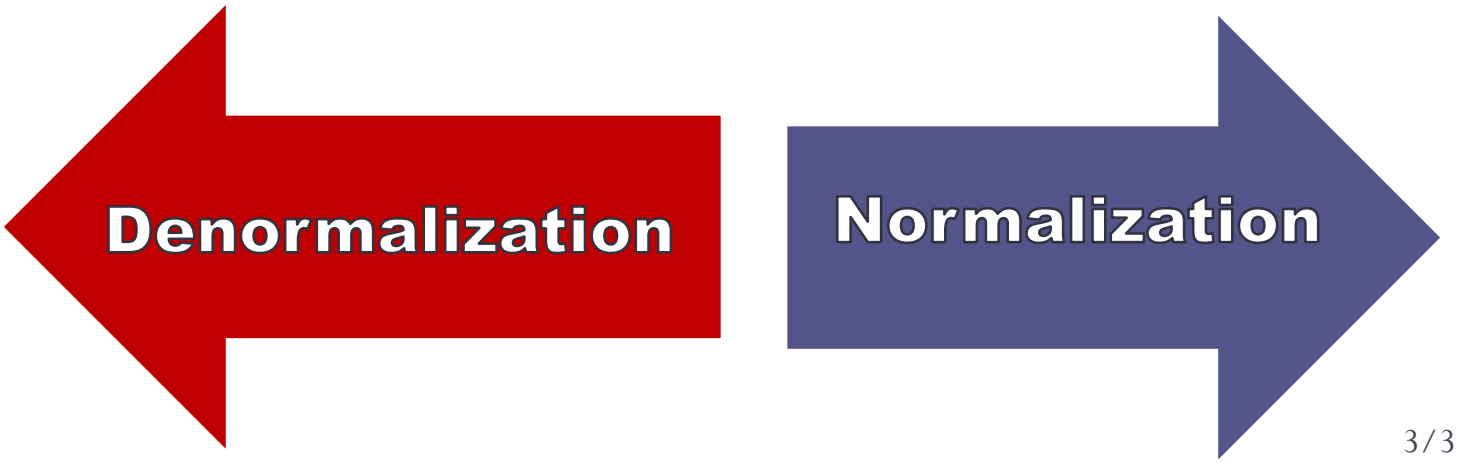
University of Bisha
Faculty Of Computing and Information Technology

Part (6)

Data Normalization & Denormalization



Outlines of Part 6 → Use Chapter 14

- Database Problems.
- Factional Dependency (FD).
- Normalization.
- Denormalization.

Database Problems
Data Redundancy and Update Anomalies

There are several problems that prevent the proper design of a relational database such as:

- Information is stored redundantly
  - Wastes storage

- Causes problems with update anomalies
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

Database Problems
Data Redundancy

Problems associated with data redundancy are illustrated by comparing the Staff and Branch relations with the StaffBranch relation.

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

- StaffBranch relation has redundant data; the details of a branch are repeated for every member of staff.

Database Problems - Update Anomalies …Cont.

Example Of An *Update* Anomaly

Consider the relation:

EMP_PROJ(Ssn, Pnumber, hours,Ename, Pname,Plocation)

Update Anomaly:

Changing the name of project number Pnumber (1) from "Productx" to "Customer-Accounting" may cause this update to be made for all 100 employees working on project no.(1).

EMP_PROJ

			Redundancy	Redundancy	
Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

Database Problems - Update Anomalies …Cont.

Example Of An *INSERT* Anomaly

Consider the relation:

EMP_PROJ(Ssn, Pnumber, hours,Ename, Pname,Plocation)

Insert Anomaly:

Cannot insert a project unless an employee is assigned to it.

Conversely

Cannot insert an employee unless an he/she is assigned to a project.

Redundancy — Redundancy

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

Database Problems - Update Anomalies …Cont.

Example Of *DELETE* Anomaly

Consider the relation:

EMP_PROJ(Ssn, Pnumber, hours,Ename, Pname,Plocation)

Delete Anomaly:

- When a project is deleted, it will result in deleting all the employees who work on that project.
- Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

