For Model Design: Normalize – Denormalize

MANOJ, NIIT

What is Data Normalization

- Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that ***avoid unnecessary duplication of data***
- The process of decomposing relations with anomalies to produce smaller, ***well-structured*** relations

Results of Normalization

- Removes the following modification *anomalies* (integrity errors) with the database:
  - Insertion
  - Deletion
  - Update

ANOMALIES

- An anomaly is an irregularity, or something which deviates from the expected or normal state.
- When designing databases, we identify three types of anomalies: Insert, Update and Delete.
- **Insertion**
  - inserting one fact in the database requires knowledge of other facts unrelated to the fact being inserted
- **Deletion**
  - Deleting one fact from the database causes loss of other unrelated data from the database
- **Update**
  - Updating the values of one fact requires multiple changes to the database

ANOMALIES EXAMPLES
TABLE: COURSE

COURSE#	SECTION#	C_NAME
CIS564	072	Database Design
CIS564	073	Database Design
CIS570	072	Oracle Forms
CIS564	074	Database Design

ANOMALIES EXAMPLES

Insertion:

Suppose our university has approved a new course called CIS563: SQL & PL/SQL.

Can this information about the new course be entered (inserted) into the table COURSE in its present form?

COURSE#	SECTION#	C_NAME
CIS564	072	Database Design
CIS564	073	Database Design
CIS570	072	Oracle Forms
CIS564	074	Database Design

ANOMALIES EXAMPLES

Deletion:

Suppose not enough students enrolled for the course CIS570 which had only one section 072. So, the school decided to drop this section and delete the section# 072 for CIS570 from the table COURSE. But then, what other relevant info also got deleted in the process?

COURSE#	SECTION#	C_NAME
CIS564	072	Database Design
CIS564	073	Database Design
CIS570	072	Oracle Forms
CIS564	074	Database Design

ANOMALIES EXAMPLES

Update:

Suppose the course name (C_Name) for CIS 564 got changed to Database Management. How many times do you have to make this change in the COURSE table in its current form?

COURSE#	SECTION#	C_NAME
CIS564	072	Database Design
CIS564	073	Database Design
CIS570	072	Oracle Forms
CIS564	074	Database Design

ANOMALIES

- So, a table (relation) is a stable (‘good’) table only if it is free from any of these anomalies at any point in time.
- You have to ensure that each and every table in a database is always free from these modification anomalies. And, how do you ensure that?
- ‘Normalization’ theory helps.

NORMAL FORMS (Types)

- ✓ 1 NF
- ✓ 2NF
- ✓ 3NF
- BCNF (Boyce-Codd Normal Form)
- 4NF
- 5NF
- *DK (Domain-Key) NF*

✓ *Mark, Indicates More Imp to Cover-in Detailed*

Functional Dependency

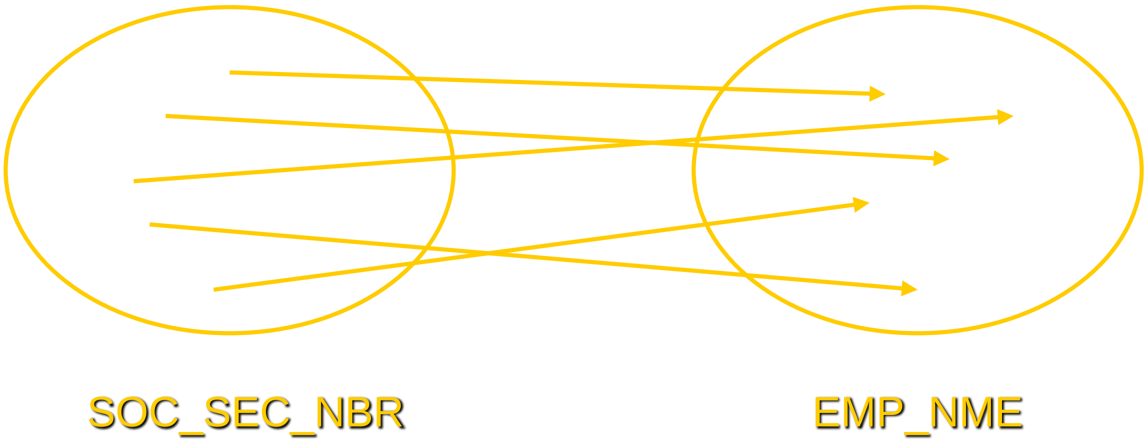
- Relationship between columns X and Y such that, given the value of X, one can *determine* the value of Y. Written as X   Y

