DATABASE METHODOLOGY

Stockholm University

# Designing Relational Databases

# Normalization

(also called analytic database design)

# Normalization

- **In this module you will learn some basics about normalization – ensuring high quality logical RDB designs**
    - Normalization defined
    - Normal forms (1NF, 2NF, 3NF)
    - Functional Dependencies
    - Stepwise normalization method
    - Update anomalies (data anomalies)

# Normalization defined

- **“A technique for producing a set of relations with desirable properties, given the data requirements of an enterprise.”** **Connolly / Begg, “Database Systems”**

- Often used as a verification method following the logical RDB design.

# Why Normalization

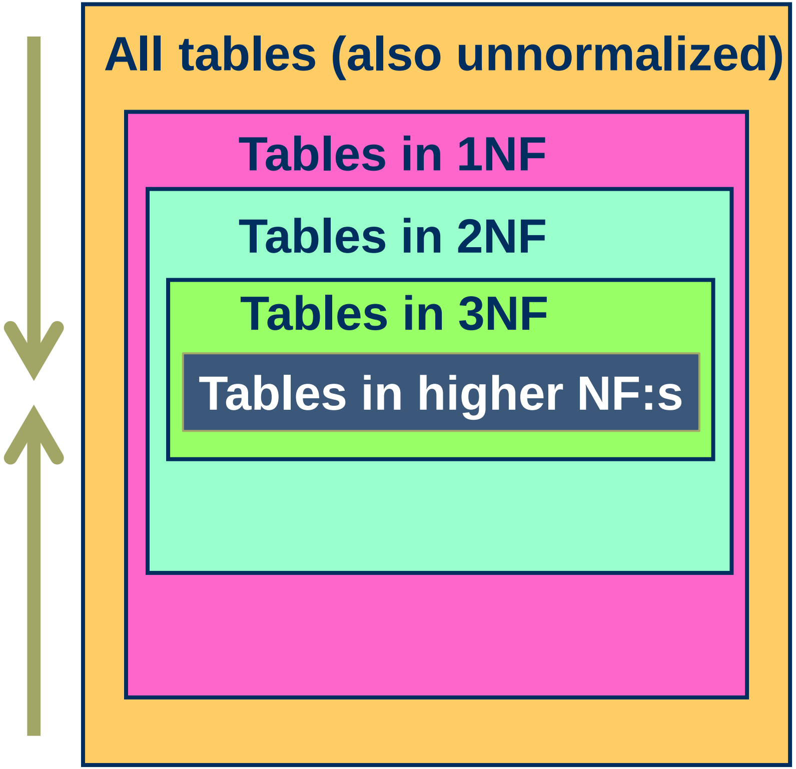
- **The Goal:**
  - To store each data item in **just one place**
    - **Benefits:**
      - The required disk space is minimized
        - » Lower cost for storing the data
      - Update anomalies are avoided
        - » Higher data quality
          - **More about this later**

# Normalization In Practice

- Find so called ***functional dependencies*** (FDs) that are not handled correctly in the current design.
- Move these FDs into their own tables
  - – leave FK:s in the original tables
    - important in order not to lose information.
  - – The so called **determinants** of the FDs (more about this later!) become the PKs in the new tables

# Normal Forms

- **Normalization is performed stepwise**
  - From lower Normal Forms (NFs) to higher
  - The most important are 1NF, 2NF, 3NF
    - The higher forms are not covered in this course



# Functional Dependencies

- A functional dependency (FD) in normalization takes the following basic form:
  - **A → B**, where A is a set of columns (perhaps only one), and B is a set of columns (perhaps only one)
  - It all means that if the row values in the columns in A are known, then we can find the row values in the columns in B.
  - We say that A ***determines*** B; A is the FDs ***determinant***

| A | B | C |
|---|---|---|
| 583 | 22 | 1 |
| 819 | 78 | 8 |
| 583 | 22 | 7 |
| 109 | 22 | 8 |

**A → B *seems* to hold in the left table.**

**A → B does not hold in the right table.**

| A | B | C |
|---|---|---|
| 583 | 22 | 1 |
| 819 | 78 | 8 |
| 583 | 32 | 7 |
| 109 | 22 | 8 |

# Functional Dependencies

- **Warning!**
  - By inspecting the contents of a table:
    - we **can falsify** a claim that a functional dependency exists
    - but we **cannot prove** that a functional dependency exists
      - there might be yet un-entered data that will falsify it
      - functional dependencies should be defined by analyzing the part of the world we are modelling
        - » That’s why normalization is also called analytic database design – we analyze which functional dependencies that exist, and make sure we are handling them correctly

# Method: 1NF – First Normal Form

- For a table to be in 1NF, every cell (i.e cross-section of row and column) must have only **one** value(*)
  - We say that all data in the table must be **atomic**
  - Any lists in cells must be **flattened**:

Unnormalized table

| A | B | C |
|---|---|---|
| 45 | 32, 33, 90 | 61 |
| 82 | 27 | 2 |
| 871 | 188 | 1002 |

The table is now in 1NF

| A | B | C |
|---|---|---|
| 45 | 32 | 61 |
| 45 | 33 | 61 |
| 45 | 90 | 61 |
| 82 | 27 | 2 |
| 871 | 188 | 1002 |

- (*) The table must also have a name and a PK

# Method: 2NF – Second Normal Form

- For a table to be in 2NF, it must be in 1NF, **and** every column that is not a part of the PK, must be **fully** functionally dependent on the PK
  - It must **not** be sufficient with a part of the PK to maintain the functional dependency (a composite PK is necessary!)