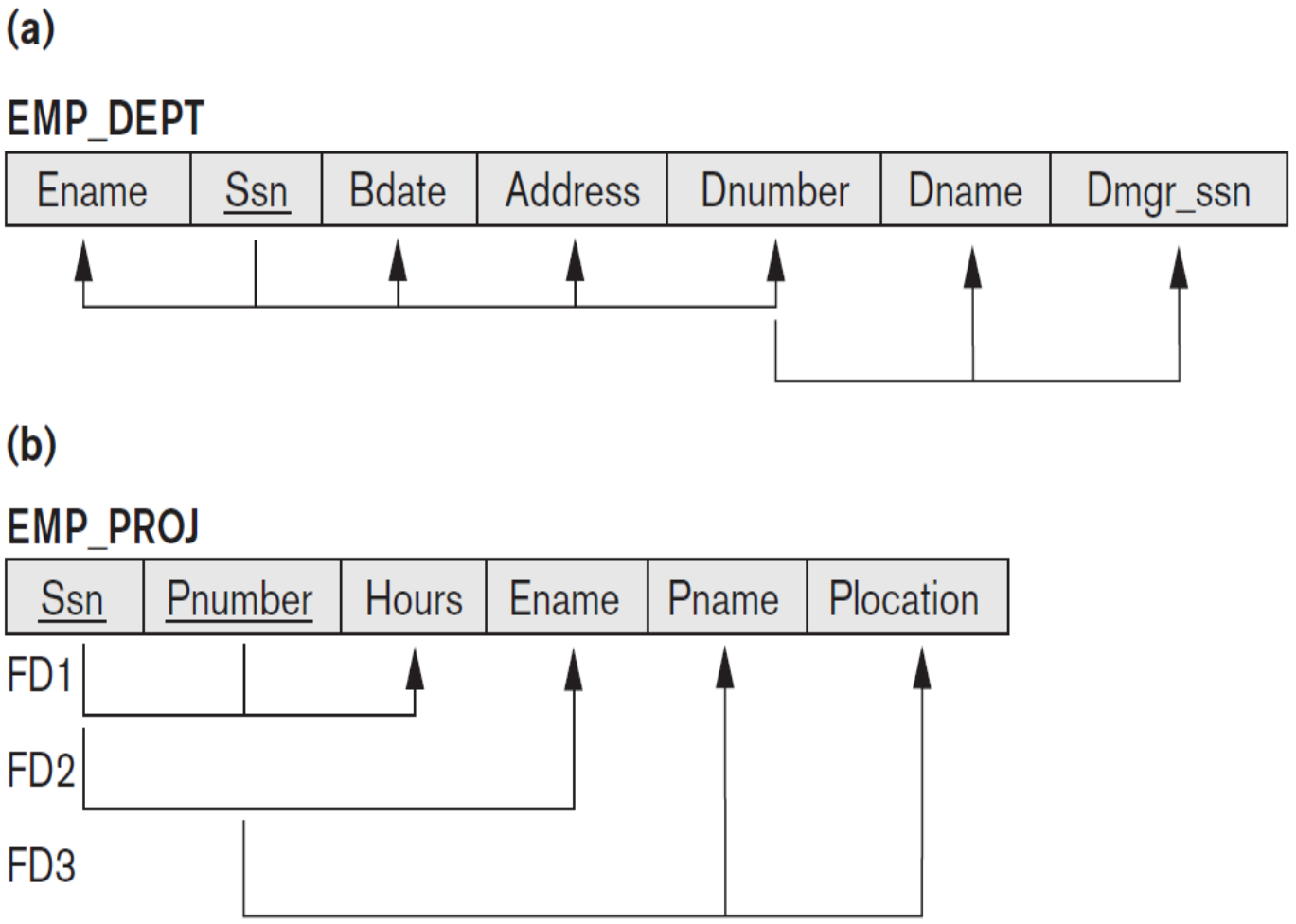
Redundancy Redundancy

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

Figure 14.3 Two relation schemas suffering from update anomalies

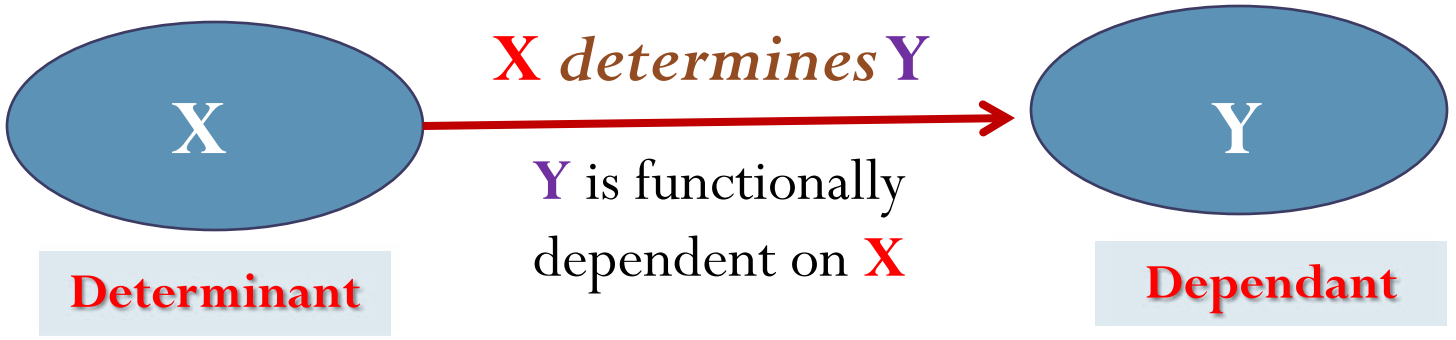
Figure 14.3
Two relation schemas suffering from update anomalies.
(a) EMP_DEPT and
(b) EMP_PROJ.



