Unit - IV

SCHEMA REFINEMENTS

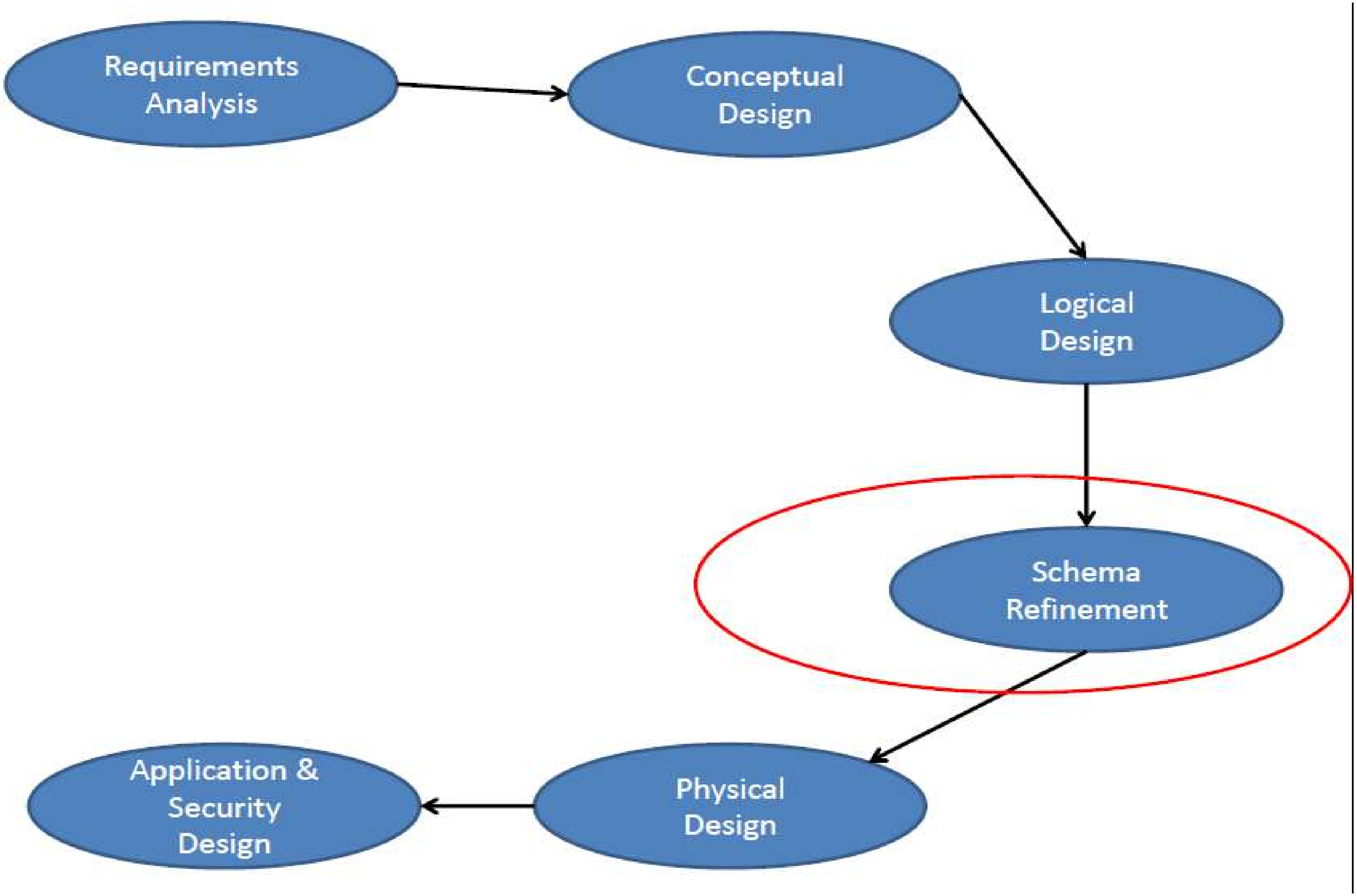
AND

NORMAL FORMS

SYLLABUS

- Introduction to Schema Refinement, Functional Dependencies, Reasoning about FD, Normal Forms, Properties of Decomposition.

