Lecture 11

Functional Dependencies and Normalization for Relational Databases

Normal Forms Based on Primary Keys

- Normalization of Relations
- Practical Use of Normal Forms
- Definitions of Keys and Attributes Participating in Keys
- First Normal Form
- Second Normal Form
- Third Normal Form

Normalization of Relations

- **Normalization:**
  - The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

Can be defined as : Removing rendundant data and preventing : Insert, Update, and delete anomalies

- **Normal form:**

When all conditions/rules are met for a specefic number of Form, then the relation schema is set to be in this normal form

  - Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

Normalization is great only for OLTP (Online transactional processing) where all read, write , update, delete transactions are going to be used.
Normalization may not be good for OLAP (Online transactional analytical processing) where all operations are 'read' this way it better for performance that normalization is not applied to avoid joining many tables.

Normalization of Relations

- 2NF, 3NF, BCNF
  - based on keys and FDs of a relation schema
- 4NF and 5NF
  - 4NF based on keys, multi-valued dependencies: MVDs;
  - 5NF based on keys, join dependencies: JDs
- Additional properties may be needed to ensure a good relational design

Practical Use of Normal Forms

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are *hard to understand* or to *detect*
- The database designers *need not* normalize to the highest possible normal form
  - (usually up to 3NF, BCNF or 4NF)

Definitions of Keys and Attributes Participating in Keys

- A **superkey** of a relation schema R = {A1, A2, ..., An} is a set of attributes S *subset-of* R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S]
- A **key** K is a **superkey** with the *additional property* that removal of any attribute from K will cause K not to be a superkey any more. Minimal Super Key

Definitions of Keys and Attributes Participating in Keys

- If a relation schema has more than one key, each is called a **candidate** key.
  - One of the candidate keys is *arbitrarily* designated to be the **primary key**, and the others are called **secondary keys**.
- A **Prime attribute** must be a member of *some* candidate key
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

Reminder: 'Domain' is all possible values an attribute can be

First Normal Form Atomicity

Atomic Domains : Each attribute (column) should contain atomic (non decomposable) values

No ordering to rows or columns

No exact duplicate rows

Attribute should contain value from same domain (eg. attribute values has to be all int or all CHAR)

- Disallows
  - composite attributes
  - multivalued attributes
  - **nested relations**; attributes whose values for an *individual tuple* are non-atomic
- Considered to be part of the definition of relation

A correctly Mapped ERD to Relational schema will by default be in 1st normal form

Normalization into 1NF

(a)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations

(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Figure 10.8
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

Dnumber	Dname	Dmgr_ssn
5	Research	-----
4		
1		

Dnumber	Dlocation

Normalization of nested relations into 1NF

(a)

EMP_PROJ

		Projs	
Ssn	Ename	Pnumber	Hours

(b)

EMP_PROJ

Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya, AliciaJ.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321	Wallace, Jennifer S.	30	20.0
		20	15.0
888665555	Borg, James E.	20	NULL

(c)

EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

Figure 10.9
Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

Second Normal Form

Relation is in second normal form if No partial dependencies exist

- Uses the concepts of **FDs, primary key**
- Definitions
  - **Prime attribute:** An attribute that is member of the primary key K
  - **Full functional dependency:** a FD Y → Z where removal of any attribute from Y means the FD does not hold any more

Partial Dependency: Proper Subset (subset not = to original set) of Key K can determine non-prime attributes (e.g: Key : {SSN,NAME}, but only SSN -> Phone (non-prime attribute)

- Examples:
  - {SSN, PNUMBER} → HOURS is a full FD since neither SSN → HOURS nor PNUMBER → HOURS hold
  - {SSN, PNUMBER} → ENAME is not a full FD (it is called a partial dependency) since SSN → ENAME also holds

Second Normal Form

- A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on the primary key
- R can be decomposed into 2NF relations via the process of 2NF normalization

