DATABASE MANAGEMENT SYSTEMS

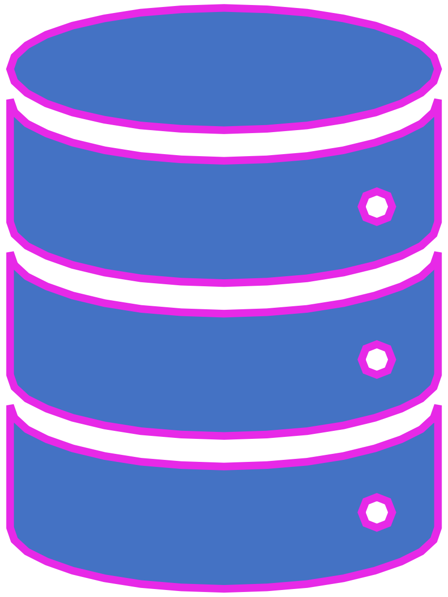
Couse code:CSC403

Prof. Juhi Janjua

Module 5 Relational-Database Design

- Pitfalls in Relational-Database designs
- Concept of normalization
- Function Dependencies
- First Normal Form, 2NF, 3NF, BCNF.

Pitfalls in Relational-Database designs



Relational database design requires that we find a "good" collection of relation schemas. A bad design may lead to

- Repetition of information.
- Inability to represent certain information.

Design Goals:

- Avoid redundant data
- Ensure that relationships among attributes are represented
- Facilitate the checking of updates for violation of database
- Integrity constraints

1.1 Semantics of the Relational Attributes must be clear

GUIDELINE 1: Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).

- Attributes of different entities (EMPLOYEEs, DEPARTMENTs, PROJECTs) should not be mixed in the same relation
- Only foreign keys should be used to refer to other entities
- Entity and relationship attributes should be kept apart as much as possible.

Bottom Line: *Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.*

Figure 14.1 A simplified COMPANY relational database schema

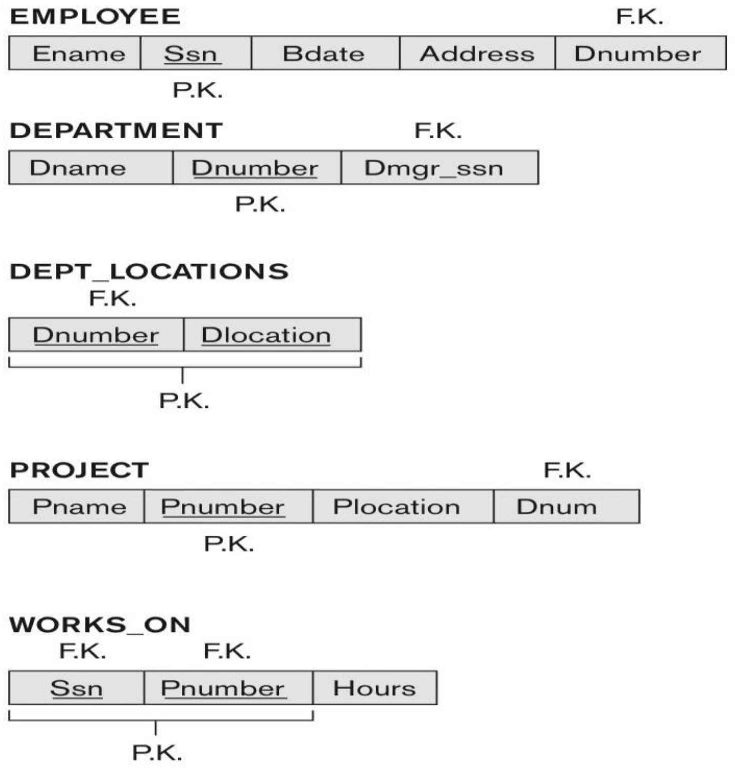


Figure 1 A simplified COMPANY relational database schema.

1.2 Redundant Information in Tuples and Update Anomalies

+ Information is stored redundantly
  - Wastes storage
  - Causes problems with update anomalies
    - *Insertion anomalies*
    - *Deletion anomalies*
    - *Modification anomalies*

EXAMPLE OF AN UPDATE ANOMALY

- Consider the relation:

  EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

- Update Anomaly:

  Changing the name of project number P1 from “Billing” to “Customer-Accounting” may cause this update to be made for all 100 employees working on project P1.

