Multivalued Dependencies

Winter 2006-2007
Lecture 22
Multivalued Attributes

• E-R schemas can have multivalued attributes
• 1NF requires only atomic attributes
  – Not a problem; translating to relational model leaves everything atomic
• Employee example:
  employee(emp_id, emp_name)
  emp_deps(emp_id, dependent)
  emp_nums(emp_id, phone_num)
• What are the requirements on these schemas for what tuples must appear?
Multivalued Attributes (2)

- Example data:

<table>
<thead>
<tr>
<th>emp_id</th>
<th>emp_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>125623</td>
<td>Rick</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>emp_id</th>
<th>dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td>125623</td>
<td>Jeff</td>
</tr>
<tr>
<td>125623</td>
<td>Alice</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>emp_id</th>
<th>phone_num</th>
</tr>
</thead>
<tbody>
<tr>
<td>125623</td>
<td>555-8888</td>
</tr>
<tr>
<td>125623</td>
<td>555-2222</td>
</tr>
</tbody>
</table>

- Every distinct value of multivalued attribute requires a separate tuple, including associated value of emp_id

- A consequence of 1NF, in fact!
  - If attributes could be nonatomic, could just store list of values in the appropriate column!
  - 1NF requires extra tuples to represent multivalues
Independent Multivalued Attributes

• Question is trickier when a schema stores several *independent* multivalued attributes

• Proposed combined schema:
  
  employee(emp_id, emp_name)
  
  emp_info(emp_id, dependent, phone_num)

• What tuples must appear in emp_info?
  
  – *emp_info* is a relation
  
  – If an employee has M dependents and N phone numbers, *emp_info* must contain M×N tuples
  
  – Every combination of the employee’s dependents and their phone numbers
Independent Multivalued Attributes

- Example data:

<table>
<thead>
<tr>
<th>emp_id</th>
<th>emp_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>125623</td>
<td>Rick</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>emp_id</th>
<th>dependent</th>
<th>phone_num</th>
</tr>
</thead>
<tbody>
<tr>
<td>125623</td>
<td>Jeff</td>
<td>555-8888</td>
</tr>
<tr>
<td>125623</td>
<td>Jeff</td>
<td>555-2222</td>
</tr>
<tr>
<td>125623</td>
<td>Alice</td>
<td>555-8888</td>
</tr>
<tr>
<td>125623</td>
<td>Alice</td>
<td>555-2222</td>
</tr>
</tbody>
</table>

- Clearly has unnecessary redundancy
- Can’t formulate functional dependencies to represent multivalued attributes
- Can’t use BCNF or 3NF decompositions to eliminate redundancy in these cases
Dependencies

- Functional dependencies rule out what tuples can appear in a relation
  - If $A \rightarrow B$ holds, then tuples cannot have the same value for $A$ but different values for $B$
  - Also called equality-generating dependencies

- Multivalued dependencies specify what tuples must be present
  - To represent a multivalued attribute’s values, a certain set of tuples must be present
  - Also called tuple-generating dependencies
Multivalued Dependencies

• Given a relation schema \( R \)
  – Attribute-sets \( \alpha \in R, \beta \in R \)
  – \( \alpha \rightarrow\beta \) is a multivalued dependency
  – “\( \alpha \) multidetermines \( \beta \)”

• Multivalued dependency \( \alpha \rightarrow\beta \) holds on \( R \) if, in any legal relation \( r(R) \):
  – For all pairs of tuples \( t_1 \) and \( t_2 \) in \( r \) such that \( t_1[\alpha] = t_2[\alpha] \)
  – There also exists tuples \( t_3 \) and \( t_4 \) in \( r \) such that:
    • \( t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha] \)
    • \( t_1[\beta] = t_3[\beta] \) and \( t_2[\beta] = t_4[\beta] \)
    • \( t_1[R - \beta] = t_4[R - \beta] \) and \( t_2[R - \beta] = t_3[R - \beta] \)
Multivalued Dependencies (2)