Database Design



Database Design

- Requirements Analysis
  - user needs; what must database do?
- Conceptual Design
  - high level description (often done with ER model)
- Logical Design
  - translate ER into DBMS data model
- Schema Refinement
  - consistency, normalization
- Physical Design - indexes, disk layout
- Security Design - who accesses what

Contents

- Introduction to Schema Refinements
- Functional Dependencies
- Normal Forms
- Properties of Decomposition

Introduction to Schema Refinements

- The Schema Refinement refers to refine the schema by using some technique.
- The best technique of schema refinement is decomposition. Normalisation or Schema Refinement is a technique of organizing the data in the database.
- It is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies.
- **Redundancy** refers to repetition of same data or duplicate copies of same data stored in different locations.
- **Anomalies:** Anomalies refers to the problems occurred after poorly planned and normalised databases where all the data is stored in one table which is sometimes called a flat file database.

The Problems of Redundancy

- *Redundancy:* root of several problems with relational schemas:
  - redundant storage, *insert/delete/update anomalies*
- *Functional dependencies:*
  - a form of *integrity constraint* that can identify schemas with such problems and suggest refinements.
- Main refinement technique: *decomposition*
  - replacing ABCD with, say, AB and BCD, or ACD and ABD.

The Problems of Redundancy

EmpDept

EID	Name	DeptID	DeptName
A01	Ali	12	Wing
A12	Eric	10	Tail
A13	Eric	12	Wing
A03	Tyler	12	Wing

- What anomalies are associated with EmpDept?
- Update Anomalies: If one copy of such repeated data is updated, an inconsistency is created unless all copies are similarly updated.
- Can we change DeptName of DeptID 12 in the first tuple? We should repeat the same for all tuples.

The Problems of Redundancy

EmpDept

<u>EID</u>	Name	DeptID	DeptName
A01	Ali	12	Wing
A12	Eric	10	Tail
A13	Eric	12	Wing
A03	Tyler	12	Wing

- Insertion Anomalies: Cannot insert a department unless an employee is assigned to it. It may not be possible to store certain information unless some other, unrelated, information is stored as well.
- Deletion Anomalies: If we delete record of A12 then the Department 10 no longer exists.

What is Schema Refinement?

- Schema Refinement is the study of what should go where in a DBMS, or, which schemas are best to describe an application.
- For example, consider this schema

EmpDept

EID	Name	DeptID	DeptName
A01	Ali	12	Wing
A12	Eric	10	Tail
A13	Eric	12	Wing
A03	Tyler	12	Wing

- **Versus this one:**

Emp

EID	Name	DeptID
A01	Ali	12
A12	Eric	10
A13	Eric	12
A03	Tyler	12

Dept

DeptID	DeptName
12	Wing
10	Tail

What's wrong?*

- The first problem students usually identify with the EmpDept schema is that it combines two different ideas: employee information and department information. But what is wrong with this?

1. If we separated the two concepts we could save space.
2. Combining the two ideas leads to some bad anomalies.

- These two problems occur because DeptID determines DeptName, but DeptID is not a key.

Decomposition: A good solution

- The standard solution to the redundancy problem is to decompose redundant schemas, e.g., EmpDept becomes

Emp

EID	Name	DeptID
A01	Ali	12
A12	Eric	10
A13	Eric	12
A03	Tyler	12

Dept

DeptID	DeptName
12	Wing
10	Tail

- The secret to understanding when and how to decompose schemas is **Functional Dependencies**, a generalization of keys.
- When we say "X determines Y" we are stating a functional dependency.

Problems related to decomposition

- Decomposing a relation schema can create more problems than it solve.
- Do we need to decompose a relation?
- Several normal forms have been proposed for the relations
- The normal form of a given relationship schema can help us to decide whether or not to decompose it further.
- If we decide to decompose further, then we have to choose a particular decomposition.

What problem(if any) does the given decomposition cause?

2 properties are important

1. Lossless-join property

It enables us to recover any instance of decomposed relation from the corresponding instance of the smaller relations.

2. Dependency preservation property

It enables us to enforce any constraint on the original relation by simply enforcing some constraints on each of the smaller relation.

Reasoning about FDs

Functional dependency

- It is denoted by X -->Y, where X and Y are attributes.
- The constraint is that for any two tuples t1 and t2 that have t1[X]=t2[X], they must also have t1[Y]=t2[Y].
- We say that there is a functional dependency from X to Y or Y is functionally dependent on X
- The abbreviation for functional dependency is FD or f.d.
- The set of attributes X is called the left-hand side of the FD, & Y is called the right-hand side of FD.

