**Normalization and Functional Dependencies**

**Anomalies (Update, Delete, Insert)**

An Insert Anomaly occurs when certain attributes cannot be inserted into the database without the presence of other attributes.

An Update Anomaly exists when one or more instances of duplicated data is updated, but not all.

A Delete Anomaly exists when certain attributes are lost because of the deletion of other attributes.

**id name address dept**

101 Rick CA D001

101 Rick CA D002

How do you fix it? 1st, 2nd, 3rd, BCNF normal form

• First Normal Form (1NF): Atomic and no repeating groups.

• Second Normal Form (2NF): 1NF and no partial dependencies

• Third Normal Form (3NF) 2NF and no transitive dependencies.

• Boyce-Codd Normal Form (BCNF) 1NF and all dependencies from full key

What are some of the concepts that we are going to discuss?

Functional dependency, F closure, X closure, Super key, Candidate key,

Armstrong axioms and inference rules, Bernstein synthesis, minimal cover,

Prime and non-prime attributes

**F closure {F}+**: The closure of F, denoted as F+, is the set of all regular FDs that can be derived from F

**X closure {X}+ :** The set of all those attributes which can be functionally determined from an attribute set is called as a closure of that attribute set.

**Superkey**: A superkey is a combination of columns that uniquely identifies any row within a RDBMS table.

**Candidate key**: A candidate key is a closely related concept where the superkey is reduced to the minimum number of columns required to uniquely identify each row.

**Minimal cover**:  Functional dependencies F covers another set of functional dependencies G, if every functional dependency in G can be inferred from F. F is a minimal cover of G if F is the smallest set of functional dependencies that cover G.

1. Right Hand Side (RHS) of all FDs should be single attribute.

2. Remove extraneous attributes.

3. Eliminate redundant functional dependencies.

**Prime/non-prime attributes**: Attributes that form a candidate key of a relation are called prime attributes. The rest of the attributes of the relation are non-prime.

**Armstrong’s axioms and inference rules:** Armstrong's axioms are a set of axioms used to infer all the functional dependencies on a relational database.  They are both sound and complete.

**Functional Dependency**

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table. The left side of FD is known as a determinant, the right side of the production is known as a dependent.

**SSN LNAME FNAME**

111 Jones Jack

111 Jones Jack

112 Jones jim

113 Jeff jim

**Name subject**

Joe prog

Joe prog

Joe econ

Jill econ

**Teacher Course Text**

Smith Data Structures Bartram

Smith Data Management Martin

Hall Data Structures Horowitz

Sly C++ Horowitz

**Dname Dnumber Dmgr\_ssn Dlocation**

Research 5 334 Bell

Research 5 334 Sugar

Research 5 334 Bell

Admin 4 335 Stafford

Headquarter 1 111 Sugar

**A B C**

1 2 3

1 2 3

2 3 4

2 4 4

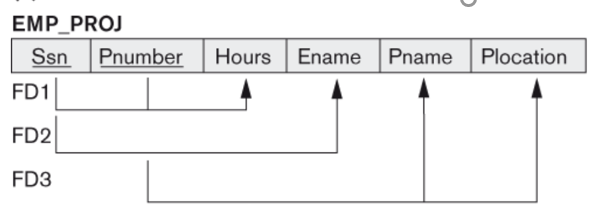
3 4 7

x->y means that y is functionally dependent on x or that x uniquely determines y.



ABC-> d, this means d is dependent fully on ABC and not a subset of the determinants

So AB->d would be wrong.



