Lecture 12: Database Design (Theory)

CS348 Spring 2025:
Introduction to Database Management

Instructor: **Xiao Hu**

Sections: 001, 002, 003

Announcements

- Appealing of Assignment 1
  - Check remark request guidelines on Piazza
  - Reach out to corresponding TA, IA (Guy), ISC (Sylvie)
  - Check sample solutions on Learn
- Milestone 1 of Group Project
  - Due on June 19
- Switch-type cut-off for assessment
  - Due on June 19
- Assignment 2
  - Coverage: Lecture 4 - Lecture 12
  - Check online office hours on Piazza
  - Due on June 24

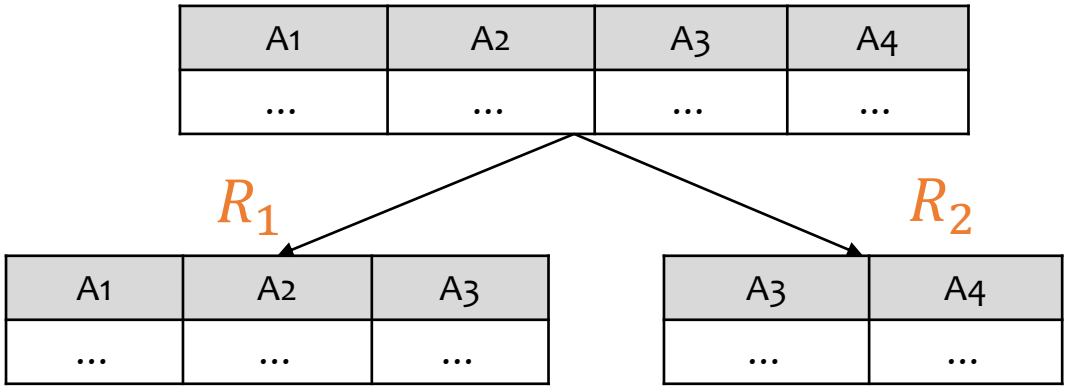
(Recap) Data Redundancy!

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

instructorID	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

- Decomposition: Break down a complex database schema into smaller, more manageable pieces while maintaining data integrity and relationships

(Recap) Decompositions



- R is decomposed into R_1 and R_2
  - Attribute (R_1) ∪ Attribute(R_2) = Attribute (R)
  - R_1 and R_2 are the projections of R onto Attribute (R_1) and Attribute(R_2)
- Any decomposition gives $R \subseteq R_1 \bowtie R_2$
  - Lossless decomposition if $R = R_1 \bowtie R_2$
  - Lossy decomposition if $R \subset R_1 \bowtie R_2$

For any tuple $(a, b, c, d) \in R$, $(a, b, c) \in R_1$ and $(c, d) \in R_2$; then $(a, b, c) \bowtie (c, d) \in R_1 \bowtie R_2$

(Recap) A good decomposition

instructorID	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

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

depName	bldng	budget
CS	DC	20000
Physics	PHY	30000

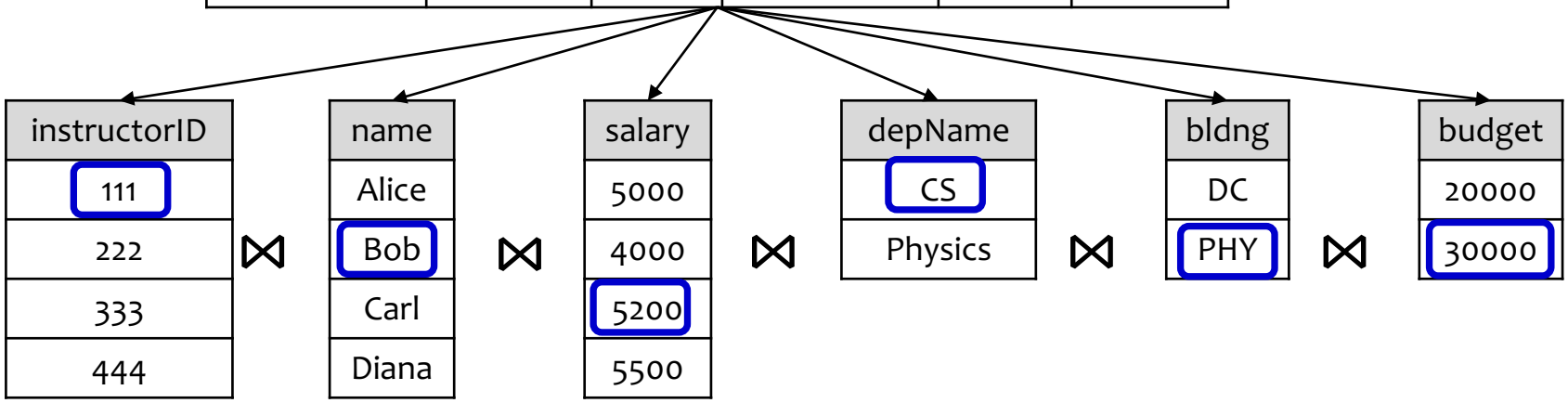
⋈

instructorID	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

Why can we recover all original tuples by joins?

