## Relational Database Design and SQL Basics

CPS 216
Advanced Database Systems

### Relational design: a review

- Identifying tuples: keys
- Generalizing the key concept: FDs
- Non-key FDs: redundancy
- Avoiding redundancy: BCNF decomposition
- Preserving FDs: 3NF

---

### BNCF = no redundancy?

- **Student** (*SID, CID, club*)
  - Suppose your classes have nothing to do with the clubs you join
  - FDs?
  - BNCF?
  - Redundancies?

<table>
<thead>
<tr>
<th>SID</th>
<th>CID</th>
<th>club</th>
</tr>
</thead>
<tbody>
<tr>
<td>140</td>
<td>CPS 216</td>
<td>ballet</td>
</tr>
<tr>
<td>140</td>
<td>CPS 216</td>
<td>sumo</td>
</tr>
<tr>
<td>123</td>
<td>CPS 214</td>
<td>tennis</td>
</tr>
<tr>
<td>123</td>
<td>CPS 214</td>
<td>sumo</td>
</tr>
<tr>
<td>123</td>
<td>CPS 216</td>
<td>chess</td>
</tr>
<tr>
<td>123</td>
<td>CPS 216</td>
<td>golf</td>
</tr>
</tbody>
</table>
Multi-valued dependencies

- A multi-valued dependency (MVD) has the form $X \rightarrow\rightarrow Y$, where $X$ and $Y$ are sets of attributes in a relation $R$
- $X \rightarrow\rightarrow Y$ means that whenever two tuples in $R$ agree on all the attributes of $X$, then we can swap their $Y$ components and get two new tuples that are also in $R$

MVD examples

$Student (SID, CID, club)$

Complete MVD + FD rules

- FD reflexivity, augmentation, and transitivity
- MVD complementation:
  If $X \rightarrow\rightarrow Y$, then $X \rightarrow\rightarrow \text{attrs}(R) - X - Y$  Try proving
- MVD augmentation:
  If $X \rightarrow\rightarrow Y$ and $V \subseteq W$, then $XW \rightarrow\rightarrow YV$  with these!?  
- MVD transitivity:
  If $X \rightarrow\rightarrow Y$ and $Y \rightarrow\rightarrow Z$, then $X \rightarrow\rightarrow Z - Y$
- Replication (FD is MVD):
  If $X \rightarrow Y$, then $X \rightarrow\rightarrow Y$
- Coalescence:
  If $X \rightarrow\rightarrow Y$ and $Z \subseteq Y$ and there is some $W$ disjoint from $Y$ such that $W \rightarrow Z$, then $X \rightarrow Z$
An elegant solution: chase

- Given a set of FDs and MVDs $D$, does another dependency $d$ (FD or MVD) follow from $D$?
- Procedure
  - Start with the hypotheses of $d$, and treat them as “seed” tuples in a relation
  - Apply the given dependencies in $D$ repeatedly
    - If we apply an FD, we infer equality of two symbols
    - If we apply an MVD, we infer more tuples
  - If we infer the conclusion of $d$, we have a proof
  - Otherwise, if nothing more can be inferred, we have a counterexample

---

Proof by chase

- In $R(A, B, C, D)$, does $A \rightarrow B$ and $B \rightarrow C$ imply $A \rightarrow C$?

---

Counterexample by chase

- In $R(A, B, C, D)$, does $A \rightarrow BC$ and $CD \rightarrow B$ imply $A \rightarrow B$?
4NF

A relation R is in Fourth Normal Form (4NF) if
- For every non-trivial MVD $X \rightarrow\rightarrow Y$ in $R$, $X$ is a super key
- That is, all FDs and MVDs follow from “key $\rightarrow$ other attributes”

- 4NF is stronger than BCNF

4NF decomposition algorithm

- Find a 4NF violation
  - A non-trivial MVD $X \rightarrow\rightarrow Y$ in $R$ where $X$ is not a super key
- Decompose $R$ into $R_1$ and $R_2$, where
  - $R_1$ has attributes $X \cup Y$
  - $R_2$ has attributes $X \cup Z$ ($Z$ contains attributes not in $X$ or $Y$)
- Repeat until all relations are in 4NF

- Almost identical to BCNF decomposition algorithm
- Any decomposition on a 4NF violation is lossless

4NF decomposition example

<table>
<thead>
<tr>
<th>ID</th>
<th>CID</th>
<th>club</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>CPS 216</td>
<td>ballet</td>
</tr>
<tr>
<td>142</td>
<td>CPS 214</td>
<td>sumo</td>
</tr>
<tr>
<td>142</td>
<td>CPS 214</td>
<td>sumo</td>
</tr>
<tr>
<td>123</td>
<td>CPS 216</td>
<td>chess</td>
</tr>
<tr>
<td>103</td>
<td>CPS 216</td>
<td>golf</td>
</tr>
</tbody>
</table>

Student (SID, CID, club)
3NF, BCNF, and 4NF

<table>
<thead>
<tr>
<th></th>
<th>3NF</th>
<th>BCNF</th>
<th>4NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preserves FDs?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Redundancy due to FDs?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Redundancy due to MVDs?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Recap

- Another source of redundancy: MVDs
- Reasoning about FDs and MVDs: chase
- Avoiding redundancy due to MVDs: 4NF

A complete design example

- Information about parts and assemblies for a manufacturing company; e.g.:
  - A bicycle consists of one frame and two wheels; the cost of assembly is $30
  - A frame is just a basic part
  - A wheel consists of one tire, one rim, and 48 spokes; the cost of assembly is $40
  - Everything has a part ID and a name
Entities and relationships

- Entities
- Relationships

Identify constraints

Design relational schema

- Entities to relations
- Relationships to relations
Encode constraints

- Part (ID, name)
- Assembly (ID, cost)
- ComposedOf (assemblyID, partID, number)
- Any missing constraints?

