Introduction to Databases

Lecture 2 – Normalization theory

Gianluca Quercini

gianluca.quercini@centralesupelec.fr

Master DSBA 2020 – 2021

CentraleSupélec

What you will learn

In this lecture you will learn:

- Why **normalization theory** is important.
- What **functional dependencies** are.
- How **functional dependencies** relate to **keys**.
- How to determine the **normal form** of a relational table.
- How to **normalize** a relational table.

Motivating example

What's bad about this table?

book_id	isbn	title	type	year	authors	publisher_name	publisher_country
book-001	978-0321197849	An introduction to database systems	Hardcover	1999	Christopher J. Date, UK	Addison-Wesley	United States
book-001	978-81780882318	An introduction to database systems	Paperback	1999	Christopher J. Date, UK	Addison-Wesley	United States
book-002	978-3446215337	Data mining	Paperback	2001	Ian H. Witten, NZ; Frank Eibe, DE	Addison-Wesley	United States
book-003	978-0132923606	Structured computer organisation	Kindle	2013	Andrew S. Tanenbaum, US	Pearson	United States

Motivating example

- **Redundancy.** Title, isbn, type of a book are stored several times.
- **Update anomalies.** Renaming a title requires updating several rows.
- **Insertion anomalies.** How to add rows *consistently*? How to add books with no publisher?
- **Deletion anomalies.** Deleting *book-003* also deletes all information about the publisher *Pearson*.

book_id	isbn	title	type	year	authors	publisher_name	publisher_country
book-001	978-0321197849	An introduction to database systems	Hardcover	1999	Christopher J. Date, UK	Addison-Wesley	United States
book-001	978-81780882318	An introduction to database systems	Paperback	1999	Christopher J. Date, UK	Addison-Wesley	United States
book-002	978-3446215337	Data mining	Paperback	2001	Ian H. Witten, NZ; Frank Eibe, DE	Addison-Wesley	United States
book-003	978-0132923606	Structured computer organisation	Kindle	2013	Andrew S. Tanenbaum, US	Pearson	United States

▸ Source

Data redundancy

Data redundancy is particularly harmful for two reasons:

- **Waste of space.** Redundant data occupy unnecessary space on disk.
  - That was a huge concern in the 70s (cost of 1 GB ~$300k). ▸ Source
  - Less of a concern today (cost of 1 GB ~$0.019).

- **Update anomalies.**
  - The publisher country of a book occurs in several rows.
  - When we update the country in a row, we need to update the others as well.

Relational database design

Good relational database design principles:

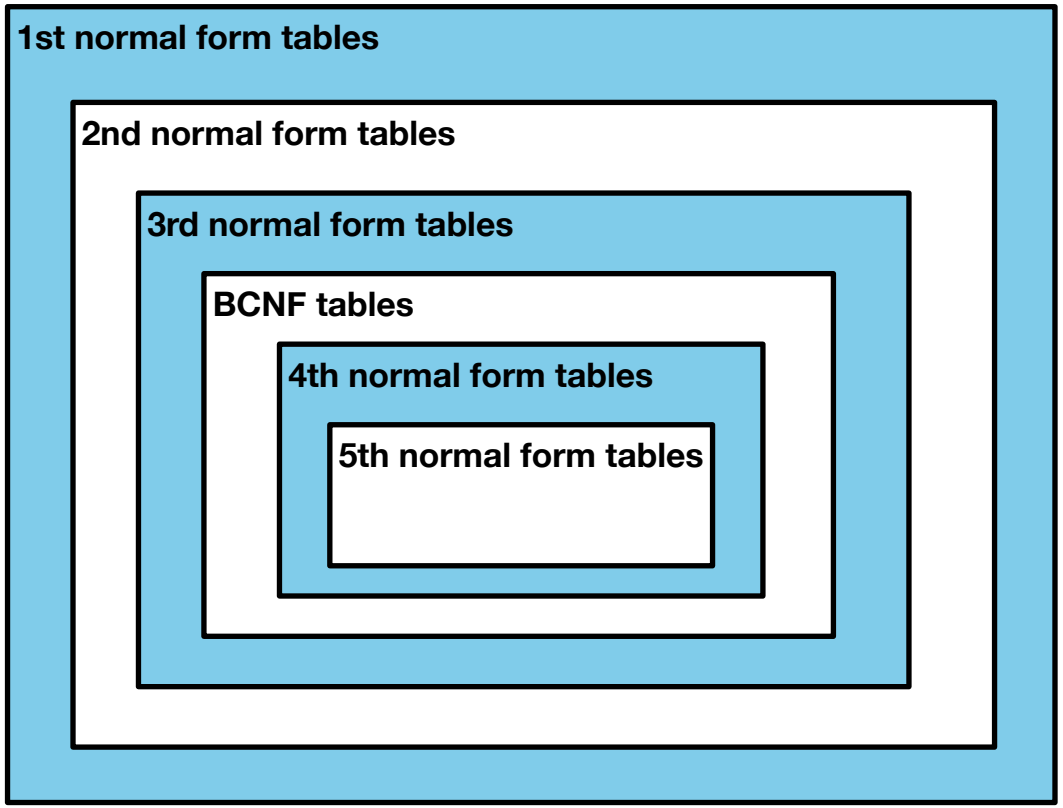
- **Minimize redundancy**, avoiding to store the same information several times.
  - Create separate tables for logically distinct information.
  - **Example.** Information on the authors should not be mixed with the information on the publishers.

- **Avoid modification anomalies**, by imposing primary and foreign key constraints.

Normalization

Definition (Normalization — Informal definition)

We define **normalization** as the process of splitting up a table so that redundancy is eliminated or reduced to a minimum.



- A table can be in one of six **normal forms**.
- A table in a particular normal form meets **some specific constraints.**
- Each normal form adds some constraints to the previous.

Normalization

Definition (Normalization — Formal definition)

Normalization is a procedure that consists in successively reducing a given collection of tables to some more desirable normal form. ▸ Source

- The normalization procedure must be **nonloss**, or **information-preserving**.
  - The output of the normalization procedure can be **mapped back** to the input.
  - To understand the normalization procedure, we need to introduce the notion of **functional dependency**.

Functional dependencies

Definition (Functional dependency)

Given a relational table, a set of column $Y = \{Y_1, \ldots, Y_n\}$ is **functionally dependent** on a set of columns $X = \{X_1, \ldots, X_m\}$, which is denoted as $X \to Y$, if **any value** $(x_1, \ldots, x_m)$ of X always implies value $(y_1, \ldots, y_n)$ of Y.

- X is called the **determinant** of the functional dependency $X \to Y$.
- Y is called the **dependent** of the functional dependency $X \to Y$.

Definition (Functional dependency — Informal definition)

In relational table, $X \to Y$ if whenever two rows agree on their X values, they also agree on their Y values. ▸ Source

Functional dependencies

Example

Consider the following table:

`Employee`(code_employee, first_name, last_name, position, salary, dept_id)

Example of functional dependencies are:

- code_employee $\rightarrow$ {first_name, last_name, position, salary, dept_id}
- position $\rightarrow$ salary
- {position, salary} $\rightarrow$ salary

- If X is a **candidate key** for a table T, then all columns of T must be functionally dependent on X.
- If $X \rightarrow Y$, and X is **not** a candidate key, then the table has some **redundancy**.
  - In our example, the salary is repeated at every row where the corresponding position occurs.

Functional dependencies

- Functional dependencies (FDs) are **semantic constraints** imposed on a table.
- The **database designer** is responsible for identifying the FDs.
- The database designer must know the **domain** of the data being modeled.

> ☞ In order to determine the FDs, the database designer **must not look at a particular instance of a table**. An FD is a constraint that all instances of a table must comply with.