(Recap) A bad decomposition

instructorID	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



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

Lossy? But I got more rows! Can't tell what's fact and what's not, so we lose information!

Outline for Today

- Functional Dependencies
- Decomposition:
  - Lossless
  - Dependency-preserving
- Boyce-Codd Normal Form (BCNF)
- Third Normal Form (3NF)

Lossless decomposition

- We should be able to reconstruct the instance of the original table from the instances of the tables in the decomposition

A decomposition $\{R_1, R_2\}$ of R is lossless if and only if the common attributes of R_1 and R_2 form a superkey for either schema, i.e., $R_1 \cap R_2 \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$

(See Lecture 11)
If X is a superkey of R, then $X \rightarrow R$

Is this a lossless decomposition?

- Consider $R = \{sno, sname, city, pno, pname, price\}$ and a decomposition (R_1, R_2) below:

$R_1 = \{sno, sname, pname\}$
$R_2 = \{pname, city, pno, price\}$

sno	sname	city	pno	pname	price
S1	Apple	K-W	P1	A	$25
S1	Apple	K-W	P2	B	$34
S2	BestBuy	London	P3	A	$5
S2	BestBuy	London	…	…	…

Lossy since $R_1 \cap R_2 = \{pname\}$ is not a superkey of either R_1 or R_2

Which one is better?

$\mathcal{F}$ includes:
project → *department*
department → *division*
project → *division*

- Consider Company = {*project*, *department*, *division*} under a set $\mathcal{F}$ of FDs, and two different decompositions:

$R_1 = \{project, department\}$
$R_2 = \{department, division\}$

$R_1 = \{project, department\}$
$R_2 = \{project, division\}$

- Both are lossless. (Why?)
- However, testing FDs is easier on one of them. (Which?)

Testing FDs

$\mathcal{F}$ includes:
project → *department*
department → *division*
Project → *division*

- Consider Company = {*project*, *department*, *division*} under a set $\mathcal{F}$ of FDs, and two different decompositions:

decomposition 1:

$R_1 = \{project, department\}$
$R_2 = \{department, division\}$

test in R_1

project → *department*
department → *division*
project → *division*

test in R_2

No need to test since it is implied by the other two FDs

decomposition 2:

$R_1 = \{project, department\}$
$R_2 = \{project, division\}$

test in R_1

project → *department*
department → *division*
project → *division*

cross-tables!

test in R_2

Dependency-preserving decomposition

- We should be able to (explicitly and implicitly) test all dependencies in each base table of the decomposition

> Given a schema R and a set $\mathcal{F}$ of FDs,
> a decomposition of R is dependency preserving
> if there is an equivalent set $\mathcal{F}'$ of FDs to $\mathcal{F}$,
> none FD in $\mathcal{F}'$ is cross-table in the decomposition.

- Given a set of FDs $\mathcal{F}$, we say $\mathcal{F}'$ is equivalent to $\mathcal{F}$ if their closures are the same, i.e., $\mathcal{F}^+ = \mathcal{F}'^+$.

Outline for Today

- Functional Dependencies
- Decomposition:
  - Lossless
  - Dependency-preserving
- Boyce-Codd Normal Form (BCNF)
- Third Normal Form (3NF)

Boyce-Codd Normal Form (BCNF)

- A relation R is in BCNF under $\mathcal{F}$ if each FD $X \to Y \in \mathcal{F}^+$ with $XY \subseteq R$ satisfies:
  - either $X \to Y$ is trivial, i.e., $Y \subseteq X$
  - or X is a super key of R, i.e., $X \to R$

- Is $R = \{sno, sname, city, pno, pname, price\}$ under $\mathcal{F}$ in BCNF?

$\mathcal{F}$ includes:

sno → *sname, city*

pno → *pname*

sno, pno → *price*

(See Lecture 11)
How to determine if K is a superkey of R?

