind | *null* |
| 4 | Chen | 11 |

**Note 1**: Joe can't really exist here (*covering*)

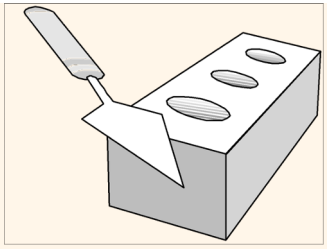
**Note 2***:* Chen couldn't exist if we'd said *disjoint*

## Hourly_Emps

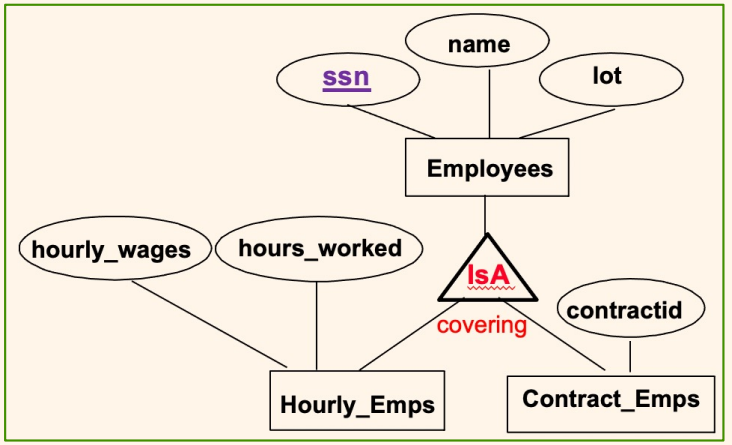
| ssn | wages | hrs_worked |
|---|---|---|
| 2 | 15 | 40 |
| 4 | 250 | 10 |

## Contract_Emps

| ssn | contract_id |
|---|---|
| 3 | 101 |
| 4 | 102 |



# *Union Table Approach*




## Employees

| ssn | name | lot |
|---|---|---|
| ~~1~~ | ~~Joe~~ | ~~12~~ |

**Note 1**: Joe can't really exist here (*covering*)

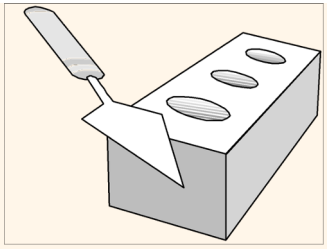
## Hourly_Emps

| ssn | name | lot | wages | hrs_worked |
|---|---|---|---|---|
| 2 | Shelly | 16 | 15 | 40 |
| 4 | Chen | 11 | 250 | 10 |

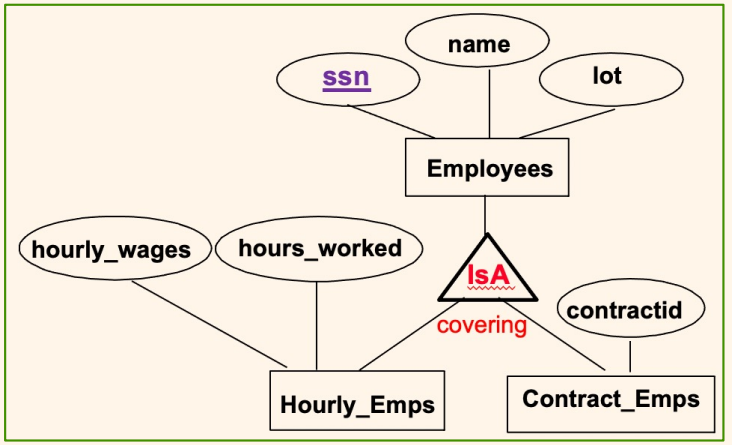
## Contract_Emps

| ssn | name | lot | contract_id |
|---|---|---|---|
| 3 | Arvind | *null* | 101 |
| 4 | Chen | 11 | 102 |

**Note 3***:* No CREATE TABLE way to prevent *ssn* duplication



# *Mashup Table Approach*

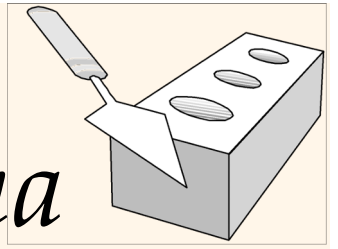



## Employees

| ssn | is_hourly | is_contract | name | lot | wages | hrs_worked | contract_id |
|---|---|---|---|---|---|---|---|
| ~~1~~ | ~~false~~ | ~~false~~ | ~~Joe~~ | ~~12~~ | ~~*null*~~ | ~~*null*~~ | ~~*null*~~ |
| 2 | true | false | Shelly | 16 | 15 | 40 | *null* |
| 3 | false | true | Arvind | *null* | *null* | *null* | 101 |
| 4 | true | true | Chen | 11 | 250 | 10 | 102 |

**Note 1**: Joe can't really exist here (*covering*)

**Note 4***:* Might want to create VIEWs on top...



# *Relational Model and E-R Schema Translation: Summary*

- Relational model: Tabular data representation.
- Simple and intuitive, *very* widely used (RDBMSs)
- Integrity constraints – specified by DBA *based on application semantics*. (DBMS prevents violations)
  - Most important ICs: Primary and foreign keys (PKs and FKs)
  - In addition, we also have domain (column type) constraints
- High-level query languages (including SQL!)
- Rules to translate E-R to relational model
  - Can be done by a human or automatically (via a tool)
  - Entities, relationships, attributes; cardinality, participation,...
  - IsA handling; composite, multi-valued, and derived attributes