Factional Dependency (FD)

❖Functional dependency describes relationship between attributes within a relation.

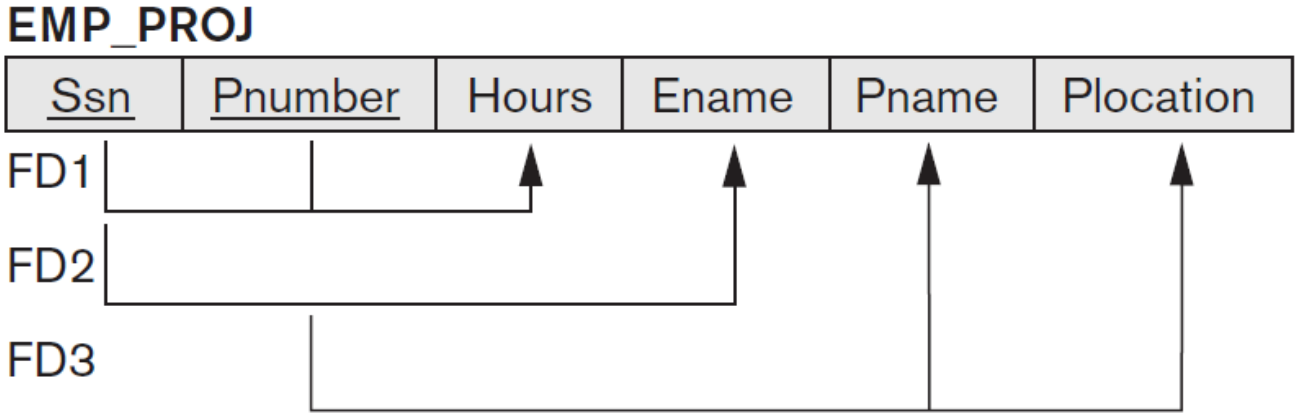
For example, if **X** and **Y** are attributes of relation R, A set of attributes **X** functionally ***determines*** a set of attributes **Y** if the value of **X** determines a unique value for **Y.** (**Y** is functionally dependent on **X**).



Factional Dependency (FD)……Cont.

- Written as X ⟶ Y; can be displayed graphically on a relation schema as in Figures. (denoted by the arrow:).

  FDs are derived from the real-world constraints on the attributes .



❖Functional dependencies (FDs)

- Are used to specify *formal measures* of the "goodness" of relational designs
- And keys are used to define **normal forms** for relations
- Are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes.

Factional Dependency (FD)..Cont.
Examples of FD constraints

- Employee ssn and project number determines the hours per week that the employee works on the project