- Multivalued dependency $\alpha \rightarrow \beta$ holds on $R$ if, in any legal relation $r(R)$:
  - For all pairs of tuples $t_1$ and $t_2$ in $r$ such that $t_1[\alpha] = t_2[\alpha]$
  - There also exists tuples $t_3$ and $t_4$ in $r$ such that:
    - $t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$
    - $t_1[\beta] = t_3[\beta]$ and $t_2[\beta] = t_4[\beta]$
    - $t_1[R - \beta] = t_4[R - \beta]$ and $t_2[R - \beta] = t_3[R - \beta]$

- Pictorially:

<table>
<thead>
<tr>
<th></th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$R - (\alpha \cup \beta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>$a_1...a_i$</td>
<td>$a_{i+1}...a_j$</td>
<td>$a_{j+1}...a_n$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>$a_1...a_i$</td>
<td>$b_{i+1}...b_j$</td>
<td>$b_{j+1}...b_n$</td>
</tr>
<tr>
<td>$t_3$</td>
<td>$a_1...a_i$</td>
<td>$a_{i+1}...a_j$</td>
<td>$b_{j+1}...b_n$</td>
</tr>
<tr>
<td>$t_4$</td>
<td>$a_1...a_i$</td>
<td>$b_{i+1}...b_j$</td>
<td>$a_{j+1}...a_n$</td>
</tr>
</tbody>
</table>
Multivalued Dependencies (3)

• Multivalued dependency:

<table>
<thead>
<tr>
<th></th>
<th>α</th>
<th>β</th>
<th>R – (α ∪ β)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>a₁…aᵢ</td>
<td>aᵢ₊₁…aⱼ</td>
<td>aⱼ₊₁…aₙ</td>
</tr>
<tr>
<td>t₂</td>
<td>a₁…aᵢ</td>
<td>bᵢ₊₁…bⱼ</td>
<td>bⱼ₊₁…bₙ</td>
</tr>
<tr>
<td>t₃</td>
<td>a₁…aᵢ</td>
<td>aᵢ₊₁…aⱼ</td>
<td>bⱼ₊₁…bₙ</td>
</tr>
<tr>
<td>t₄</td>
<td>a₁…aᵢ</td>
<td>bᵢ₊₁…bⱼ</td>
<td>aⱼ₊₁…aₙ</td>
</tr>
</tbody>
</table>

• If α →→ β then R – (α ∪ β) is independent of this
  – Every distinct value of β must be associated once with every distinct value of R – (α ∪ β)

• Let γ = R – (α ∪ β)
  – If α →→ β then also α →→ γ
  – α →→ β implies α →→ γ
  – Sometimes written α →→ β | γ
Trivial Multivalued Dependencies

• $\alpha \rightarrow \beta$ is a trivial multivalued dependency on $R$ if all relations $r(R)$ satisfy the dependency

• Specifically, $\alpha \rightarrow \beta$ is trivial if $\beta \subseteq \alpha$, or if $\alpha \cup \beta = R$

• Employee examples:
  – For schema $\text{emp_deps}(\text{emp_id}, \text{dependent})$, $\text{emp_id} \rightarrow \text{dependent}$ is trivial
  – For $\text{emp_info}(\text{emp_id}, \text{dependent}, \text{phone_num})$, $\text{emp_id} \rightarrow \text{dependent}$ is not trivial
Inference Rules

• Can reason about multivalued dependencies, just like functional dependencies

• Example inference rules:
  – Complementation rule:
    • If $\alpha \rightarrow \beta$ holds on $R$, then $\alpha \rightarrow R - (\alpha \cup \beta)$ holds
  – Multivalued augmentation rule:
    • If $\alpha \rightarrow \beta$ holds, and $\gamma \supseteq \delta$, then $\alpha\gamma \rightarrow \beta\delta$
  – Multivalued transitivity rule:
    • If $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$ holds, then $\alpha \rightarrow \gamma - \beta$ holds
  – Coalescence rule:
    • If $\alpha \rightarrow \beta$ and there exists $\gamma$ such that $\gamma \cap \beta = \emptyset$, and $\gamma \rightarrow \delta$, and $\beta \supseteq \delta$, then $\alpha \rightarrow \delta$
Functional Dependencies

