AN HONORS UNIVERSITY IN MARYLAND

CMSC 461, Database Management Systems
Spring 2018

Lecture 11 - Chapter 8 Relational Database Design Part 1

These slides are based on “Database System Concepts” 6th edition book and are a modified version of the slides which accompany the book (http://codex.cs.yale.edu/avi/db-book/db6/slide-dir/index.html), in addition to the 2009/2012 CMSC 461 slides by Dr. Kalpakis

Dr. Jennifer Sleeman

Logistics

- HW3 released today due 3/12/2018
- Phase 2 due Wednesday 3/7/2018
- Midterm 3/14/2018

Lecture Outline

- Relational Database Design
- First Normal Form (1NF)
- Functional Dependencies
- Boyce-Codd (BCNF)
- Third Normal Form (intro)

Lecture Outline

- ***Relational Database Design***
- First Normal Form (1NF)
- Functional Dependencies
- Boyce-Codd (BCNF)
- Third Normal Form (intro)

Relational Database Design

- We want to move from the E-R diagram to a set of relations
  - Eliminate redundancy
  - Ensure design is complete
  - Ensure information is easily retrievable
- Going to learn about
  - normal form
  - functional dependencies

Based on and image from "Database System Concepts" book and slides, 6th edition

Design Alternatives: Combining Schemas

- Recall the instructor and department relations

ID	name	dept_name	salary
12121	Wu	Finance	90000.00
76766	Crick	Biology	72000.00

Instructor

dept_name	building	budget
Biology	Watson	90000.00
Comp. Sci.	Taylor	100000.00
Elec. Eng.	Taylor	85000.00
Finance	Painter	120000.00
History	Painter	50000.00
Music	Packard	80000.00
Physics	Watson	70000.00

Department

Based on and image from "Database System Concepts" book and slides, 6th edition

Design Alternatives: Combining Schemas

- Suppose we combine instructor and department into inst_dept
  - No connection to relationship set inst_dept
- Why is this a bad idea?

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

Based on and image from “Database System Concepts” book and slides, 6th edition

Design Alternatives: Combining Schemas

- Repetition of information

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

Based on and image from "Database System Concepts" book and slides, 6th edition

Combining Schemas without Repetition?

- Consider combining relations:

  sec_class(sec_id, building, room_number) and

  section(course_id, sec_id, semester, year)

sec_id	building	room_number
1	Painter	514

course_id	sec_id	semester	year
BIO-101	1	Summer	2009

- into one relation

  section(course_id, sec_id, semester, year, building, room_number)

course_id	sec_id	semester	year	building	room_number
BIO-101	1	Summer	2009	Painter	514

Based on and image from "Database System Concepts" book and slides, 6th edition

Design Alternatives: Smaller schemas

Given inst_dept was the result of our design…..

How would we know to split up (***decompose***) it into *instructor* and *department*?

Based on and image from "Database System Concepts" book and slides, 6th edition

Design Alternatives: Smaller schemas

- What can we say about this data?

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

Based on and image from "Database System Concepts" book and slides, 6th edition

Design Alternatives: Smaller schemas

- Write a rule:

  "if there were a schema (*dept_name, building, budget*), then *dept_name* would be able to serve as the primary key"

Design Alternatives: Smaller schemas

- Denote as a ***functional dependency***:

  $$dept\_name \rightarrow building, budget$$

- In *inst_dept*, because *dept_name* is not a primary key, the building and budget of a department may have to be repeated
  - This indicates the need to decompose *inst_dept*

Based on and image from "Database System Concepts" book and slides, 6th edition

Design Alternatives: Smaller schemas

Not all decompositions are good.