- Similarly to key constraints, FDs are part of the **integrity constraints** of a database.

Functional dependencies

Exercise

Given the following table:

`Book` (book_id, isbn, title, type, authors, publisher_name, publisher_country)

can you list some of the functional dependencies?

- The complete set of FDs can be **very large**.
- But some FDs imply other FDs.
- Since we want a DBMS to enforce FDs, **we should avoid redundant FDs.**

Functional dependencies

Exercise

Given the following table:

`Book` (book_id, isbn, title, type, authors, publisher_name, publisher_country)

can you list some of the functional dependencies?

- The complete set of FDs can be **very large**.
- But some FDs imply other FDs.
- Since we want a DBMS to enforce FDs, **we should avoid redundant FDs.**

Functional dependencies

Exercise

Given the following table:

`Book` (book_id, isbn, title, type, authors, publisher_name, publisher_country)

can you list some of the functional dependencies?

- The complete set of FDs can be **very large**.
- But some FDs imply other FDs.
- Since we want a DBMS to enforce FDs, **we should avoid redundant FDs.**

☞ But how can we determine a non-redundant set of FDs?

Trivial FDs

Some FDs are **trivial** and should not be included in the final set.

Definition (Trivial functional dependency)

A FD is **trivial** if it cannot possibly fail to be satisfied. ▸ Source

Definition (Trivial functional dependency)

An FD is **trivial** if and only if the **right side is a subset of the left side**.

Example

$\{\text{position, salary}\} \rightarrow \text{salary}$

☞ Trivial FDs account for a small part of the redundant FDs. To go further, we need to introduce the notion of **closure**.

Closure of a set of FDs

Definition (Closure)

Given a set $\mathcal{F}$ of FDs, the **closure** of $\mathcal{F}$, denoted as $\mathcal{F}^+$, is the set of all FDs **implied** by $\mathcal{F}$.

- We can use **Armstrong's axioms** to compute $\mathcal{F}^+$ from $\mathcal{F}$.

Armstrong's axioms

Let A, B and C be three arbitrary sets of columns of a given table. The following inference rules apply:

- **Reflexivity.** If $B \subseteq A$, then $A \to B$.
- **Augmentation.** If $A \to B$, then $AC \to BC$.
- **Transitivity.** If $A \to B$ and $B \to C$ then $A \to C$.

☞ AB is a shorthand for $A \cup B$

Closure of a set of columns

- Computing $\mathcal{F}^+$ with Armstrong's axioms is inefficient.
- Instead we find the set $Z_{\mathcal{F}}^+$ (the **closure** of Z given $\mathcal{F}$) of all columns that are functionally dependent on Z.
  - Z is any subset of columns of the table.

Computing $Z_{\mathcal{F}}^+$ from $\mathcal{F}$

1: $Z_{\mathcal{F}}^+ := Z$
2: **while** True **do**
3: **for all** $X \to Y \in \mathcal{F}$ **do**
4: **if** $X \subseteq Z_{\mathcal{F}}^+$ **then**
5: $Z_{\mathcal{F}}^+ := Z_{\mathcal{F}}^+ \cup Y$
6: **end if**
7: **end for**
8: **if** $Z_{\mathcal{F}}^+$ did not change in this iteration **then**
9: exit loop
10: **end if**
11: **end while**

Closure of a set of attributes

Example

- We're given a table with four columns: A, B, C, D.
- We're given the following set $\mathcal{F}$ of functional dependencies:

$$A \to BC,\ E \to CF,\ B \to E,\ CD \to EF.$$

- We compute $Z^+_{\mathcal{F}}$, where $Z = \{A, B\}$.
- We initialize $Z^+_{\mathcal{F}} = \{A, B\}$.