Two relation schemas suffering from update anomalies

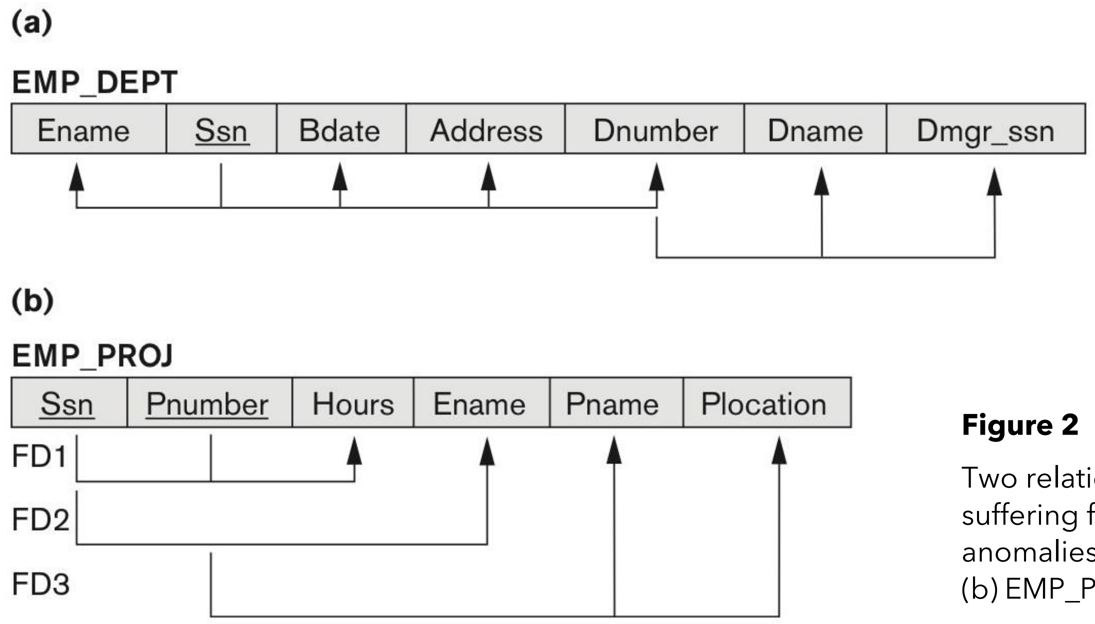


Figure 2

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

EXAMPLE OF AN INSERT ANOMALY

- Consider the relation:

  EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

- 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.

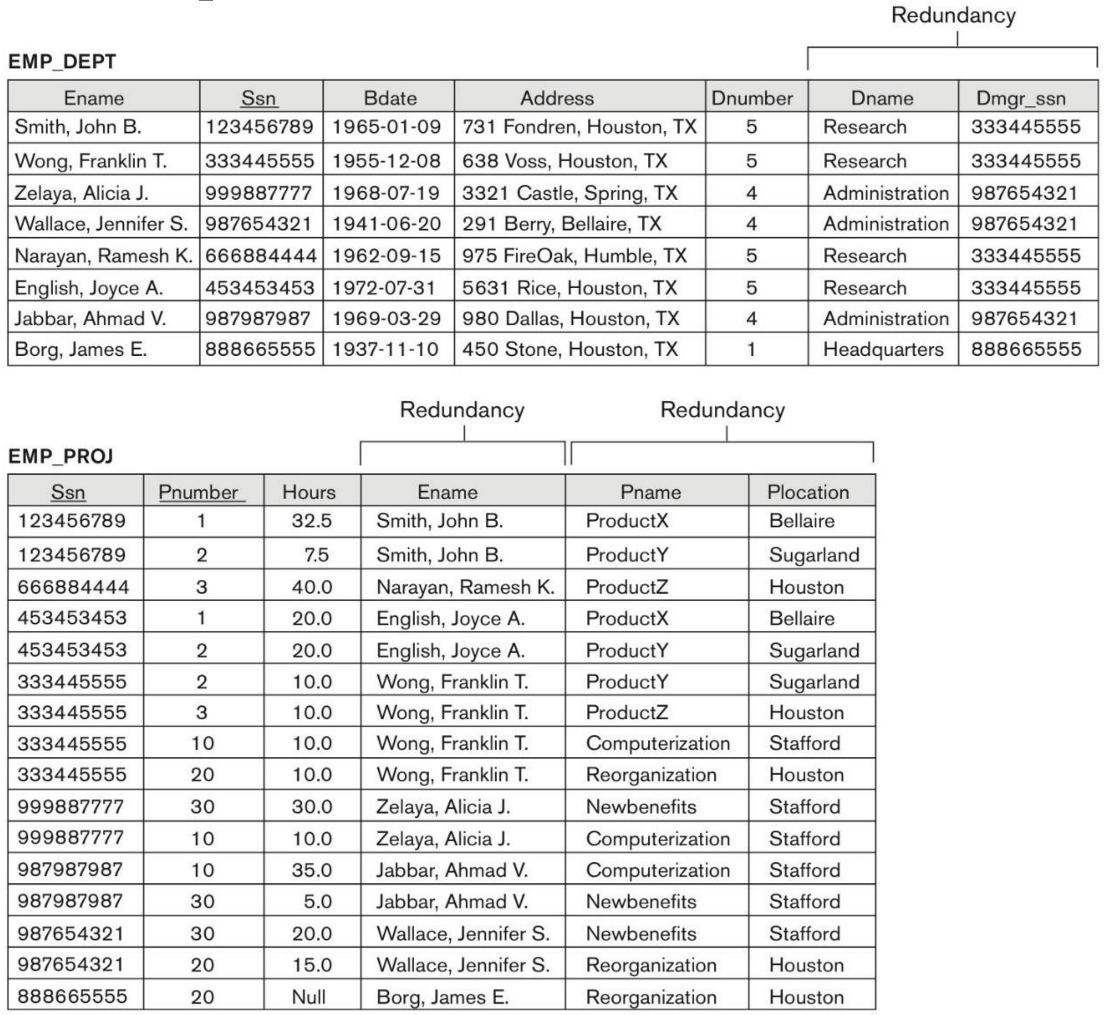
EXAMPLE OF A DELETE ANOMALY

- Consider the relation:

  EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

- 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.

