

UNIT 4

Database Normalization

(4 Hours) — Database Management System (CAC255) · BCA II Year / IV Semester

⚠ Important Note — Please Read

These notes were prepared quickly from the course syllabus and standard textbook knowledge. **They may contain errors, typing mistakes or incomplete points.** Please take care: cross-check definitions, formulas, SQL syntax and examples with your class notes, the prescribed textbook and your teacher before relying on them for exams or practicals.

4.1 Definition and Importance of Normalization

Normalization is the process of organizing the tables and columns of a relational database to **reduce data redundancy** and to remove undesirable **anomalies**. Large tables are decomposed into smaller, well-structured tables connected by keys, without losing information.

Why is normalization needed? — Anomalies

Consider the badly designed table **StudentCourse**, where student and course details are repeated in every row:

SID	SName	CID	CName	Grade
1	Ram	C1	DBMS	A
1	Ram	C2	Java	B
2	Sita	C1	DBMS	A

Anomaly	Meaning	Example in the table above
Insertion anomaly	Cannot insert some data without other data.	A new course cannot be stored until some student enrolls in it.
Update anomaly	Changing one fact requires changes in many rows; missing one gives inconsistency.	Renaming DBMS to “Database Systems” must be done in every row.
Deletion anomaly	Deleting one fact accidentally deletes other facts.	Deleting Sita’s only enrollment also deletes her name.

Importance: normalization reduces redundancy, saves storage, prevents anomalies, keeps data consistent, and makes the design clearer and easier to maintain.

*Note: Too much normalization can slow queries because more joins are required; sometimes designers deliberately **denormalize** for speed.*

4.2 Functional Dependencies

A **functional dependency (FD)** $X \rightarrow Y$ means that if two tuples have the same value of X they must have the same value of Y ; in other words **X determines Y** . X is called the **determinant**. Example: $SID \rightarrow SName$.

- **Trivial FD:** Y is a subset of X , e.g. $\{SID, SName\} \rightarrow SName$.
- **Non-trivial FD:** Y is not a subset of X , e.g. $SID \rightarrow SName$.
- **Full functional dependency:** Y depends on the whole of a composite key X and not on any proper subset of it.
- **Partial dependency:** a non-key attribute depends on *part* of a composite candidate key.
- **Transitive dependency:** $X \rightarrow Y$ and $Y \rightarrow Z$ (with Y not a key), so $X \rightarrow Z$ indirectly.

Armstrong's Axioms and Rules

Rule	Statement
Reflexivity	If $Y \subseteq X$ then $X \rightarrow Y$.
Augmentation	If $X \rightarrow Y$ then $XZ \rightarrow YZ$.
Transitivity	If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$.
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 $X \rightarrow Y$ and $WY \rightarrow Z$ then $WX \rightarrow Z$.

Closure of an attribute set and finding candidate keys

The **closure** X^+ is the set of all attributes that can be determined from X using the given FDs. If X^+ contains all attributes of the relation, then X is a super key; a minimal such X is a candidate key.

Example 4.1 — Find the candidate key

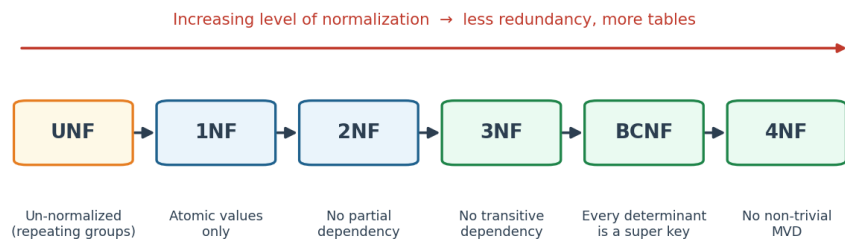
$R(A, B, C, D)$ with FDs: $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$.

Solution

$A^+ = \{A\} \rightarrow \text{add } B (A \rightarrow B) \rightarrow \text{add } C (B \rightarrow C) \rightarrow \text{add } D (C \rightarrow D) = \{A, B, C, D\} = \text{all attributes.}$

No smaller subset exists, so **A is the candidate key.**

4.3 Normal Forms



Each higher form assumes all lower forms are already satisfied.

Figure 4.1 — Progression of normal forms

First Normal Form (1NF)

A relation is in 1NF if every attribute holds only **atomic (single) values** — no multi-valued attributes or repeating groups.

