ADMINISTRIVIA

• WQ6 due tonight

• HW7 due Wednesday
DATABASE DESIGN
PROCESS

Conceptual Model:

Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details
Physical Schema
ELIMINATING ANOMALIES

Main idea:

X → A is OK if X is a (super)key

X → A is bad otherwise

• Need to decompose the table, but how?

Boyce-Codd Normal Form
There are no “bad” FDs:

**Definition.** A relation R is in BCNF if:

Whenever $X \rightarrow B$ is a non-trivial dependency, then $X$ is a superkey.

Equivalently:

**Definition.** A relation R is in BCNF if:

$\forall X$, either $X^+ = X$ or $X^+ = \{\text{all attributes}\}$
**BCNF DECOMPOSITION**

**ALGORITHM**

```
Normalize(R)
    find X s.t.: X ≠ X⁺ and X⁺ ≠ [all attributes]
    if (not found) then “R is in BCNF”
    let Y = X⁺ - X;   Z = [all attributes] - X⁺
    decompose R into R1(X ∪ Y) and R2(X ∪ Z)
    Normalize(R1);  Normalize(R2);
```
EXAMPLE: BCNF

R(A,B,C,D)

A → B
B → C

R(A,B,C,D)
EXAMPLE: BCNF

Recall: find $X$ s.t. $X \subsetneq X^+ \subsetneq \text{[all-attrs]}$

$R(A,B,C,D)$

$A \rightarrow B$

$B \rightarrow C$
EXAMPLE: BCNF

\[ R(A,B,C,D) \]

\[ A \rightarrow B \]

\[ B \rightarrow C \]

\[ A^+ = ABC \neq ABCD \]
EXAMPLE: BCNF

\[ R(A,B,C,D) \]

\[ A^+ = ABC \neq ABCD \]

\[ R_1(A,B,C) \]

\[ R_2(A,D) \]

\[ A \rightarrow B \]
\[ B \rightarrow C \]
EXAMPLE: BCNF

\[ R(A, B, C, D) \]
\[ A^+ = ABC \neq ABCD \]

\[ R_1(A, B, C) \]
\[ R_2(A, D) \]
R(A,B,C,D)

**EXAMPLE: BCNF**

\[ R(A,B,C,D) \]
\[ A^+ = ABC \neq ABCD \]
\[ R_1(A,B,C) \]
\[ B^+ = BC \neq ABC \]
\[ R_2(A,D) \]
What happens if in R we first pick B⁺? Or AB⁺?
R(A,B,C,D)

**EXAMPLE: BCNF**

\[ R(A,B,C,D) \]
\[ A^+ = ABC \neq ABCD \]

\[ R_1(A,B,C) \]
\[ B^+ = BC \neq ABC \]

\[ R_{11}(B,C) \]
\[ R_{12}(A,B) \]

\[ R_2(A,D) \]

What are the keys?

A → B
B → C
EXAMPLE: BCNF

R(A,B,C,D)

A → B
B → C

R(A,B,C,D)
A⁺ = ABC ≠ ABCD

R₁(A,B,C)
B⁺ = BC ≠ ABC

R₁₁(B,C)

R₂(A,D)

R₁₂(A,B)

What are the keys?
R(A,B,C,D)

EXAMPLE: BCNF

R(A,B,C,D)
A⁺ = ABC ≠ ABCD

R₁(A,B,C)
B⁺ = BC ≠ ABC

R₁₁(B,C)

R₁₂(A,B)

R₂(A,D)

What are the keys?

A → B
B → C
DECOMPOSITIONS IN GENERAL

\[ R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p) \]

\[ S_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \]
\[ S_2(A_1, \ldots, A_n, C_1, \ldots, C_p) \]

\[ S_1 = \text{projection of } R \text{ on } A_1, \ldots, A_n, B_1, \ldots, B_m \]
\[ S_2 = \text{projection of } R \text{ on } A_1, \ldots, A_n, C_1, \ldots, C_p \]

and \( R \) is a subset of \( S_1 \times S_2 \)
## LOSSLESS DECOMPOSITION

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

### First Decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
</tr>
</tbody>
</table>

### Second Decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>
What is lossy here?

Lossy Decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

Name | Category
--- | ---
Gizmo | Gadget
OneClick | Camera
Gizmo | Camera

<table>
<thead>
<tr>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>
# LOSSY DECOMPOSITION

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>
DECOMPOSITION IN GENERAL

Let: \( S_1 = \text{projection of } R \text{ on } A_1, \ldots, A_n, B_1, \ldots, B_m \)

\( S_2 = \text{projection of } R \text{ on } A_1, \ldots, A_n, C_1, \ldots, C_p \)

The decomposition is called **lossless** if \( R = S_1 \bowtie S_2 \)

Fact: If \( A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m \) then the decomposition is lossless

It follows that every BCNF decomposition is lossless
IS THIS LOSSLESS?