Closure of a set of attributes

Example

- We're given a table with four columns: A, B, C, D.
- We're given the following set $\mathcal{F}$ of functional dependencies:

$$A \to BC,\ E \to CF,\ B \to E,\ CD \to EF.$$

- We compute $Z^+_{\mathcal{F}}$, where $Z = \{A, B\}$.
- We initialize $Z^+_{\mathcal{F}} = \{A, B\}$.

First iteration — First FD

- $Z^+_{\mathcal{F}} = \{A, B\}$
- We consider the FD $A \to BC$.
- $\{A\} \subset Z^+_{\mathcal{F}}$; then $Z^+_{\mathcal{F}} = Z^+_{\mathcal{F}} \cup \{B, C\} = \{A, B, C\}$.

Closure of a set of attributes

Example

- We're given a table with four columns: A, B, C, D.
- We're given the following set $\mathcal{F}$ of functional dependencies:

$$A \to BC,\ E \to CF,\ B \to E,\ CD \to EF.$$

- We compute $Z^+_{\mathcal{F}}$, where $Z = \{A, B\}$.
- We initialize $Z^+_{\mathcal{F}} = \{A, B\}$.

First iteration — Second FD

- $Z^+_{\mathcal{F}} = \{A, B, C\}$
- We consider the FD $E \to CF$.
- $\{E\} \not\subseteq Z^+_{\mathcal{F}}$; then $Z^+_{\mathcal{F}} = \{A, B, C\}$.

Closure of a set of attributes

Example

- We're given a table with four columns: A, B, C, D.
- We're given the following set $\mathcal{F}$ of functional dependencies:

$$A \to BC,\ E \to CF,\ B \to E,\ CD \to EF.$$

- We compute $Z^+_{\mathcal{F}}$, where $Z = \{A, B\}$.
- We initialize $Z^+_{\mathcal{F}} = \{A, B\}$.

First iteration — Third FD

- $Z^+_{\mathcal{F}} = \{A, B, C\}$
- We consider the FD $B \to E$.
- $\{B\} \subset Z^+_{\mathcal{F}}$; then $Z^+_{\mathcal{F}} = Z^+_{\mathcal{F}} \cup \{E\} = \{A, B, C, E\}$.

Closure of a set of attributes

Example

- We're given a table with four columns: A, B, C, D.
- We're given the following set $\mathcal{F}$ of functional dependencies:

$$A \to BC,\ E \to CF,\ B \to E,\ CD \to EF.$$

- We compute $Z^+_{\mathcal{F}}$, where $Z = \{A, B\}$.
- We initialize $Z^+_{\mathcal{F}} = \{A, B\}$.

First iteration — Fourth FD

- $Z^+_{\mathcal{F}} = \{A, B, C, E\}$
- We consider the FD $CD \to EF$.
- $\{CD\} \not\subseteq Z^+_{\mathcal{F}}$; then $Z^+_{\mathcal{F}} = \{A, B, C, E\}$.

Closure of a set of attributes

Example

- We're given a table with four columns: A, B, C, D.
- We're given the following set $\mathcal{F}$ of functional dependencies:

$$A \to BC,\ E \to CF,\ B \to E,\ CD \to EF.$$

- We compute $Z^+_{\mathcal{F}}$, where $Z = \{A, B\}$.
- We initialize $Z^+_{\mathcal{F}} = \{A, B\}$.

Second iteration — First FD

- $Z^+_{\mathcal{F}} = \{A, B, C, E\}$
- We consider the FD $A \to BC$.
- $\{A\} \subset Z^+_{\mathcal{F}}$; then $Z^+_{\mathcal{F}} \cup \{B, C\} = \{A, B, C, E\}$.

Closure of a set of attributes

Example

- We're given a table with four columns: A, B, C, D.
- We're given the following set $\mathcal{F}$ of functional dependencies:

$$A \to BC,\ E \to CF,\ B \to E,\ CD \to EF.$$

- We compute $Z^+_{\mathcal{F}}$, where $Z = \{A, B\}$.
- We initialize $Z^+_{\mathcal{F}} = \{A, B\}$.