Suppose we decompose:

employee(ID, name, street, city, salary)

into

employee1 (ID, name)
employee2 (name, street, city, salary)

A 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
⋮			

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
⋮				

Based on and image from “Database System Concepts” book and slides, 6th edition

Lossless Join Decomposition

Decomposition of $R = (A, B, C)$

$R_1 = (A, B) \quad R_2 = (B, C)$

A	B	C
α	1	A
β	2	B

r

A	B
α	1
β	2

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

B	C
1	A
2	B

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

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

A	B	C
α	1	A
β	2	B

Based on and image from "Database System Concepts" book and slides, 6th edition

Another Example

- Consider the relation schema:
  - *Lending-schema = (branchName, branchCity, assets, customerName, loanNumber, amount)*

branch-name	*branch-city*	*assets*	*customer-name*	*loan-number*	*amount*
Downtown	Brooklyn	9000000	Jones	L-17	1000
Redwood	Palo Alto	2100000	Smith	L-23	2000
Perryridge	Horseneck	1700000	Hayes	L-15	1500
Downtown	Brooklyn	9000000	Jackson	L-14	1500

Based on and image from "Database System Concepts" book and slides, 6th edition

Another Example

- Redundancy:
  - Data for branchName, branchCity, assets are repeated for each loan that a branch makes
  - Wastes space
  - Complicates updating, introducing possibility of inconsistency of assets value
- Null values
  - Cannot store information about a branch if no loans exist
  - Can use null values, but they are difficult to handle.

Decomposition

Decompose the relation schema Lending-schema into:

BranchSchema = (branchName, branchCity,assets)

LoanSchema = (customerName, loanNumber,branchName, amount)

All attributes of an original schema (R) must appear in the decomposition (R_1,R_2):

$$R = R_1 \cup R_2$$

Lossless-join decomposition.
For all possible relations *r* on schema *R*

$$r = \Pi_{R1}(r) \bowtie \Pi_{R2}(r)$$

Based on and image from "Database System Concepts" book and slides, 6th edition

Lecture Outline

- Relational Database Design
- ***First Normal Form (1NF)***
- Functional Dependencies
- Boyce-Codd (BCNF)
- Third Normal Form

First Normal Form (1NF)

- In the relational model attributes have no substructure
  - composite attributes - each component becomes its own attribute
  - multivalued attributes - one tuple for each item
- 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 CS001 where department is combined with employee number
    - Fine line: ‘CS101’ might be a course identifier and could be interpreted as atomic

First Normal Form (1NF)

- 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

First Normal Form (1NF)

- A relational schema R is in ***first normal form*** if the domains of all attributes of R are atomic

Based on and image from "Database System Concepts" book and slides, 6th edition

First Normal Form (1NF)

- Atomicity is actually a property of how the elements of the domain are used
  - Example: 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

Is it atomic?

- Address
- Course ID (CS-101)
- Student Name
- SSN
- ISBN Number

Convert it to 1NF

ID	Color
1	Blue,Red
2	Yellow,Brown
3	Orange,Green

ID	Name	Address	Order
1	John Doe	13101 Brandley Lane	Cell Phone
2	Jackie Click	4531 Tinker Road	Charger
3	Brad Dunkin	8593 Gerwin Avenue	Cell Phone Case

Based on and image from "Database System Concepts" book and slides, 6th edition

Pitfalls in Relational Database Design

- Relational database design requires that we find a “good” collection of relation schemas
- A bad design may lead to
  - Repetition of Information
  - Inability to represent certain information
- Design Goals:
  - Avoid redundant data
  - Ensure that relationships among attributes are represented
  - Facilitate the checking of updates for violation of database integrity constraints

Based on and image from “Database System Concepts” book and slides, 6th edition

Goal: Devise a Theory for the following:

- Decide whether a particular relation R is in “good” form
- In the case that a relation R is not in “good” form, decompose it into a set of relations $\{R_1, R_2, ..., R_n\}$ such that
  - each relation is in good form
  - the decomposition is a lossless-join decomposition
- This theory is based on:
  - functional dependencies
  - multivalued dependencies

Based on and image from “Database System Concepts” book and slides, 6th edition

Lecture Outline

- Relational Database Design
- First Normal Form (1NF)
- ***Functional Dependencies***
- Boyce-Codd (BCNF)
- Third Normal Form

Functional Dependencies

- Constraints on the set of legal relations
- Require that the value for a certain set of attributes determines uniquely the value for another set of attributes
- A functional dependency is a generalization of the notion of a key

