CS 4440 A

Emerging Database Technologies

Lecture 4
01/22/24

Announcements

Assignment 1 (technology review) due next Monday (Jan 29)

Start looking for project groups!

- Use the ”Search for Teammates” feature in Piazza
- Self-signup for groups on canvas (under People->Project Groups tab)
- Project proposal due Feb 7 (not graded)

Anonymous course feedback form: https://forms.gle/N1z9QSLyjiFHe9sU8

Recap

- Design theory
  - Functional dependency (FD)
  - Trivial FDs
  - Splitting/combining rule
  - Closure of attributes
  - Armstrong's axioms
  - Minimal basis
  - Projection of FDs

title, year → length, genre, studioName

title	year	length	genre	studioName	starName
Ponyo	2008	103	anime	Ghibli	Yuria Nara
Ponyo	2008	103	anime	Ghibli	Hiroki Doi
Oldboy	2003	120	mystery	Show East	Choi Min-Sik

$AB \rightarrow C$

$BC \rightarrow AD$

$D \rightarrow E$

$CF \rightarrow B$

1. $AB \rightarrow C$ (given)
2. $BC \rightarrow AD$ (given)
3. $AB \rightarrow BC$ (Augmentation on 1)
4. $AB \rightarrow AD$ (Transitivity on 2,3)
5. $AD \rightarrow D$ (Reflexivity)

Anomalies

- Occurs when we try to cram too much information into a single relation

  1. Redundancy: information is repeated unnecessarily
  2. Update anomaly: only updating the first tuple may leave the second tuple incorrect

Movies1

title	year	length	genre	studioName	starName
Ponyo	2008	103	anime	Ghibli	Yuria Nara
Ponyo	2008	103	anime	Ghibli	Hiroki Doi
Oldboy	2003	120	mystery	Show East	Choi Min-Sik

Anomalies

- Occurs when we try to cram too much information into a single relation

Movies1

title	year	length	genre	studioName	starName
Ponyo	2008	103	anime	Ghibli	Yuria Nara
Ponyo	2008	103	anime	Ghibli	Hiroki Doi
Oldboy	2003	120	mystery	Show East	Choi Min-Sik

3. Deletion anomaly: removing the movie star Choi Min-Sik will also remove the movie information of Oldboy

Decomposing relations

- The accepted way to eliminate anomalies is to decompose relations

Movies2 No redundancy or update anomalies

title	year	length	genre	studioName
Ponyo	2008	103	anime	Ghibli
Oldboy	2003	120	mystery	Show East

Movies3 No deletion anomalies

title	year	starName
Ponyo	2008	Yuria Nara
Ponyo	2008	Hiroki Doi
Oldboy	2003	Choi Min-Sik

Decomposing relations

- The accepted way to eliminate anomalies is to decompose relations

Movies2

title	year	length	genre	studioName
Ponyo	2008	103	anime	Ghibli
Oldboy	2003	120	mystery	Show East

Movies3

title	year	starName
Ponyo	2008	Yuria Nara
Ponyo	2008	Hiroki Doi
Oldboy	2003	Choi Min-Sik

This is OK because title and year form a key of a movie and cannot be more succinct; if one of the year changes, the movie is a different one

Boyce-Codd Normal Form (BCNF)

A relation satisfying BCNF does not have the discussed anomalies

- Holds when the left side of every nontrivial FD is a superkey
- Equivalently, the left side of every nontrivial FD must contain a key

Boyce-Codd Normal Form (BCNF)

A relation satisfying BCNF does not have the discussed anomalies

- Movies1 does not satisfy BCNF
  - There is a nontrivial FD title year → length genre studioName, but the only key is {title, year, starName}
- Movies2 satisfies BCNF
  - Again the nontrivial FD is title year → length genre studioName, but the key is now {title, year}
- Movies3 satisfies BCNF
  - There is no nontrivial FD

Movies2

title	year	length	genre	studioName
Ponyo	2008	103	anime	Ghibli
Oldboy	2003	120	mystery	Show East

Movies3

title	year	starName
Ponyo	2008	Yuria Nara
Ponyo	2008	Hiroki Doi
Oldboy	2003	Choi Min-Sik

Boyce-Codd Normal Form (BCNF)