If we decompose $R$ into $\Pi_{S_1}(R)$, $\Pi_{S_2}(R)$, $\Pi_{S_3}(R)$, …
Is it true that $S_1 \bowtie S_2 \bowtie S_3 \bowtie … = R$?

That is true if we can show that:

$R \subseteq S_1 \bowtie S_2 \bowtie S_3 \bowtie …$ always holds (why?)

$R \supseteq S_1 \bowtie S_2 \bowtie S_3 \bowtie …$ neet to check
THE CHASE TEST FOR LOSSLESS JOIN

Example from textbook Ch. 3.4.2

R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)
R satisfies: A \rightarrow B, B \rightarrow C, CD \rightarrow A

S1 = \Pi_{AD}(R), S2 = \Pi_{AC}(R), S3 = \Pi_{BCD}(R),

hence R \subseteq S1 \bowtie S2 \bowtie S3

Need to check: R \supseteq S1 \bowtie S2 \bowtie S3
Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

\[ R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D) \]

R satisfies: \( A \rightarrow B, B \rightarrow C, CD \rightarrow A \)

S1 = \( \Pi_{AD}(R) \), S2 = \( \Pi_{AC}(R) \), S3 = \( \Pi_{BCD}(R) \),

therefore \( R \subseteq S1 \bowtie S2 \bowtie S3 \)

Need to check: \( R \supseteq S1 \bowtie S2 \bowtie S3 \)

Suppose \((a,b,c,d) \in S1 \bowtie S2 \bowtie S3\) Is it also in R?
THE CHASE TEST FOR LOSSLESS JOIN

Example from textbook Ch. 3.4.2

\[ R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D) \]
R satisfies: \( A \rightarrow B \), \( B \rightarrow C \), \( CD \rightarrow A \)

\[ S1 = \Pi_{AD}(R), S2 = \Pi_{AC}(R), S3 = \Pi_{BCD}(R), \]
hence \( R \subseteq S1 \bowtie S2 \bowtie S3 \)
Need to check: \( R \supseteq S1 \bowtie S2 \bowtie S3 \)
Suppose \( (a,b,c,d) \in S1 \bowtie S2 \bowtie S3 \) Is it also in \( R \)?
R must contain the following tuples:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
</tr>
</tbody>
</table>

Why ?
\( (a,d) \in S1 = \Pi_{AD}(R) \)
Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

\[ R(A,B,C,D) = S_1(A,D) \bowtie S_2(A,C) \bowtie S_3(B,C,D) \]

R satisfies: \( A \rightarrow B, \ B \rightarrow C, \ CD \rightarrow A \)

\( S_1 = \Pi_{AD}(R), \ S_2 = \Pi_{AC}(R), \ S_3 = \Pi_{BCD}(R), \)
hence \( R \subseteq S_1 \bowtie S_2 \bowtie S_3 \)

Need to check: \( R \supseteq S_1 \bowtie S_2 \bowtie S_3 \)

Suppose \((a,b,c,d) \in S_1 \bowtie S_2 \bowtie S_3\) Is it also in R?

R must contain the following tuples:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
<td>c</td>
<td>d2</td>
</tr>
</tbody>
</table>

Why?
Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

\[ R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D) \]
R satisfies: \( A \rightarrow B, \ B \rightarrow C, \ CD \rightarrow A \)

\( S1 = \Pi_{AD}(R) \), \( S2 = \Pi_{AC}(R) \), \( S3 = \Pi_{BCD}(R) \),

\[ R \subseteq S1 \bowtie S2 \bowtie S3 \]

Need to check: \( R \supseteq S1 \bowtie S2 \bowtie S3 \)

Suppose \((a,b,c,d) \in S1 \bowtie S2 \bowtie S3\) Is it also in \( R \)?

