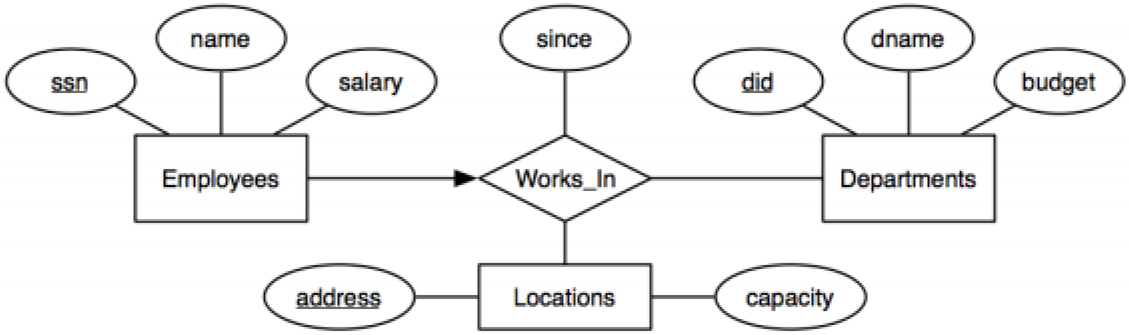
CS186 Week 13 - ER & Functional Dependencies

ER to SQL




```
CREATE TABLE Employees(
  ______________________________,
  ______________________________,
  ______________________________,
  ______________________________);
CREATE TABLE Departments(
  ______________________________,
  ______________________________,
  ______________________________,
  ______________________________);
CREATE TABLE Locations(
  ______________________________,
  ______________________________,
  ______________________________);
CREATE TABLE Works_In(
  ______________________________,
  ______________________________,
  ______________________________,
  ______________________________,
  ______________________________,
  ________________________________________________________,
  ________________________________________________________,
  ________________________________________________________);
```

Functional Dependencies Review

Function Dependency: X→Y

- Given any two tuples in *r*, if the X values are the same, then the Y values must also be the same, but not vice versa.
- An FD is a statement about *all* allowable relations and is identified based on *application semantics*. Given some instance *r1* of R, we can check if *r1* violates some FD *f*, but we cannot determine if *f* holds over R.

FDs vs. Keys

- If “K→all attributes of R”, then K is a *superkey* for R (does not require K to be *minimal*.)
- FDs are a generalization of keys.

FD Closure: F+ = closure of F = set of all FDs that are implied by F, including “trivial dependencies”

Armstrong’s Axioms

- Reflexivity: if X ⊇ Y , then X → Y
- Augmentation: if X → Y , then XZ → Y Z for any Z
- Transitivity: if X → Y and Y → Z , then X → Z

Useful rules (can be derived from AA)

- Union: if X → Y and X → Z , then X → YZ
- Decomposition: if X → Y Z , then X → Y and X → Z

Common mistake (!): if XA → Y A , you CAN’T infer X → Y

Attribute Closure X+: what attributes can X determine?

- `initialize X+ := X`
- `Repeat until no change:`
    - `if U → V in F such that U is in X+, then add V to X+`
- Example: R = ABCDE, F = { B → CD, D → E, B → A, E → C, AD → B }
    - B -> B -> BCD -> BCDA -> BCDAE
    - B+ = {ABCDE}, which means B is a key of R!

Boyce-Codd Normal Form

- Definition: Relation R with FDs F is in BCNF if, for all X → A in F+
    - A belongs to X (called a trivial FD), or
    - X is a superkey for R.
- i.e. R is in BCNF if the only non-trivial FDs over R are the key constraints.

Decomposing a schema into BCNF

- What is the motivation of decomposing a schema into BCNF?
- If X → A is a FD that violates BCNF, then decompose R into R - A and XA
- Repeat if necessary.
- Example: R = ABCEG; F = {AB->C, AC->B, BC->A, E->G}.
  Decompose R into BCNF.
    - AB->C => Decompose ABCEG: ABEG, ABC
    - E->G => Decompose ABEG: ABE, EG
    - Final relations: ABE, EG, ABC

Decomposition properties

- Lossless-join
- Dependency preserving

FD Problem 1

Consider the **Works_In(<u>S</u>sn, L**ot, **D**id, s**I**nce**)** relation from our previous example**.** If **S** (ssn) is a key for this relationship, what is the functional dependency we can infer from that? Abbreviate the attribute with the bolded/capitalized letter.

If employees in the same department are given the same parking lot, what additional functional dependency can we infer?

FD Problem 2

*Abbreviate attributes with the bolded/capitalized letter (e.g. F = Flight_no)
Flight schema

```
Flights(Flight_no, Date, fRom, To, Plane_id), ForeignKey(Plane_id)
Planes(Plane_id, tYpe)
Seat(Seat_no, Plane_id, Legroom), ForeignKey(Plane_id)
```

1. Find the set of functional dependencies.

2. Expand the FDs found above using Armstrong's axioms (you can omit the trivial and non interesting dependencies).

FD Problem 3

1. Now consider the attribute set R = ABCDE and the FD set F = {AB → C, A → D, D → E, AC → B}. Compute the attribute closure for each of A, AB, B, and D.
   - *A:*
   - *AB:*
   - *B:*
   - *D:*

2. Decompose R = ABCDEFG into BCNF, given the functional dependency set F = {AB → CD, C → EF, G → A, G → F, CE → F}.