Functional Dependencies

EmpDept

EID	Name	DeptID	DeptName
A01	Ali	12	Wing
A12	Eric	10	Tail
A13	Eric	12	Wing
A03	Tyler	12	Wing

- A key like EID has another property: If two rows have the same EID, then they have the same value of every other attribute. We say EID **functionally determines** all other attributes and write this **Functional Dependency (FD)**:

  EID → Name, DeptID, DeptName
- Is **Name → DeptID** true?
  - No, because rows 2 and 3 have the same Name but not the same DeptID.

Functional Dependencies (cond.,)

EmpDept

EID	Name	DeptID	DeptName
A01	Ali	12	Wing
A12	Eric	10	Tail
A13	Eric	12	Wing
A03	Tyler	12	Wing

- Do you see any more FDs in EmpDept?
  - Yes, the FD **DeptID → DeptName**

- **DEFINITION:** If A and B are sets of attributes in a relation, we say that **A (functionally) determines B**, or **A → B is a Functional Dependency (FD)**, the value of a row on A functionally determines its value on B.
- There are two special kinds of FDs:
  - Key FDs, **X → A** where X contains a key

Reasoning about FDs

EmpDept(<u>EID</u>, Name, DeptID, DeptName)

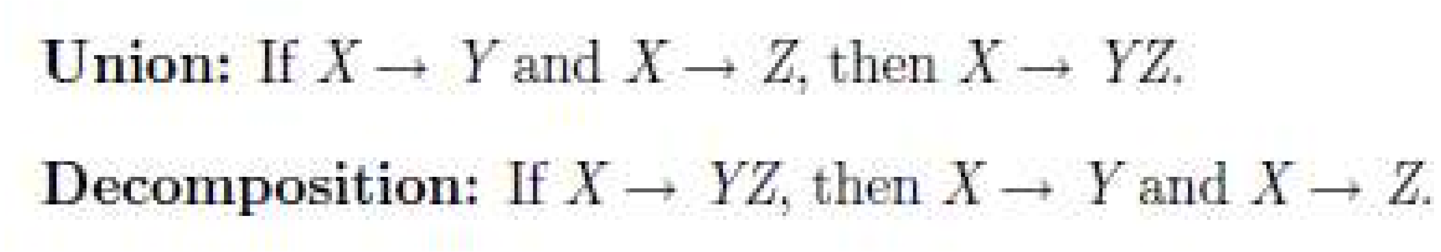
- Two natural FDs are

  EID → DeptID and **DeptID → DeptName**

- These two FDs imply the FD **EID → DeptName**
  - Because if two tuples agree on EID, then by the first FD they agree on DeptID, then by the second FD they agree on DeptName.
- The set of FDs implied by a given set F of FDs is called the closure of F and is denoted F^+

Armstrong's Axioms

- The closure of F can be computed using these axioms
  - Reflexivity: If **X** $\supseteq$ **Y**, then **X** → **Y**
  - Augmentation: If **X** → **Y**, then **XZ** → **YZ** for any Z
  - Transitivity: If **X** → **Y** and **Y** → **Z** then **X** → **Z**
- Armstrong's axioms are sound (they generate only FDs in F^+ when applied to FDs in F) and complete (repeated application of these axioms will generate all FDs in F^+).



Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$.

Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$.

Pseudo transitivity : if x→y and yz→w then xz→w

NORMAL FORMS

Normal forms

Well structured relation

- A relation that contain minimum redundancy and allow users to safely insert, delete, update records in a table is called well structured.

Normal form

- It is a state of a relation obtained by applying simple rules regarding FDs.

Normalization

- The process of decomposing a relation which having the anomalies into smaller relation to produce well structured relations is called normalization.

First Normal Form (1 N F)

Each attribute must be atomic

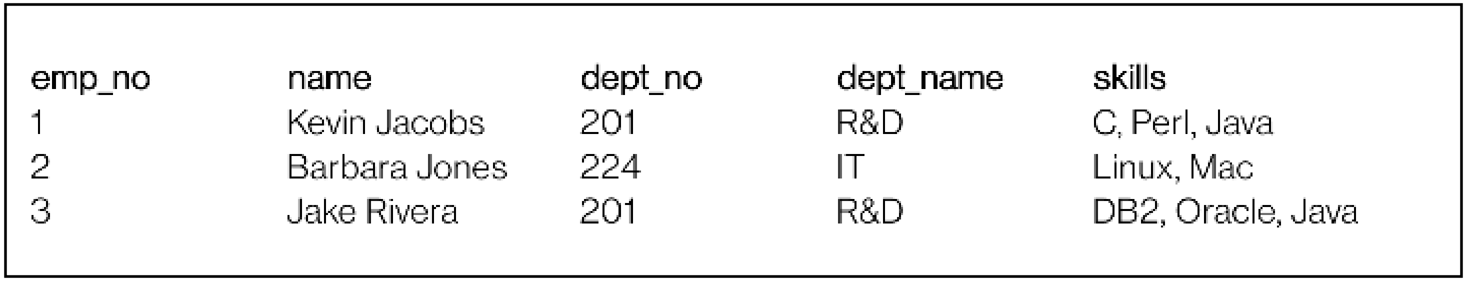
- No repeating columns within a row.
- No multi-valued columns.

