Normalization

CS 4750
Database Systems

[A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Ch.7]
[Ricardo and Urban, Database Illuminated, Ch.6]
[https://www.w3schools.in/dbms/database-normalization/]

Recap: FD

Consider a student_lecture relation

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

Assume that there is exactly one teaching assistant assigned to each student for every course

1. Determine all functional dependencies of the relation
2. Give an example of a super key and a candidate key

Recap: FD

Consider a student_lecture relation

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

1. Determine all functional dependencies of the relation

Recap: FD

Consider a student_lecture relation

S_id	Address	Course	Teaching_assistant
1234	57 Hockanum Blvd	Database Systems	Minnie
2345	1400 E. Bellows	Database Systems	Humpty
3456	900 S. Detroit	Cloud Computing	Dumpty
1234	57 Hockanum Blvd	Web Programming Lang.	Mickey
5678	2131 Forest Lake Ln.	Software Analysis	Minnie

2. Give an example of a super key and a candidate key

Recap: Attribute Closure

Given a relation R (A, B, C, D, E, F, G) with the following FDs

FDs = { A → BC, E → CF, B → E, CD → EF, A → G }

Find the closure of A (A+)

revisit

General Design Guidelines

- Semantics of attributes should self-evident
- Avoid redundancy – between tuples, relations
- Avoid NULL values in tuples
- If certain tuples should not exist, don't allow them

Database design = process or organizing data into a database model by considering data needs to be stored and the interrelationship of the data

Database design is about
characterizing data and the organizing data

How to describe properties we know or see in the data

How to organize data to promote ease of use and efficiency

Normalization

- Normalization = technique of organizing data in a database
- Two purposes:
  - Eliminating redundant data
    - Avoid storing the same data in multiple tables
  - Ensuring data dependencies make sense
    - Store data logically – only related data in a table, nothing else
- Need to refine schema

Schema Refinement

- Constraints, in particular functional dependencies, cause problems
- Must understand when and how constraints cause redundancy
- Refinement is needed when redundancy exists
- Decomposition – main refinement technique
  - Example: replace ABCD with AB and BCD or ACD and ABD
  - Judgment call:
    - Is there a reason to decompose a relation?
    - What problems (if any) does the decomposition cause?

Decomposition

Suppose a relation R contains attribute A_1, …, A_n. A decomposition of R consists of replacing R by two or more relations such that

- Each new relation schema contains a subset of the attributes of R (and no attribute that do not appear in R)
- Every attribute of R appears as an attribute of at least one of the new relations

Tradeoff: must consider these issues vs. redundancy

Three potential problems:

- Some queries become more expensive
- Given instances of the decomposed relations, we may not be able to reconstruct the original relation
- Checking some dependencies may require joining the instances of the decomposed relations

Properties of Decomposition

Lossless join

- Employee = R1 ⋈ R2   (⋈ "natural join")
- No gain or loose columns
- R1 ∩ R2 ≠ { }
- R1 ∩ R2 → R1 or R1 ∩ R2 → R2 (R1 ∩ R2 is a super key of R1 or R2)

Dependency preserving

- Every dependency is in the same relation (thus, when checking a dependency, no need to join tables)

No redundancy

- (solved through the normal forms)

Lossless-Join Decomposition

Employee

computingID	name	year	hourly_rate	hours_worked
ht1y	Humpty	4	12	20
dt2y	Dumpty	3	10	20
md3y	Mickey	4	12	15
mn4e	Minnie	4	12	16
dh5h	Duhhuh	3	10	10

Employee $= R1 \bowtie R2$

No gain or loose columns

$R1 \cap R2 \neq \{\ \}$

Super key

= R1

computingID	name	year	hours_worked
ht1y	Humpty	4	20
dt2y	Dumpty	3	20
md3y	Mickey	4	15
mn4e	Minnie	4	16
dh5h	Duhhuh	3	10

$\bowtie$ R2

year	hourly_rate
4	12
3	10

Lossless-Join Decomposition

R1

computingID	name	year	hours_worked
ht1y	Humpty	4	20
dt2y	Dumpty	3	20
md3y	Mickey	4	15
mn4e	Minnie	4	16
dh5h	Duhhuh	3	10

$\bowtie$

R2

year	hourly_rate
4	12
3	10

=

Employee

computingID	name	year	hourly_rate	hours_worked
ht1y	Humpty	4	12	20
dt2y	Dumpty	3	10	20
md3y	Mickey	4	12	15
mn4e	Minnie	4	12	16
dh5h	Duhhuh	3	10	10

Exercise: Lossless Join decomposition?

A	B	C
1	2	3
4	5	6
7	2	8

decomposed

A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8

natural join

A	B	C
1	2	3
1	2	8
4	5	6
7	2	3
7	2	8

Exercise: Lossless Join decomposition?