Compute BCNF decomposition

Repeat the following until all relations are in BCNF

- Step 1: Find a BCNF violation
  - A relation R in the decomposition
  - A non-trivial FD $X \rightarrow Y$ in $\mathcal{F}^+$ with $XY \subseteq R$, where X is not a super key of R

- Step 2: Decompose R into R_1 and R_2
  - R_1 has attributes $X \cup Y$;
  - R_2 has attributes $X \cup Z$, where Z contains all attributes of R that are in neither X nor Y

Example of BCNF decomposition

(See Lecture 11)
How to determine if K is a superkey of R?

$\mathcal{F}$ includes:
sno → *sname, city*
pno → *pname*
sno, pno → *price*

R = {*sno, sname, city, pno, pname, price*}

BCNF violation: *sno* → *sname, city*

R2= {*sno, pno, pname, price*}

pno → *pname*
sno, pno → *price*

BCNF violation: *pno* → *pname*

R1 = {*sno, sname, city*}

sno → *sname, city*

In BCNF

R3= {*pno, sno, price*}

sno, pno → *price*

In BCNF

R4 = {*pno, pname*}

pno → *pname*

In BCNF

BCNF helps remove redundancy

$\mathcal{F}$ includes:

sno → *sname, city*

pno → *pname*

sno, pno → *price*

sno	*sname*	*city*	*pno*	*pname*	*price*
S1	Apple	K-W	P1	A	$25
S1	Apple	K-W	P2	B	$34
S1	Apple	K-W	P3	A	$20
S2	BBuy	London	…	…	…

BCNF violation: *sno* → *sname, city*

sno	*pno*	*pname*	*price*
S1	P1	A	$25
S1	P2	B	$34
S1	P3	A	$20
S2	…	…	…

sno	*sname*	*city*
S1	Apple	K-W
S2	BBuy	London
..	…	…

Another example

$\mathcal{F}$ includes:
uid → *uname*, *twitterid*
twitterid → *uid*
uid, *gid* → *fromDate*

UserJoinsGroup (*uid*, *uname*, *twitterid*, *gid*, *fromDate*)

BCNF violation: *uid* → *uname*, *twitterid*

User (*uid*, *uname*, *twitterid*)

uid → *uname*, *twitterid*
twitterid → *uid*

In BCNF (why?)

Member (*uid*, *gid*, *fromDate*)

uid, *gid* → *fromDate*

In BCNF (why?)

An alternative

$\mathcal{F}$ includes:
uid → *uname*, *twitterid*
twitterid → *uid*
uid, *gid* → *fromDate*

UserJoinsGroup (*uid*, *uname*, *twitterid*, *gid*, *fromDate*)

(See Lecture 11)
What is closure of $\mathcal{F}$?

BCNF violation: *twitterid* → *uid*

No FDs in $\mathcal{F}$ violate BCNF here. But we need to check all the FDs in $\mathcal{F}^+$!!

UserID (*twitterid*, *uid*)

twitterid → *uid*

In BCNF

UserJoinsGroup (*twitterid*, *uname*, *gid*, *fromDate*)

twitterid → *uname*
twitterid, *gid* → *fromDate*

BCNF violation: *twitterid* → *uname*

UserName (*twitterid*, *uname*)

In BCNF

Member (*twitterid*, *gid*, *fromDate*)

In BCNF

Is BCNF lossless?

YES!

Given a non-trivial FD $X \to Y$ in $\mathcal{F}^+$ with $XY \subseteq R$, where X is not a super key of R, we need to prove:

Let $Z = \text{Attribute}(R) - X - Y$

- Anything we project always comes back in the join:

$$R \subseteq (\pi_{XY} R) \bowtie (\pi_{XZ} R)$$

  - Sure; and it doesn't depend on the FD

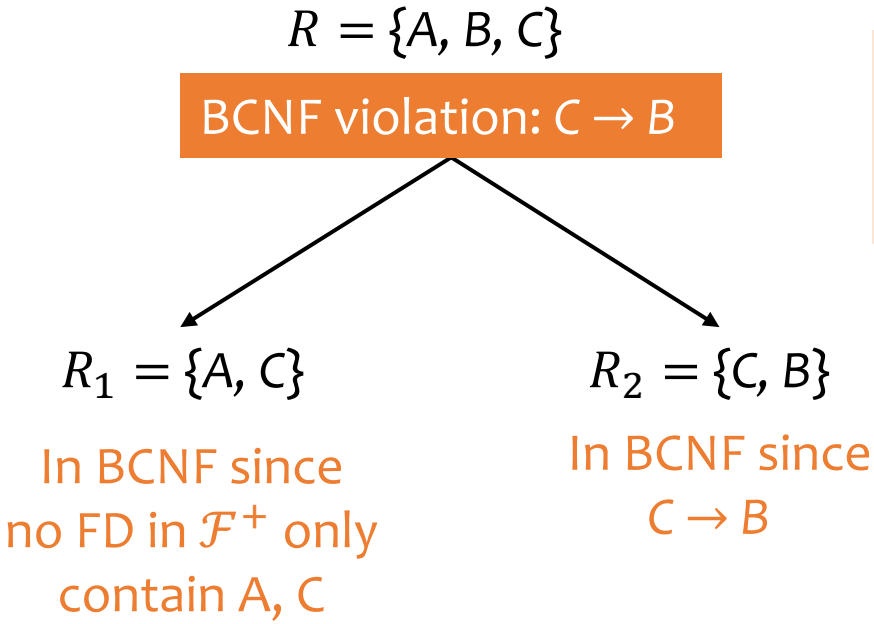
- Anything that comes back in the join must be in the original relation:

$$R \supseteq (\pi_{XY} R) \bowtie (\pi_{XZ} R)$$

  - Proof uses the fact that $X \to Y$

Is BCNF dependency-preserving?

- NO!
- Consider a simple example R under $\mathcal{F}$:



$\mathcal{F}$ includes:
$A, B \rightarrow C$
$C \rightarrow B$

$A, B \rightarrow C$ is cross-table and cannot be tested directly

Outline for Today

- Functional Dependencies
- Decomposition:
  - Lossless
  - Dependency-preserving
- Boyce-Codd Normal Form (BCNF)
  - Lossless decomposition
  - Not necessarily dependency-preserving decomposition
- Third Normal Form (3NF)
  - Lossless decomposition
  - Dependency-preserving decomposition

Third normal form (3NF)

- A relation R is in 3NF under $\mathcal{F}$ if each FD $X \to Y \in \mathcal{F}^+$ with $XY \subseteq R$ satisfies:
  - either $X \to Y$ is trivial, i.e., $Y \subseteq X$,
  - or X is a super key of R, i.e., $X \to R$ or,
  - or each attribute in $Y - X$ is in a key of R

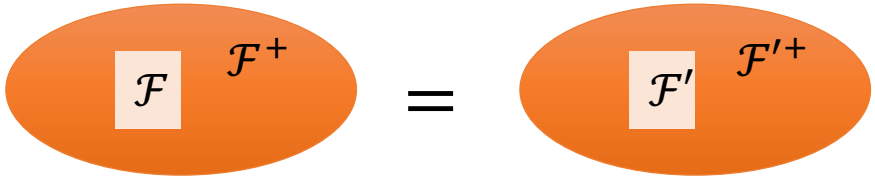
BCNF only allows the first two cases, so 3NF is looser than BCNF

- Is $R = \{A, B, C\}$ under $\mathcal{F}$ in 3NF?
  - $A, B \to C$ is satisfied since AB is a super key
  - $C \to B$ *is* satisfied since B is part of key $\{A, B\}$

$\mathcal{F}$ includes:
$A, B \to C$
$C \to B$

Compute 3NF decomposition

- Step 1: Finding the minimal cover of the FD set $\mathcal{F}$



  - Given a set of FDs $\mathcal{F}$, we say $\mathcal{F}'$ is equivalent to $\mathcal{F}$ if their closures are the same, i.e., $\mathcal{F}^+ = \mathcal{F}'^+$.
  - The smallest equivalent set of FDs

- Step 2: Decompose based on the minimal cover

Minimal cover of $\mathcal{F}$

A set of FDs $\mathcal{F}$ is minimal if

- every RHS of a FD in $\mathcal{F}$ is a single attribute

- Consider $R = \{sno, sname, city, pno, pname, price, ptype\}$ under a set $\mathcal{F}$ of FDs

$\mathcal{F}$ includes:

sno → *sname, city*

pno → *pname*

sno, pno → *price*

sno, pname → *price*

sno, pname → *ptype*

Fail condition 1

Minimal cover of $\mathcal{F}$

A set of FDs $\mathcal{F}$ is minimal if

- every RHS of a FD in $\mathcal{F}$ is a single attribute
- No FD $X \to A$ *with* $Z \subseteq X$ such that
$$(\mathcal{F} - \{X \to A\} + \{Z \to A\})^+ = \mathcal{F}^+$$

How to check efficiently?
if A is in the attribute closure of Z under $\mathcal{F}$

No redundant attributes in X

- Consider $R = \{sno, sname, city, pno, pname, price, ptype\}$ under a set $\mathcal{F}$ of FDs