Remember

- *r* is a relation
- *r(R)* is the schema for the relation r
- *R* denotes the set of attributes
- K represents the set of attributes that is the superkey
- A superkey – set of attributes that uniquely identify a tuple

Functional Dependencies

- *Let R be a relation schema*

  $\alpha \subseteq R$ and $\beta \subseteq R$

- *The functional dependency*

  $\alpha \rightarrow \beta$

- *holds on R if and only if for any legal relations r(R), whenever any two tuples t_1 and t_2 of r agree on the attributes **α**, they also agree on the attributes **β**. That is*

  $$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

- *Example: Consider r(A,B) with the following instance of r*

1	4
1	5
3	7

- *On this instance, A⟶B does NOT hold, but B⟶A does hold*

Functional Dependencies

- Functional dependencies allow us to express constraints that cannot be expressed using superkeys. Consider the schema:
  inst_dept (ID, name, salary, dept_name, building, budget)

We expect these functional dependencies to hold:

$dept\_name \rightarrow building$

but would not expect the following to hold:

$dept\_name \rightarrow salary$

Based on and image from "Database System Concepts" book and slides, 6th edition

Functional Dependencies

- We use functional dependencies to:
  - test relations to see if they are legal under a given set of functional dependencies
    - If a relation r is legal under a set F of functional dependencies, we say that r ***satisfies*** F
  - specify constraints on the set of legal relations
- We say that F ***holds on*** R if all legal relations on R satisfy the set of functional dependencies F
- Note: A specific instance of a relation schema may satisfy a functional dependency even if the functional dependency does not hold on all legal instances
  - For example, a specific instance of *instructor* may sometimes satisfy

$$name \longrightarrow ID$$

Based on and image from "Database System Concepts" book and slides, 6th edition

Functional Dependencies

- A functional dependency is trivial if it is satisfied by all instances of a relation
  - Example:
    - *ID, name* $\rightarrow$ *ID*
    - *name* $\rightarrow$ *name*
  - In general, $\alpha \rightarrow \beta$ is trivial if $\beta \subseteq \alpha$

Functional Dependencies Examples

Assume schema:

student(student_id, first_name, last_name, major, SSN)

Which are true in regards to functional dependencies:

student_id → *last_name*
last_name → *student_id*
student_id → *last_name, major, SSN, student_id*
SSN → *student_id, last_name, major, SSN*
first_name→ *last_name*
last_name → *last_name*

Functional Dependencies Examples

Assume schema:

student(student_id, first_name, last_name, major, SSN)

Which are true in regards to functional dependencies:

student_id → last_name TRUE
last_name → student_id FALSE
student_id → last_name, major, SSN, student_id TRUE
SSN → student_id, last_name, major, SSN TRUE
first_name→ last_name FALSE
last_name → last_name TRUE

Based on and image from "Database System Concepts" book and slides, 6th edition

Closure of a Set of Functional Dependencies

- Given a set F of functional dependencies, there are certain other functional dependencies that are logically implied by *F*
  - For example:
    Given a schema r(A,B,C)
    If $A \rightarrow B$ and $B \rightarrow C$
    then we can infer that $A \rightarrow C$
- The set of all functional dependencies logically implied by *F* is the closure of *F*
- We denote the closure of *F* by F^+
- F^+ is a superset of *F*

Based on and image from "Database System Concepts" book and slides, 6th edition

Lecture Outline

- Relational Database Design
- First Normal Form (1NF)
- Functional Dependencies
- ***Boyce-Codd (BCNF)***
- Third Normal Form (Intro)

Boyce-Codd Normal Form

A relation schema R is in BCNF with respect to a set F of functional dependencies if for all functional dependencies in F^+ of the form

$$\alpha \rightarrow \beta$$

where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:

- $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
- α is a superkey for R

Example schema *not* in BCNF:

instr_dept (*ID, name, salary, dept_name, building, budget*)

because *dept_name* → *building, budget*
holds on *instr_dept*, but *dept_name* is not a superkey

Based on and image from "Database System Concepts" book and slides, 6th edition

Boyce-Codd Normal Form

Are these schemas in BCNF:

instructor (*ID, name, dept_name, salary*)
ID$\rightarrow$ *name,dept_name,salary*

department(dept_name,building,budget)
dept_name$\rightarrow$ *building, budget*

Why or why not?