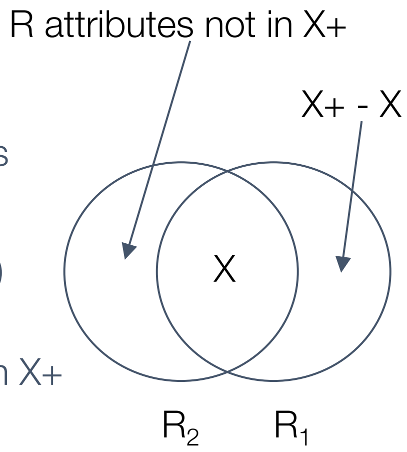
- Any two-attribute relation is in BCNF
  - If there are no nontrivial FD's BCNF holds
  - If $A \rightarrow B$ holds, but not $B \rightarrow A$, the only nontrivial FD has A (i.e., the key) on the left
  - Symmetric case when $B \rightarrow A$ holds, but not $A \rightarrow B$
  - If both $A \rightarrow B$ and $B \rightarrow A$ hold, any nontrivial FD has A or B (both are keys) on the left

```
Employee(empId, ssn)
```

empId → ssn
ssn → empId

Decomposition into BCNF

- Repeatedly decompose relations so that
  - Each subset relation is in BCNF
  - The original relation can be reconstructed from the decomposed relations
- Algorithm: given relation R
  - Find an FD X → Y that violates BCNF (here X and Y are sets of attributes)
  - Then compute the closure X+
  - The decomposed relations are R_1 = X+ and R_2 = X and R attributes not in X+
  - Recursively decompose R_1 and R_2




```
Movies1(title,year,length,genre,studioName,starName)
```

R_1 R_2

```
Movies2(title,year,length,genre,studioName)
```

```
Movies3(title,year,starName)
```

Decomposition into BCNF

- In general, there can be multiple decompositions

```
R(title,year,studioName,president,presAddr)
```

R's FDs

title year → studioName
studioName → president
president → presAddr

Decomposition into BCNF

- In general, there can be multiple decompositions

```
R(title,year,studioName,president,presAddr)
```

R's FDs

Key

BCNF violations

title year → studioName
studioName → president
president → presAddr

Decomposition into BCNF

- In general, there can be multiple decompositions

`R(title,year,studioName,president,presAddr)`

`R1(studioName,president,presAddr)`

`R2(title,year,studioName)`

Key

R's FDs

title year → studioName
studioName → president
president → presAddr

BCNF violations

Decomposition into BCNF

- In general, there can be multiple decompositions

`R(title,year,studioName,president,presAddr)`

`R1(studioName,president,presAddr)`

`R2(title,year,studioName)`

R1's FDs

Key

studioName → president
president → presAddr

BCNF violation

Decomposition into BCNF

- In general, there can be multiple decompositions

`R(title,year,studioName,president,presAddr)`

`R1(studioName,president,presAddr)` `R2(title,year,studioName)`

`R3(president,presAddr)` `R4(president,studioName)`

R1's FDs

Key

studioName → president
president → presAddr

BCNF violation

Q: Is this algorithm guaranteed to terminate successfully?

Decomposition into BCNF

- The algorithm eventually terminates successfully because decomposed relations have strictly fewer attributes, and any relation with two attributes are in BCNF

Exercise #1

- What are the BCNF violations of the FDs?
- Decompose into relations satisfying BCNF

$\boxed{R(A,\ B,\ C,\ D)}$

FD's: $AB \rightarrow C$, $C \rightarrow D$, $D \rightarrow A$

Desirable properties of decomposition

We want the decomposition to have

(1) Elimination of anomalies

(2) Recoverability of information: can we recover the original relation by joining?

(3) Preservation of dependencies: if we check the projected FD's in the decomposed relations, does the reconstructed original relation satisfy the original FD's?

- The BCNF algorithm gives (1) and (2), but not necessarily (3)
- 3NF algorithm (covered later) gives (2) and (3), but not necessarily (1)
- In fact, there is no way to get all three at once!

Recovery of information

- Why not decompose a relation into any 2-attribute relations, which use BCNF?

A	B	C
1	2	3
2	2	4
3	3	5

A	B
1	2
2	2
3	3

B	C
2	3
2	4
3	5

Recovery of information

- Why not decompose a relation into any 2-attribute relations, which use BCNF?