Normalizing into 2NF and 3NF

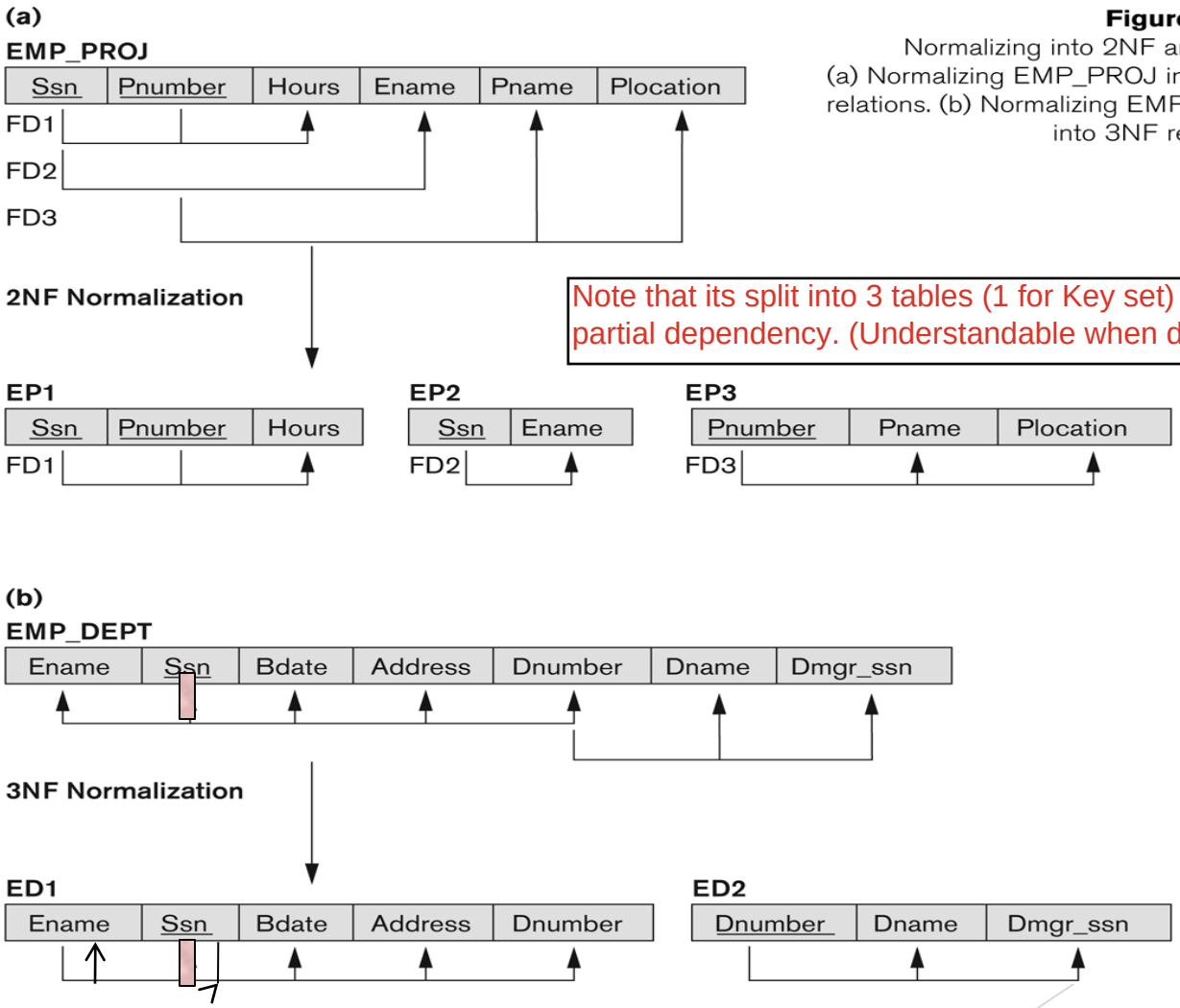


Figure 10.10
Normalizing into 2NF and 3NF. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

Note that its split into 3 tables (1 for Key set) and 2 for each partial dependency. (Understandable when doing it yourself)

Third Normal Form

- Definition:
  - **Transitive functional dependency:** a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$
- Examples:
  - SSN $\rightarrow$ DMGRSSN is a **transitive** FD

  Since

  SSN $\rightarrow$ DNUMBER and DNUMBER $\rightarrow$ DMGRSSN hold
  - SSN $\rightarrow$ ENAME is **non-transitive**

    Since there is no set of attributes X where

    SSN $\rightarrow$ X and X $\rightarrow$ ENAME

Third Normal Form

No transitive dependency in relation. reason: This can cause Insert, Update, delete anomalies still because some data will be redundant because some attributes (non-prime) may determine another attributes.