$\mathcal{F}$ includes:
sno → *sname, city*
pno → *pname*
sno, pno → *price*
sno, pname → *price*
pno, pname → *ptype*

Fail condition 2: removing *pname*
since *pno* → *pname*
and *pno* → *ptype* together imply it

Minimal cover of $\mathcal{F}$

A set of FDs $\mathcal{F}$ is minimal if

- every RHS of a FD in $\mathcal{F}$ is a single attribute
- No FD $X \to A$ *with* $Z \subseteq X$ such that
$$(\mathcal{F} - \{X \to A\} + \{Z \to A\})^+ = \mathcal{F}^+$$
- No FD $X \to A$ such that $(\mathcal{F} - \{X \to A\})^+ = \mathcal{F}^+$

How to check efficiently?
if A is in the attribute closure of X under $\mathcal{F} - \{X \to A\}$

No redundant FD in $\mathcal{F}$

- Consider $R = \{sno, sname, city, pno, pname, price, ptype\}$ under a set $\mathcal{F}$ of FDs

$\mathcal{F}$ includes:

sno → *sname, city*
pno → *pname*
sno, pno → *price*
sno, pname → *price*
pno, pname → *ptype*

Fail condition 3: *pno* → *pname* and *sno, pname* → *price* can imply it

Compute minimal cover of $\mathcal{F}$

Repeat the following steps until $\mathcal{F}$ does not change

- Step 1: Replace $X \to YZ$ with $X \to Y$ and $X \to Z$
- Step 2: Remove B from the LHS of $X \to A$ if A is in the attribute closure of $X - \{B\}$ under $\mathcal{F}$
- Step 3: Remove $X \to A$ if A is in the attribute closure of X under $\mathcal{F} - \{X \to A\}$

Why equivalent set of FDs?

Example of minimal cover

- Consider $R = \{sno, sname, city, pno, pname, price, ptype\}$ under a set $\mathcal{F}$ of FDs

$\mathcal{F}$ includes:

- *sno* → *sname, city*
- *pno* → *pname*
- *sno, pno* → *price*
- *sno, pname* → *price*
- *pno, pname* → *ptype*

Split it into *sno* → *sname* and *sno* → *city*

Remove it
as it is implied by *pno* → *pname*
and *sno, pname* → *price*

Update it by *pno* → *ptype*
as it is implied by pno → *pname*
and pno → *ptype*

$\mathcal{F}^*$ includes:

- *sno* → *sname*
- *sno* → *city*
- *pno* → *pname*
- *sno, pname* → *price*
- *pno* → *ptype*

Compute 3NF decomposition

Given a relation R with a set $\mathcal{F}$ of FDs:

Step 1: Find a minimal cover $\mathcal{F}^*$ for $\mathcal{F}$

Step 2: For every $X \rightarrow Y$ in $\mathcal{F}^*$, add a relation {X, Y} to the decomposition

Step 3: If no relation contains a key for R, add a relation containing an arbitrary key for R to the decomposition

Example for 3NF decomposition

- Consider $R = \{sno, sname, city, pno, pname, price, ptype\}$ under a set $\mathcal{F}$ of FDs

$\mathcal{F}$ includes:

sno → *sname, city*
pno → *pname*
sno, pno → *price*
sno, pname → *price*
pno, pname → *ptype*

$\mathcal{F}^*$ includes:

sno → *sname*
sno → *city*
pno → *pname*
sno, pname → *price*
pno → *ptype*

R_1 (*sno, sname*)
R_2 (*sno, city*)
R_3 (*pno, pname*)
R_4 (*sno, pname, price*)
R_5 (*pno, ptype*)

R_6 (*sno, pno*)

(Optional) possible optimization: combine R_1 and R_2

Add the key (sno,pno)

(See Lecture 11)
How to determine if K is or is not a key of R?

Summary (this week)

- Functional dependencies: provide clues towards the elimination of (some) redundancies in a schema
  - Armstrong's axioms
  - Closure of FDs
  - Closure of attributes

- Decomposition (this lecture):
  - BCNF: lossless (but not necessarily dependency-preserving)
  - 3NF: lossless and dependency-preserving (but with more redundancy)

Where are we now?

- Overview (Lecture 1)
- Relational data model and Relational Algebra (Lectures 2 - 3)
- SQL (Lectures 4 - 8)
- Database Design (Lectures 9 - 12)

- Database Internals
  - Storage & Indexing (Lectures 13 - 15)
  - Query Processing & Optimization (Lectures 16 - 18)
  - Transactions (Lectures 19 - 20)

What's next?

- Midterm Review