Relational Database Design Theory (I)

CS348 Spring 2023

Instructor: Sujaya Maiyya

Sections: **002 and 004 only**

Announcements

- Assignment 2 is released
  - Due by June 20th

Lectures on Relational Algebra & SQL

- Main SQL clauses for querying and data manipulation
  - Founded on relational algebra
- Constraints: Primary Keys, Foreign Keys, Not NULL, General Assertions and CHECKs
- Triggers

Achieve Integrity of Database

- Views
  - When materialized also a way to achieve performance

Ease of Programming

- Indexes
  - Fast access to some data

Performance

- Recursion & programming

Enhanced functionality

Lectures on Entity/Relationship (ER) Model

- Often users do not directly design relational tables
- ER Model: An even higher-level data model
  - Convert requirements in plaintext to ER diagrams
- Convert ER diagrams into relational model
  - Convert relational model into SQL DDL commands

Next 2 Lectures: Relational Database Design Theory

- Theory of Normal Forms (TNF): Given a set of constraints about the real-world facts that an app will store, how can we formally separate "good" and "bad" relational db schemas?

Design 1

InstDep					
iID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000
…	…	…	…	…	…

If each department identified by depName has as associated (bldng, budget) Design 1, intuitively, is a bad design with redundancy.

Design 2

Inst			
iID	name	salary	depName
111	Alice	5000	CS
222	Bob	4000	Physics
…	…	….	…

Dep		
depName	bldng	budget
CS	DC	20000
Physics	PHY	30000

- Goal of TNF: make the above intuition formal.

Outline For Today

1. Application Constraints and Decompositions
2. Functional Dependencies
3. Boyce-Codd Normal Form (BCNF) & BCNF Decomposition Alg.
4. Dependency Preservation and 3rd Normal Form

Outline For Today

1. Application Constraints and Decompositions
2. Functional Dependencies

This lecture

3. Boyce-Codd Normal Form (BCNF) & BCNF Decomposition Alg.
4. Dependency Preservation and 3rd Normal Form

Next lecture

Application Constraints

- Consider a simple university DB:

Instructors Departments Courses Students

- Independent of stored data: there will be external app. constraints. E.g:
    - Each instructor has 1 name, salary, and department
    - Each department has 1 building
    - Each student can have 1 advisor from each department
    - Instructor i's set of addresses are independent of the departments of i
        - *High-level idea: A "good" DB makes such constraints explicit*

Application Constraints

- Instructors: iIDs, names, salaries, departments (w/ unique iIDs)
- Departments: names, building, budget (w/ unique names)

b/c iID is key

- Constraint 1: Each instructor has 1 name, salary, and department ✓
- Constraint 2: Each department has 1 building and 1 associated budget X

b/c depName is not key

- Possible Design: 1 large table InstDep with one row for each instructor

InstDep					
iID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000
...	...	...	...	...	...

- Problem: redundant data replication. (CS, DC, 20000) repeated k times if there are k instructors in CS.

Problems of Redundancy

InstDep					
iID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000
…	…	…	…	…	…

- Harder to keep db consistent when facts are stored multiple times. E.g:
- If CS's building changed to E4 => need to update 3 rows
- Suppose Bob is the only instructor in Physics and retires (a delete):
  - Deletion of Bob's tuple: Physics department, which will continue to exist, is deleted unless extra work is done
- If new department (w/out yet an instructor) is added: new row w/ NULLs

Redundancy Is Determined By App. Constraints

- Courses: cID, term, iID, capacity

Course			
cID	term	iID	capacity
CS348	S23	Sujaya	100
CS341	F22	Lap Chi	80
CS348	S21	Semih	100
CS348	W20	Xi	100
CS350	W19	Salem	130

- Unclear if this is redundant or not. Depends on external app constraint:
- If courses have 1 associated capacity (independent of term): Redundant
- Otherwise, repetition may be necessary and reflects similarity across entities

Redundancy Is Determined By App. Constraints

- Courses: cID, term, iID, capacity

Course			
cID	term	iID	capacity
$\boxed{\text{CS348}}$	S23	Sujaya	$\boxed{100}$
CS341	F22	Lap Chi	80
$\boxed{\text{CS348}}$	S21	Semih	$\boxed{100}$
$\boxed{\text{CS348}}$	W20	Xi	$\boxed{100}$
CS350	W19	Salem	130
CS348	W23	David	200

- Unclear if this is redundant or not. Depends on external app constraint:
- If courses have 1 associated capacity (independent of term): Redundant
- Otherwise, repetition may be necessary and reflects similarity across entities
- Takeaway: Constraints are external to the db/app and need to be inputs in a db design theory.