Based on and image from "Database System Concepts" book and slides, 6th edition

Boyce-Codd Normal Form

Are these schemas in BCNF:

instructor (<u>*ID,*</u> *name, dept_name, salary*)
ID$\rightarrow$ *name,dept_name,salary*

YES – ID is superkey

department(dept_name,building,budget)
dept_name$\rightarrow$ *building, budget*

YES – dept_name is superkey

Based on and image from "Database System Concepts" book and slides, 6th edition

Decomposing a Schema into BCNF

- Suppose we have a schema R and a non-trivial dependency $\alpha \rightarrow \beta$ causes a violation of BCNF.

  We decompose R into:

  $(\alpha \cup \beta)$

  $(R - (\beta - \alpha))$

- In our example:

  instr_dept (*ID, name, salary, dept_name, building, budget*)

  α = *dept_name*

  β = *building, budget*

  and *inst_dept* is replaced by

  $(\alpha \cup \beta)$ = (*dept_name, building, budget*)

  $(R - (\beta - \alpha))$ = (*ID, name, salary, dept_name*)

Convert it to BCNF

Schema:

Student(ID,Name,AdvisorID,AdvisorName)

What are the functional dependencies?

Convert it to BCNF

Schema:

Student(ID,Name,AdvisorID,AdvisorName)

What are the functional dependencies?

ID -> Name
AdvisorID -> AdvisorName

What uniquely identifies the tuples?

Based on and image from "Database System Concepts" book and slides, 6th edition

Convert it to BCNF

Schema:

Student(ID,Name,AdvisorID,AdvisorName)

What are the functional dependencies?

ID -> Name
AdvisorID -> AdvisorName

What uniquely identifies the tuples?
(ID,AdvisorID)

Is there a BCNF violation?

Convert it to BCNF

Schema:

Student(ID,Name,AdvisorID,AdvisorName)

What are the functional dependencies?

ID -> Name
AdvisorID -> AdvisorName

What is the primary key?
(ID,AdvisorID)

Is there a BCNF violation? YES!

Based on and image from "Database System Concepts" book and slides, 6th edition

Convert it to BCNF

Schema:
Student(ID,Name,AdvisorID,AdvisorName)

What are the functional dependencies?
ID -> Name
AdvisorID -> AdvisorName

What is the primary key?
(ID,AdvisorID)

Is there a BCNF violation? YES!

Use ID-> Name to decompose R- (Name-ID)= (ID,AdvisorID,AdvisorName) and ID union Name = (ID,Name)

Based on and image from "Database System Concepts" book and slides, 6th edition

BCNF and Dependency Preservation

- Constraints, including functional dependencies, are costly to check in practice unless they pertain to only one relation
- If it is sufficient to test only those dependencies on each individual relation of a decomposition in order to ensure that all functional dependencies hold, then that decomposition is dependency preserving
- Because it is not always possible to achieve both BCNF and dependency preservation, we consider a weaker normal form, known as third normal form

Lecture Outline

- Relational Database Design
- First Normal Form (1NF)
- Functional Dependencies
- Boyce-Codd (BCNF)
- ***Third Normal Form (Intro)***

Third Normal Form

- A relation schema R is in **third normal form (3NF)** if for all:

  $\alpha \rightarrow \beta$ in F^+

  at least one of the following holds:
  - $\alpha \rightarrow \beta$ is trivial
  - α is a superkey for R
  - Each attribute A in $\beta - \alpha$ is contained in a candidate key for R.

  (**NOTE**: each attribute may be in a different candidate key)
- If a relation is in BCNF it is in 3NF (since in BCNF one of the first two conditions above must hold).
- Third condition is a minimal relaxation of BCNF to ensure dependency preservation (will see why later).