R must contain the following tuples:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
<td>c</td>
<td>d2</td>
<td></td>
</tr>
<tr>
<td>a3</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

Why?

- \((a,d) \in S1 = \Pi_{AD}(R)\)
- \((a,c) \in S2 = \Pi_{BD}(R)\)
- \((b,c,d) \in S3 = \Pi_{BCD}(R)\)
Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

\[ R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D) \]

R satisfies: \( A \rightarrow B, B \rightarrow C, CD \rightarrow A \)

\( S1 = \Pi_{AD}(R), S2 = \Pi_{AC}(R), S3 = \Pi_{BCD}(R), \)

hence \( R \subseteq S1 \bowtie S2 \bowtie S3 \)

Need to check: \( R \supseteq S1 \bowtie S2 \bowtie S3 \)

Suppose \( (a,b,c,d) \in S1 \bowtie S2 \bowtie S3 \) Is it also in \( R \)?

R must contain the following tuples:

“Chase” them (apply FDs):

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
<td>c</td>
<td>d2</td>
</tr>
<tr>
<td>a3</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

Why ?

\( (a,d) \in S1 = \Pi_{AD}(R) \)
\( (a,c) \in S2 = \Pi_{BD}(R) \)
\( (b,c,d) \in S3 = \Pi_{BCD}(R) \)
Example from textbook Ch. 3.4.2

THE CHASE TEST FOR LOSSLESS JOIN

R(A,B,C,D) = S1(A,D) \Join S2(A,C) \Join S3(B,C,D)
R satisfies: A\rightarrow B, B\rightarrow C, CD\rightarrow A

S1 = \Pi_{AD}(R), S2 = \Pi_{AC}(R), S3 = \Pi_{BCD}(R),
hence R \subseteq S1 \Join S2 \Join S3
Need to check: R \supseteq S1 \Join S2 \Join S3
Suppose (a,b,c,d) \in S1 \Join S2 \Join S3 Is it also in R?
R must contain the following tuples:

“Chase” them (apply FDs):

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>b1</td>
<td>c</td>
<td>d2</td>
</tr>
<tr>
<td>a3</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

Why ?
(a,d) \in S1 = \Pi_{AD}(R)
(a,c) \in S2 = \Pi_{BD}(R)
(b,c,d) \in S3 = \Pi_{BCD}(R)
Example from textbook Ch. 3.4.2

**THE CHASE TEST FOR LOSSLESS JOIN**

\[ R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D) \]

R satisfies: \( A \rightarrow B, B \rightarrow C, CD \rightarrow A \)

S1 = \( \Pi_{AD}(R) \), S2 = \( \Pi_{AC}(R) \), S3 = \( \Pi_{BCD}(R) \),

hence \( R \subseteq S1 \bowtie S2 \bowtie S3 \)

Need to check: \( R \supseteq S1 \bowtie S2 \bowtie S3 \)

Suppose \((a,b,c,d) \in S1 \bowtie S2 \bowtie S3\) Is it also in R?

R must contain the following tuples:

"Chase" them (apply FDs):

\[
\begin{array}{cccc}
A & B & C & D \\
\hline
a & b1 & c1 & d \\
a & b2 & c & d2 \\
a3 & b & c & d \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & C & D \\
\hline
a & b1 & c & d \\
a & b1 & c & d2 \\
a3 & b & c & d \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & C & D \\
\hline
a & b1 & c & d \\
a & b1 & c & d2 \\
a & b & c & d \\
\end{array}
\]

Hence R contains \((a,b,c,d)\)
SCHEMA REFINEMENTS = NORMAL FORMS

• 1st Normal Form = all tables are flat
• 2nd Normal Form = no FD with “non-prime” attributes
  • Obsolete
  • Prime attributes: attributes part of a key
• Boyce Codd Normal Form = no “bad” FDs
  • Are there problems with BCNF?
DEPENDENCY PRESERVATION

- Bookings(title, theatre, city)
  - FD:
    - theatre -> city
    - title, city -> theatre
- What are the keys?
DEPENDENCY PRESERVATION

• Bookings(title, theatre, city)
  • FD:
    • theatre -> city
    • title, city -> theatre
• What are the keys?
  • None of the single attributes
  • {title, city}, {theatre, title}
• BCNF?
DEPENDENCY PRESERVATION