SSN, Pnumber-> Hours

SSN -> Ename

Pnumber -> Pname, Plocation

**Keys (Super key and candidate key)**

Emp\_SSN Emp\_Number Emp\_Name

--------- ---------- --------

1 226 jack

9 227 jill

3 228 jim

4 229 jim

Super key

Empssn, emp\_number, Emp\_name

Empssn, emp\_number

Emp\_ssn, emp\_name

Emp\_number, emp\_name

Emp\_ssn,

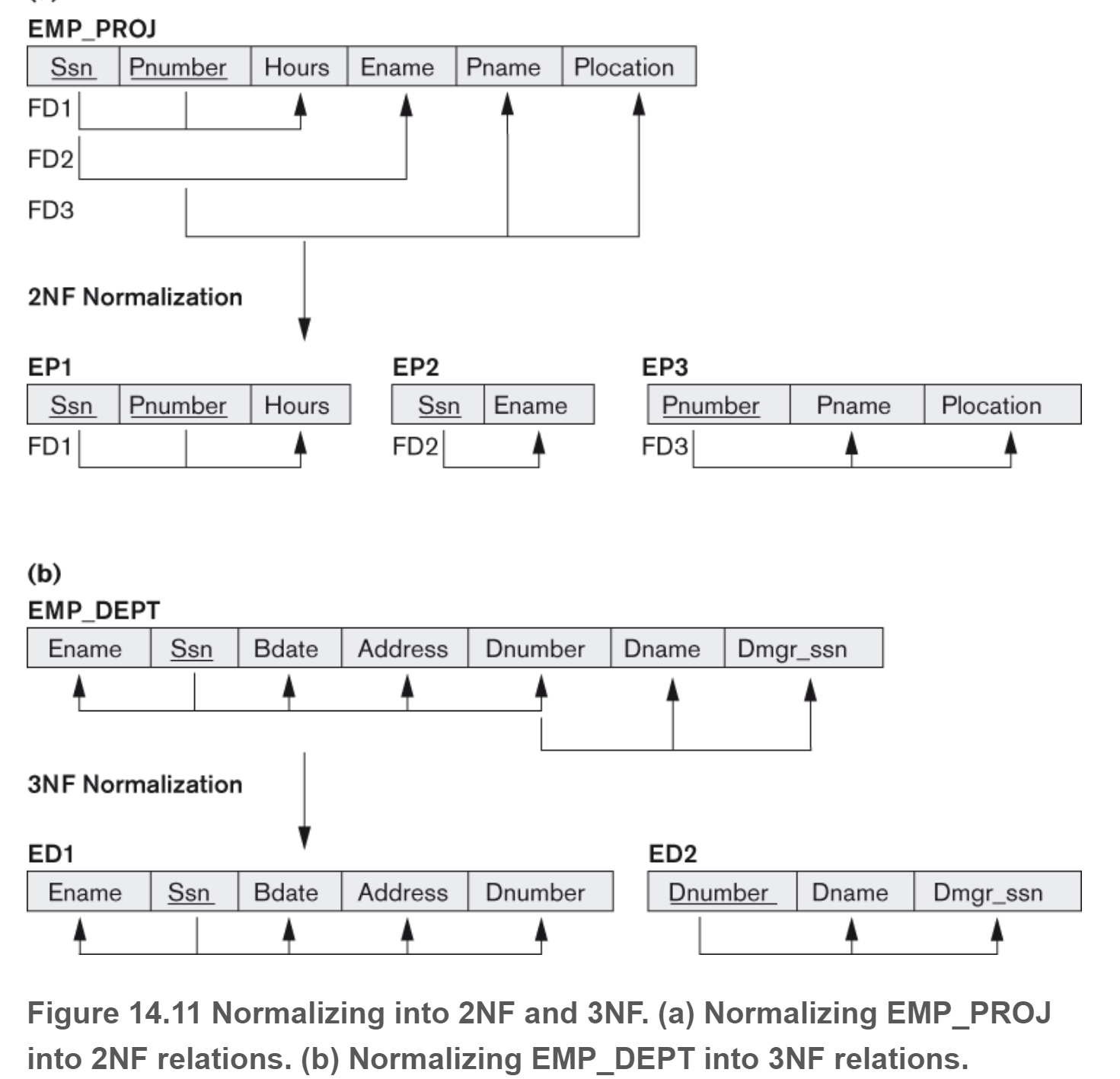
Emp\_number

Candidate key is because you can thin it out and you still have uniqueness

Emp\_snn

Emp\_number

**Decomposition**



**Applying Armstrong’s axioms and inference rules**



• Reflexivity: If Y ⊆ X, then X → Y

• Augmentation: If X → Y , then XZ → Y Z

• Transitivity: If X → Y and Y → Z, then X → Z

• Union: If X → Y and X → Z, then X → Y Z

• Decomposition: If X → Y Z, then X → Y and X → Z

• Pseudotransitivity: If X → Y and W Y → Z, then W X → Z

• Composition: If X → Y and Z → W, then XZ → Y W

**X closure**

The set of all those attributes which can be functionally determined from an attribute set is called as a closure of that attribute set.

