The Entity Relationship Model and Database Design

CPS352: Database Systems

Simon Miner
Gordon College
Last Revised: 9/20/12
Agenda

• Check-in
• The Entity Relationship Model
• Group Exercise
• Database Design Principles
  • Functional Dependencies
Check-in
The Entity Relationship Model
Background

- Entity relationship model as a conceptual database design tool
  - Not a DBMS implementation
    - No “entity relationship databases” available

- Entity relationship (E-R) diagrams help us think about the structure of a data model
  - Can be translated into relational schemas
  - Which then can be implemented in a DBMS
  - Analogous to use case or class diagrams in OO design
Entity Concepts

- **Entity** – an object being represented (along with its details)
- **Entity set** – the set of all objects of a given kind
- **Attribute** – individual fact about an entity
  - Often simple (atomic) and single-valued
  - Can be composite
  - Sometimes multi-valued
  - Can be derived from other attributes
    - Not necessarily stored with the entity, but calculated when needed
- **Domain** – set of possible values for an attribute
- **Keys** – set of attributes that uniquely identifies an entity
  - Superkeys, candidate keys, and primary key
Entities in E-R Diagrams

- Entity set represented by rectangular box containing name of entity
- Attributes represented by ellipses containing attribute names
  - Primary key attribute(s) underlined
  - Composite attributes displayed with a hierarchical structure
  - Multivalued attributes enclosed in double ellipses
  - Derived attributes enclosed in a dashed ellipse
- Attributes connected by lines to entity set
Relationship Concepts

- Relationship – the connection between two or more entities
  - A relationship with more than two entities can always be converted to a new entity plus relationships between the new and original entities
  - A relationship can be between an entity and itself

- Relationship set – set of all relationships of a given type
  - A subset of the Cartesian product of the entity sets
  - Degree of a relationship set is how many entities are involved in it (i.e. binary, ternary, quadranary, etc.)

- Descriptive attribute – a property of a relationship that does not apply to its associated entities
  - When a relationship of more than two entities is converted into a new entity, the original relationship’s descriptive attributes become the new entity’s attributes
Relationships in E-R Diagrams

- Relationship sets represented by diamonds
  - Connected with associated entities by solid lines (potentially doubled or decorated with arrows)
  - Descriptive attributes depicted the same as entity attributes

- Converting a ternary+ relationship to a new entity
Mapping Constraints

- Restrictions as to what kind of subsets are possible in a relationship set

- Mapping cardinalities – how many entities in each entity set can participate in the relationship

- Participation constraints – when an entity in one entity set must participate in a relationship

- Existence dependencies – when an entity in one entity set of a relationship is dependent on the existence of an entity in the other entity set

- Primary keys for relationship sets
Mapping Cardinalities

- **One to one**
  - Any member of either entity set involved can participate in at most one instance of the relationship set
  - Often represented by arrow heads pointing to both entities arrow in E-R diagrams

- **One to many / Many to one**
  - Basically the same concept (just in opposite directions)
  - Entities in the “one” entity set can participate in multiple relationships
  - Entities in the “many” entity set can participate in at most one
  - Often represented by an arrow head pointing to the “one” in E-R diagrams

- **Many to many**
  - Entities in either entity set can participate in multiple relationships
  - Often represented by a solid line to all entities in the relationship (no arrow heads)
Participation Constraints

- Total participation constraint
  - When the underlying of a relationship dictates that every entity in an entity set *must* participate in an instance of the relationship

- Represented by a double line between the relationship and the entity that must participate

- Example: every borrower must have a category
Existence Dependencies

- Weak entity set – an entity set in which each entity is dependent on the existence of an entity from another entity set
  - Has a partial key or discriminator which must be combined with attributes from the strong entity to uniquely identify it (no superkey)
  - If the dominant strong entity is deleted, the subordinate weak entity ceases to exist
- Example: Fines owed by borrowers
Weak Entities in E-R Diagrams

- Weak entity set represented by a double box
- Existence dependency relationship represented by a double diamond
- Partial key attributes underlined using a dashed line
# Primary Keys for Relationship Sets

<table>
<thead>
<tr>
<th>Mapping Cardinality</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Many to many</td>
<td>Union of key attributes in all involved entities</td>
</tr>
<tr>
<td>One to many</td>
<td>Primary key of the “many” entity</td>
</tr>
<tr>
<td>Many to one</td>
<td>Primary key of the “many” entity</td>
</tr>
<tr>
<td>One to one</td>
<td>Primary key of either of the entities</td>
</tr>
</tbody>
</table>
Converting to the Relational Model

• Any database scheme consisting of entities and relationships can be represented by a series of tables
  • One for each entity set
  • One for each relationship set
    • Except when the relationship can be “folded” into an entity
Converting Entities to Tables

- **Strong entity set**
  - One row for each entity
  - One column for each attribute

- **Weak entity set**
  - One row for each entity
  - One column for each attribute
  - Add column(s) for the primary key of the strong entity on which the weak entity depends
Converting Relationships to Tables

• Relationship set
  • One row for each relationship
  • One column for each descriptive attribute
  • Column(s) for primary key attributes of each participating entity set