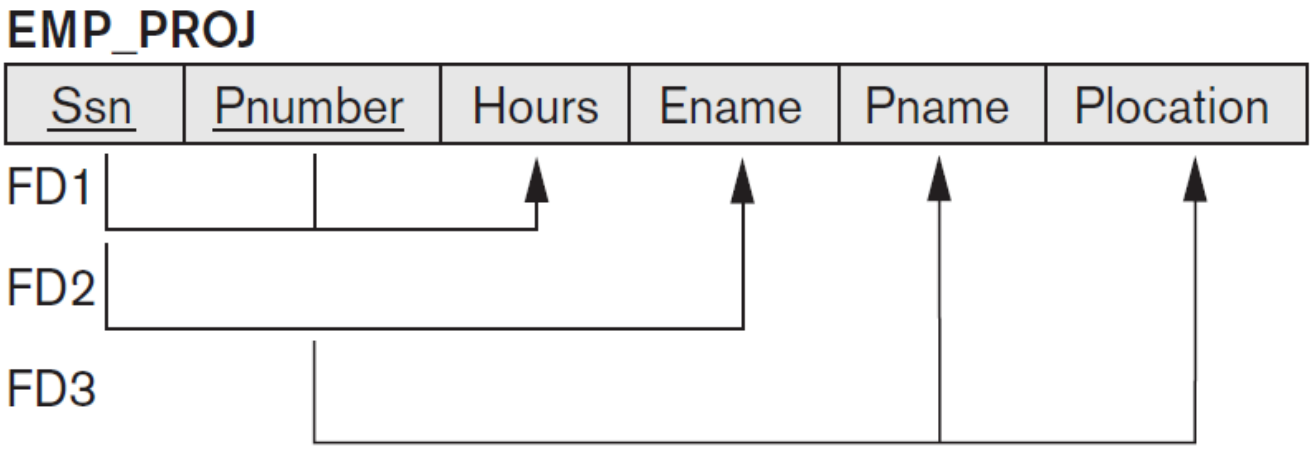
  {SSN, PNUMBER} -> HOURS… **(FD1)**

- Social security number determines employee name

  SSN -> ENAME…… **(FD2)**

- Project number determines project name and location

  PNUMBER -> {PNAME, PLOCATION}…. **(FD3)**



Factional Dependency (FD)..Cont.
Factional Dependency Types

We will discuss two types of Factional Dependencies:

- ***Partial Dependency.***
- ***Transitive Dependency***

❑***Partial Dependency*** occurs when a non-prime attribute is functionally dependent on part of a candidate key (Composite key).

❑***Transitive Dependency***

When an indirect relationship causes functional dependency it is called Transitive Dependency.

If A → B and B → C R is true, then A → C is a transitive dependency.

Factional Dependency (FD)..Cont.
Partial Dependency Example

Consider Staff relation:

staffNo, sName → branchNo

True - each value of (staffNo, sName) is associated with a single value of branchNo.

However, branchNo is also functionally dependent on a subset of (staffNo, sName), namely staffNo. Example above is a *partial dependency*

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Factional Dependency (FD)..Cont.
Transitive Dependencies Example

Consider functional dependencies in the StaffBranch relation

staffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

Transitive dependency, branchNo → bAddress exists on staffNo via branchNo.

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Normalization

Normalization:

The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

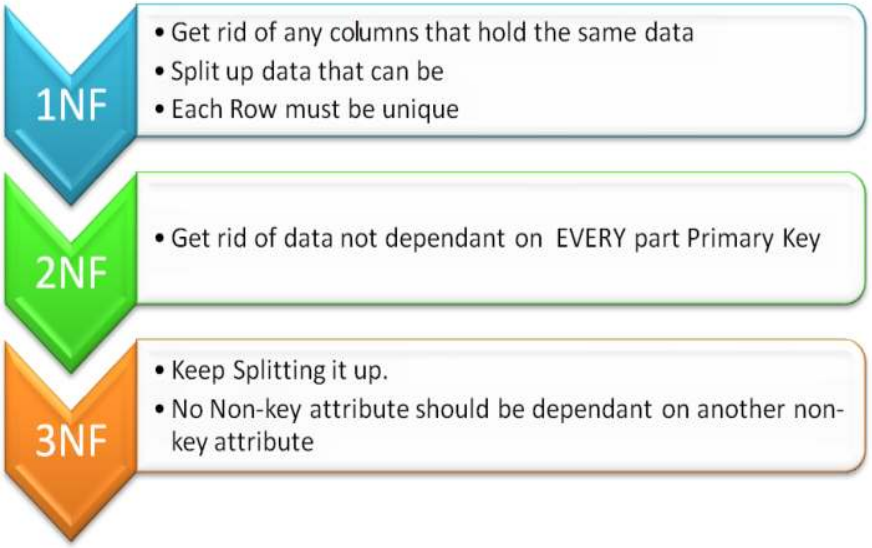
Normal form:

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

Normalization

The database normalization process is divided into following the normal form:

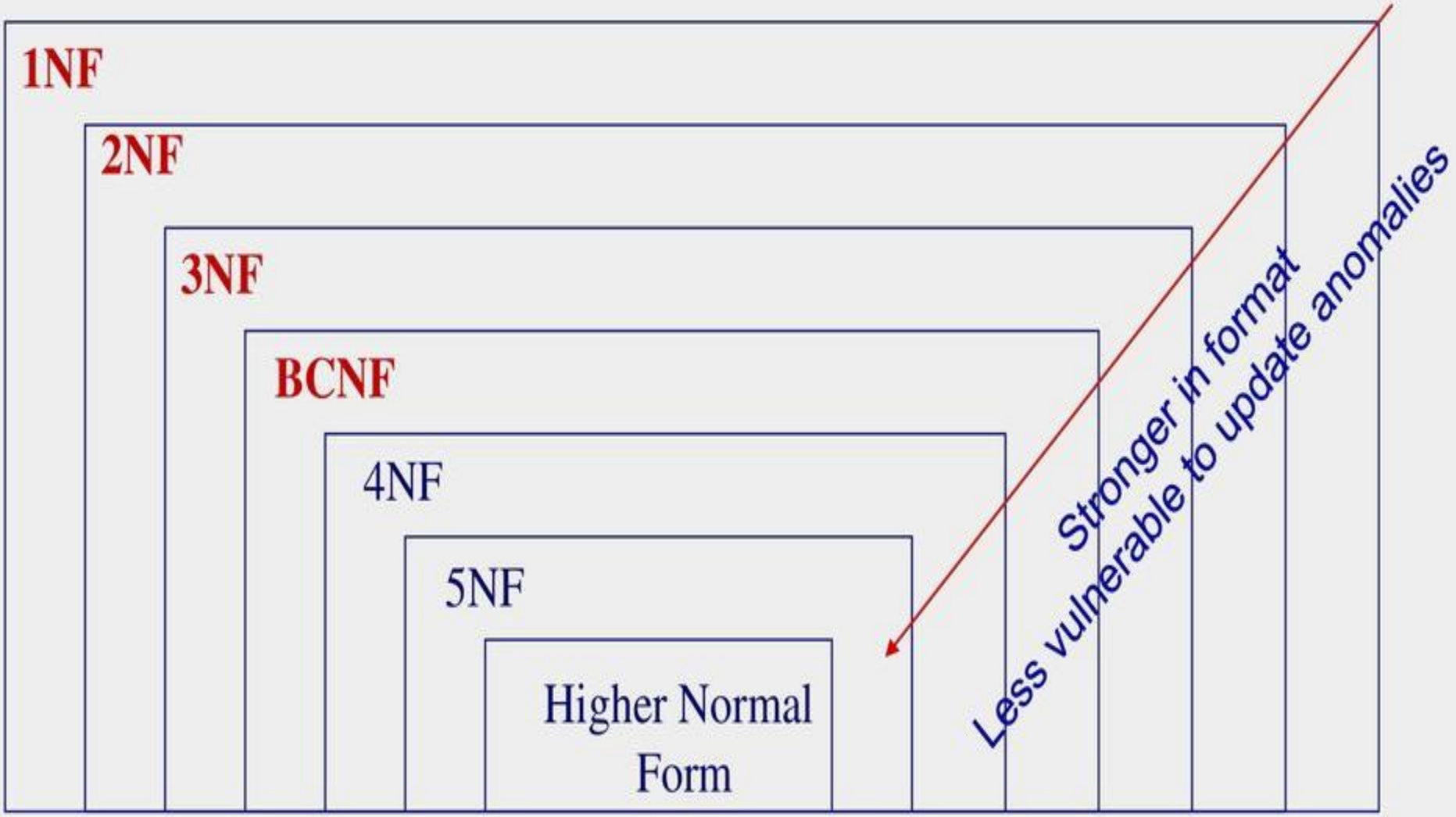
- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- BCNF
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)



The database designers *need not* normalize to the highest possible normal form
(usually up to 3NF, BCNF, 4NFor 5NF)

Normalization…Cont.

The Process of Normalization



Normalization…….Cont.

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.

Normalization…….Cont.
First Normal Form

Disallows

- **Composite** attributes - **INF**(must be atomic (or indivisible) values.
- **Multivalued** attributes (**Nested relations) - INF** must be single values.

Considered to be part of the definition of relation

Normalization……Cont.

First Normal Form - Composite attributes Example

Consider the Department relation shown in Figure 14.9(b).

We assume that each department can have *a number of* locations.

As we can see, this is not in 1NF because Dlocations is not an atomic attribute.

(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 14.9
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Sample state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

Normalization……Cont.

First Normal Form - Composite attributes Example

To achieve first normal form for such a relation:

- **Remove** the attribute Dlocations that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key Dnumber of DEPARTMENT.

- The primary key of this newly formed relation is the combination {Dnumber, Dlocation}, as shown in Figure 14.2.

- A distinct tuple in DEPT_LOCATIONS exists for *each location of a department. This decomposes* the non-1NF relation into two 1NF relations.

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

First Normal Form – ….Cont.

Multivalued attributes (Nested relations) Example

- Multivalued attributes that are called ***nested*** relations because each tuple can have a relation within it.
- Figure 14.10 shows how the EMP_PROJ relation could appear if nesting is allowed.

(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, Alicia J.	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

Figure 14.10
Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Sample extension of the

First Normal Form – ….Cont.