i.e., for a given value of X we can obtain (or look up) a specific value of X

- X is called the *determinant* of Y
- Y is said to be *functionally dependent* on X

Functional Dependency

- Example
  - SOC_SEC_NBR → EMP_NME



-One and only one EMP_NME for a specific SOC_SEC_NBR

- SOC_SEC_NBR is the *determinant* of EMP_NME

- EMP_NME is functionally *dependent* on SOC_SEC_NBR

1 NF

A table is in 1NF if there are no repeating groups in the table. In other words, a table is in 1NF if all non-key fields are functionally dependent on the primary key (PK). That is, for each given value of PK, we always get only one value of the non-key field(s).

Is the following table COURSE in 1NF?

Course

COURSE#	SECTION#	C_NAME
CIS564	072	Database Design
CIS564	073	Database Design
CIS570	072	Oracle Forms
CIS564	074	Database Design

1NF

But, didn't we just conclude that COURSE is a 'bad' table (the way it is structured) as it suffers from all the three anomalies we talked about?

So, what's the problem?

COURSE#	SECTION#	C_NAME
CIS564	072	Database Design
CIS564	073	Database Design
CIS570	072	Oracle Forms
CIS564	074	Database Design

Partial Dependency

- Occurs when a column in a table only depends on part of a concatenated key

Example

COURSE (<u>COURSE# + SECTION#</u>, C-NAME

• C_Name only depends upon the Course# not the Section#. It is partially dependent upon the primary key.

• A table is in 2NF if it is in 1NF and has no partial dependencies.

2NF

- How do you resolve partial dependency?

  • Decompose the problematic table into smaller tables.
  • Must be a ‘loss-less’ decomposition. That is, you must be able to put the decomposed tables back together again to arrive at the original information.
  • Remember *Foreign Keys*!

2NF

OFFERED_COURSE

COURSE#	SECTION#
CIS564	072
CIS564	073
CIS564	074
CIS570	072

COURSE

COURSE#	C_NAME
CIS564	Database Design
CIS570	Oracle Forms

2NF

- Are the two (decomposed) tables COURSE and OFFEERED_COURSE are 2NF?
- Do these two tables have any modification anomalies?
  - Can you now readily enter the info that a new approved course CIS563?
  - Can you now delete the section# 072 for CIS570 without losing the info tat CIS570 exists?
  - How many times do you have to change the name of a given course?

Transitive Dependency

Table: Student-Dorm-Fee

SID	DORM	FEE
101	Oracle	1000
102	Oracle	1000
103	DB2	800
104	DB2	800
105	Sybase	500

Transitive Dependency

- Is the table Student-Dorm-Fee in 2NF?
- Does this table have any modification anomalies?
  - Insertion?
  - Deletion?
  - Update?

Transitive Dependency

•Occurs when a non-key attribute is functionally dependent on one or more non-key attributes.

Example: HOUSING (<u>SID</u>, DORM, FEE)

PRIMARY KEY: SID

FUNCTIONAL DEPENDENCIES:

SID	→	BUILDING
SID	→	FEE
DORM	→	FEE

• A table is in 3NF if it is in 2NF and has no transitive dependencies

3NF

- Besides SID, FEE is also functionally dependent on DORM which is a non-key attribute.

- A table is in 3NF if it is in 2NF and has no transitive Dependencies.

3NF

- How do you resolve transitive dependency?

  - Decompose the problematic table into smaller tables.
  - Must be a ‘loss-less’ decomposition. That is, you must be able to put the decomposed tables back together again to arrive at the original information.
  - Remember *Foreign Keys*!