1NF simplifies attributes

- Queries become easier.

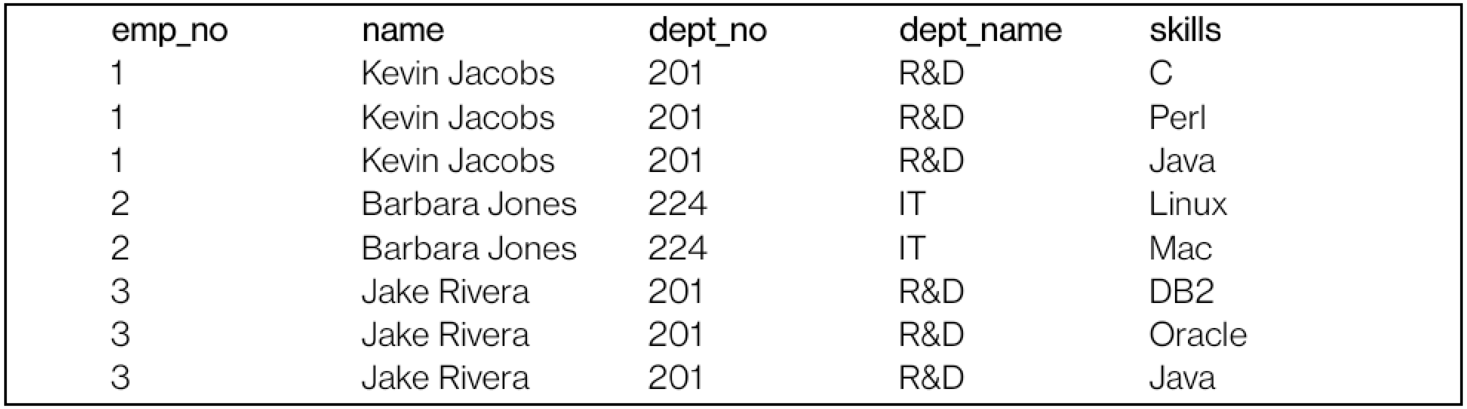
1 N F

Employee (unnormalized)



emp_no	name	dept_no	dept_name	skills
1	Kevin Jacobs	201	R&D	C, Perl, Java
2	Barbara Jones	224	IT	Linux, Mac
3	Jake Rivera	201	R&D	DB2, Oracle, Java

Employee (1NF)



emp_no	name	dept_no	dept_name	skills
1	Kevin Jacobs	201	R&D	C
1	Kevin Jacobs	201	R&D	Perl
1	Kevin Jacobs	201	R&D	Java
2	Barbara Jones	224	IT	Linux
2	Barbara Jones	224	IT	Mac
3	Jake Rivera	201	R&D	DB2
3	Jake Rivera	201	R&D	Oracle
3	Jake Rivera	201	R&D	Java

Second Normal Form (2 N F)