- Bookings(title, theatre, city)
  - FD:
    - theatre -> city
    - title, city -> theatre
- What are the keys?
  - None of the single attributes
  - {title, city}, {theatre, title}
- BCNF?
  - No, {theatre} is neither a trivial dependency nor a superkey
  - Decompose?
**DEPENDENCY PRESERVATION**

- **Bookings**(title, theatre, city)
  - FD:
    - theatre -> city
    - title, city -> theatre
  - What are the keys?
    - None of the single attributes
    - \{title, city\}, \{theatre, title\}
  - **BCNF?**
    - No, \{theatre\} is neither a trivial dependency nor a superkey
    - Decompose? R1(theatre, city) R2(theatre, title)
    - What’s wrong? *(think of FDs)*
DEPENDENCY PRESERVATION

• **Bookings**(title, theatre, city)
  • FD:
    • theatre -> city
    • title, city -> theatre

• What are the keys?
  • None of the single attributes
  • {title, city}, {theatre, title}

• **BCNF**?
  • No, {theatre} is neither a trivial dependency nor a superkey
  • Decompose? R1(theatre, city) R2(theatre, title)
  • What’s wrong? (think of FDs)
  • We can’t guarantee title, city -> theatre with *simple* constraints
    (now need to join)
NORMA L FORMS

• 3\textsuperscript{rd} Normal form
  • Allows tables with BCNF violations if a decomposition separates an FD
  • Can result in redundancy

• 4\textsuperscript{th} Normal form
  • Multi-valued dependencies
    • Incorporate info about attributes in neither A nor B
    • All MVDs are also FDs
  • Apply BCNF alg with MVD and FD
NORMAL FORMS

• 5\textsuperscript{th} Normal Form
  • Join dependency
    • Lossless/exact joining
    • Join independent Tables

• 6\textsuperscript{th} Normal Form
  • Only allow trivial join dependencies
  • Only need key/tuple constraints to represent all constraints
KEY POINTS

• Produce and verify FDs, superkeys, keys
• Be able to decompose a table into BCNF
• Flaws of 1NF & BCNF
• Identify loss and be able to apply the chase test
IMPLEMENTATION

We learned about how to normalize tables to avoid anomalies.

How can we implement normalization in SQL if we can’t modify existing tables?

- This might be due to legacy applications that rely on previous schemas to run.
- Can recover original tables via join on demand and we want those available to queries.
VIEWS

A **view** in SQL =

- A table computed from other tables, s.t., whenever the base tables are updated, the view is updated too

More generally:

- A **view** is derived data that keeps track of changes in the original data

Compare:

- A **function** computes a value from other values, but does not keep track of changes to the inputs
A SIMPLE VIEW

Create a view that returns for each store the prices of products purchased at that store

CREATE VIEW StorePrice AS
    SELECT DISTINCT x.store, y.price
    FROM Purchase x, Product y
    WHERE x.product = y.pname
WE USE A VIEW LIKE ANY TABLE

A "high end" store is a store that sells some products over 1000.

For each customer, return all the high end stores that they visit.

```
SELECT DISTINCT u.customer, u.store
FROM Purchase u, StorePrice v
WHERE u.store = v.store
    AND v.price > 1000
```
TYPES OF VIEWS

Virtual views
- Computed only on-demand – slow at runtime
- Always up to date

Materialized views
- Pre-computed offline – fast at runtime
- May have stale data (must recompute or update)

The key components of physical tuning of databases are the selection of materialized views and indexes
MATERIALIZED VIEWS

CREATE MATERIALIZED VIEW View_name

BUILD [IMMEDIATE/DEFERRED]

REFRESH [FAST/COMPLETE/FORCE]

ON [COMMIT/Demand]

AS Sql_query

• **Immediate v deferred**
  • Build immediately, or after a query

• **Fast v. Complete v. Force**
  • Level of refresh – log based v. complete rebuild

• **Commit v. Demand**
  • Commit: after data is added
  • Demand: after conditions are set (time is common)
CONCLUSION

Poor schemas can lead to bugs and inefficiency

E/R diagrams are means to structurally visualize and design relational schemas

Normalization is a principled way of converting schemas into a form that avoid such problems

BCNF is one of the most widely used normalized form in practice