A	B	C
1	2	3
2	2	4
3	3	5

A	B
1	2
2	2
3	3

$\bowtie$

B	C
2	3
2	4
3	5

$\neq$ information is lost!

A	B	C
1	2	3
1	2	4
2	2	3
2	2	4
3	3	5

Lossless join

- A decomposition of R where joining them gives back R (i.e., recovers information)
- The BCNF decomposition algorithm gives a lossless join

A	B	C
1	2	4
2	2	4
3	3	5

A is the key
B → C, which is a BCNF violation

A	B
1	2
2	2
3	3

B	C
2	4
3	5

Lossless join

- A decomposition of R where joining them gives back R (i.e., recovers information)
- The BCNF decomposition algorithm gives a lossless join

A	B	C
1	2	3
2	2	4
3	3	5

A is the key
B → C, which is a BCNF violation

Can this happen?

A	B
1	2
2	2
3	3

B	C
2	3
2	4
3	5

Recall this decomposition loses information

Lossless join

- A decomposition of R where joining them gives back R (i.e., recovers information)
- The BCNF decomposition algorithm gives a lossless join

A	B	C
1	2	3
2	2	4
3	3	5

A is the key
B → C, which is a BCNF violation

Can this happen?
No, because B → C

A	B
1	2
2	2
3	3

B	C
2	3
2	4
3	5

Recall this decomposition loses information

Dependency preservation

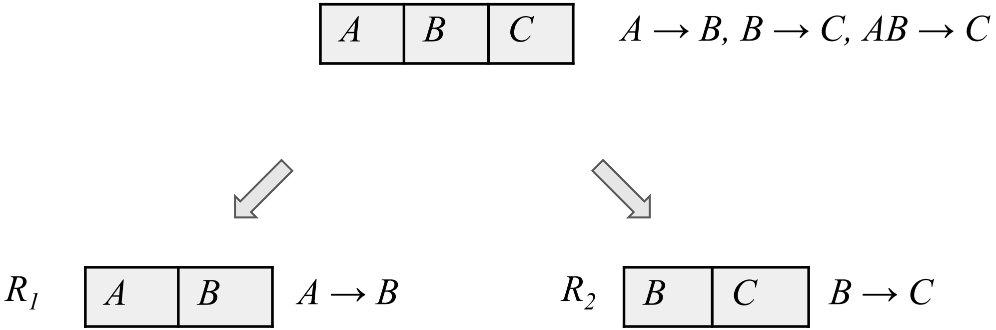
- We can check all the FD’s in the original relation by checking the FD’s in the decomposed relations

A	B	C

$A \rightarrow B,\ B \rightarrow C,\ AB \rightarrow C$

Dependency preservation

- We can check all the FD's in the original relation by checking the FD's in the decomposed relations



Dependency preservation

- We can check all the FD's in the original relation by checking the FD's in the decomposed relations

A	*B*	*C*

$A \to B,\ B \to C,\ AB \to C$

R_1

A	*B*
1	2
3	2

$A \to B$

R_2

B	*C*
2	3
3	3

$B \to C$

FDs satisfied here

Dependency preservation

- We can check all the FD's in the original relation by checking the FD's in the decomposed relations

A	B	C

$A \rightarrow B,\ B \rightarrow C,\ AB \rightarrow C$

This FD is still satisfied

R_1

A	B
1	2
3	2

$A \rightarrow B$

R_2

B	C
2	3
3	3

$B \rightarrow C$

BCNF and dependency preservation

- A BCNF decomposition has lossless-join, but may not have dependency-preservation

- Example:
    - Suppose R(A,B,C) has the FD's $B \rightarrow C$ and $AC \rightarrow B$
    - The keys are AB and AC
    - Therefore, $B \rightarrow C$ is a BCNF violation

BCNF and dependency preservation

- Suppose R(A,B,C) has the FD’s B → C and AC → B
  - The keys are AB and AC
  - Therefore, B → C is a BCNF violation
- The BCNF decomposition is thus R_1(A, B) and R_2(B, C)
- The projected FD is B → C

R_1

A	B

R_2

B	C

BCNF and dependency preservation

- Suppose R(A,B,C) has the FD's $B \rightarrow C$ and $AC \rightarrow B$
  - The keys are AB and AC
  - Therefore, $B \rightarrow C$ is a BCNF violation
- The BCNF decomposition is thus $R_1(A, B)$ and $R_2(B, C)$
- The projected FD is $B \rightarrow C$
- Now suppose we insert tuples into R_1 and R_2