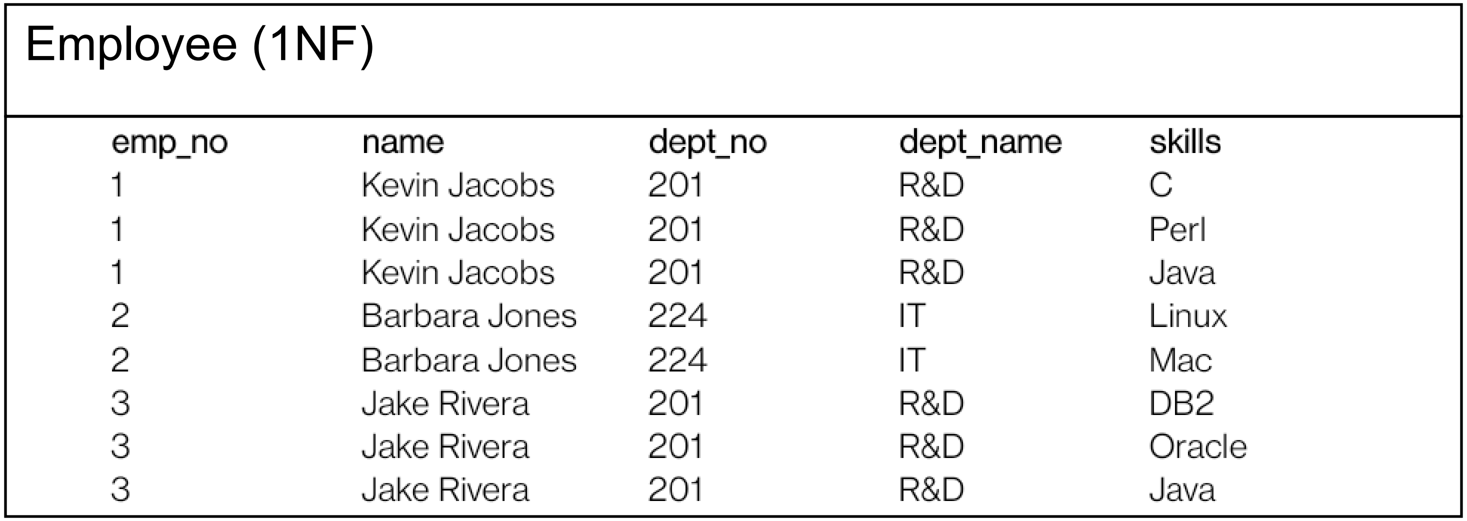
Each attribute must be functionally dependent on the primary key.

- Functional dependence - the property of one or more attributes that uniquely determines the value of other attributes.
- Any non-dependent attributes are moved into a smaller (subset) table.

2NF improves data integrity.

- Prevents update, insert, and delete anomalies.

Functional Dependence



Employee (1NF)

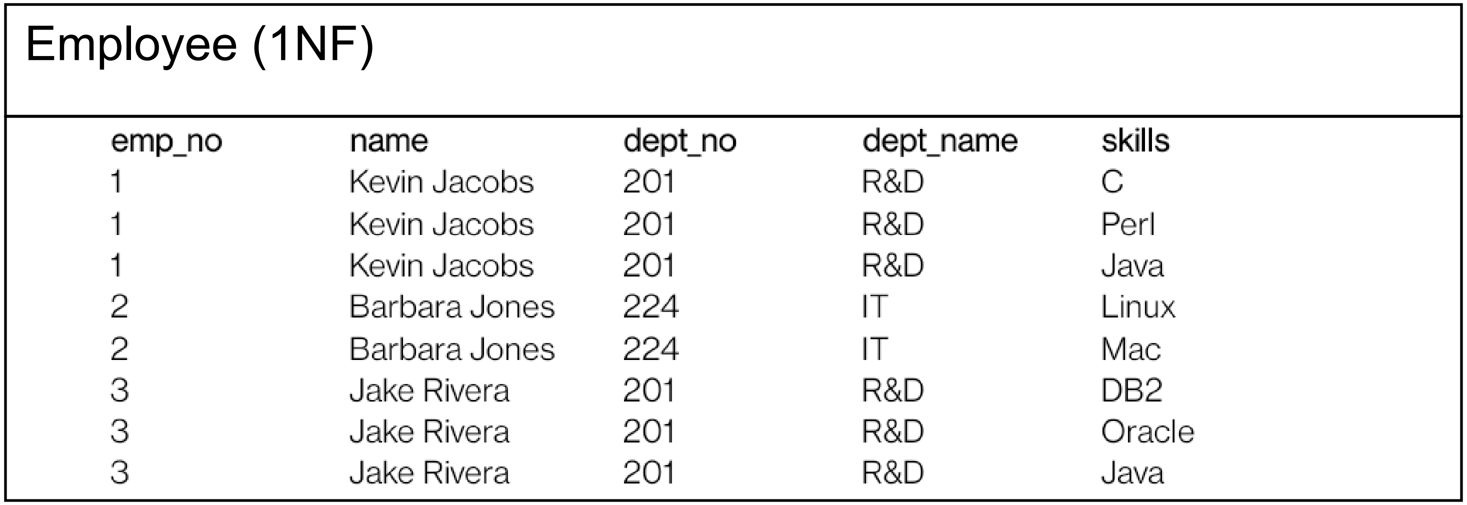
emp_no	name	dept_no	dept_name	skills
1	Kevin Jacobs	201	R&D	C
1	Kevin Jacobs	201	R&D	Perl
1	Kevin Jacobs	201	R&D	Java
2	Barbara Jones	224	IT	Linux
2	Barbara Jones	224	IT	Mac
3	Jake Rivera	201	R&D	DB2
3	Jake Rivera	201	R&D	Oracle
3	Jake Rivera	201	R&D	Java

Name, dept_no, and dept_name are functionally dependent on emp_no.