A table in 1NF, but not 2NF, ColB alone determines ColC.

| ColA | ColB | ColC | ColD |
|---|---|---|---|
| Kim | 2002 | 36 | 89 |
| Mel | 2002 | 36 | 45 |
| Mel | 2009 | 33 | 56 |
| Jim | 2009 | 33 | 09 |
| Ian | 2002 | 36 | 67 |
| Ian | 2004 | 36 | 76 |

| ColA | ColB | ColD |
|---|---|---|
| Kim | 2002 | 89 |
| Mel | 2002 | 45 |
| Mel | 2009 | 56 |
| Jim | 2009 | 09 |
| Ian | 2002 | 67 |
| Ian | 2004 | 76 |

The tables are now in 2NF. ColB in the original table is now an FK to ColB in the new table.

| ColB | ColC |
|---|---|
| 2002 | 36 |
| 2009 | 33 |
| 2004 | 36 |

**ColB → ColC now has its own table**

# Method: 3NF – Third Normal Form

- For a table to be in 3NF, it must be in 2NF, **and** every column that is not a part of the PK, must **only be directly** functionally dependent on the PK
  - There must not be any non-PK column that transitively determines other non-PK columns

A table in 2NF, but not 3NF,
ColC transitively determines ColD.

| ColA | ColB | ColC | ColD |
|---|---|---|---|
| Wof | 2002 | 101 | 95 |
| Dig | 2002 | 77 | 45 |
| Del | 2009 | 77 | 45 |
| Wof | 2009 | 17 | 89 |
| Del | 2002 | 101 | 95 |
| Sno | 2004 | 77 | 45 |

| ColA | ColB | ColC |
|---|---|---|
| Wof | 2002 | 101 |
| Dig | 2002 | 77 |
| Del | 2009 | 77 |
| Wof | 2009 | 17 |
| Del | 2002 | 101 |
| Sno | 2004 | 77 |

The tables are now in 3NF. ColC in the original table is now an FK to ColC in the new table.

| ColC | ColD |
|---|---|
| 101 | 95 |
| 77 | 45 |
| 17 | 89 |

**ColC → ColD now has its own table**

# Normalization Method - Summary

- For each table in the database:
  Work **stepwise** from unnormalized (0NF) to 3NF
  - **0NF to 1NF:**
    - Make sure that all cells have atomic values (no lists)
    - Make sure the table has a name and a PK assigned
  - **1NF to 2NF:**
    - Eliminate **partial** functional dependencies, where non-PK columns are **not fully** dependent of the whole PK, by creating new tables as necessary and leaving FKs in the original table
  - **2NF to 3NF:**
    - Eliminate **transitive** functional dependencies, where non-PK columns are **not only** dependent directly of the whole PK, **but also** via some other non-PK column(s), by creating new tables as necessary, and leaving FKs in the original table

# Update Anomalies – Poor Normalization

- **Insertion anomalies:**
  - Say we need to insert the ColC-value for the ColB-value 2005. Then we at least must also enter a ColA-value, since ColA cannot be NULL (it is part of the PK). **What value?**
- **Deletion anomalies:**
  - If we delete the row with the composite PK value Ian + 2004, then we lose the information that the ColC-value for the ColB value 2004 is 36.
- **Update anomalies:**
  - What if the ColC value for ColB = 2002 changes? Then we need to update the ColC value for all rows where ColB = 2002

Table not in 2NF (**ColB → ColC**)

| ColA | ColB | ColC | ColD |
|---|---|---|---|
| Kim | 2002 | 36 | 89 |
| Mel | 2002 | 36 | 45 |
| Mel | 2009 | 33 | 56 |
| Jim | 2009 | 33 | 09 |
| Ian | 2002 | 36 | 67 |
| Ian | 2004 | 36 | 76 |

**Solution:**

**Next slide!**

# Update Anomalies – Good Normalization

- **Insertion anomalies:**
  - Say we need to insert the ColC-value for the ColB-value 2005.
    - **Just insert a new row into the new table!**
- **Deletion anomalies:**
  - Delete the row with the composite PK value Ian + 2004.
    - **The info about ColB = 2004 is still there in the new table!**
- **Update anomalies:**
  - What if the ColC value for ColB = 2002 changes?
    - **We can change it in one single place in the new table!**

| ColA | ColB | ColD |
|---|---|---|
| Kim | 2002 | 89 |
| Mel | 2002 | 45 |
| Mel | 2009 | 56 |
| Jim | 2009 | 09 |
| Ian | 2002 | 67 |

| ColB | ColC |
|---|---|
| 2002 | 68 |
| 2009 | 33 |
| 2004 | 36 |
| 2005 | 71 |

# Normalization

- **In this module you learnt some basics about normalization, a technique for ensuring high quality logical RDB designs**
  - We defined normalization
  - Talked about Normal forms (1NF, 2NF, 3NF)
  - And Functional Dependencies
  - We showed a stepwise normalization method
  - And explained update anomalies (data anomalies)

## Medverkande

Anders Thelemyr – Lärare

Lars In de Betou – Mediepedagog

Inspelat 2015-09-03
Institutionen för data- och systemvetenskap, DSV

Stockholms universitet