Lecture 2.2. Normalization

What is Normalization?

Normalization is database design technique that:

> - Organizes tables to **reduce redundancy** (duplicate data)
> - Reduces **dependency** (data depending on wrong attributes)
> - **Splits** big tables into smaller, related tables
> - Links them using **relationships** (usually foreign keys)
> - Goal: **eliminate useless/duplicate data** and ensure data is stored **logically**

One fact = one place

Benefits of Normalization

- **Improved data integrity**
  - Same fact is not stored in 10 places with different values
- **Smaller databases**
  - No repeated long strings everywhere
- **Better performane**
  - Less duplicate data → smaller I/O
- **Fewer indexes to maintain**
- **Less management** (simplier schema, easier updates)
- **Prevents data modifications anomalies**:
  - Insert anomaly - can`t insert something because other fields missing
  - Update anomaly - must update same info in many rows
  - Delete anomaly - deleting one row accidentally deletes important info

---

Normal Forms - hierarchy

- **1NF** - weakest
- **2NF** - stronger than 1NF
- **3NF** - stronger than 2NF
- **BCNF** - stronger than 3NF

And:

- If a relation is in **BCNF** → it is **automatically in 3NF**
- In **3NF → automatically in 2NF**
- In **2NF → automatically in 1NF**

First Normal Form (1NF)

Definition:

> **All values are atomic** (single-valued) - no lists, no reapiting groups.
> Values are **simple**, not arrays/sets
> Each row is uniquely identifiable (usually by a **primary key**)

Exmple of NOT 1NF:

Name	Birth_date	Certifications	Certification1	Certification2	Certification3
John	1990/02/12	PBI, MCSA, MTA	PBI	MCSA	MTA
Kelly	1988/10/01	Excel	Excel	NULL	NULL
Bryan	...	PBI, Excel	PBI	Excel	NULL
Rob	...	PBI	PBI	NULL	NULL

Problems:

- `Certifications` column contains multiple values in one field (violates atomicity)
- `Certifications1/2/3` are reapiting groups - structure depends on “max 3 certs”
- If someone gets a 4th certification → design breaks
- NULLs appear just to “fill grid”

Bringing it to 1NF:

Learner_ID	Name	Birth_date	Certification_ID	Certification_name	Vendor_name	Date_acquired
1	John	1990/02/12	1	PBI	Microsoft	2017/01/08
1	John	1990/02/12	2	MCSA	Microsoft	2020/04/16
1	John	1990/02/12	3	MTA	Microsoft	2021/05/12
2	Kelly	1988/10/01	4	Excel	Microsoft	2018/06/03
3	Bryan	1991/01/15	1	PBI	Microsoft	2019/12/11
3	Bryan	1991/01/15	4	Excel	Microsoft	2020/04/07
4	Rob	1986/08/07	1	PBI	Microsoft	2022/01/15

We create a table where **each row has exactly one certification per learner**

Now:

- Each column has single values only
- Still some redundancy, but 1NF condition is satisfied
- An outer join of normalized tables later can reproduce original info

---

Second Normal Form (2NF)

Definition:

> It`s **already in 1NF,** all non-key attributes are fully funcionally dependent on the entire primary key

Key phrase:

> No **partial dependency** of non-key columns on part of a composite primary key

> if the **primary key is a single column**, the table is **automatically in 2NF** (because there is no “part of the key”

Where 2NF problem appears in our 1NF table

If we use **(Learner_ID, Certification_ID)** as composite primary key in the 1NF table:

- **Name**, **Birth_date** depend only on **Learner_ID** (part of the key).
- **Certification_name**, **Vendor_name** depend only on **Certification_ID** (other part of the key).
- Both are **partial dependencies** → violates 2NF.