(emp_no -> name, dept_no, dept_name)

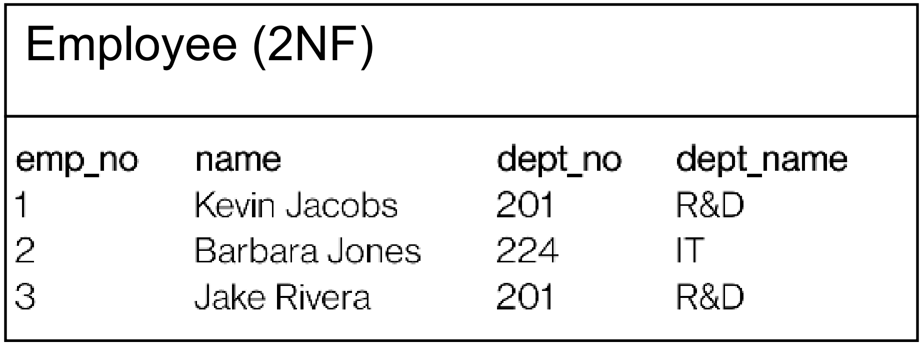
Skills is not functionally dependent on emp_no since it is not unique to each emp_no.

2 N F



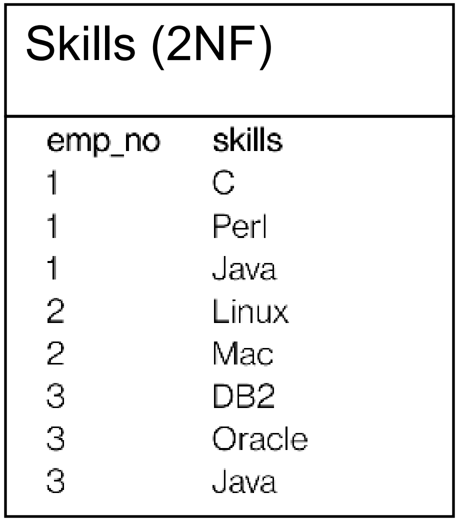
Employee (1NF)

emp_no	name	dept_no	dept_name	skills
1	Kevin Jacobs	201	R&D	C
1	Kevin Jacobs	201	R&D	Perl
1	Kevin Jacobs	201	R&D	Java
2	Barbara Jones	224	IT	Linux
2	Barbara Jones	224	IT	Mac
3	Jake Rivera	201	R&D	DB2
3	Jake Rivera	201	R&D	Oracle
3	Jake Rivera	201	R&D	Java



Employee (2NF)

emp_no	name	dept_no	dept_name
1	Kevin Jacobs	201	R&D
2	Barbara Jones	224	IT
3	Jake Rivera	201	R&D



Skills (2NF)

emp_no	skills
1	C
1	Perl
1	Java
2	Linux
2	Mac
3	DB2
3	Oracle
3	Java

Third Normal Form (3NF)

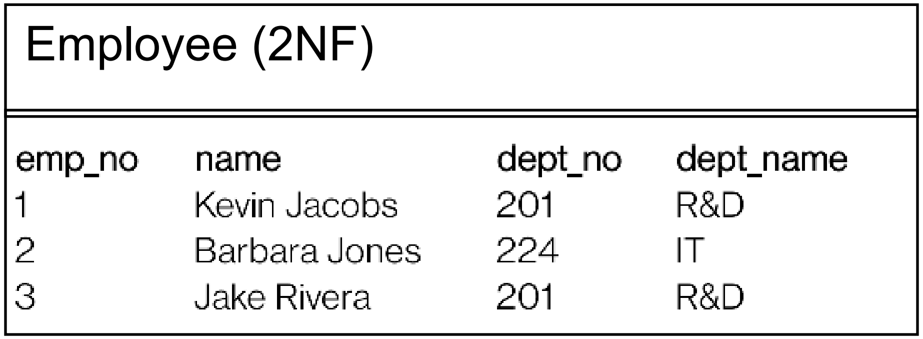
Remove transitive dependencies.