A	B	C
1	2	3
4	5	6
7	2	3

decomposed

A	B
1	2
4	5
7	2

B	C
2	3
5	6

natural join

A	B	C
1	2	3
4	5	6
7	2	3

Normalization

- **1NF**: each column is atomic, flat
- **2NF**: 1NF + no partial dependency -- *[outdated]*
- **3NF**: 2NF + lossless-join + dependency preserving -- *[our focus]*
- **BCNF**: 1NF + lossless join + redundancy free -- *[our focus]*
- **4NF:** no multi-valued dependency -- *[out of CS 4750 scope]*
- **5NF**: 4NF + cannot be further non loss decomposed -- *[out of CS 4750 scope – too complicated]*
- **6NF**: 5NF + every join dependency is trivial -- *[out of CS 4750 scope – somewhat unrealistic]*

First Normal Form (1NF)

- Every attribute/column has a single (atomic) value
- Values stored in a column should be of the same domain
- The order in which data is stored does not matter

computingID	name	phone	department
ht1y	Humpty	111-111-1111	Computer Science, Math
dt2y	Dumpty	222-222-2222	Biology
md3			
mn4			

computingID	name	phone	department
ht1y	Humpty	111-111-1111	Computer Science
ht1y	Humpty	111-111-1111	Math
dt2y	Dumpty	222-222-2222	Biology
md3y	Mickey	333-333-3333	Computer Science
mn4e	Minnie	444-444-4444	Computer Science

Second Normal Form (2NF)

- 1NF + no partial dependency (FDs)

Suppose we also know course → instructor

computingID	course	grade	instructor
ht1y	cs1	B+	someone1
dt2y	cs1	A-	someone1
dt2y	cs2	A	someone2
md3y	cs1	A	someone1
mn4e	cs2	B	someone2
md3y	cs2	A	someone2

Let's simplify the name: R (A, B, C, D), AB is a candidate key
FDs { AB → C, AB → D, B → D }

Since B is part of a candidate key, D depends on a part of a key
"Partial dependency"

Note: many-to-many relationship

Second Normal Form (2NF)

To convert the table into 2NF, decompose the table to remove partial dependency

computingID	course	grade	instructor
ht1y	cs1	B+	someone1
dt2y	cs1	A-	someone1
dt2y	cs2	A	someone2
		A	someone1
		B	
		A	

computingID	course	grade
ht1y	cs1	B+
dt2y	cs1	A-
dt2y	cs2	A
md3y	cs1	A
mn4e	cs2	B
md3y	cs2	A

course	instructor
cs1	someone1
cs2	someone2

Non-key attributes must depend upon the whole of the candidate key

Third Normal Form (3NF)

- 2NF + lossless-join + dependency preserving

No transitive dependencies

computingID	course	grade	textbook_id	title
ht1y	cs1	B+	book1	Intro to Python
dt2y	cs1	A-	book1	Intro to Python
dt2y	cs2	A	book2	Intro to Java
md3y	cs1	A	book1	Intro to Python
mn4e	cs2	B	book2	Intro to Java
md3y	cs2	A	book2	Intro to Java

Suppose we want to keep track of textbook_id and title for the course

Let's simplify the name: R (A, B, C, D, E), AB is a candidate key
FDs { AB → C, AB → D, D → E }

Since AB → D, D → E, D and E are non keys -- "transitive dependency"

Problem with dependency … Fix it!

Third Normal Form (3NF)

To convert the table into 3NF, decompose the table to remove transitive dependency

computingID	course	grade	textbook_id	title
ht1y	cs1	B+	book1	Intro to Python
dt2y	cs1	A-	book1	Intro to Python
dt2y	cs2	A	book2	Intro to Java
			book1	Intro to Python
			book2	Intro to Java

computingID	course	grade	textbook_id
ht1y	cs1	B+	book1
dt2y	cs1	A-	book1
dt2y	cs2	A	book2
md3y	cs1	A	book1
mn4e	cs2	B	book2
md3y	cs2	A	book2

textbook_id	title
book1	Intro to Python
book2	Intro to Java

Reduce redundant data; ensure data integrity;
no transitive dependency; dependency is in the same relation

Preserve all FDs but allow anomalies (may have redundancy)

Boyce-Codd Normal Form (BCNF)

- 1NF + lossless-join + redundant free
- For every non-trivial dependency, $X \rightarrow A$, X is a superkey.

All dependencies must be from full key

computingID	course	instructor
ht1y	cs1	someone1
dt2y	cs1	someone1
dt2y	cs2	someone2
md3y	cs1	someone3
mn4e	cs2	someone2
md3y	cs2	someone2

Suppose we know that instructor → course

Cannot have a non-key implies a key

Let's simplify the name: R (A, B, C), AB is a candidate key
FDs { AB → C, C → B }

Since C → B, non-key implies a (part of) key

Not satisfy BCNF -- Fix it!

Boyce-Codd Normal Form (BCNF)

To convert the table into BCNF, decompose the table to remove non-keys that imply a key – to make all dependencies from a key

computingID	course	instructor
ht1y	cs1	someone1
dt2y	cs1	someone1
dt2y	cs2	someone2
		meone3
		meone2
		meon

computingID	instructor
ht1y	someone1
dt2y	someone1
dt2y	someone2
md3y	someone3
mn4e	someone2
md3y	someone2

Given

FDs { $\underline{AB \rightarrow C}$, C → B }

lost FD

instructor	course
someone1	cs1
someone2	cs2
someone3	cs1

Remove redundant data; ensure data integrity; may have dependency across relations (need to join to check dependency)

No transitive FDs, no non-key dependencies, but can lose FDs

Recap: Normal Forms

What is the main aspect of 1NF?

Which among 3NF and BCNF is dependency preserving?

Which among 3NF and BCNF is redundancy free?

What is common between 3NF and BCNF?

Wrap-Up

- Properties of Decomposition
  - Lossless join
  - Dependency preserving
  - No redundancy
- Overview of normal forms

What's next?

- 3NF and decomposition
- BCNF and decomposition