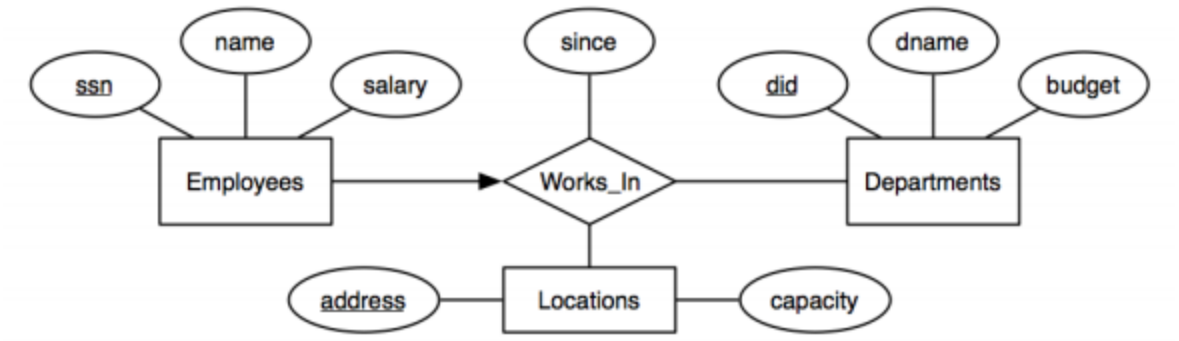
CS186 Week 13 - ER & Functional Dependencies Solutions

ER to SQL




```
CREATE TABLE Employees(
 ssn CHAR(11),
 name CHAR(100),
 salary REAL,
 PRIMARY KEY(ssn));
CREATE TABLE Departments(
 did INTEGER,
 dname CHAR(100),
 budget REAL,
 PRIMARY KEY(did));
CREATE TABLE Locations(
 address CHAR(200),
 capacity INTEGER,
 PRIMARY KEY(address));
CREATE TABLE Works_In(
 ssn CHAR(11),
 did INTEGER,
 address CHAR(200),
 since DATE,
 PRIMARY KEY(ssn),
 FOREIGN KEY(ssn) REFERENCES Employees,
 FOREIGN KEY(did) REFERENCES Departments,
 FOREIGN KEY(address) REFERENCES Locations);
```

FD Problem 1

1. Consider the **Works_In(Ssn, Lot, Did, sInce)** 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.

S → SLDI

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

D → L

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.

From the key constraints we get the following functional dependencies:
FD → FDRTP
P → PY
SP → SPL

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

Using decomposition and transitivity (FD → FDRTP , P → PY) we can obtain:
FD → Y
Using decomposition, augmentation and transitivity (FD → FDRTP , SP → SPL) we can obtain:
FDS → L

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: {A, D, E}*
- *AB: {A, B, C, D, E}*
- *B: {B}*
- *D: {D, E}*

2. Decompose R = ABCDEFG into BCNF, given the functional dependency set F = {AB → CD,

C → EF, G → A, G → F, CE → F}.

AB→CD => decompose ABCDEFG into ABCD, ABEFG
G→A => decompose ABEFG into AG, BEFG
G→F => decompose BEFG into FG, BEG
Final relations: **ABCD, AG, FG, BEG**.