R_1

A	B
4	1
4	2

R_2

B	C
1	3
2	3

BCNF and dependency preservation

- Suppose R(A,B,C) has the FD's B → C and AC → B
  - The keys are AB and AC
  - Therefore, B → C is a BCNF violation
- The BCNF decomposition is thus R_1(A, B) and R_2(B, C)
- The projected FD is B → C
- Now suppose we insert tuples into R_1 and R_2
- However, AC → B is no longer satisfied
- So the dependency is not preserved

R_1

A	B
4	1
4	2

R_2

B	C
1	3
2	3

$R_1 \bowtie R_2$

A	B	C
4	1	3
4	2	3

Third normal form (3NF)

- Intuition: slightly relax BCNF by allowing relations that cannot be decomposed into BCNF relations without losing the ability to check the FD's
- Definition: whenever $A_1\ A_2 \ldots A_n \rightarrow B_1\ B_2 \ldots B_m$ is a nontrivial FD, either
  - $\{A_1\ A_2 \ldots A_n\}$ is a superkey or
  - Those of B's not among the A's are members of some keys (i.e., they are prime)
- In previous example,
  - We had R(A,B,C) and the FD's $B \rightarrow C$ and $AC \rightarrow B$
  - The keys are AB and AC
  - $B \rightarrow C$ is a BCNF violation, but not a 3NF violation because C is prime (part of the key AC)

A 3NF decomposition algorithm

- Given relation R and FD's F,
  - Find minimal basis for F, say G
  - For each FD $X \rightarrow A$ in G, use XA as the schema of one of the decomposed relations
  - Eliminate a relation if it is a subset of another
  - If none of the resulting schemas are superkeys, add one more relation whose schema is a key for R
- Previous example

A	B	C
1	2	3
3	2	3

F: $B \rightarrow C$, $AC \rightarrow B$

Why 3NF satisfy dependency preservation: each FD of the minimal basis has all its attributes in some relation in the decomposition

A 3NF decomposition algorithm

- Given relation R and FD's F,
  - Find minimal basis for F, say G
  - For each FD $X \rightarrow A$ in G, use XA as the schema of one of the decomposed relations
  - Eliminate a relation if it is a subset of another (textbook also implicitly uses this step)
  - If none of the resulting schemas are superkeys, add one more relation whose schema is a key for R
- Previous example

A	B	C
1	2	3
3	2	3

F: $B \rightarrow C$, $AC \rightarrow B$

G: $B \rightarrow C$, $AC \rightarrow B$

Keys: AC, AB

A 3NF decomposition algorithm

- Given relation R and FD’s F,
  - Find minimal basis for F, say G
  - For each FD $X \rightarrow A$ in G, use XA as the schema of one of the decomposed relations
  - Eliminate a relation if it is a subset of another (textbook also implicitly uses this step)
  - If none of the resulting schemas are superkeys, add one more relation whose schema is a key for R
- Previous example

A	B	C
1	2	3
3	2	3

F: $B \rightarrow C$, $AC \rightarrow B$
G: $B \rightarrow C$, $AC \rightarrow B$
Keys: AC, AB

No decomposition because

- R_1(A,B,C), R_2(B,C) are produced, but R_2 is a subset of R_1
- R_1(A,B,C) is trivially a superkey

Another example

- R(A,B,C,D,E) with FD's AB $\rightarrow$ C, C $\rightarrow$ B, A $\rightarrow$ D

Another example

- R(A,B,C,D,E) with FD’s $AB \rightarrow C$, $C \rightarrow B$, $A \rightarrow D$
- Convince yourself that the FD’s form a minimal basis

Another example

- $R(A,B,C,D,E)$ with FD's $AB \to C$, $C \to B$, $A \to D$
- Convince yourself that the FD's form a minimal basis
- Generate relations $R_1(A,B,C)$, $R_2(B,C)$, $R_3(A,D)$
- Remove $R_2(B,C)$ (subset of $R_1(A,B,C)$)