Sample states for EMP_DEPT and EMP_PROJ


Redundancy
Redundancy
Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

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 3

Sample states for EMP_DEPT and EMP_PROJ resulting from applying NATURAL JOIN to the relations in Figure 14.2. These may be stored as base relations for performance reasons.

Guideline for Redundant Information in Tuples and Update Anomalies

- GUIDELINE 2:

  Design a schema that does not suffer from the insertion, deletion and update anomalies.

  If there are any anomalies present, then note them so that applications can be made to take them into account.

1.3 Null Values in Tuples

+ GUIDELINE 3:
  - Relations should be designed such that their tuples will have as few NULL values as possible
  - Attributes that are NULL frequently could be placed in separate relations (with the primary key)
+ Reasons for nulls:
  - Attribute not applicable or invalid
  - Attribute value unknown (may exist)
  - Value known to exist, but unavailable

1.4 Generation of Spurious Tuples – avoid at any cost

+ Bad designs for a relational database may result in erroneous results for certain JOIN operations
+ The "lossless join" property is used to guarantee meaningful results for join operations

+ GUIDELINE 4:
  - The relations should be designed to satisfy the lossless join condition.
  - No spurious tuples should be generated by doing a natural-join of any relations.

Spurious Tuples

- There are two important properties of decompositions:
  a) Non-additive or losslessness of the corresponding join
  b) Preservation of the functional dependencies.

- Note that:

  Property (a) is extremely important and *cannot* be sacrificed.

  Property (b) is less stringent and may be sacrificed.

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.

$$X \rightarrow Y$$

- The left side of FD is known as a determinant, the right side of the production is known as a dependent.

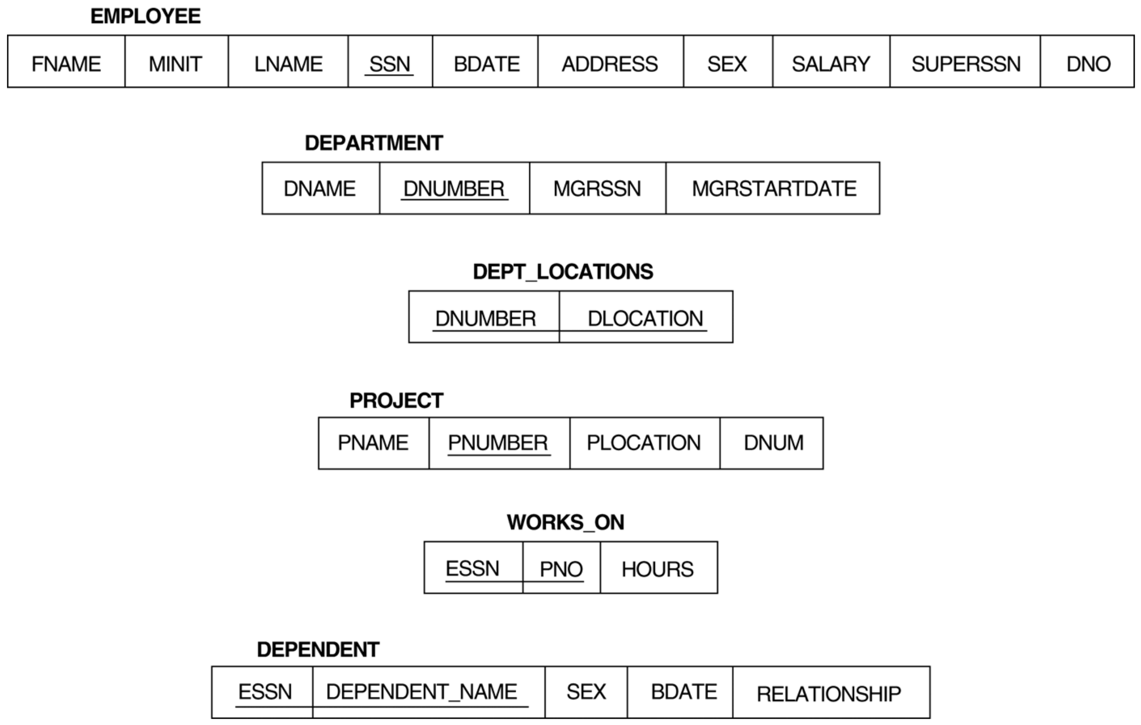
Examples of FD

- Social security number determines employee name

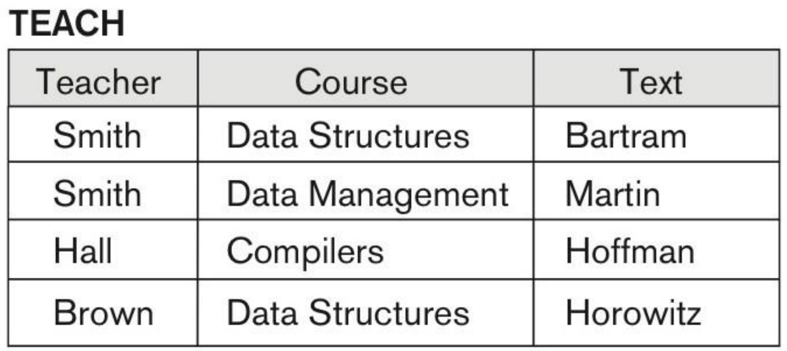
  SSN → ENAME
- Project number determines project name and location