Multivalued attributes (Nested relations) Example

To normalize this into 1NF:

- Remove the nested relation attributes into a new relation and *propagate the primary key into it*.

- *the primary key of the new relation will combine the* partial key with the primary key of the original relation.

- Decomposition and primary key propagation yield the schemas EMP_PROJ1 and EMP_PROJ2, as shown in Figure 14.10(c).

(a)

EMP_PROJ

Ssn	Ename	Projs	
		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, Alicia J.	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 14.10
Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Sample 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.

First Normal Form – ….Cont.

Example

Example : normalize the following relation to 1NF

roll_no	name	subject
101	Akon	OS, CN
103	Ckon	Java
102	Bkon	C, C++

Solution

roll_no	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	Java
102	Bkon	C
102	Bkon	C++

Normalization……Cont.
Second Normal Form

A relation is in ***2NF*** if:

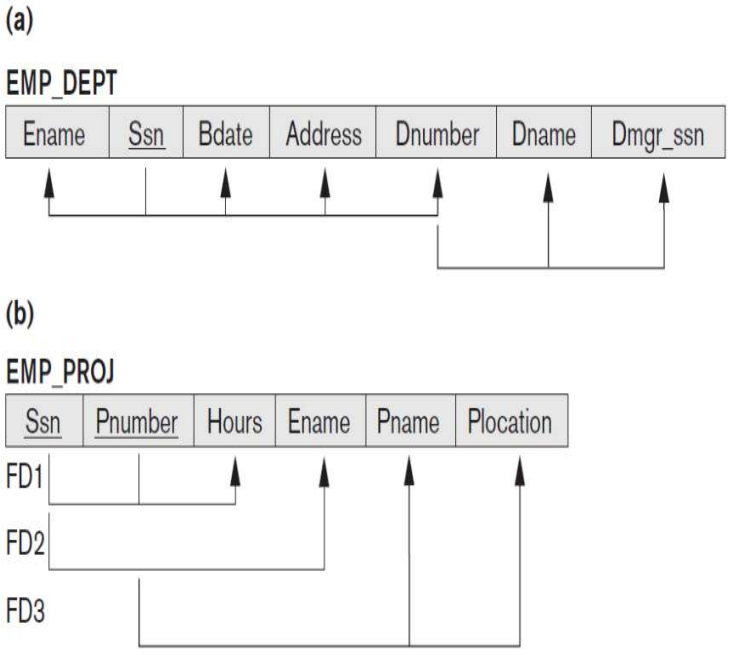
- It is in 1NF.
- Every non-key attribute is fully dependent on each candidate key. (That is, we don't have any partial functional dependency.).

Examples:

- {SSN, PNUMBER} -> HOURS is a full FD.

- {SSN, PNUMBER} -> ENAME is not a full FD (it is called a partial dependency) .see Figure 14.3 (b).

Figure 14.3
Two relation schemas suffering from update anomalies.
(a) EMP_DEPT and
(b) EMP_PROJ.



Normalization……Cont.
Second Normal Form….Cont

Process for 2NF:

- Remove the partially dependent attribute(s) from the relation.
- Create new table(s) with removed dependency.

Examples:

The functional dependencies FD1, FD2, and FD3 in Figure 14.3(b) lead to the decomposition of EMP_PROJ into the three relation schemas EP1, EP2, and EP3 shown in Figure 14.11(a), each of which is in 2NF.

