IT420: Database Management and Organization

Entity-Relationship Model to Relational Model
20 January 2006
Adina Crăiniceanu
www.ca.usna.edu/~adina

Database Design Process

- Requirements analysis
- Conceptual design: Entity-Relationship Model
- Logical design: transform ER model into relational schema
- Schema refinement: Normalization
- Physical tuning

Goals

- Transform ER model to relational model
ER Model - Entities

- Attributes
- Identifiers: no two instances with the same value for identifier
- Weak entity: its existence depends on another entity
- ID-Dependent entity: weak + identifier contains another identifier

ER Model - Relationships

- HAS-A
  - Maximum cardinality
  - Minimum cardinality
- IS-A (super-type/sub-type)
  - Exclusive/inclusive

Relational Model

- Tables
- Integrity constraints
  - Primary key
  - Candidate key
  - Foreign key
ER to Relational

- Transform entities in tables
- Transform relationships using foreign keys
- Specify logic for enforcing minimum cardinalities

Create a Table for Each Entity

EMPLOYEE (EmployeeNumber, EmployeeName, Phone, Email, HireDate, ReviewDate, EmpCode)

- Primary key is designated by key symbol
- Note shadowless table

Select the Primary Key

- The ideal primary key is short, numeric and fixed
- Surrogate keys meet the ideal, but have no meaning to users
Specify Candidate (Alternate) Keys

- candidate key = alternate key
- Candidate keys: alternate identifiers of unique rows in a table
- ERwin uses \textit{AKn.m} notation, where \( n \) is the number of the alternate key, and \( m \) is the column number in that alternate key

Specify Candidate (Alternate) Keys

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>CUSTOMER</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeNumber</td>
<td>CustomerNumber</td>
</tr>
<tr>
<td>EmployeeName</td>
<td>Name (AK1,1)</td>
</tr>
<tr>
<td>Phone</td>
<td>City (AK1,2)</td>
</tr>
<tr>
<td>Email (AK1,1)</td>
<td>Phone</td>
</tr>
<tr>
<td>HireDate</td>
<td>Email (AK2,1)</td>
</tr>
<tr>
<td>ReviewDate</td>
<td></td>
</tr>
<tr>
<td>EmpCode</td>
<td></td>
</tr>
</tbody>
</table>

Specify Column Properties: Data Type

- Generic Data Types:
  - CHAR(n)
  - VARCHAR(n)
  - DATE
  - TIME
  - MONEY
  - INTEGER
  - DECIMAL
Specify Column Properties: SQL Server Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>Binary, length 0 to 8,096 bytes.</td>
</tr>
<tr>
<td>Date</td>
<td>Date, year length 14,000 years.</td>
</tr>
<tr>
<td>Datetime</td>
<td>8-byte datetime. Range from 1700 through December 31, 9999, with an accuracy of thousands of a second.</td>
</tr>
<tr>
<td>Image</td>
<td>Variable-length binary data. Maximum length 2,047,483,647 bytes.</td>
</tr>
<tr>
<td>Integer</td>
<td>4-byte integer. Value range from -2,147,483,648 through +2,147,483,647.</td>
</tr>
<tr>
<td>Money</td>
<td>8-byte money. Range from +8,999,999.99 through +9,000,000.00.</td>
</tr>
<tr>
<td>Smalldate</td>
<td>1-byte date. Range from January 1, 1900, through June 6, 2030, with an accuracy of one day.</td>
</tr>
<tr>
<td>Smallint</td>
<td>2-byte integer. Range from -32,768 through 32,767.</td>
</tr>
<tr>
<td>Smoney</td>
<td>8-byte money. Range from 0.0 through 9,999,999.99.</td>
</tr>
<tr>
<td>Varbinary</td>
<td>Variable-length character, length 0 to 8,096 bytes.</td>
</tr>
</tbody>
</table>

Specify Column Properties: Oracle Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>Binary large object. Up to 4 gigabytes in length.</td>
</tr>
<tr>
<td>CHAR(max)</td>
<td>Fixed-length character field of length 1. Maximum 2,000 characters.</td>
</tr>
<tr>
<td>DATE</td>
<td>7-byte field containing both date and time.</td>
</tr>
<tr>
<td>NUMBER(n)</td>
<td>Whole number of length n.</td>
</tr>
<tr>
<td>NUMBER(*r)</td>
<td>Fixed-length character field up to n characters long. Minimum value of -1,000.</td>
</tr>
<tr>
<td>NUMBER(<em>r,</em>)</td>
<td>Variable-length character field up to n characters long. Minimum value of -1,000.</td>
</tr>
</tbody>
</table>

Specify Column Properties: Null Status

- **Null status** indicates whether or not the value of the column can be NULL.
Specify Column Properties:
Default Value

- A **default value** is the value supplied by the DBMS when a new row is created