Fixing to 2NF – splitting tables

1. Learner

Learner_ID (PK)	Name	Birth_date
1	John	1990/02/12
2	Kelly	1988/10/01
3	Bryan	1991/01/15
4	Rob	1986/08/07

2. Certification

Certification_ID (PK)	Certification_name	Vendor_name
1	PBI	Microsoft
2	MCSA	Microsoft
3	MTA	Microsoft
4	Excel	Microsoft

3. **Learner_Certification** (bridge)

Learner_ID (PK part, FK)	Certification_ID (PK part, FK)	Date_acquired
1	1	2017/01/08
1	2	2020/04/16
1	3	2021/05/12
2	4	2018/06/03
3	1	2019/12/11
3	4	2020/04/07
4	1	2022/01/15

Now:

- In **Learner_Certification**, non-key column `Date_acquired` depends on **whole composite key** (Learner_ID + Certification_ID).
- No attribute depends only on Learner_ID or only on Certification_ID.
- Non-key attributes in Learner/Certification tables depend on their **single-column PK** → 2NF satisfied.

Exam hint:

> If you see composite key (A,B) and some column depends only on A or only on B → **2NF violation**.

Third Normal Form (3NF)

Defenition

> It satisfies **all conditions of 2NF,** and there is **no transitive dependency of non-key attributes** on the key

Transitive dependency:

- Key → attribute X → attribute Y
- Where X is non-key and Y is non-key
- Then Y is transitively dependent on the key via X

Example:

In Certification table (2NF version):

Certification_ID (PK)	Certification_name	Vendor_name
1	PBI	Microsoft
2	MCSA	Microsoft
3	MTA	Microsoft
4	Excel	Microsoft

Here:

- PK = `Certification_ID`
- `Certification_name` depends on `Certification_ID` ✅
- `Vendor_name` also depends on `Certification_ID`, BUT logically:
  - `Certification_ID` → `Vendor_name`
  - `Vendor_name` is describing **Vendor**, which is its own entity.

Normalization to 3NF:

1. New Vendor table:

Vendor_ID (PK)	Vendor_name
1	Microsoft
2	Amazon

2. **Update Certification** to use Vendor_ID:

Certification_ID (PK)	Certification_name	Vendor_ID (FK)
1	PBI	1
2	MCSA	1
3	MTA	1
4	Excel	1

Now:

- Certification_ID → Vendor_ID
- Vendor_ID → Vendor_name (in Vendor table)
- But in **Certification** table, all non-key attributes (`Certification_name`, `Vendor_ID`) depend **directly on key** and none depends on another non-key attribute.
- Vendor_name is in separate table, so **no transitive dependency inside a single relation**.

Summary for 1NF–3NF

- **1NF:**
  - Single atomic value in each column
  - Each row has a **unique identifier**
- **2NF:**
  - Satisfy all **1NF** conditions
  - Remove **partial dependencies** (each non-key depends on whole key)
- **3NF:**
  - Satisfy all **2NF** conditions
  - Remove **transitive dependency** of non-key attributes on key

How to Normalize a Table in Steps

- **Check 1NF**:
  - Any columns with lists (e.g. "A,B,C") or repeating groups (`Phone1`, `Phone2`)?
  - If yes → create new table(s) so that **each cell is atomic**.
- **Check 2NF** (only if composite PK):
  - Is PK composed of several columns?
  - Does any non-key attribute depend only on part of the key?
  - If yes → move that attribute (and its determinant) into a new table.
- **Check 3NF**:
  - Is there an attribute A (non-key) that determines attribute B (non-key)?
  - Key → A → B pattern?
  - If yes → split so that A and B live in different tables (A with key, B in table with A as key/foreign key).
- **Redraw final schema**:
  - Show all tables and keys
  - Indicate foreign key relationships

Typical exam questions

MCQ examples:

1. A table where a column contains values like `"Math, Physics"` violates:
   - A) 1NF
   - B) 2NF
   - C) 3NF
   - D) BCNF

   → **Answer: A (1NF)**

2. In 2NF, which dependency is **not allowed**?
    - A) Key → Non-key
    - B) Composite key → Non-key
    - C) Part of composite key → Non-key
    - D) Non-key → Non-key

      → **Answer: C** (partial dependency)
3. 3NF specifically removes:
    - A) Repeating groups
    - B) Transitive dependencies
    - C) Multi-valued dependencies
    - D) Join dependencies

      → **Answer: B**

Open question examples:

- “Explain the difference between 1NF, 2NF, and 3NF using an example.”
- “What is a transitive dependency? How does 3NF eliminate it?”
- “Why does normalization help prevent update anomalies? Give an example.”