Multivalued Dependencies

Winter 2006-2007

Lecture 22

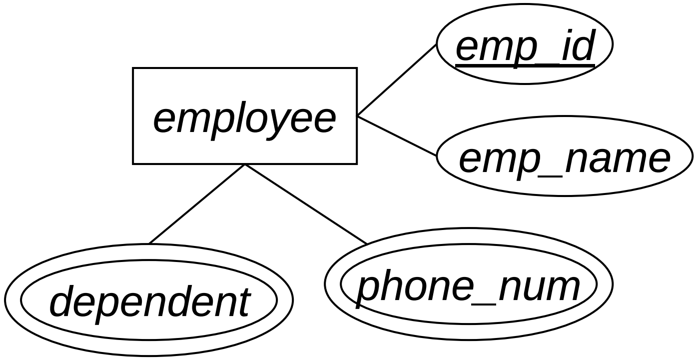
Multivalued Attributes

- 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(*<u>emp_id</u>*, *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:

emp_id	*emp_name*
125623	Rick

employee

emp_id	*dependent*
125623	Jeff
125623	Alice

emp_deps

emp_id	*phone_num*
125623	555-8888
125623	555-2222

emp_nums

  – 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(*<u>emp_id</u>*, *emp_name*)

  emp_info(*emp_id*, *dependent*, *phone_num*)
- What tuples must appear in *emp_info* ?
  - *emp_info* is a <u>relation</u>
  - 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:

emp_id	*emp_name*
125623	Rick

employee

emp_id	*dependent*	*phone_num*
125623	Jeff	555-8888
125623	Jeff	555-2222
125623	Alice	555-8888
125623	Alice	555-2222

emp_info

- 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 same value for A but different values for B
  - Also called <u>equality-generating dependencies</u>
- <u>Multivalued dependencies</u> specify what tuples must be present
  - To represent a multivalued attribute's values, a certain set of tuples *must* be present
  - Also called <u>tuple-generating dependencies</u>

Multivalued Dependencies

- Given a relation schema R
  - Attribute-sets $\alpha \in R$, $\beta \in R$
  - $\alpha \twoheadrightarrow \beta$ is a multivalued dependency
  - "α multidetermines β"
- Multivalued dependency $\alpha \twoheadrightarrow \beta$ 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 $\mathrm{t}_2[R - \beta] = t_3[R - \beta]$

Multivalued Dependencies (2)

- Multivalued dependency $\alpha \twoheadrightarrow \beta$ 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:

	α	β	$R - (\alpha \cup \beta)$
t_1	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
t_2	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$b_{j+1} \ldots b_n$
t_3	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_4	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$a_{j+1} \ldots a_n$

Multivalued Dependencies (3)

- Multivalued dependency:

	α	β	$R - (\alpha \cup \beta)$
t_1	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
t_2	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$b_{j+1} \ldots b_n$
t_3	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_4	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$a_{j+1} \ldots a_n$

- If $\alpha \twoheadrightarrow \beta$ then $R - (\alpha \cup \beta)$ is independent of this
  - Every distinct value of β must be associated once with every distinct value of $R - (\alpha \cup \beta)$
- Let $\gamma = R - (\alpha \cup \beta)$
  - If $\alpha \twoheadrightarrow \beta$ then also $\alpha \twoheadrightarrow \gamma$
  - $\alpha \twoheadrightarrow \beta$ implies $\alpha \twoheadrightarrow \gamma$
  - Sometimes written $\alpha \twoheadrightarrow \beta \mid \gamma$

Trivial Multivalued Dependencies

- $\alpha \twoheadrightarrow \beta$ is a trivial multivalued dependency on *R* if all relations *r*(*R*) satisfy the dependency
- Specifically, $\alpha \twoheadrightarrow \beta$ is trivial if $\beta \subseteq \alpha$, or if $\alpha \cup \beta = R$
- Employee examples:
  - For schema *emp_deps*(*emp_id*, *dependent*), *emp_id* $\twoheadrightarrow$ *dependent* is trivial
  - For *emp_info*(*emp_id*, *dependent*, *phone_num*), *emp_id* $\twoheadrightarrow$ *dependent* is not trivial

Inference Rules

- Can reason about multivalued dependencies, just like functional dependencies
- Example inference rules:
  - Complementation rule:
    - If $\alpha \twoheadrightarrow \beta$ holds on *R*, then $\alpha \twoheadrightarrow R - (\alpha \cup \beta)$ holds
  - Multivalued augmentation rule:
    - If $\alpha \twoheadrightarrow \beta$ holds, and $\gamma \supseteq \delta$, then $\alpha\gamma \twoheadrightarrow \beta\delta$
  - Multivalued transitivity rule:
    - If $\alpha \twoheadrightarrow \beta$ and $\beta \twoheadrightarrow \gamma$ holds, then $\alpha \twoheadrightarrow \gamma - \beta$ holds
  - Coalescence rule:
    - If $\alpha \twoheadrightarrow \beta$ and there exists γ 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 $\alpha \to \beta$, then $\alpha \twoheadrightarrow \beta$ too
  - Caveat: each value of α has at most one associated value for β
- 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 $\alpha \twoheadrightarrow \beta \cap R$, where $\alpha \subseteq R$, and $\alpha \twoheadrightarrow \beta$ 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 $\alpha \twoheadrightarrow \beta$ in D^+, where $\alpha \in R$ and $\beta \in R$, at least one of the following holds:
    - $\alpha \twoheadrightarrow \beta$ is a trivial multivalued dependency
    - α is a superkey for *R*
  - Note: If $\alpha \rightarrow \beta$ then $\alpha \twoheadrightarrow \beta$
- 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 α is not a superkey for *R*
  - If $\alpha \rightarrow \beta$ then $\alpha \twoheadrightarrow \beta$

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 $\alpha \twoheadrightarrow \beta$ in D^+ where $\alpha \subseteq R$ and $\beta \subseteq R$, and α 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:

  employee(<u>*emp_id*</u>, *emp_name*)

  emp_info(*emp_id*, *dependent*, *phone_num*)

  – Also have these dependencies:

    - $emp\_id \rightarrow emp\_name$
    - $emp\_id \twoheadrightarrow dependent$
    - $emp\_id \twoheadrightarrow phone\_num$

- *emp_info* is not in 4NF

  – Following rules for 4NF decomposition produces:

  (*emp_id*, *dependent*)

  (*emp_id*, *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^+ :

$$R_1 \cap R_2 \twoheadrightarrow R_1$$

$$R_1 \cap R_2 \twoheadrightarrow R_2$$

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 <u>denormalize</u> 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 <u>unique-role assumption</u>:
  - 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 **`NATURAL JOIN`** syntax for most queries

Unique-Role Assumption (2)

- A common example:

  assignment(<u>*id*</u>, *shortname*, *url*, *perfectscore*, *due*)

  submission(<u>*id*</u>, *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(<u>*asn_id*</u>, *shortname*, *url*, *perfectscore*, *due*)

  submission(<u>*sub_id*</u>, *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:

  employee(<u>*emp_id*</u>, *emp_name*, *salary*)

  manages(<u>*emp_id*</u>, *manager_id*)

  - *manager_id* has same domain as *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?