  PNUMBER → {PNAME, PLOCATION}
- Employee ssn and project number determines the hours per week that the employee works on the project

  {SSN, PNUMBER} → HOURS


EMPLOYEE
FNAME MINIT LNAME SSN BDATE ADDRESS SEX SALARY SUPERSSN DNO
DEPARTMENT
DNAME DNUMBER MGRSSN MGRSTARTDATE
DEPT_LOCATIONS
DNUMBER DLOCATION
PROJECT
PNAME PNUMBER PLOCATION DNUM
WORKS_ON
ESSN PNO HOURS
DEPENDENT
ESSN DEPENDENT_NAME SEX BDATE RELATIONSHIP

Ruling Out FDs



TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

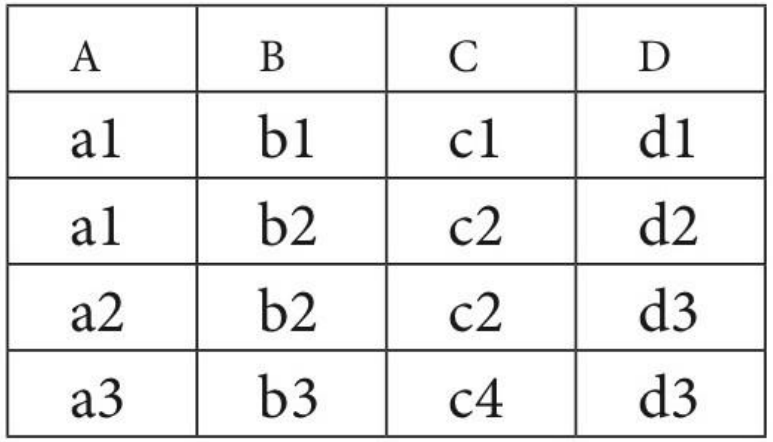
Note that given the state of the TEACH relation, we can say that the FD:

Text → Course may exist.

However, the FDs

Teacher → Course, Teacher → Text and Couse → Text are ruled out.

What FDs may exist?



A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

- A relation *R*(A, B, C, D) with its extension.
- Which FDs *may exist* in this relation?

Armstrong’s axioms/properties of functional dependencies

- **Reflexivity:** If Y is a subset of X, then X→Y holds by reflexivity rule
- For example, {ssn, fname} → fname is valid.
- **Augmentation:** If X → Y is a valid dependency, then XZ → YZ is also valid by the augmentation rule.
- For example, If {ssn, fname} → dno is valid, hence {ssn, fname, superssn} → {dno, superssn} is also valid.
- **Transitivity:** If X → Y and Y → Z are both valid dependencies, then X→Z is also valid by the Transitivity rule.
- For example, ssn → dno & dno → superssn, then ssn → superssn is also valid.

Types of Functional dependencies in DBMS:

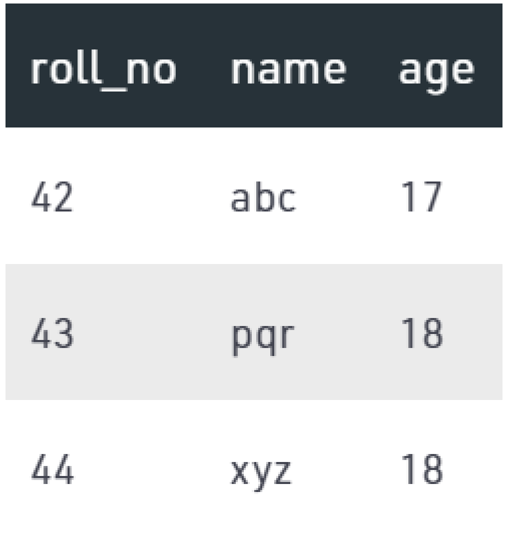
1. Trivial functional dependency
2. Non-Trivial functional dependency
3. Multivalued functional dependency
4. Transitive functional dependency

Trivial functional dependency

- In **Trivial Functional Dependency**, a dependent is always a subset of the determinant.

i.e. If **X** → **Y and Y is the subset of X**, then it is called trivial functional dependency.

- Here, {roll_no, name} → name is a trivial functional dependency, since the dependent name is a subset of determinant set {roll_no, name}
- Similarly, roll_no → roll_no is also an example of trivial functional dependency.



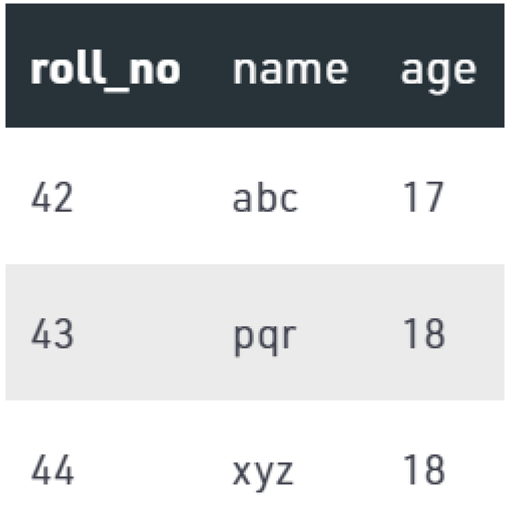
roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Non-trivial Functional Dependency

- In **Non-trivial functional dependency**, the dependent is strictly not a subset of the determinant.

i.e. If **X → Y and Y is not a subset of X**, then it is called Non-trivial functional dependency.