- Transitive dependence - two separate entities exist within one table.
- Any transitive dependencies are moved into a smaller (subset) table.

3NF further improves data integrity.

- Prevents update, insert, and delete anomalies.

Transitive Dependence

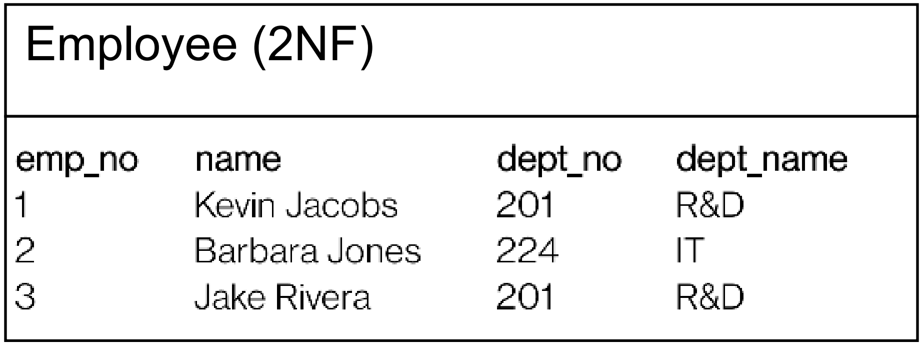


Employee (2NF)

emp_no	name	dept_no	dept_name
1	Kevin Jacobs	201	R&D
2	Barbara Jones	224	IT
3	Jake Rivera	201	R&D

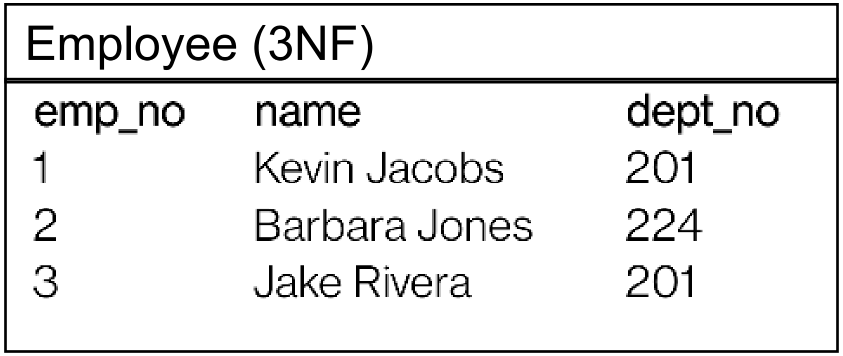
Dept_no and dept_name are functionally dependent on emp_no however, department can be considered a separate entity.

3 N F



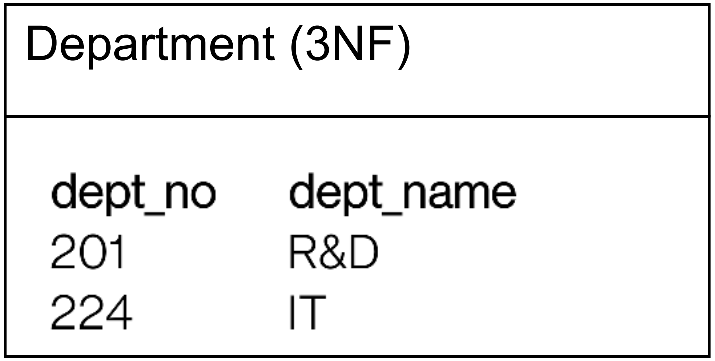
Employee (2NF)

emp_no	name	dept_no	dept_name
1	Kevin Jacobs	201	R&D
2	Barbara Jones	224	IT
3	Jake Rivera	201	R&D