• “Folding” in one to one and one to many relationships
  • Into the many entity by including the foreign key of the “one” entity and any attributes
  • These will be null for an entity that is not in any relationship
Generalized and Specialized Entities

• An entity set may contain multiple groups of similar entities with common and distinct attributes
  • Example: different kinds of borrowers for students, faculty/staff, and community members

• Converting generalized/specialized entities to tables
  • One big table
  • One table per group
  • One generalized table with common attributes and one specialized table per group
Group Exercise

Complete Practice Exercise 7.1
On page 315 of *Database System Concepts*
Introduction

• Terminology review
  • Relation scheme – set of attributes for some relation (R, R₁, R₂)
  • Relation – the actual data stored in some relation scheme (r, r₁, r₂)
  • Tuple – a single actual row in the relation (t, t₁, t₂)

• Changes to the library database schema
  • We make the following updates for this discussion
  • Add the following attributes to the book relation
    • copy_number – a library can have multiple copies of a book
    • accession_number – unique number (ID) assigned to a copy of a book when the library acquires it
  • New book and checked_out relation scheme
    • Book( catalog_number, copy_number, accession_number, title, author )
    • Checked_out( borrower_id, catalog_number, copy_number, date_due )
Database Design Issues

• Designing a database is a balancing act

• One the one extreme, you can have a *universal relation* (in which all attributes reside within a single relation scheme)
  • Everything
    borrower_id, last_name, first_name, // from borrower
    call_number, copy_number,
    accession_number, title, author    // from book
    date_due                            // from checked_out

• Leads to numerous anomalies with changing data in the database
Decomposition

- *Decomposition* is the process of breaking up an original scheme into two or more schemes
  - Each attribute of the original scheme appears in at least one of the new schemes
- But this can be taken too far
  - Borrower( borrower_id, last_name, first_name )
  - Book( call_number, copy_number, accession_number, title, author )
  - Checked_out( date_due )
- Leads to *lossy-join* problems
Lossless-Join

- Part of the middle ground in the balancing act
  - Allows decomposition of the Everything relation
  - Preserves connections between the tuples of the participating relations
  - So that the natural join of the new relations = the original relation

- Formal definition
  - For some relation scheme $R$ decomposed into two or more schemes ($R_1$, $R_2$, … $R_n$)
  - Where $R = R_1 \cup R_2 \cup \ldots \cup R_n$
  - A lossless-join decomposition means that for every legal instance $r$ of $R$ decomposed into $r_1$, $r_2$, … $r_n$ of $R_1$, $R_2$, and $R_n$
  - $r = r_1 \mid X \mid r_2 \mid X \mid \ldots \mid X \mid r_n$
Database Design Goal

• Decide whether a particular relation \( R \) is in “good” form.

• In the case that a relation \( R \) is not in “good” form, decompose it into a set of relations \( \{R_1, R_2, ..., R_n\} \) such that
  • each relation is in good form
  • the decomposition is a lossless-join decomposition

• Our theory is based on:
  • functional dependencies
  • multivalued dependencies
Functional Dependency (FD)

- When the value of a certain set of attributes uniquely determines the value for another set of attributes
  - Generalization of the notion of a key
  - A way to find “good” relations
  - A → B (read: A determines B)

- Formal definition
  - For some relation scheme R and attribute sets A (A ⊆ R) and B (B ⊆ R)
  - A → B if for any legal relation on R
    - If there are two tuples t₁ and t₂ such that t₁(A) = t₂(A)
    - It must be the case that t₂(A) = t₂(B)
Finding Functional Dependencies

- From keys of an entity
- From relationships between entities
- Implied functional dependencies
FDs from Entity Keys

A → BC
FDs from One to Many / Many to One Relationships

A → BC
W → XY
A → BCMWXY
FDs from One to One Relationships

A → BC
W → XY
A → BCMWXYZ
W → XYMABC
FDs from Many to Many Relationships

A → BC
W → XY
AW → M
# Implied Functional Dependencies

- **Initial set of FDs logically implies other FDs**
  - If $A \rightarrow B$ and $B \rightarrow C$, then $B \rightarrow C$

- **Closure**
  - If $F$ is the set of functional dependencies we develop from the logic of the underlying reality
  - Then $F^+$ (the *transitive closure* of $F$) is the set consisting of all the dependencies of $F$, plus all the dependencies they imply
Rules for Computing $F^+$

- We can find $F^+$, the closure of $F$, by repeatedly applying Armstrong’s Axioms:
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ (reflexivity)
  - Trivial dependency
  - if $\alpha \rightarrow \beta$, then $\gamma \alpha \rightarrow \gamma \beta$ (augmentation)
  - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ (transitivity)

- Additional rules (inferred from Armstrong’s Axioms)
  - If $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$, then $\alpha \rightarrow \beta \gamma$ (union)
  - If $\alpha \rightarrow \beta \gamma$, then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ (decomposition)
  - If $\alpha \rightarrow \beta$ and $\gamma \beta \rightarrow \delta$, then $\alpha \gamma \rightarrow \delta$ (pseudotransitivity)