Apply relational design theory

- Part (ID, name)
  - ID is a key
- Assembly (ID, cost)
  - ID is a key
- ComposedOf (assemblyID, partID, number)
  - \{assemblyID, partID\} is a key
- 3NF? BCNF? 4NF?

Populate schema with data

<table>
<thead>
<tr>
<th>Part</th>
<th>Assembly</th>
<th>ComposedOf</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>name</td>
<td>ID</td>
</tr>
<tr>
<td>1</td>
<td>bicycle</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>frame</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>wheel</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>tire</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>rim</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>spoke</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>cost</th>
<th>assemblyID</th>
<th>partID</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>3</td>
<td>6</td>
<td>48</td>
</tr>
</tbody>
</table>

... ... ...

... ... ...

... ... ...
Good design principles

- Avoid redundancy
- Avoid decomposing too much
- KISS
  - Focus on the task and avoid over-design

SQL

- SQL: Structured Query Language
  - Pronounced “S-Q-L” or “sequel”
  - The query language of every commercial DBMS
- A brief history
  - System R
  - SQL89
  - SQL92 (SQL2)
  - SQL3 (still under construction)

Table creation

- CREATE TABLE table_name
  (... , column_name, column_type, ...);
- Example
  - create table Student (SID integer, name varchar(30), email varchar(30), age integer, GPA float);
  - create table Course (CID char(10), title varchar(100));
  - create table Enroll (SID integer, CID char(10));

SQL is case insensitive
Key declaration

- At most one PRIMARY KEY per table
  - Typically implies a primary index
  - Rows are stored inside the index, typically sorted by primary key value
- Any number of UNIQUE keys per table
  - Typically implies a secondary index
  - Pointers to rows are stored inside the index

Key declaration examples

- create table Student
  (SID integer primary key,
   name varchar(30),
   email varchar(30) unique,
   age integer, GPA float);
- create table Course
  (CID char(10) primary key,
   title varchar(100));
- create table Enroll
  (SID integer, CID char(10),
   primary key(SID, CID));

SFW queries

- SELECT $A_1$, $A_2$, …, $A_n$
  FROM $R_1$, $R_2$, …, $R_m$
  WHERE condition;
- Also called an SPJ (select-project-join) query
- Equivalent (more or less) to relational algebra query
Example: reading a table

- SELECT * FROM Student;
  - "*" is a shorthand for all columns
  - WHERE clause is optional

Example: selection and projection

- Names of students under 18
- When was Lisa born?

- SELECT list can contain calculations
- String literals are enclosed in single quotes (case sensitive)

Example: join

- SIDs and names of students taking courses with the word “Database” in their titles

- Okay to omit the table_name in table_name.column_name if column name is unique
- Many, many more built-in predicates such as LIKE
Example: rename

- SIDs of all pairs of classmates

  - AS is optional; in fact Oracle doesn’t like it in the FROM clause

Set versus bag semantics

- Set
  - No duplicates
  - Relational model uses set semantics

- Bag
  - Duplicates allowed
  - Number of duplicates is significant
  - SQL uses bag semantics by default

Set versus bag example

\[ \pi_{\text{SID}} (\text{Enroll}) \]

SELECT SID FROM Enroll;
A case for bag semantics

- Efficiency
- Which one is more useful?
- Besides, SQL provides the option of set semantics with DISTINCT

Example: forcing set semantics

- SIDs of all pairs of classmates
  - SELECT e1.SID as SID1, e2.SID as SID2
    FROM Enroll as e1, Enroll as e2
    WHERE e1.CID = e2.CID
    AND e1.SID > e2.SID;
  - Duplicates?
    - SELECT DISTINCT e1.SID as SID1, e2.SID as SID2
      FROM Enroll as e1, Enroll as e2
      WHERE e1.CID = e2.CID
      AND e1.SID > e2.SID;
  - No duplicates

Operational semantics of SFW

- SELECT [DISTINCT] E₁, E₂, …, Eₙ
  FROM R₁, R₂, …, Rₘ
  WHERE condition;
- For each t₁ in R₁:
  - For each t₂ in R₂: … …
  - For each tₘ in Rₘ:
    - If condition is true over t₁, t₂, …, tₘ:
      Compute and output E₁, E₂, …, Eₙ
    - If DISTINCT is present
      Eliminate duplicates in output
What’s next

More SQL
  • Set/bag operations
  • Joins
  • Subqueries
  • Aggregates
  • NULL
  • Modification statements