Solution To Redundancy: Decompositions

InstDep

iID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000

Inst

iID	name	salary	depName
111	Alice	5000	CS
222	Bob	4000	Physics
333	Carl	5200	CS
444	Diana	5500	CS

Dep

depName	bldng	budget
CS	DC	20000
Physics	PHY	30000

Requirement for Decompositions (1)

- R1 (Lossless): If R is decomposed into R1 and R2, then:

$$R = R1 \bowtie R2$$

- Lossless-ness achieved by decomposing on an appropriate key

Inst			
iID	name	salary	depName
111	Alice	5000	CS
222	Bob	4000	Physics
333	Carl	5200	CS
444	Diana	5500	CS

$\bowtie$

Dep		
depName	bldng	budget
CS	DC	20000
Physics	PHY	30000

RESULT					
iID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000

=

InstDep					
iID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000

Example Lossy Decomposition

InstDep					
ID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000

ID
111
222
333
444

$\bowtie$

name
Alice
Bob
Carl
Diana

$\bowtie$

salary
5000
4000
5200
5500

$\bowtie$

depName
CS
Physics

$\bowtie$

bldng
DC
PHY

$\bowtie$

budget
20000
30000

RESULT					
ID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
111	Bob	5200	CS	PHY	30000
…	…	…	…	…	…

X Can’t tell what’s fact and what’s not.

Requirement for Decompositions (2)

- R2 (Locality of Constraints): If the app had a constraint C, we would prefer to check C in a single relation
- Will discuss more in 3rd Normal Form (next lecture)

High level question we answer in this topic:

How to decompose a database to be lossless & (preferably) retain locality of constraints?

Outline For Today

1. Application Constraints and Decompositions
2. Functional Dependencies
3. Boyce-Codd Normal Form (BCNF) & BCNF Decomposition Alg.
4. Dependency Preservation and 3rd Normal Form

Functional dependencies

- A functional dependency (FD) is a constraint between two sets of attributes in a relation

- FD has the form $X \rightarrow Y$, where X and Y are sets of attributes in a relation R
- $X \rightarrow Y$ means that whenever two tuples in R agree on all the attributes in X, they must also agree on all attributes in Y

X	Y	Z
a	b	c
a	b	?
…	…	…

Must be b

Could be anything

  - If X is a superkey of R , then X → R (all the attributes)

Functional dependencies: Formal definition

Formally: Let t[A] be a tuple t's projection on attributes A

- Dfn: Let X, Y be sets of attributes. An fd $X \longrightarrow Y$ holds in a

  relation R if given t_1 and $t_2 \in R$ s.t. :

  If $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$ holds.

If $X \rightarrow Y$, we say X determines Y

Example FDs

- Captures generalized uniqueness constraints (beyond keys):

InstDep					
iID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000
…	…	…	…	…	…

- Constraint 1: Each iID has 1 name and salary
  - iID → name, salary
- Constraint 2: Each depName has 1 building & 1 associated budget
  - depName → bldng, budget
- Key constraints: Each iID, depName is unique in InstDep
  - iID, depName → name, salary, bldgn, budget

Some FD Vocabulary

- We take FDs as given, i.e., cannot be inferred from a relation instance.
- FDs limit legal instances of a relation $R(A_1, \ldots, A_m)$
- Given a set $\mathcal{F}$ of fds on R, on all *legal instances of R*, each $F \in \mathcal{F}$ hold.

InstDep					
iID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
111	Alice	5000	Biology	BIO	50000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
444	Diana	5500	CS	DC	20000

- Suppose $\mathcal{F}$: (i) iID $\rightarrow$ name, salary; (ii) depName $\rightarrow$ bldng, budget
- E.g: The above instance is a legal instance
- E.g: iID $\rightarrow$ name, salary holds on the above instance.
- Won't need this vocabulary much in lecture.

Implied FDs: Armstrong's Axioms

- A set of fds can imply other fds via 3 intuitive rules: Armstrong's Axioms

1. Reflexivity: If $Y \subseteq X$, then $X \rightarrow Y$ (trivially)
   - iID, name → iID
   - English: Each iID and name value determine a unique iID value
2. Augmentation: if $X \rightarrow Y$, then $XZ \rightarrow YZ$ (trivially)
   - If iID → salary then iID, name → salary, name
   - English: if each iID determines a unique salary value, then each (iID, name) value pair determines a unique (salary, name) value

InstDep					
<u>iID</u>	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
…	…	…	…	…	…

Implied FDs: Armstrong's Axioms

3. Transitivity: if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

- Suppose each instructor can be in a single department and each dep has a single budget
- FD1: iID → depName FD2: depName → budget, then

  iID → budget
- English: If each iID value determines a unique depName value, which in turn determines a unique budget value, then each iID value determines a unique budget value.

InstDep					
iID	name	salary	depName	bldng	budget
111	Alice	5000	CS	DC	20000
222	Bob	4000	Physics	PHY	30000
333	Carl	5200	CS	DC	20000
…	…	…	…	…	…

Other Rules Implied by Armstrong's Axioms

1. Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

   Proof:

   i. $X \rightarrow YZ$

   ii. $YZ \rightarrow Y$ (by reflexivity); $YZ \rightarrow Z$ (by reflexivity)

   iii. $X \rightarrow Y$ (by transitivity); $X \rightarrow Z$ (by transitivity)

2. Union: If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$ (Prove as exercise)
3. Pseudo-transitivity: If $X \rightarrow Y$ and $YZ \rightarrow T$ then $XZ \rightarrow T$ (Prove as exercise)

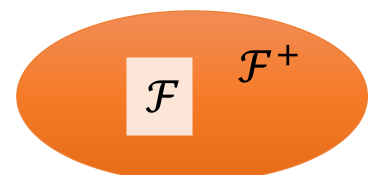
Using these rules, you can prove or disprove a (derived) FD given a set of (base) FDs

Closure of FD sets: $\mathcal{F}^+$

- How do we know what additional FDs hold in a schema?