Figure 14.11 (a) Normalizing into 2NF

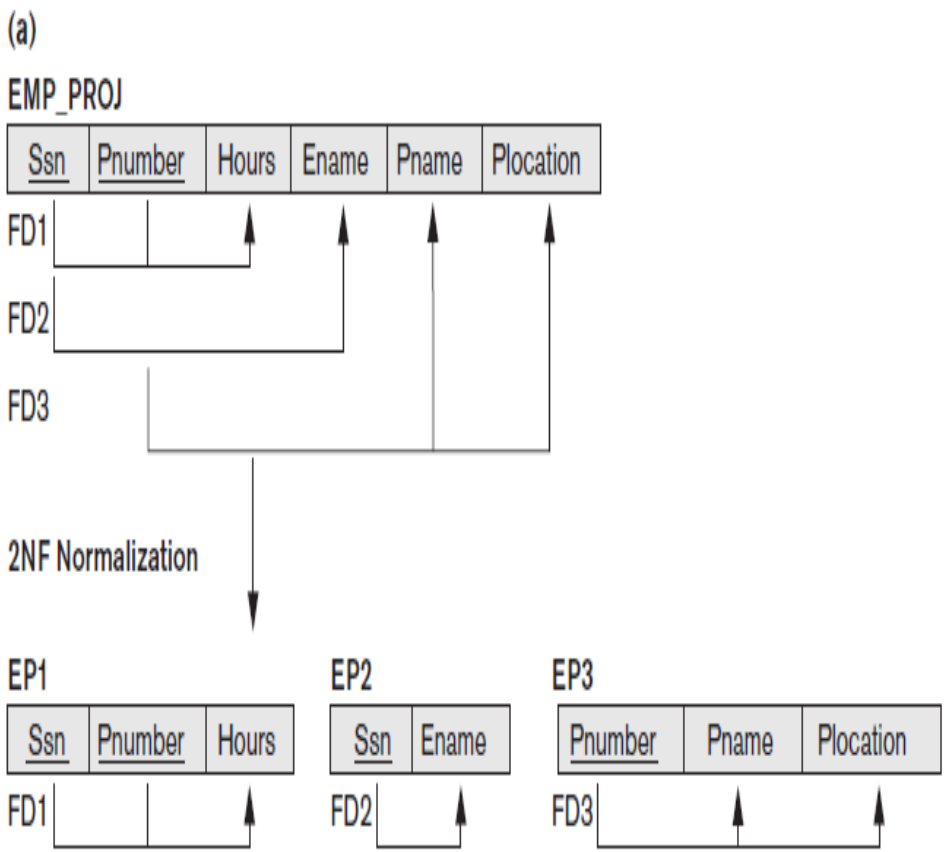


Figure 14.11
(a) Normalizing EMP_PROJ into 2NF relations.

Second Normal Form – ….Cont.

Example

Example : normalize the following relation to 2NF

Order	Customer	Contact Person	Total
1	Rishabh	Manish	134.23
2	Preeti	Rohan	521.24
3	Rishabh	Manish	1042.42
4	Rishabh	Manish	928.53

Solution

Customer	Contact Person
Rishabh	Manish
Preeti	Rohan

Order	Customer	Total
1	Rishabh	134.23
2	Preeti	521.24
3	Rishabh	1042.42
4	Rishabh	928.53

Normalization……Cont.
Third Normal Form

A relation is in ***3NF*** if:

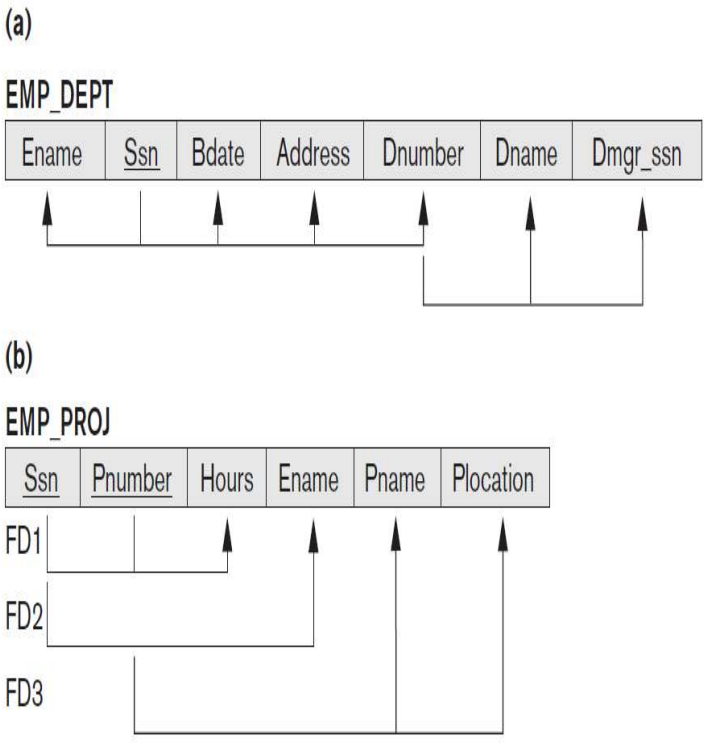
- It is in first and second normal form.
- In which no non-primary-key attribute is transitively dependent on any candidate key.

Examples:

- SSN -> DMGRSSN is a **transitive** FD Since SSN -> DNUMBER and DNUMBER -> DMGRSSN hold
- SSN -> ENAME is **non-transitive**

Since there is no set of attributes X where SSN -> X and X - >ENAME see Figure 14.3 (a).

Figure 14.3
Two relation schemas suffering from update anomalies.
(a) EMP_DEPT and
(b) EMP_PROJ.



Normalization……Cont.
Third Normal Form….Cont.