Second iteration — Second FD

- $Z^+_{\mathcal{F}} = \{A, B, C, E\}$
- We consider the FD $E \to CF$.
- $\{E\} \subset Z^+_{\mathcal{F}}$; then $Z^+_{\mathcal{F}} \cup \{C, F\} = \{A, B, C, E, F\}$.

Closure of a set of attributes

Example

- We're given a table with four columns: A, B, C, D.
- We're given the following set $\mathcal{F}$ of functional dependencies:

$$A \to BC,\ E \to CF,\ B \to E,\ CD \to EF.$$

- We compute $Z^+_{\mathcal{F}}$, where $Z = \{A, B\}$.
- We initialize $Z^+_{\mathcal{F}} = \{A, B\}$.

Second iteration — Third FD

- $Z^+_{\mathcal{F}} = \{A, B, C, E\}$
- We consider the FD $B \to E$.
- $\{B\} \subset Z^+_{\mathcal{F}}$; then $Z^+_{\mathcal{F}} \cup \{E\} = \{A, B, C, E, F\}$.

Closure of a set of attributes

Example

- We're given a table with four columns: A, B, C, D.
- We're given the following set $\mathcal{F}$ of functional dependencies:

$$A \to BC,\ E \to CF,\ B \to E,\ CD \to EF.$$

- We compute $Z^+_{\mathcal{F}}$, where $Z = \{A, B\}$.
- We initialize $Z^+_{\mathcal{F}} = \{A, B\}$.

Second iteration — Fourth FD

- $Z^+_{\mathcal{F}} = \{A, B, C, E\}$
- We consider the FD $CD \to EF$.
- $\{CD\} \not\subseteq Z^+_{\mathcal{F}}$; then $Z^+_{\mathcal{F}} = \{A, B, C, E, F\}$.

Closure of a set of attributes

Example

- We're given a table with four columns: A, B, C, D.
- We're given the following set $\mathcal{F}$ of functional dependencies:

$$A \to BC,\ E \to CF,\ B \to E,\ CD \to EF.$$

- We compute $Z^+_{\mathcal{F}}$, where $Z = \{A, B\}$.
- We initialize $Z^+_{\mathcal{F}} = \{A, B\}$.

Third iteration

- We loop over the four FDs again, but this time $Z^+_{\mathcal{F}}$ won't change.
- We exit the loop.

Closure of a set of attributes

First corollary

Given a set $\mathcal{F}$ of FDs, a specific FD $X \to Y$ follows from $\mathcal{F}$ (i.e., it is **redundant**) if and only if $Y \subseteq X^+_{\mathcal{F}}$.

- If K is a **superkey**, then: $K \to A$ for each column A of the table (from the definition).

Second corollary

A set of columns K is a **superkey** if and only if, given a set of functional dependencies $\mathcal{F}$, $K^+_{\mathcal{F}}$ is the set of all columns of the table.

☞ Given a set $\mathcal{F}$ of FDs, how do we find a set of FDs $\mathcal{G}$ that is **equivalent** and **minimal** (or, **irreducible**)?

Equivalence

Definition (Equivalence)

Given two sets of FDs $\mathcal{F}$ and $\mathcal{G}$, $\mathcal{F}$ and $\mathcal{G}$ are **equivalent** if and only if $\mathcal{F}^+ = \mathcal{G}^+$.

