IIT Madras
BSc Degree

Database Management Systems

Module 21: Relational Database Design/1

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ac.in

Week Recap

- Discussed relational algebra with examples
- Introduced tuple relational and domain relational calculus
- Illustrated equivalence of algebra and calculus
- Introduced the Design Process for Database Systems
- Elucidated the E-R Model for real world representation with entities, entity sets, attributes, and relationships
- Illustrated ER Diagram notation for ER Models
- Discussed translation of ER Models to Relational Schema and extended features of ER Model
- Deliberated on various design issues

Module Objectives

- To identify the features of good relational design
- To familiarize with the First Normal Form

Module Outline

- Features of Good Relational Design
- Atomic Domains and First Normal Form

Features of Good Relational Design

Good Relational Design

- Reflects *real-world structure* of the problem
- Can represent *all expected data* over time
- Avoids *redundant storage* of data items
- Provides *efficient access* to data
- Supports the *maintenance of data integrity* over time
- *Clean, consistent, and easy* to understand
- Note: *These objectives are sometimes contradictory!*

What is a Good Schema?

instructor_with_department

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

- *ID*: Key
- *building*, *budget*: Redundant Information
- *name*, *salary*, *dept_name*: No Redundant Information

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

department

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

What is a Good Schema? (2)

- Consider combining relations
  - $sec\_class(\underline{sec\_id}, building, room\_number)$ and
  - $section(\underline{course\_id}, \underline{sec\_id}, semester, year)$

  into one relation
  - $section(\underline{course\_id}, \underline{sec\_id}, semester, year, building, room\_number)$
- No repetition in this case

Redundancy and Anomaly

- **Redundancy**: having multiple copies of same data in the database.
  - This problem arises when a database is not normalized
  - It leads to anomalies
- **Anomaly**: inconsistencies that can arise due to data changes in a database with insertion, deletion, and update
  - These problems occur in poorly planned, un-normalised databases where all the data is stored in one table (a flat-file database)

  There can be three kinds of anomalies
  - *Insertions Anomaly*
  - *Deletion Anomaly*
  - *Update Anomaly*

Redundancy and Anomaly (2)

- **Insertions Anomaly**
  - When the insertion of a data record is not possible without adding some additional unrelated data to the record
  - We cannot add an Instructor in *instructor_with_department* if the *department* does not have a *building* or *budget*
- **Deletion Anomaly**
  - When deletion of a data record results in losing some unrelated information that was stored as part of the record that was deleted from a table
  - We delete the last Instructor of a Department from *instructor_with_department*, we lose *building* and *budget* information
- **Update Anomaly**
  - When a data is changed, which could involve many records having to be changed, leading to the possibility of some changes being made incorrectly
  - When the *budget* changes for a Department having large number of Instructors in *instructor_with_department* application may miss some of them

Redundancy and Anomaly (3)

- We have observed the following:
  - **Redundancy $\Rightarrow$ Anomaly**
  - Relations *instructor* and *department* is better than *instructor_with_department*
- What causes redundancy?
  - **Dependency $\Rightarrow$ Redundancy**
  - *dept_name* uniquely decides *building* and *budget*. A department cannot have two different budget or building. So *building* and *budget* **depends on** *dept_name*
- How to remove, or at least minimize, redundancy?
  - Decompose (partition) the relation into smaller relations
  - *instructor_with_department* can be decomposed into *instructor* and *department*
  - **Good Decomposition $\Rightarrow$ Minimization of Dependency**
- Is every decomposition good?
  - No. It needs to preserve information, honour the dependencies, be efficient etc.
  - Various schemes of normalization ensure good decomposition
  - **Normalization $\Rightarrow$ Good Decomposition**

Decomposition

- Suppose we had started with *inst_dept*. How would we know to split up (**decompose**) it into *instructor* and *department*?
- Write a rule "if there were a schema *(dept_name, building, budget)*, then *dept_name* would be a candidate key"
- Denote as a **functional dependency**: $dept\_name \rightarrow building, budget$
- In *inst_dept*, because *dept_name* is not a candidate key, the *building* and *budget* of a *department* may have to be repeated.
  - This indicates the need to decompose *inst_dept*

Decomposition (2)

- Not all decompositions are good
- Suppose we decompose
  employee(ID, name, street, city, salary) into
  - *employee1 (<u>ID</u>, name)*
  - *employee2 (name, street, city, salary)*
- Note that if *name* can be duplicate, then *employee*2 is a weak entity set and cannot exist without an identifying relationship
- Consequently, this decomposition cannot preserve the information
- The next slide shows how we lose information – we cannot reconstruct the *original employee* relation – and so, this is a **lossy decomposition**.