- A relation schema R is in **third normal form (3NF)** if it is in 2NF *and* no non-prime attribute A in R is transitively dependent on the primary key
- R can be decomposed into 3NF relations via the process of 3NF normalization
- NOTE:
  - In X → Y and Y → Z, with X as the primary key, we consider this a problem only if Y is not a candidate key.
  - When Y is a candidate key, there is no problem with the transitive dependency .
  - E.g., Consider EMP (SSN, Emp#, Salary).

    Here, SSN → Emp# → Salary and Emp# is a candidate key.

Normal Forms Defined Informally

- **1st normal form**

  All attributes depend on **the key**

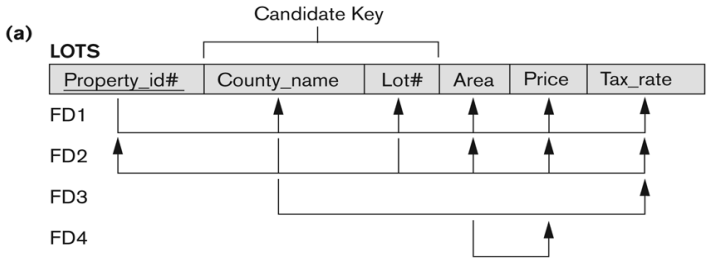
- **2nd normal form**

  All attributes depend on **the whole key**

- **3rd normal form**

  All attributes depend on **nothing but the key**

Successive Normalization of LOTS into 2NF and 3NF



SUMMARY OF NORMAL FORMS based on Primary Keys

Table 10.1
Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

Example

Consider the universal relation R
{A,B,C,D,E,F,G,H,I,J} and the set of FDs

FD = { {A,B} → C, A→ {D,E} , B → F,

F → {G,H}, D → {I,J}}

What is the key for R? Decompose R into 2NF, then 3NF relations.

Solution

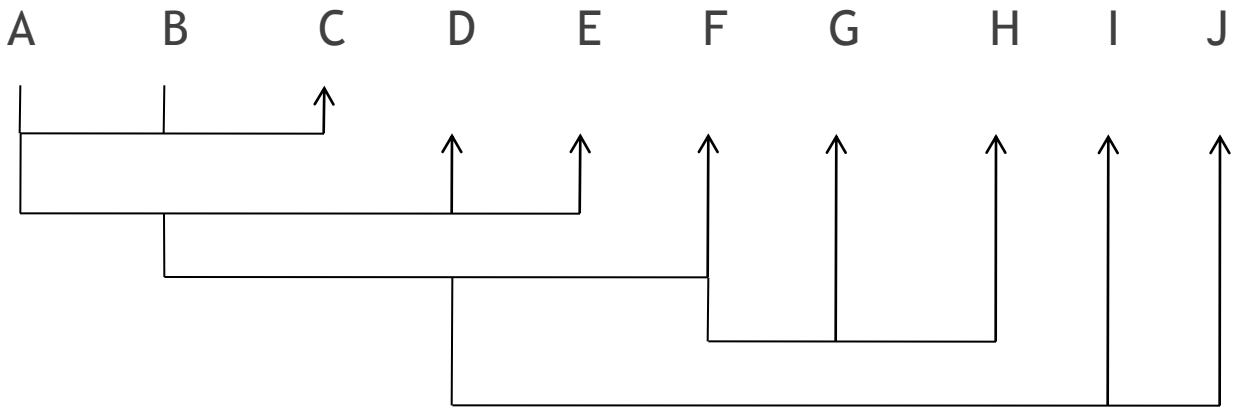
- The key of R :

{A,B}+ = {A,B,C} , {A}+ = {A,D,E} , {B}+ = {B,F},

{F}+ = {F,G,H}, {D}+ = {D,I,J}

{A,B}+ = {A,D,E,B,F,C,G,H,I,J}

So AB is the key



$\{A,B\} \rightarrow C,$

$A \rightarrow \{D,E\}$, $D \rightarrow \{I,J\}$

$B \rightarrow F$, $F \rightarrow \{G,H\}$

- 2NF:

  {A,B} → C Table1(<u>A,B</u>,C)

  A→ {D,E} and D → {I,J} so Table2(<u>A</u>,D,E,I,J)

  B → F, F → {G,H} so Table3(<u>B</u>,F,G,H)

- 3NF:

Table1 is in 3NF

Table2 decompose into :

  Table21(<u>A</u>,D,E) , Table22(<u>D</u>,I,J)

Table3 decompose into

  Table31(<u>B</u>,F) , Table32(<u>F</u>,G,H)