Theorem

$\mathcal{F}^+ = \mathcal{G}^+$ if and only if for any FD $X \to Y \in \mathcal{F} \cup \mathcal{G}$, we have $\mathcal{F}_X^+ = \mathcal{G}_X^+$

Proof

- Let $F' = \{X \to F_X^+ \text{ s.t. } X \to Y \in F \cup G\}$
- Let $G' = \{X \to G_X^+ \text{ s.t. } X \to Y \in F \cup G\}$
- Clearly, $F'^+ = F^+$ and $G'^+ = G^+$.
- If $F^+ = G^+ \implies F' = G' \implies F_X^+ = G_X^+$
- If $F_X^+ + = G_X^+ \implies F'^+ = G'^+ \implies F^+ = G^+$

Minimality

Definition (Minimality)

A set of FDs $\mathcal{F}$ is **minimal** if and only if the following is true:

- Every FD $X \to Y$ is in **canonical form**.
  - Y consists of exactly one attribute.
- Every FD $X \to Y$ in $\mathcal{F}$ is **left-irreducible**.
  - No attribute can be removed from X without changing $\mathcal{F}^+$.
- Every FD $X \to Y$ in $\mathcal{F}$ is **non-redundant**
  - $X \to Y$ cannot be removed from $\mathcal{F}$ without changing $\mathcal{F}^+$.

☞ This definition gives us a procedure to minimize a set of functional dependencies. We'll take a closer look at this procedure in *Tutorial 2*.

First Normal Form — 1NF

Definition (First Normal Form — 1NF)

A **table** is in **first normal form** if it meets the following conditions:

1. Each row is unique (identified by a **primary key**).
2. There are **no duplicate columns**.
3. Each cell contains a **single value** (lists are not allowed).

First Normal Form — 1NF

Question. The table Book is not in 1NF. Why?

Example

```
Book (book_id, isbn, title, type, year, authors,
      publisher_name, publisher_country)

("book-001", 978-0321197849, "An introduction to database systems",
 "Hardcover", 1999, "Christopher J. Date, UK", "Addison-Wesley", "US" )

("book-001", 978-8178082318, "An introduction to database systems",
 "Paperback", 1999, "Christopher J. Date, UK", "Addison-Wesley", "US" )

("book-002", 978-3446215337, "Data Mining",
 "Paperback", 2001, "Ian H. Witten, New Zealand"; "Frank Eibe, Germany",
 "Hanser Fachbuch", "Germany" )
```

First Normal Form — 1NF

Question. The table Book is not in 1NF. Why?

- **Answer.** Column *authors* contains a list of values.

Example

```
Book (book_id, isbn, title, type, year, authors,
      publisher_name, publisher_country)

("book-001", 978-0321197849, "An introduction to database systems",
 "Hardcover", 1999, "Christopher J. Date, UK", "Addison-Wesley", "US" )

("book-001", 978-8178082318, "An introduction to database systems",
 "Paperback", 1999, "Christopher J. Date, UK", "Addison-Wesley", "US" )

("book-002", 978-3446215337, "Data Mining",
 "Paperback", 2001, "Ian H. Witten, New Zealand"; "Frank Eibe, Germany",
 "Hanser Fachbuch", "Germany" )
```

First Normal Form — 1NF

- We create a new table `BookAuthor` that contains the authors of the books.
- The two tables `Book` and `BookAuthor` are in 1NF.
- But there is still a lot of **redundancy**. **Where**?

Example

```
Book       (book_id, isbn, title, type, year, publisher_name,
            publisher_country)
BookAuthor (book_id, aut_id, aut_name, aut_country)
```

First Normal Form — 1NF

- We create a new table `BookAuthor` that contains the authors of the books.
- The two tables `Book` and `BookAuthor` are in 1NF.
- But there is still a lot of **redundancy**. **Where**?
  - title of the book is repeated for each edition.
  - same goes for the publisher name and country.
  - the author country is repeated for each book of that author.

Example

```
Book       (book_id, isbn, title, type, year, publisher_name,
            publisher_country)
BookAuthor (book_id, aut_id, aut_name, aut_country)
```

Second Normal Form — 2NF

Definition (Prime attribute)

A **prime** column is one that belongs to at least one candidate key. Conversely, a **non-prime** column is one that does not belong to any candidate key.

Definition (Second Normal Form — 2NF)

A table is in **second normal form** if it meets the following conditions:

- The table is in 1NF.
- All non-prime columns are functionally dependent on **all** the columns of **each** candidate key.