- A set of FDs $\mathcal{F}$ logically implies a FD $X \rightarrow Y$ if $X \rightarrow Y$ holds in all instances of R that satisfy $\mathcal{F}$

- The closure of a FD set $\mathcal{F}$ (denoted $\mathcal{F}^+$):
  - The set of all FDs that are logically implied by $\mathcal{F}$
  - Informally, $\mathcal{F}^+$ includes all of the FDs in $\mathcal{F}$, i.e., $\mathcal{F} \subseteq F^+$, plus any dependencies they imply.



$\mathcal{F}^+$: Closure of $\mathcal{F}$ (example)

Dfn: Let $\mathcal{F}$ be a set of fds. The closure $\mathcal{F}^+$ of $\mathcal{F}$ is the set of all fds implied by $\mathcal{F}$.

- Ex: $\mathcal{F}$: iID→name, depName; depName→bldng
- $\mathcal{F}^+$: $\mathcal{F}$ ∪ iID→iID; iID,email→name,email (trivial ones) … ∪
  iID→bldng (transitivity) etc..

InstDep				
iID	name	email	depName	bldng
$\boxed{111}$	Alice	alice@gmail	CS	$\boxed{\text{DC}}$
$\boxed{111}$	Alice	alice@hotmai l	CS	$\boxed{\text{DC}}$
222	Bob	bob@gmail	Physics	PHY
222	Bob	bob@hotmail	Physics	PHY
333	Carl	carl@gmail	CS	DC
…	…	…	…	…

Exercise Showing an FD is in $\mathcal{F}^+$

- Consider an Inst_Proj relation of instructors and their research projects

InstProj						
iID	name	projID	projName	projDep	hours	funds

- Given: (i) iID → name; (ii) projID → projName, projDep;

  (iii) iID, projID → hours; (iv) projDep, hours → funds;

- Prove iID, projID → funds

1. iID, projID → hours (by fd iii)
2. projID → projName, projDep (by fd ii)
3. iID, projID → hours, projName, projDep (by reflexivity, union & decomposition of 1 & 2)
4. iID, projID → funds (by transitivity of 3, and fd iv) (+ decomposition)

How To Compute $\mathcal{F}^+$ from $\mathcal{F}$

$F^+ = F$
repeat
    for each functional dependency f in F^+
        apply reflexivity and augmentation rules on f
        add the resulting functional dependencies to F^+
    for each pair of functional dependencies f_1 and f_2 in F^+
        if f_1 and f_2 can be combined using transitivity
            add the resulting functional dependency to F^+
until F^+ does not change any further

Figure 8.7 A procedure to compute F^+.

Attribute closure

- The closure of attributes Z in a relation R (denoted Z^+) with respect to a set of FDs, $\mathcal{F}$, is the set of all attributes $\{A_1, A_2, \ldots\}$ functionally determined by Z (that is, $Z \rightarrow A_1 A_2 \ldots$)

- Algorithm for computing the closure Compute$Z^+(Z, \mathcal{F})$:
  - Start with closure $= Z$
  - If $X \rightarrow Y$ is in $\mathcal{F}$ and X is already in the closure, then also add Y to the closure
  - Repeat until no new attributes can be added

Example for computing attribute closure

Given relation R(ABCDEFG)

Compute$Z^+(\{B, F\}, \mathcal{F})$:

$\mathcal{F}$ includes:

$A, B \rightarrow F$

$A \rightarrow C$

$B \rightarrow E, D$

$D, F \rightarrow G$

FD	Z^+
initial	B, F
$B \rightarrow E, D$	B, F, E, D
$D, F \rightarrow G$	B, F, E, D, G

$$B, F \rightarrow E, D, G$$

Using attribute closure

Given a relation R and set of FD's $\mathcal{F}$

- Does another FD $X \rightarrow Y$ follow from $\mathcal{F}$?
  - Compute X^+ with respect to $\mathcal{F}$
  - If $Y \subseteq X^+$, then $X \rightarrow Y$ follows from $\mathcal{F}$

- Is K a key of R?
  - Compute K^+ with respect to $\mathcal{F}$
  - If K^+ contains all the attributes of R, K is a super key
  - Still need to verify that K is *minimal* (how?)
    - Hint: check the attribute closure of its proper subset.
    - i.e., Check that for no set X formed by removing attributes from K is $K^+$the set of all attributes

Design Theory

- Detect anomalies: Functional dependencies
  - Closure of FDs (rules, e.g. Armstrong's axioms)
  - Attribute closure
- Repair anomalies: Schema decomposition
  - (next lecture)