Relational Database Design

Ch 14: 14.1, 14.2 New

Ch 15: 15.1, 15.2 Old

Company Database

- why this design?
- many designs possible
- DB design theory.

Goodness test

1) logical
   * ease of understanding
   * easy to formulate queries

2) physical
   * storage requirement should be as minimal as possible.
   * easy to retrieve & easy to update

Emp - Dept

Ename, SSN, Bdate, Address, Dnumber, Dname, Dmgr-SSN

Problems (Informal design consideration)

* Insert : new dept with no employees.
* Delete : delete the last emp of a dept
  ⇒ you lost the department.
* Update : Redundancy : mistake in data entry
  - Reserch not Research
  - different mgrs for same dept.

- DO NOT combine 2 independent entities into one entity.

- minimize redundancy & NULL values in tables: do not combine EMP & Dependent

- design should not generate false data.

↓

EMP - PROJ :

R: SSN, Pnumber, Hours, Ename, Pname, Plocations

Too much redundancy in ↑

R1: EMP_LOCS : Ename Plocation

R2: Emp-Proj-LOCS : SSN, Pnumber, Hours, Pname, Plocation

$R1 \bowtie R2 \neq R$

Formal Design Considerations

1) Start with a relation that has all the attributes.

emp_proj:

(SSN, Pno, Hours, Ename, Pname, Plocation)

2) Specify the functional dependencies (fds) and the multivalued dependencies (mvds)
   - based on real-world relationships.
   - input by database designer
   - property of data.

SSN ⟶ EName

* SSN functionally determines Ename
* Ename is functionally dependent on SSN.

Pno ⟶ Pname

Pno ⟶ Plocation

SSN, Pno ⟶ Hours

SSN	Ename
1234	John
5678	Mary
9876	John

Is this valid table?

If SSN = 1234 then Ename = John.

⟺ If Ename ≠ John then SSN ≠ 1234.

SSN → Ename

If ename is John then SSN = 1234. ✗

if A then B	$A \Rightarrow B$	}	equivalent statements.
not B then not A	$\neg B \Rightarrow \neg A$		

Example: if today is Sunday then school is closed.
if school is open (not closed) $\Rightarrow$ today is not Sunday.

if school is closed $\Rightarrow$ today is Sunday.

3) decompose the table (Emp-Proj) with all the attributes into relations (i.e., tables) that are in normal form. (NF)

* a set of tables in NF does not suffer from insert, delete, update, join anomalies.

Original table : R
Divide into R1, R2 where R1, R2 in NF.
Then R = R1 ⋈ R2