Second Normal Form — 2NF

Functional dependencies

```
Book       (book_id, isbn, title, type, year, publisher_name,
            publisher_country)
BookAuthor (book_id, aut_id, aut_name, aut_country)
```

Functional dependencies in table `Book` (not exhaustive!):

- *book_id* $\rightarrow$ *title*
- *isbn* $\rightarrow$ *book_id*, *type*, *publisher_name*
- *publisher_name* $\rightarrow$ *publisher_country*

Functional dependencies in table `BookAuthor`:

- *aut_id* $\rightarrow$ *aut_name*, *aut_country*

Second Normal Form — 2NF

- Candidate key in table `Book`: $\{isbn, year\}$.
- Table `Book` is not in 2NF:
  - $isbn \rightarrow title$; $isbn \rightarrow type$; $isbn \rightarrow publisher\_name$.
- **Question:** table `BookAuthor` is not in 2NF. Why?
  - Answer: $aut\_id \rightarrow aut\_name$; $aut\_id \rightarrow aut\_country$.

Example

```
Book       (book_id, isbn, title, type, year, publisher_name,
            publisher_country)
BookAuthor (book_id, aut_id, aut_name, aut_country)
```

Second Normal Form — 2NF

- Candidate key in table `Book`: $\{isbn, year\}$.
- Table `Book` is not in 2NF:
  - $isbn \rightarrow title$; $isbn \rightarrow type$; $isbn \rightarrow publisher\_name$.
- **Question:** table `BookAuthor` is not in 2NF. Why?
  - **Answer.** $aut\_id \rightarrow aut\_name$; $aut\_id \rightarrow aut\_country$.

Example

```
Book       (book_id, isbn, title, type, year, publisher_name,
            publisher_country)
BookAuthor (book_id, aut_id, aut_name, aut_country)
```

Second Normal Form — 2NF

- Candidate key in table `Book`: $\{isbn, year\}$.
- Table `Book` is not in 2NF:
  - $isbn \rightarrow title$; $isbn \rightarrow type$; $isbn \rightarrow publisher\_name$.
- **Question:** table `BookAuthor` is not in 2NF. Why?
  - **Answer.** $aut\_id \rightarrow aut\_name$; $aut\_id \rightarrow aut\_country$.

Example

```
Book       (book_id, isbn, title, type, year, publisher_name,
            publisher_country)
BookAuthor (book_id, aut_id, aut_name, aut_country)
```

Second Normal Form — 2NF

- Candidate key in table `Book`: $\{isbn, year\}$.
- Table `Book` is not in 2NF:
  - $isbn \to title$; $isbn \to type$; $isbn \to publisher\_name$.
- **Question:** table `BookAuthor` is not in 2NF. Why?
  - **Answer.** $aut\_id \to aut\_name$; $aut\_id \to aut\_country$.

Example

```
Book       (book_id, isbn, title, type, year, publisher_name,
            publisher_country)
BookAuthor (book_id, aut_id, aut_name, aut_country)
```

☞ How do we turn this database into 2NF?

Second Normal Form (2NF)

- The following tables are in 2NF.
- Still redundant data
  - *publisher_name* $\rightarrow$ *publisher_country*

.

Example

```
Book              (book_id, isbn, title, type, publisher_name,
                   publisher_country)
PublicationYear   (isbn, year)
BookAuthor        (book_id, aut_id)
Author            (aut_id, aut_name, aut_country)
```

Third Normal Form — 3NF

Definition (Third Normal Form — 3NF)

A table is in **third normal form** if it meets the following conditions:

- The table is in 2NF.
- No non-prime column depends on non-prime columns.

The table `Book` is not in 3NF. Why?