- roll_no → name is a non-trivial functional dependency, since the dependent name is not a subset of determinant roll_no



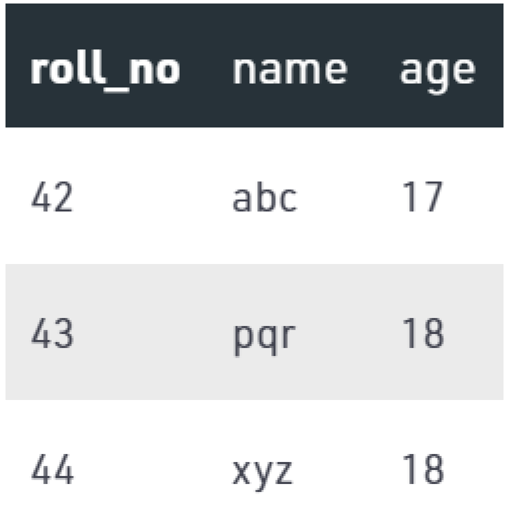
roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Multivalued Functional Dependency

- In **Multivalued functional dependency**, entities of the dependent set are not dependent on each other.

i.e. If **a → {b, c}** and there exists **no functional dependency between b and c**, then it is called a multivalued functional dependency.

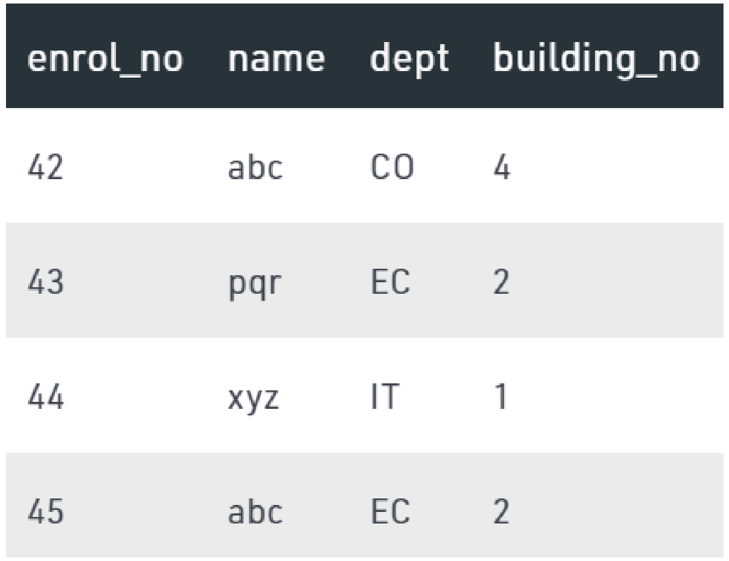
- Here, roll_no → {name, age} is a multivalued functional dependency, since the dependents name & age are not dependent on each other(i.e. name → age or age → name doesn't exist !)



roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Transitive Functional Dependency

+ In transitive functional dependency, dependent is indirectly dependent on determinant.
+ i.e. If **a** → **b & b** → **c,** then according to axiom of transitivity, **a** → **c**. This is a **transitive functional dependency.**
+ Here, enrol_no → dept and dept → building_no,
+ Hence, according to the axiom of transitivity, enrol_no → building_no is a valid functional dependency. This is an indirect functional dependency, hence called Transitive functional dependency.