Employee (3NF)

emp_no	name	dept_no
1	Kevin Jacobs	201
2	Barbara Jones	224
3	Jake Rivera	201



Department (3NF)

dept_no	dept_name
201	R&D
224	IT

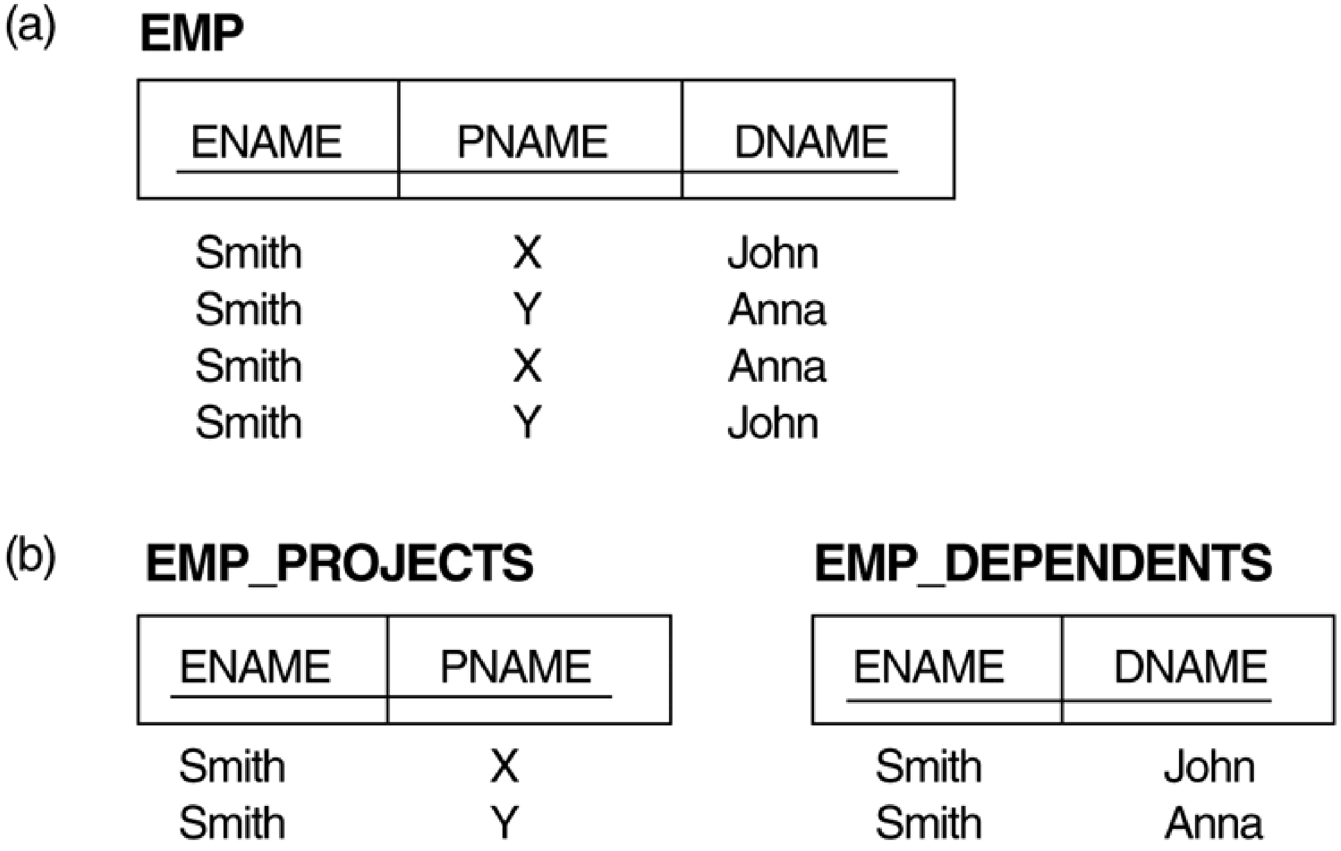
BCNF(Boyce Codd Normal form)

- BCNF is a Advanced version of 3NF.It is Stricter than 3NF.
- A table id Functional Dependency X->Y is the Super key of the Table.
- For BCNF Table should be in 3NF.

Fourth Normal Form (4 N F)

- 4th Normal Form
  - BCNF with no multi valued dependencies
  - Create separate tables for each separate functional dependency

- (a) The EMP relation with two MVDs: ENAME —>> PNAME and ENAME —>> DNAME. (b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.



(a) **EMP**

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

(b) **EMP_PROJECTS**

ENAME	PNAME
Smith	X
Smith	Y

EMP_DEPENDENTS

ENAME	DNAME
Smith	John
Smith	Anna

Normal Form Comparisons

- 4NF ⊂ BCNF ⊂ 3NF

Property	3NF	BCNF	4NF
eliminates FD redundancies	most	yes	yes
eliminates MVD redundancies	no	no	yes
preserves FDs	yes	maybe	maybe
preserves MVDs	maybe	maybe	no

End of Unit 4