Decomposition (3): Lossy Decomposition

ID	*name*	*street*	*city*	*salary*
⋮				
57766	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

employee

ID	*name*
⋮	
57766	Kim
98776	Kim
⋮	

name	*street*	*city*	*salary*
⋮			
Kim	Main	Perryridge	75000
Kim	North	Hampton	67000
⋮			

Duplicate names cause the problem here.

natural join

ID	*name*	*street*	*city*	*salary*
⋮				
57766	Kim	Main	Perryridge	75000
57766	Kim	North	Hampton	67000
98776	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

Decomposition (4): Lossless-Join Decomposition

- **Lossless Join Decomposition**
- Decomposition of $R = (A, B, C)$
  $R_1 = (A, B),\ R_2 = (B, C)$

r

A	B	C
α	1	A
β	2	B

$\prod_{A,B}(r)$

A	B
α	1
β	2

$\prod_{B,C}(r)$

B	C
1	A
2	B

$\prod_{A,B}(r) \bowtie \prod_{B,C}(r)$

A	B	C
α	1	A
β	2	B

Decomposition (5): Lossless-Join Decomposition

- **Lossless Join Decomposition** is a decomposition of a relation R into relations R_1, R_2 such that if we perform natural join of two smaller relations it will return the original relation

$$R_1 \cup R_2 = R, R_1 \cap R_2 \neq \phi$$

$$\forall r \in R, r_1 = \sqcap_{R_1}(r), r_2 = \sqcap_{R_2}(r)$$

$$r_1 \bowtie r_2 = r$$

- This is effective in removing redundancy from databases while preserving the original data
- In other words by lossless decomposition it becomes feasible to reconstruct the relation R from decomposed tables R_1 and R_2 by using Joins

Atomic Domains and First Normal Form

First Normal Form (1NF)

- A domain is **atomic** if its elements are considered to be indivisible units
  - Examples of non-atomic domains:
    - Set of names, composite attributes
    - Identification numbers like CS101 that can be broken up into parts
- A relational schema R is in **First Normal Form (INF)** if
  - the domains of all attributes of R are *atomic*
  - the value of each attribute contains only a *single value* from that domain
- Non-atomic values complicate storage and encourage redundant (repeated) storage of data
  - Example: Set of accounts stored with each customer, and set of owners stored with each account
  - *We assume all relations are in first normal form*

First Normal Form (2)

- *Atomicity* is actually a property of how the elements of the domain are used
  - Strings would normally be considered indivisible
  - Suppose that students are given roll numbers which are strings of the form *CS0012* or *EE1127*
  - **If the first two characters are extracted to find the department, the domain of roll numbers is not atomic**
  - *Doing so is a bad idea*
    - Leads to encoding of information in application program rather than in the database

First Normal Form (3)

- The following is not in 1NF

Customer

Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025, 192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53; 182-929-2929
789	John	Doe	555-808-9633

  - A telephone number is composite
  - Telephone number is multi-valued

First Normal Form (4)

- Consider:

Customer

Customer ID	First Name	Surname	Telephone Number1	Telephone Number2
123	Pooja	Singh	555-861-2025	192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53	182-929-2929
789	John	Doe	555-808-9633	

  - is in 1NF if telephone number is not considered composite
  - However, conceptually, we have two attributes for the same concept
    - Arbitrary and meaningless ordering of attributes
    - How to search telephone numbers
    - Why only two numbers?

First Normal Form (5)

- Is the following in 1NF?

Customer

Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025
123	Pooja	Singh	192-122-1111
456	San	Zhang	182-929-2929
456	San	Zhang	(555) 403-1659 Ext. 53
789	John	Doe	555-808-9633

  - Duplicated information
  - ID is no more the key. Key is (ID, Telephone Number)

First Normal Form (6)

- Better to have 2 relations:

Customer Name

Customer ID	First Name	Surname
123	Pooja	Singh
456	San	Zhang
789	John	Doe

Customer Telephone Number

Customer ID	Telephone Number
123	555-861-2025
123	192-122-1111
456	(555) 403-1659 Ext. 53
456	182-929-2929
789	555-808-9633

  - One-to-Many relationship between parent and child relations
  - Incidentally, satisfies 2NF and 3NF
- Decomposition helps to attain 1NF for the embedded one-to-many relationship

Module Summary

- Identified the features of good relational design
- Familiarized with the First Normal Form

Slides used in this presentation are borrowed from http://db-book.com/ with kind permission of the authors.
Edited and new slides are marked with "PPD".