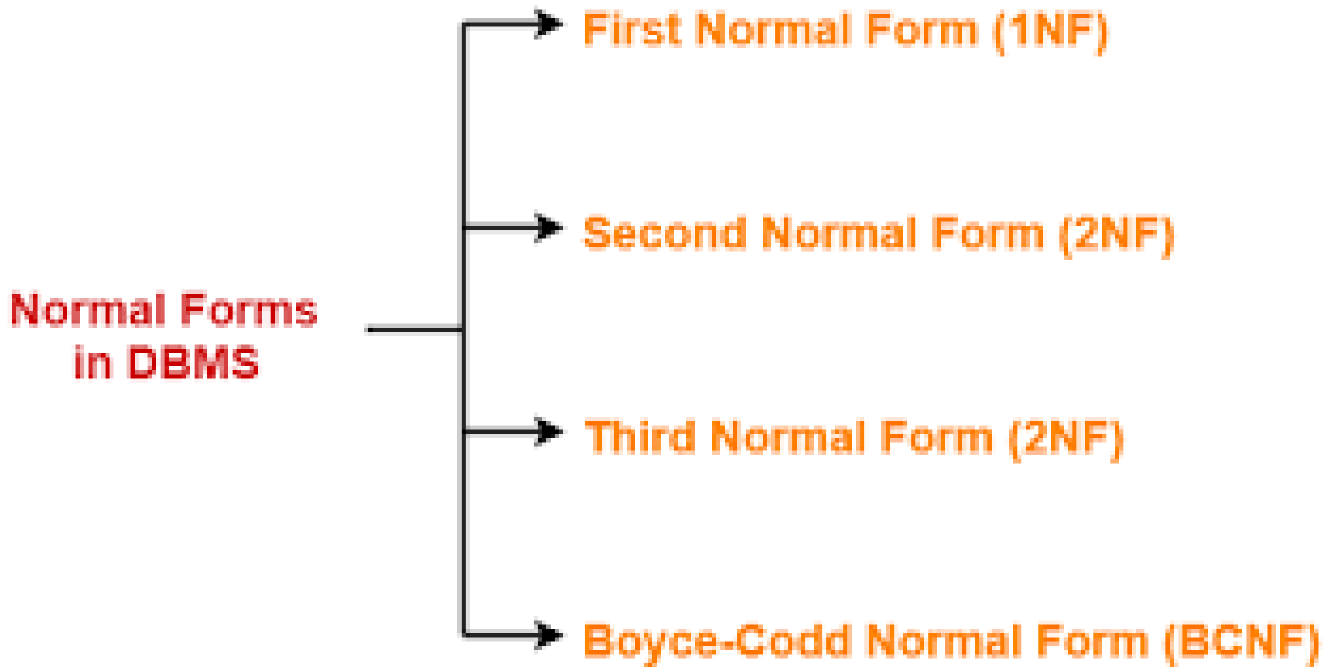


enrol_no	name	dept	building_no
42	abc	CO	4
43	pqr	EC	2
44	xyz	IT	1
45	abc	EC	2

Normalization

- The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations.
- It minimizes **redundancy** from a relation or set of relations.
  - Redundancy in relation may cause insertion, deletion and updation anomalies.
- **Normal forms** are used to eliminate or reduce redundancy in database tables.

Normal forms



First Normal Form

+ 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 a relation
+ Most RDBMSs allow only those relations to be defined that are in First Normal Form

Normalization into 1NF

Normalization into 1NF.

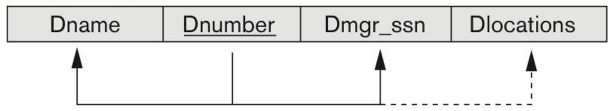
(a) A relation schema that is not in 1NF.

(b) Sample state of relation DEPARTMENT.

(c) 1NF version of the same relation with redundancy.

(a)

DEPARTMENT



(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

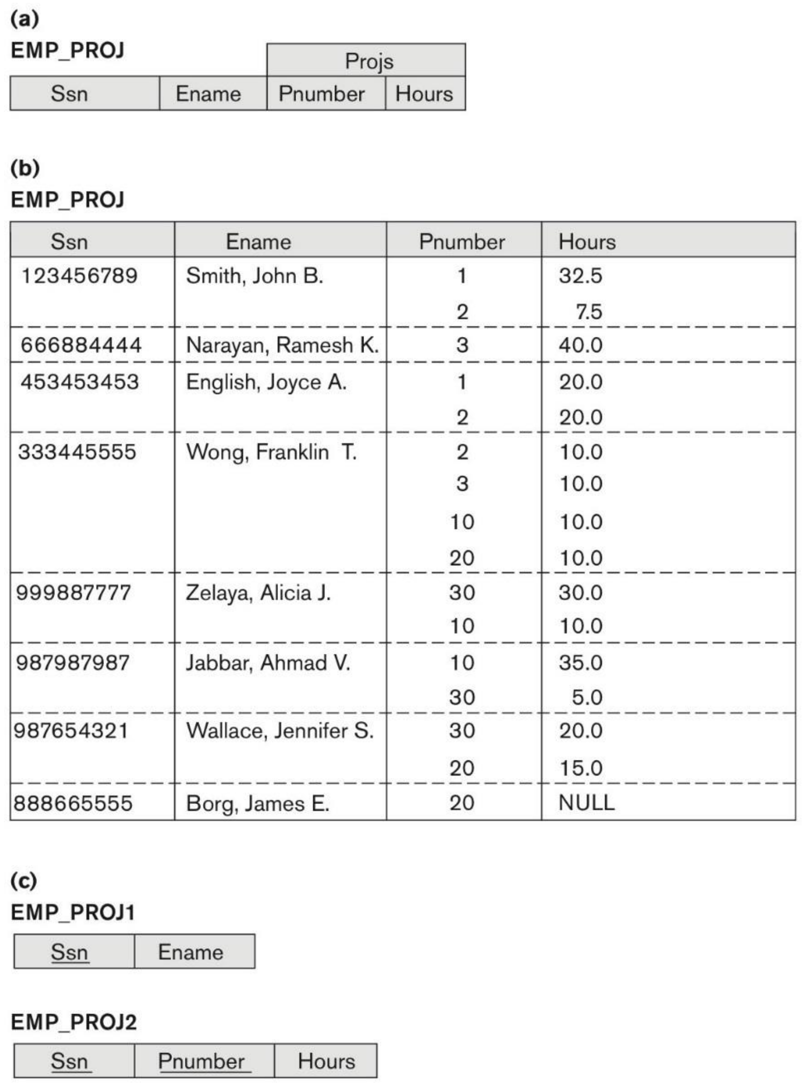
Normalizing nested relations into 1NF

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.