Another example

- R(A,B,C,D,E) with FD’s AB → C, C → B, A → D
- Convince yourself that the FD’s form a minimal basis
- Generate relations R_1(A,B,C), R_2(B,C), R_3(A,D)
- Remove R_2(B,C) (subset of R_1(A,B,C))
- Keys are ABE and ACE, so no relations are superkeys
- Add R_4(A,B,E) or R_4(A,C,E)

Another example

- $R(A,B,C,D,E)$ with FD's $AB \to C$, $C \to B$, $A \to D$
- Convince yourself that the FD's form a minimal basis
- Generate relations $R_1(A,B,C)$, $R_2(B,C)$, $R_3(A,D)$
- Remove $R_2(B,C)$ (subset of $R_1(A,B,C)$)
- Keys are ABE and ACE, so no relations are superkeys
- Add $R_4(A,B,E)$ or $R_4(A,C,E)$

Exercise #2

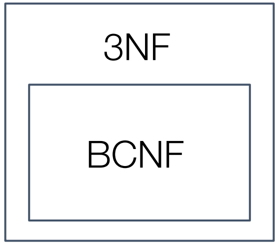
- What are the 3NF violations of the FDs?
- Decompose into relations satisfying 3NF

$\boxed{R(A,\ B,\ C,\ D)}$

FD's: $AB \rightarrow C$, $C \rightarrow D$, $D \rightarrow A$

BCNF versus 3NF

- Given a non-trivial FD X → B (X is a set of attributes)
  - BCNF: X must be a superkey
  - 3NF: X must be a superkey or B is prime
- Use 3NF over BCNF if you need dependency preservation
- However, 3NF may not remove all redundancies and anomalies



3NF relation:

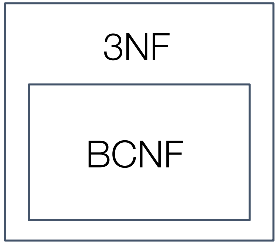
A	B	C
1	2	3
3	2	3
2	3	1

F: B → C, AC → B

Can have redundancy and update anomalies

BCNF versus 3NF

- Given a non-trivial FD X → B (X is a set of attributes)
  - BCNF: X must be a superkey
  - 3NF: X must be a superkey or B is prime
- Use 3NF over BCNF if you need dependency preservation
- However, 3NF may not remove all redundancies and anomalies



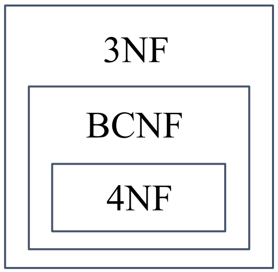
3NF relation:

A	B	C
1	2	3
3	2	3
2	3	1

F: B → C, AC → B

Can have deletion anomalies

Further Readings (Chapter 3.6)



- Multivalued Dependencies (MVD)
  - Two sets of attributes are independent
  - A generalization of FDs
- 4NF
  - Removes MVD redundancies

Property	3NF	BCNF	4NF
Lossless join	Yes	Yes	Yes
Eliminates FD redundancies	No	Yes	Yes
Eliminates MVD redundancies	No	No	Yes
Preserves FD’s	Yes	No	No
Preserves MVD’s	No	No	No

And beyond 4NF?

	UNF (1970)	1NF (1971)	2NF (1971)	3NF (1971)	EKNF (1982)	BCNF (1974)	4NF (1977)	ETNF (2012)	5NF (1979)	DKNF (1981)	6NF (2003)
Primary key (no duplicate tuples)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
No repeating groups	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Atomic columns (cells have single value)	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
No partial dependencies (values depend on the whole of every Candidate key)	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
No transitive dependencies (values depend only on Candidate keys)	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency involves either a superkey or an elementary key's subkey	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	N/A
No redundancy from any functional dependency	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	N/A
Every non-trivial, multi-value dependency has a superkey	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	N/A
A component of every explicit join dependency is a superkey[8]	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	N/A
Every non-trivial join dependency is implied by a candidate key	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	N/A
Every constraint is a consequence of domain constraints and key constraints	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	N/A
Every join dependency is trivial	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

https://en.wikipedia.org/wiki/Database_normalization

Summary

- Good schema design is important
  - Avoid redundancy and anomalies
  - Functional dependencies
- The solution is to decompose relations
  - BCNF gives elimination of anomalies and lossless join
  - 3NF gives lossless join and dependency preservation
- BCNF is intuitive and most widely used in practice