3NF

STUDENT_DORM

SID	DORM
101	Oracle
102	Oracle
103	DB2
104	DB2
105	Sybase

DOM_FEE

DORM	FEE
Oracle	1000
DB2	800
Sybase	500

3NF

- Are the two (decomposed) tables STUDENT_DORM and DORM_FEE in 2NF?
- Are they in 3NF?
- Do these two tables have any modification anomalies?

Data Analyst's Oath

EVERY NON-KEY COLUMN IN A TABLE MUST BE *FUNCTIONALLY DEPENDENT* UPON THE *ENTIRE* KEY AND *NOTHING* BUT THE KEY!

Other Normal Forms

- There are additional normal forms which do not often occur in actual practice. However, these situations *can* occur in practice so it is necessary to understand them. These are:
  - Boyce-Codd Normal Form
  - Fourth Normal Form
  - Fifth Normal Form
- We will deal with these normal forms if time allows. You must, however, fully understand 1ST through 3RD NF.
- Domain/Key normal form is a different approach and we will not deal with it in this course.

Practical Problems of Normalization

- **Derivable Data:** As **"**Rule of thumb", Do NOT include derivable (computable) data in the baseline *Logical* database design schema. - sometimes derivable data in your design, mainly to enhance the performance of your application
- **No calculated values**. Calculated values minimise burden at Runtime, but a normalized database lacks them. Denormalization is one such approach
- **Problem of Multi-Join: I**t daunting to pull together everything needed for a certain query. A query joining 4,5, 7 or even 12 tables may be required, sometimes hard to code at runtime, hard to debug, and dangerous to alter.
- **Performance.** When face Multi-Join jungle you always face performance problems. A JOIN is a very expensive operation on Multi-tables

What is De-Normalization

- De-Normalize, Overcome Normalize Issues in terms of Performance
- De-normalization is the process of attempting to optimize the read performance of a database by adding **redundant** data or by grouping data.
- In many cases, de-normalization will address **performance** or **scalability** in relational databases
- De-normalizing means **adding columns** to tables that provide values you would otherwise have to calculate as needed.
- Calculations are made within a row, and **totals**, **averages** and other **aggregations** are made between child and parent tables.

When decide to use de-normalize

- When you need to analyze the data access requirements of the applications in your environment and their actual performance characteristics.
- **Considering the following, when decide on De-normalization:**
  - *What are the critical transactions, and response time?*
  - *How often are the transactions executed? How many rows do they access each time?*
  - *How big are the most frequently accessed tables?*
  - *Do any processes compute summaries?*
  - *Where is the data physically located?*

De-normalization Techniques

- The most prevalent De-normalization techniques are:
  - Adding redundant columns
  - Adding derived columns
  - Collapsing (or Combining) tables

De-normalization Techniques

- De-normalization can improve performance by:
  - *Minimizing the need for joins*
  - *Reducing the number of foreign keys on tables*
  - *Reducing the number of indexes, saving storage space and reducing data modification time*
  - *Pre-computing aggregate values, that is, computing them at data modification time rather than at select time*
  - *Reducing the number of tables (in some cases)*

Disadvantages of Denormalization

- It usually speeds retrieval, but can slow data modification (Updates)
- It is always application-specific and needs to be re-evaluated if the application changes.
- It can increase the size of tables.
- More Memory to process Big Size Rows
- In some instances, it simplifies coding; in others, it makes coding more complex.

General Comparison btw Normalize to De-Normalize

Normalize Vs Denormalize

Normalize	Denormalize
• FAST RESPONSE	• FAST RESPONSE
• FAST UPDATE	• SLOW UPDATE
• EFFICIENT STORAGE	• IN-EFFICIENT STORAGE
• RELATIONAL MODELS (TYPICALLY)	• MULTI-DIMENSIONAL MODELS (TYPICALLY)
• THIRD NORMAL FORM OR HIGHER	• THIRD NORMAL FORM OR LOWER