(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

(c)

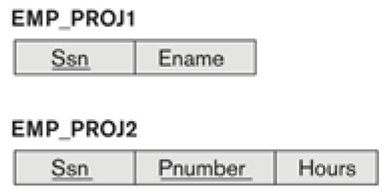
EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

Second Normal Form


EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

- 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
- Examples:

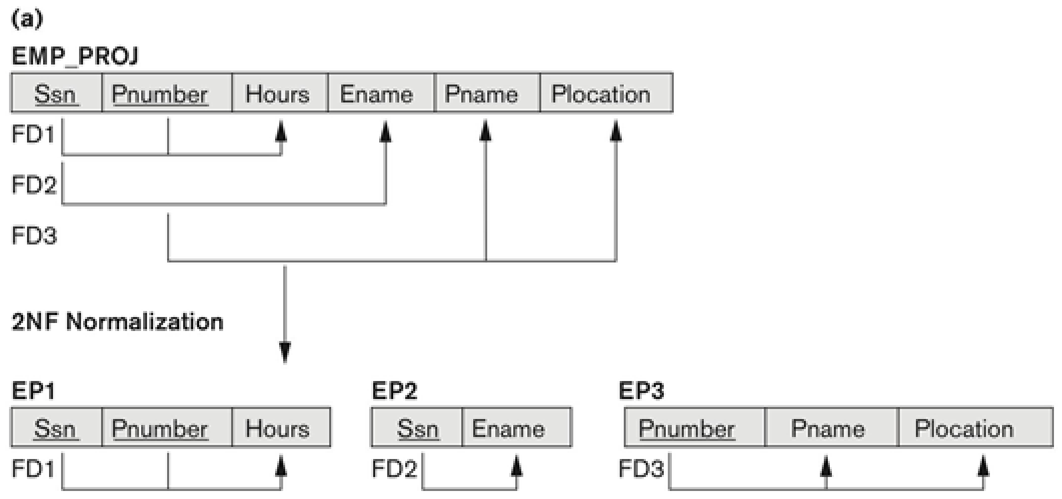
  {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold

  {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):**
  - It should be in the First Normal form.
  - If every non-prime attribute A in R is fully functionally dependent on the primary key

Normalizing into 2NF



Third Normal Form

- Definition:

  Transitive functional dependency: a FD X -> Z that can be derived from two FDs X -> Y and Y -> Z

- 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

Third Normal Form

- 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
- NOTE:

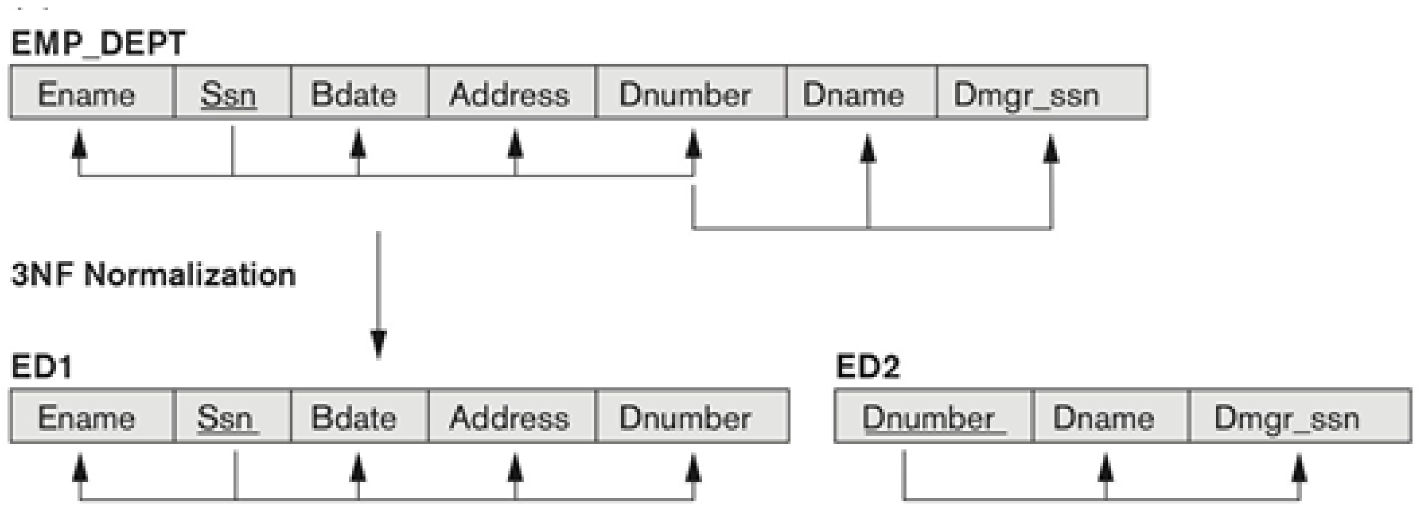
  In X -> Y and Y -> Z, with X as the primary key, we consider this a problem only if Y is not a candidate key.

  When Y is a candidate key, there is no problem with the transitive dependency .

  E.g., Consider EMP (SSN, Emp#, Salary).