- Functional dependencies are also multivalued dependencies
  - If $\alpha \rightarrow \beta$, then $\alpha \rightarrow\!\!\!\!\!\!\rightarrow \beta$ too
  - Caveat: each value of $\alpha$ has at most one associated value for $\beta$

- Usually, functional dependencies are not stated as multivalued dependencies
  - Because of additional caveat; not obvious in notation
  - Also because functional dependencies are just easier to reason about!
Closures and Restrictions

• For a set $D$ of functional and multivalued dependencies, can compute closure $D^+$
  – Use inference rules for both functional and multivalued dependencies to compute closure
• Sometimes need the restriction of $D^+$ to a relation schema $R$, too
• The restriction of $D$ to a schema $R$ includes:
  – All functional dependencies in $D^+$ that include only attributes in $R$
  – All multivalued dependencies of the form $\alpha \rightarrow \beta \cap R$, where $\alpha \subseteq R$, and $\alpha \rightarrow \beta$ is in $D^+$
Fourth Normal Form

• Given:
  – Relation schema \( R \)
  – Set of functional and multivalued dependencies \( D \)

• \( R \) is in 4NF with respect to \( D \) if:
  – For all multivalued dependencies \( \alpha \rightarrow \beta \) in \( D^+ \), where \( \alpha \in R \) and \( \beta \in R \), at least one of the following holds:
    • \( \alpha \rightarrow \beta \) is a trivial multivalued dependency
    • \( \alpha \) is a superkey for \( R \)
  – Note: If \( \alpha \rightarrow \beta \) then \( \alpha \rightarrow \beta \)

• A database design is in 4NF if all schemas in the design are in 4NF
4NF and BCNF

- Main difference between 4NF and BCNF is use of multivalued dependencies instead of functional dependencies
- Every schema in 4NF is also in BCNF
  - If a schema is not in BCNF then there is a nontrivial functional dependency \( \alpha \rightarrow \beta \) such that \( \alpha \) is not a superkey for \( R \)
  - If \( \alpha \rightarrow \beta \) then \( \alpha \rightarrow \beta \)
4NF Decompositions

• Decomposition rule very similar to BCNF
• If schema $R$ is not in 4NF with respect to a set of multivalued dependencies $D$:
  – There is some nontrivial dependency $\alpha \rightarrow \beta$ in $D^+$ where $\alpha \subseteq R$ and $\beta \subseteq R$, and $\alpha$ is not a superkey of $R$
    • Also constrain that $\alpha \cap \beta = \emptyset$
  – Replace $R$ with two new schemas:
    • $R_1 = (\alpha \cup \beta)$
    • $R_2 = (R - \beta)$
Employee Information Example

• Combined schema:
  \[\text{employee}(\text{emp}_\text{id}, \text{emp\_name})\]
  \[\text{emp\_info}(\text{emp}_\text{id}, \text{dependent}, \text{phone\_num})\]
  – Also have these dependencies:
    • \(\text{emp}_\text{id} \rightarrow \text{emp\_name}\)
    • \(\text{emp}_\text{id} \rightarrow \text{dependent}\)
    • \(\text{emp}_\text{id} \rightarrow \text{phone\_num}\)

• \text{emp\_info} is not in 4NF
  – Following rules for 4NF decomposition produces:
    \((\text{emp}_\text{id}, \text{dependent})\)
    \((\text{emp}_\text{id}, \text{phone\_num})\)
Lossless Decompositions

• Can also define lossless decomposition with multivalued dependencies
  – $R_1$ and $R_2$ form a lossless decomposition of $R$ if at least one of these dependencies is in $D^+$:
    \[
    R_1 \cap R_2 \rightarrow R_1 \\
    R_1 \cap R_2 \rightarrow R_2
    \]
Beyond Fourth Normal Form?