1. A → B

2. B → C

3. BC → D

Then we can compute the closure of A with respect to M in the following way:

i A → A ( by reflexivity rule )

ii A → AB ( by (i) and 1 )

iii A → ABC ( by (ii), 2, and transitivity rule )

iv A → ABCD ( by (iii), and 3 ) Therefore, A + = ABCD.

**Cover**

Example 1

We have the following functional dependencies (F):

• AB → C

• A → B

we want to know if we can simplify to the following (H):

• A → C

• A → B

we can replace F by H.

Example 2

We have the following functional dependencies (F):

• A → BC

• B → C

we want to know if we can simplify to the following (H):

• A → B

• B → C

we can replace F by H.

Remove all trivial functional dependencies

Union Simplification

RHS Simplification

LHS Simplification

1. E → G

2. G → S

3. E → S

Using the union rule, we combine 1 and 3 and get

1. E → GS

2. G → S

Simplifying RHS, we get: 1. E → G 2. G → S

**Find the X closure**

R(ABCDE} F={A->D, D->B, B->C, E->B}



R(ABCDE) F={A->BC, CD->E, B->D, E->A}

**Find all the keys**

1. R(A,B,C,D) F={ A->B, C->D}
2. R(A,B,C,D) F={ AB->C, B->A }
3. R(A, B, C, D) F={ A->B B->C C->D CD->B BC->A}
4. R(A,B,C,D,E,F} F={ DF->C, BC->F, E->A , ABC->E}

**What is the minimal cover**

1. Singleton RHS
2. No extraneous attributes on the left hand side (Do Closure on one and see if the other ones shows up)
3. No Redundancy (Do X closure and see if the RHS comes up)
4. ABC->D

AB->C

BC->A

B->A

A->C

1. a->bc,

b->c

a->b

Ab->c

1. a->bc

b->ac

c->ab

1. A → BC

B → C

A → B

AB → C



1. A→B,

ABCD→E

EF→GH

ACDF→EG

**Normal forms**

1st normal form

1. The information in each field must be atomic (single values, not multiple values)
2. Columns should hold dissimilar information (this means no repeating groups, no multivalues)
3. There has to be a primary key

Example:

John James, java, C should be decomposed

111, John, James, java

111, John James, C

2nd normal form

* + Find candidate keys, Identify prime and non-prime attributes
  + Partial dependency is not allowed,
    - part of candidate key in the FDs is determining non-prime attribute

Example:

abc->d abc is a candidate key, abc is prime, d is non-prime

a->d is a partial dependency because a is a partial key.

d is fully functional on abc. We say fully because we have a composite key as opposed to just functional.



3rd normal form

* + Find candidate keys, Identify prime and non-prime attributes
  + x->a either x should be a superkey or a should be a prime attribute for every functional dependency

Example:

**No Transitive dependency**

A->B and B->C then A->C (This would not be a normal form because B is not a superkey or C is not a prime

Stid name deptID deptName

11 John 1 comp

12 john 1 comp

13 jim 2 bus

Stid, name, deptID, deptName

Stdid->name

Stdid->deptID

Deptid->deptname

Stdid->departmentname

When you have a column that is dependent on another non key item and the nonkey item is dependent on the key. In other words, there cannot be any kind of interdependency among nonkey attributes.

BCNF normal form

* + Find candidate keys, Identify prime and non-prime attributes
  + x->a x is superkey

What normal form?

1. R(A,B,C) A->B, B->C
2. R(A,B,C,D) AB->CD, C->B
3. R(A,B,C,D) AB->C, B->C, C->B
4. R(A,B,C,D) A->B, B->C, C->D, D->A
5. R(A,B,C,D) AB->C, C->D
6. R(A,B,C) AB->C, C->A
7. EmpDept(ssn, depid, dname) ssn->deptid ssn->dname
8. Empproj( ssn, p#, fname, pname) ssn, p#->fname, pname ssn->fname