Example

`Book` (`book_id`, `isbn`, `title`, `type`, `publisher_name`, `publisher_country`)

Third Normal Form — 3NF

Example

```
Book              (book_id, title)
                  f.d.: book_id --> title
BookEdition       (isbn, book_id, type, publisher_name)
                  f.d.: isbn --> book_id, type, publisher_name
Publisher         (publisher_name, publisher_country)
                  f.d.: publisher_name --> publisher_country
PublicationYear   (isbn, year)
                  f.d.: none
BookAuthor        (isbn, aut_id)
                  f.d.: none
Author            (aut_id, aut_name, aut_country)
                  f.d.: aut_id --> aut_name, aut_contry
```

Boyce-Codd Normal Form — BCNF

Another way to define 3NF

In a table in third normal form, a non-prime column must provide a fact about the key, use the whole key and nothing but the key (so help me Codd!) — William Kent, 1983

- 3NF still leaves the door open to data redundancy.

Definition (Boyce-Codd Normal Form)

A table is in **Boyce-Codd normal form** (BCNF) if, for every functional dependency $X \to Y$, X is a superkey.

Boyce-Codd Normal Form

Example

`Course (course_name, major_name, lecturer_id)`

- A teacher only teaches one course. ($lecturer\_id \rightarrow course\_name$).
- A course can be taught in two different majors.
- Candidate keys are $\{course\_name, major\_name\}$ and $\{lecturer\_id, major\_name\}$.
- Table `Course` is not in BCNF ($lecturer\_id$ is not a superkey).
- The following example is in BCNF.

Example

`Course` (`course_name`, `lecturer_id`)
`Lecturer` (`lecturer_id`, `major_name`)

Fourth Normal Form

Definition (Fourth Normal Form)

A table is in **fourth normal form** (4NF) if it does not contain two or more independent multi-valued facts about an entity.

Example

```
Employee (emp_id, emp_skill, emp_language)
```

- *emp_skill* and *emp_language* are **multi-valued**.
  - An employee can have many skills and speak many languages.
- *emp_skill* and *emp_language* are **independent**.
  - Skills do not depend on the spoken languages.
- **Uncertainty** as to how the values are stored in the table.

Fourth Normal Form

Disjoint format

```
(1, "programming", NULL)
(1, "network admin", NULL)
(1, NULL, "French")
(1, NULL, "English")
(1, NULL, "German")
```

Random mix format

```
(1, "programming", "French")
(1, "network admin", "English")
(1, "network admin", "German")
```

Cross-product format

```
(1, "programming", "French")    (1, "network admin", "French")
(1, "programming", "English")   (1, "network admin", "English")
(1, "programming", "German")    (1, "network admin", "German")
```

Tables in 4NF

EmployeeSkill (<u>emp_id</u>, emp_skill)
EmployeeLanguage (<u>emp_id</u>, emp_language)

Fifth Normal Form

Definition (Fifth Normal Form)

A table is in **fifth normal form** (5NF) if its information content cannot be reconstructed from several smaller tables.

Example

`SaleRepresentative (agent, company, product)`

1. Companies make products.
2. Agents represent companies.
3. An agent representing a company sells all products made by that company.

The columns *company* and *product* are not independent. The table is in 4NF.

Fifth Normal Form

Content of the table SaleRepresentative

```
("Smith", "Ford", "car")     ("Jones", "Ford", "car")
("Smith", "Ford", "truck")   ("Jones", "Ford", "truck")
("Smith", "GM", "car")       ("Brown", "Toyota", "car")
("Smith", "GM", "truck")     ("Brown", "Toyota", "bus")
```

Tables in 5NF

```
AgentCompany   (agent, company)
CompanyProduct (company, product)
```

Content of AgentCompany

```
("Smith", "Ford")
("Smith", "GM")
("Jones", "Ford")
("Brown", "Toyota")
```

Content of CompanyProduct

```
("Ford", "car")
("Ford", "truck")
("GM", "car")
("GM", "truck")
("Toyota", "car")
("Toyota", "bus")
```

References

- Date, Christopher John. *An introduction to database systems*. Pearson Education India, 2004. ▸ Click here