• Several other normal forms with various constraints
• Some define new dependencies, such as:
  – Join dependencies: a more general form of multivalued dependencies
  – Inclusion dependencies: for representing foreign key constraints
• Fifth normal form (5NF) includes join dependencies
  – Also known as project-join normal form (PJNF)
• Domain-key normal form (DKNF) is an even more general normal form
  – Takes domain constraints into account
  – Can state other normal forms as special cases of DKNF
• Higher normal forms are much harder to reason about
  – Not widely used, although often good goals to aim for!
Normalized Schemas

• Normalized schemas have certain features
  – Usually more relations, to eliminate redundancy
  – Usually need to join several relations together to retrieve a particular result
  – Manipulating the data is usually easier, because there is very little redundancy

• Sometimes a database application needs high performance query support
  – Normalized database’s layout doesn’t facilitate fast query operations
Faster Query Evaluation

- Sometimes database designers will **denormalize** a database schema
  - Intentionally design a schema to violate a normal form, in order to be faster
  - Usually requires more development effort, to keep redundant data synchronized
  - For systems requiring high transaction throughput, may be the only option!
  - Should usually consider this as a last resort

  - Try other options first!
Materialized Views

• Materialized views can provide a denormalized view of a normalized schema
  – Create a normalized schema
  – Define widely used views against schema
  – The database stores the view’s results on disk
• Database itself keeps view in sync with underlying tables
  – Unlike a denormalized schema, developers don’t have to worry about managing things
• Not all databases provide materialized views
Materialized Views (2)

• Materialized views can become expensive if data changes frequently
  – Especially when a value in the underlying relation is duplicated multiple times in the view
• One solution: update view periodically
  – e.g. view is recomputed and stored hourly or daily
• For applications where underlying data changes very frequently, a denormalized schema may be faster
Unique-Role Assumption

• SQL requires every table has unique column names
• Column names indicate what values represent
  – Different tables can have columns with the same name, but different meanings
• The unique-role assumption:
  – Every attribute name in a schema has a unique meaning in the database
  – Some obvious benefits:
    • Clearly indicates foreign key relationships between tables
    • No ambiguity in what an attribute name means in a query, or in its result
    • Can use \texttt{NATURAL JOIN} syntax for most queries
Unique-Role Assumption (2)

• A common example:
  
  assignment(id, shortname, url, perfectscore, due)
  submission(id, assignment, graded, score)
  
  – For each table, id is the unique identifier for that table’s rows
  – Foreign key relationships become awkward
    • submission.assignment is foreign key to assignment.id

• A better design:
  
  assignment(asn_id, shortname, url, perfectscore, due)
  submission(sub_id, asn_id, graded, score)
  
  – Follows unique-role assumption guideline
Unique-Role Assumption (3)

• One situation where unique-role assumption is harder to follow:
  – A relation with a foreign key reference to itself
• Example:
  \[ \text{employee}(\text{emp_id}, \text{emp_name}, \text{salary}) \]
  \[ \text{manages}(\text{emp_id}, \text{manager_id}) \]
  – \text{manager_id} has same domain as \text{emp_id}
  – Simply can’t give both attributes the same name!
    (Nor would you necessarily want to…)
• Usually occurs infrequently in schema designs
  – Only certain circumstances where relations have foreign key references to themselves
General Naming Concerns

• Unique-role assumption is one specific aspect of a more general concern
• Database schemas contain lots of names!
  – Table names, column names
  – Possibly constraint names and stored procedure names, as well
  – Names become hard to change after a while
• Database schemas should follow a unified naming convention
  – Singular vs. plural table names
    • e.g. employee vs. employees
  – Unique-role assumption for column names, etc.
Review

- Covered normal forms for database schemas
  - Patterns to follow that make for “good” schemas
- Goals:
  - Representation must be complete
  - Eliminate unnecessary redundancy
- Formalized the concept using dependencies
  - Functional dependencies, multivalued dependencies
  - Rules of inference for dependencies
  - In some cases, algorithms for generating normalized schemas
Review, and Preview!

• Designing normalized schemas can involve trade-offs
  – Can mean slower performance, but easier development and maintenance
  – Some ways to work around slow performance
  – Sometimes, only choice is a denormalized schema

• Next time:
  – How to maximize performance of query evaluation?
  – How do databases actually evaluate queries?
  – What tools do databases provide to improve performance?