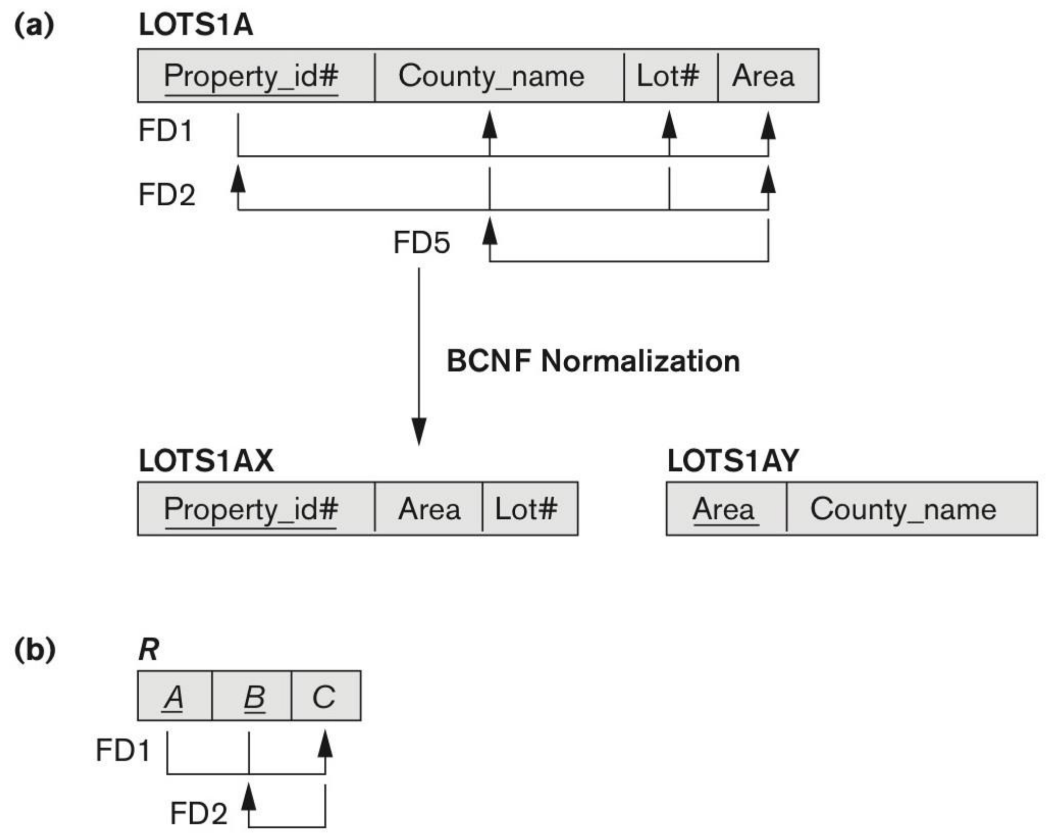
  - Here, SSN -> Emp# -> Salary and Emp# is a candidate key.

Normalizing into 3NF



BCNF (Boyce-Codd Normal Form)

- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever an **FD X** → **A** holds in R, then **X is a superkey** of R
- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF
- Hence BCNF is considered a stronger form of 3NF
- The goal is to have each relation in BCNF (or 3NF)



Example

Boyce-Codd normal form.

(a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition.

(b) A schematic relation with FDs; it is in 3NF, but not in BCNF due to the f.d. C → B.

Example

+ To make the table comply with BCNF we can break the table in three tables like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above:

emp_id -> emp_nationality

emp_dept -> {dept_type, dept_no_of_emp}

emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

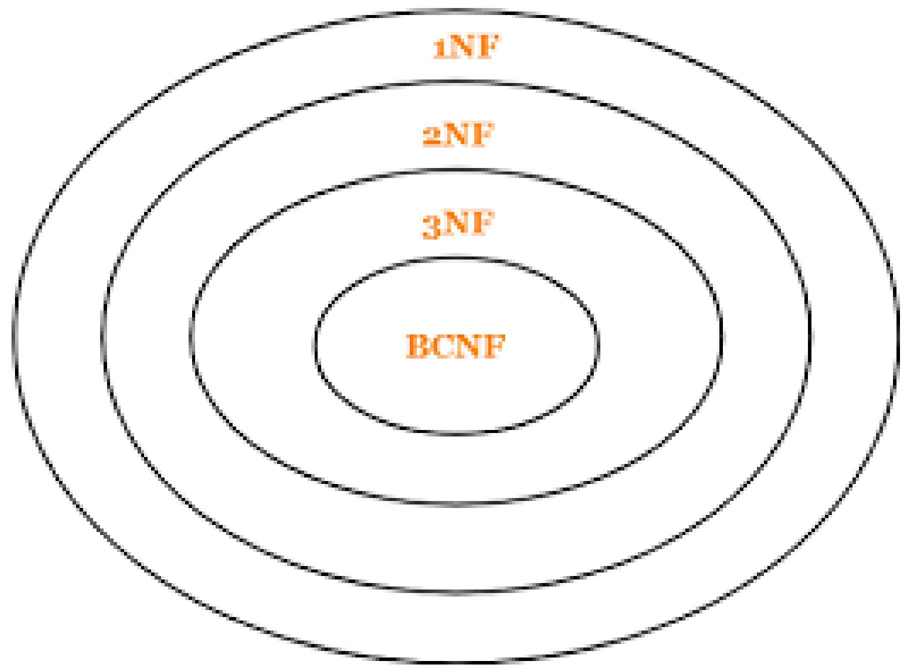
emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Normal Forms


1NF
2NF
3NF
BCNF