Process for 3NF:

- Eliminate all dependent attributes in transitive relationship(s) from each of the tables that have a transitive relationship.
- Create new table(s) with removed dependency.

Examples:

We can normalize EMP_DEPT by decomposing it into the two 3NF relation schemas ED1 and ED2 shown in Figure 14.11(b).

Figure 14.11 Normalizing into 3NF

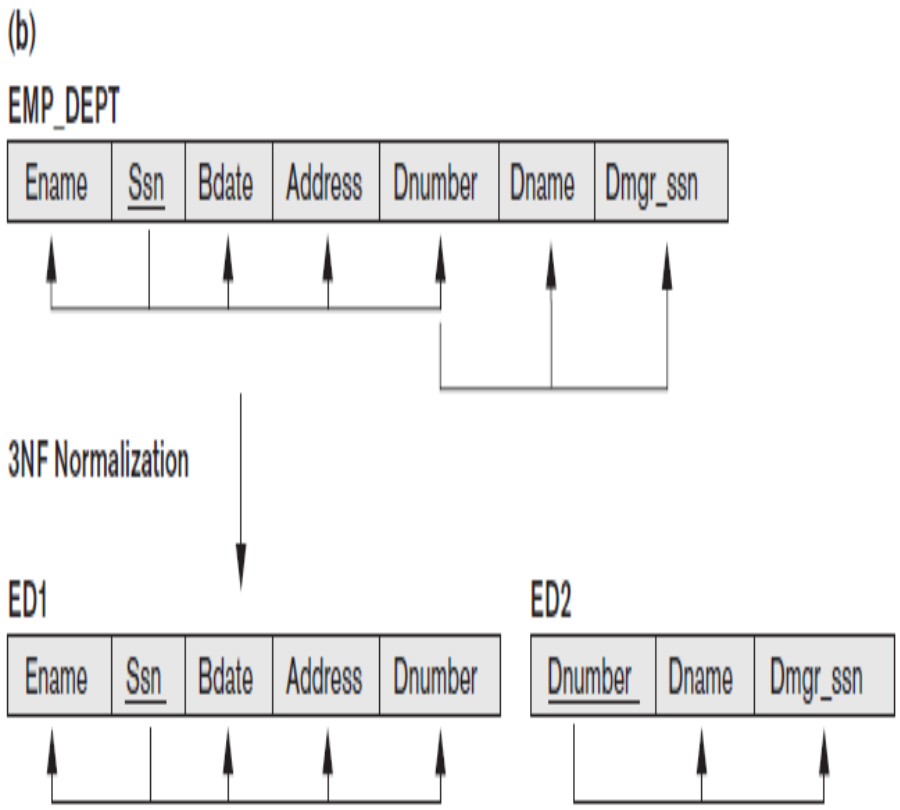


Figure 14.11
Normalizing into 3NF. (b) Normalizing EMP_DEPT into 3NF relations.

Third Normal Form – ….Cont.

Example

Example : normalize the following relation to 3NF

TABLE_BOOK_DETAIL

Book ID	Genre ID	Genre Type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

Solution

TABLE_BOOK

Book ID	Genre ID	Price
1	1	25.99
2	2	14.99
3	1	10.00
4	3	12.99
5	2	17.99

TABLE_GENRE

Genre ID	Genre Type
1	Gardening
2	Sports
3	Travel

Denormalization

Denormalization is a database optimization technique in which we add redundant data to one or more tables. This can help us avoid costly joins in a relational database.

De-Normalization is the opposite process of normalization where the data from multiple tables are combined into one table, so that data retrieval will be faster.

Note that denormalization does not mean not doing normalization. It is an optimization technique that is applied after doing normalization.

Denormalization

Pros of Denormalization:-

- Retrieving data is faster since we do fewer joins
- Queries to retrieve can be simpler, since we need to look at fewer tables.

Cons of Denormalization:-

- Denormalization can make update and insert code harder to write.
- Data redundancy requires more storage.

Normalization Vs Denormalization

Basis for comparison	Normalization	Denormalization
Basic	Normalization is the process of dividing the data into multiple tables	Denormalization is the process of combining the data so that it can be queried speedily.
Purpose	To reduce the data redundancy and inconsistency.	To achieve the faster execution of the queries through introducing redundancy.
Data integrity	Maintained (any addition or deletion of data from the table will not create any mismatch in the relationship of the tables.)	May not retain
Redundancy	Eliminated	Added
Number of tables	Increases and hence the joins to get the result.	Decreases(and hence reduces the number of joins. Hence the performance of the query is faster here compared to normalized tables.)


Thank you