<table>
<thead>
<tr>
<th>Table</th>
<th>Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITEM</td>
<td>ItemNumber</td>
<td>Surrogate key</td>
</tr>
<tr>
<td>ITEM</td>
<td>Category</td>
<td>None</td>
</tr>
<tr>
<td>ITEM</td>
<td>ItemPrefix</td>
<td>If Category = &quot;Portable&quot; then &quot;P&quot;</td>
</tr>
<tr>
<td>ITEM</td>
<td>ItemPrefix</td>
<td>If Category = &quot;Imported&quot; then &quot;Y&quot;</td>
</tr>
<tr>
<td>ITEM</td>
<td>ItemPrefix</td>
<td>If Category = &quot;One-off&quot; then &quot;O&quot;</td>
</tr>
<tr>
<td>ITEM</td>
<td></td>
<td>Otherwise = &quot;N&quot;</td>
</tr>
</tbody>
</table>
| ITEM  | ApprovingAgent | If ItemPrefix = "Y" then "Purchasing"
| ITEM  |             | Otherwise = "Purchasing/MD"          |
| ITEM  | ShippingMethod | If ItemPrefix = "P" then "Next Day"
|       |             | Otherwise = "Ground"                 |

Specify Column Properties:
Data Constraints

- **Data constraints** are limitations on data values:
  - **Domain constraint** - Column values must be in a given set of specific values
  - **Range constraint** - Column values must be within a given range of values
  - **Intra-relation constraint** – Column values are limited by comparison to values in other columns in the same table
  - **Inter-relation constraint** - Column values are limited by comparison to values in other columns in other tables

Create Relationships:
1:1 Strong Entity Relationships

- Place the key of one entity in the other entity as a foreign key:
  - Either design will work – no parent, no child
  - Minimum cardinality considerations may be important:
    - O-M will require a different design that M-O
Create Relationships:
1:1 Strong Entity Relationships

(a) With Foreign Key in LOCKER

(b) With Foreign Key in CLUB_MEMBER

Create Relationships:
1:N Strong Entity Relationships

- “Place the key of the parent in the child”

Create Relationships:
1:N Strong Entity Relationships

(a) 1:N Strong Entity Relationship

(b) Placing the Key of the Parent in the Child
Create Relationships:
N:M Strong Entity Relationships

- In an N:M strong entity relationship there is no place for the foreign key in either table:
  - A COMPANY may supply many PARTs
  - A PART may be supplied by many COMPANYs

Create Relationships:
N:M Strong Entity Relationships

- Create an intersection table:
  - The primary keys of each table → composite primary key for intersection table
  - Each table’s primary key becomes a foreign key linking back to that table

Create Relationships:
N:M Strong Entity Relationships

COMPANY_PART_INT (CompanyName, PartNumber)
Relationships Using ID-Dependent Entities:
Four Uses for ID-Dependent Entities

- Representing N:M Relationships
  - We just discussed this
- Association Relationships
- Multivalued Attributes
- Archtype/Instance Relationships

An intersection table:
- Holds the relationships between two strong entities in an N:M relationship
- Contains only the primary keys of the two entities:
  - As a composite primary key
  - As foreign keys

An association table:
- Has all the characteristics of an intersection table
- PLUS it has one or more columns of attributes specific to the associations of the other two entities
Relationships Using ID-Dependent Entities: Multivalued Attributes

ER model

- COMPANY:
  - CompanyID
  - CompanyName
  - CompanyValue

- PHONE:
  - Contact
  - CompanyName
  - PhoneNumber

As a set of tables

- COMPANY:
  - CompanyID
  - CompanyName
  - CompanyValue

- PHONE:
  - Contact
  - CompanyName
  - PhoneNumber

Relationships Using ID-Dependent Entities: Archetype/Instance Pattern

As a data model

- CLASS:
  - StudentNumber
  - StudentName
  - StudentCourse
  - Taks

- SECTION:
  - StudentNumber
  - Professor
  - Course
  - ClassDays
  - ClassTimes
Relationships Using ID-Dependent Entities: Archetype/Instance Pattern

Relationships Using Weak Entities: Archetype/Instance Pattern

Relationships Using Weak Entities: Archetype/Instance Pattern
Mixed Entity Relationships: The Line-Item Pattern

Subtype Relationships
Subtype Relationships

As a data model

As a set of tables

Recursive Relationships: 1:1 Recursive Relationships

As a data model

As a table

Recursive Relationships: 1:1 Recursive Relationships

As a data model
Recursive Relationships:
1:N Recursive Relationships

As a data model

As a table

Recursive Relationships:
N:M Recursive Relationships

As a data model

As a set of tables

Recursive Relationships:
N:M Recursive Relationships

As a data model

As a set of tables
Minimum cardinality:
**Parent Required**

DEPARTMENT (DepartmentName, BudgetCode, ManagerName)
EMPLOYEE (EmployeeNumber, EmployeeName, DepartmentName)

- On Insert: child OK
- On Delete:
  - delete child (cascade) or prohibit
- On Update:
  - update child (cascade) or prohibit

Minimum cardinality:
**Child Required**

- More difficult to enforce (write code)
- Tricky:
  - A department must have some employee
  - EMPLOYEE has DepartmentName as FK, NOT NULL