Example 4.2 — Not in 1NF

Student(SID, Name, Phone) where Ram has Phone = “9801, 9841” in one cell.

Solution — convert to 1NF

Store each phone in a separate row: (1, Ram, 9801) and (1, Ram, 9841). Better: separate table StudentPhone(SID, Phone).

Second Normal Form (2NF)

A relation is in 2NF if it is in 1NF and **no non-prime attribute is partially dependent** on any candidate key (every non-key attribute depends on the *whole* key). 2NF matters only when the key is composite.

Example 4.3 — Not in 2NF

Enrollment(SID, CID, SName, CName, Grade) with FDs: $SID \rightarrow SName$, $CID \rightarrow CName$, $(SID, CID) \rightarrow Grade$.

SName depends only on SID and CName only on CID — partial dependencies.

Solution — decompose into 2NF

Student(SID, SName) | Course(CID, CName) | Enrollment(SID, CID, Grade)

Third Normal Form (3NF)

A relation is in 3NF if it is in 2NF and has **no transitive dependency** of a non-prime attribute on the key. Formally, for every FD $X \rightarrow A$, either X is a super key or A is a prime attribute.

Example 4.4 — Not in 3NF

Employee(EmpID, EName, DeptID, DeptName) with FDs: $\text{EmpID} \rightarrow \text{DeptID}$ and $\text{DeptID} \rightarrow \text{DeptName}$.

So $\text{EmpID} \rightarrow \text{DeptName}$ is transitive.

Solution — decompose into 3NF

Employee(EmpID, EName, DeptID) | Department(DeptID, DeptName)

Boyce-Codd Normal Form (BCNF)

A relation is in BCNF if, for **every non-trivial FD $X \rightarrow Y$** , X is a **super key**. BCNF is stricter than 3NF: every BCNF relation is in 3NF, but not vice-versa.

Example 4.5 — In 3NF but not in BCNF

R(Student, Subject, Teacher) with FDs: $(\text{Student}, \text{Subject}) \rightarrow \text{Teacher}$ and **$\text{Teacher} \rightarrow \text{Subject}$** (each teacher teaches one subject).

Candidate keys: (Student, Subject) and (Student, Teacher). Since Teacher is not a super key, $\text{Teacher} \rightarrow \text{Subject}$ violates BCNF, although Subject is a prime attribute so 3NF holds.

Solution — decompose into BCNF

R1(Teacher, Subject) | R2(Student, Teacher)

Fourth Normal Form (4NF)

A relation is in 4NF if it is in BCNF and has **no non-trivial multi-valued dependency (MVD)** except on a super key. A multi-valued dependency $X \twoheadrightarrow Y$ means that for each X value there is a set of Y values independent of the other attributes.

Example 4.6 — Multi-valued dependency

Person(Name, Hobby, Language): Ram has hobbies {Chess, Music} and speaks {Nepali, English}. To keep them independent, 4 rows are needed: (Ram, Chess, Nepali), (Ram, Chess, English), (Ram, Music, Nepali), (Ram, Music, English).
Here Name $\Rightarrow$ Hobby and Name $\Rightarrow$ Language.

Solution — decompose into 4NF

PersonHobby(Name, Hobby) | PersonLanguage(Name, Language)

Comparison of Normal Forms

Form	Condition to satisfy	Removes
1NF	Atomic values only	Repeating groups, multi-valued attributes
2NF	1NF + no partial dependency	Partial dependencies
3NF	2NF + no transitive dependency	Transitive dependencies
BCNF	Every determinant is a super key	Remaining FD anomalies
4NF	BCNF + no non-trivial MVD	Multi-valued dependencies

Properties of a good decomposition

- **Lossless-join:** joining the decomposed tables gives back exactly the original table (no spurious tuples).
- **Dependency preservation:** all original FDs can still be enforced on the smaller tables. 3NF can always achieve both; BCNF may lose dependency preservation.

Key Takeaways — Unit 4

- ✓ Normalization removes redundancy and insertion, update and deletion anomalies.
- ✓ A functional dependency $X \rightarrow Y$ means X determines Y; use attribute closure to find keys.
- ✓ 1NF: atomic values. 2NF: no partial dependency. 3NF: no transitive dependency.
- ✓ BCNF: every determinant must be a super key. 4NF: no non-trivial multi-valued dependency.
- ✓ Decomposition should be lossless; over-normalization may need extra joins.

— End of Unit 4 —

Reminder: these notes may contain errors — verify with your textbook and teacher.