Databases and Database Design

CE 199/299 Database Systems for Engineering and Management

Arpad Horvath
Department of Civil and Environmental Engineering
University of California, Berkeley
October 12, 2000
Sources for this Presentation

- Professor Daniel Rehak and Rebecca Buchheit, Department of Civil and Environmental Engineering, Carnegie Mellon University (http://fen.ce.cmu.edu/cae/), Copyright, All rights reserved

Objectives of Presentation

- Basic terminology
- Utility of databases
- Relational data model
- Database design using dependency diagrams
Terminology

- **Data sources**
- **Database**: persistent collection of data
- **Database Management System (DBMS)**: software that controls access to the database
- **Database Administrator (DBA)**: person who controls database
- **Data Model**: general structure of the data in the database
- **Data Language**: commands used to define the data model and give users access to the database
Interaction and Feedback

- **Transaction**: non-decomposable unit of data manipulation
  - example: purchasing an airline ticket on-line
  - typically small and fast for commercial applications
  - may be long and involved in engineering applications

- **Rollback**: if any part of a transaction fails, all completed parts are “rolled back” or undone
  - example: if you haven’t provided your credit card number, airline ticket purchase on-line transaction fails
  - rollback ensures integrity of database
  - automatically done by DBMS
Utility of Databases

- Data have value independent of use
- Organized approach to data management (e.g., data mining)
- Eliminate redundancy in data
- Share data
- Archive data
- Security of data
- Integrity of data
Relational Database Model

- Informal description of how database is organized
- Database
  - Database is a collection of tables (relations)
  - Data are stored in tables
- Tables
  - Each table has a name
  - Each table has a set of columns (fields) and rows of data (records)
  - All operations process a table to produce a new table
  - Each table has a fixed number of columns
  - Each table has an arbitrary number of rows
- Based on set theory
- SQL (Structured Query Language)
  - DBMS independent language
Database Columns (Fields)

- Columns
  - Each column has a name
  - Columns are accessed by name
  - No standard column ordering
  - Does not make sense to say “the third column” like it does in a “paper” table or spreadsheet
  - Data in a column belongs to a particular domain
    - Columns are the “attributes” of the dataset
    - Each value in a column is from the same domain
    - Each value in a column is of the same data type
Database Rows (Records)

- **Rows**
  - Each row entry is either a simple value or empty ("null")
  - Rows are sets of values for the columns (attribute values)
  - **Primary key**: a set of columns that uniquely identifies each row
  - Each row must be unique given the primary key (no duplicates)
  - Rows are referenced by the primary key
  - Row order cannot be determined by the user
  - Does not make sense to say “the fourth row” like it does in a “paper” table or spreadsheet
Data Types

- Each row value is an instance of a primitive data type
  - Integer
  - Real (e.g., number, currency)
  - Character (e.g., text, hyperlink, yes/no)
  - Date/Time

- No complex types in standard DBMS (matrix, drawing)
  - MS Access will allow drawings and some objects
  - Object oriented databases may allow objects and structures

- Non existent value is “null”
Database Design Goals

- Create a balanced design which is good for all users
- Based on a set of assumptions about the world being modeled
  - e.g., might assume bridge has one designer, or multiple designers
- Determine the data to be stored
- Determine the relations among the data
- Determine the operations to be performed
- Specify the structure of the tables
Database Design Process

1. Identify all the data items
2. Identify all the dependencies
3. Design tables to represent the stated items and dependencies
4. Verify the design
5. Implement the database
6. Design the queries
7. Test and revise
1a. Identify All Data Items

- Describe each item to be stored
  - items should be primitives
  - example: better to store first name and last name separately

- Determine the data type for each item
  - text, currency, date, etc.

- Determine the range of allowable values for each item
  - non-negative?
  - greater than zero?
  - decimal points?
  - any of the 50 state abbreviations
  - zip code between 00000 and 99999
  - phone number
1b. Turn Data Items into Attributes

- Each attribute should have:
  - a meaningful name
  - a description of what the attribute means or what kind of data make up the attribute
  - a domain
    - the data type of the attribute
    - the range or a list of allowable values of the attribute
2. Identify All the Dependencies

- Assume a set of relationships between data items
  - a model of the world
  - may have to make assumptions
  - these assumptions should be listed clearly

- Turn these relationships into dependencies
  - single-valued: there is one and only one value of ‘x’ for every value of ‘y’
    - a person Y receives a grade X for a course in a semester
    - a person Y has a birth date X
  - multi-valued: there are zero (or one) or more values of ‘x’ for every value of ‘y’
    - a student Y enrolls in one or more classes (X) each semester
    - a person Y has zero or more sisters
Single-Valued (One-to-One) Dependencies

- Draw a single-headed arrow for single-valued dependencies
  - A person has one and only one birth date
  - A student has one and only one final grade for a course
Multi-Valued (One-to-Many) Dependencies

- Draw a double-headed arrow between multi-valued dependencies

  - A student can enroll in one or more classes
  - A person has zero or more sisters
Independent vs. Dependent Attributes

- Some attributes are independent
  - your client’s phone number does not dependent on when you are scheduled to meet him
  - your client still exists whether or not you have an appointment with him

- Some attributes are dependent
  - the length of a superstructure span on a bridge is dependent on the structure of the bridge
  - the superstructure of a bridge would not exist if the bridge itself was not there
Dependent vs. Independent Attribute Representation

- Start a new bubble around an independent attribute
  - properties of that attribute are attached to the new bubble
  - properties that are dependent on other attributes are attached to the old bubble
  - Each appointment is with one or more clients. Each appointment with one or more clients has a time. Each client has a single phone number.
Each bridge has a unique bridge-id.
Each bridge may have zero or more bridge-names.
Each bridge has one or more designers and one or more owners.
Each bridge consists of one or more spans.
Each bridge was erected at some location in some year (i.e., year-built) and was removed from that location in some year (i.e., year-demolished)
This ignores the possibility of incremental replacement of parts (e.g., spans), which might have different designers.
Each bridge is described by zero or more references. The reference is considered non-decomposable.

Each span of a bridge has a principal type and material. The spatial ordering of spans is not considered.

Each span of a bridge has a clearance.

Each span of a bridge has a length.

Each span of a bridge has zero or more lanes.
Each lane of a span of a bridge has a lane-width. The total-width is a transitive dependency and thus is ignored.

Each span of a bridge has zero or more walks.

Each walk of a span of a bridge has a walk-width.

Each owner is identified by a unique owner-id and has an owner-name.

Each designer is identified by a unique designer-id and has a designer-name.

Each designer has a birth-date and zero or more degrees each consisting of a degree, institution and year.
Bridge Example Animation

• View the step-by-step dependency diagram creation process at http://fen.ce.cmu.edu/cae99/content/caelectures/dbms/depend/depend.html
3a. Design the Tables

- Draw a dependency diagram
- Each dependency statement is a part of the diagram
- Each statement is a single path through the diagram
- Tables are formed by traversing the dependency diagram
3b. Traversing the Diagram

- Choose an attribute at the end of a path
- Follow the chain of arrows upwards
  - each multi-valued dependency on the path becomes a primary key for the table
  - combine all single-valued attributes at at first level up into a single table
  - all attributes on the path should be included in the table
  - stop when you reach a bubble that has no arrows coming into it
  - each path becomes a separate table
- Mark off your traversed path
- Repeat until all paths have been traversed
Resulting Tables from Bridge Example

Syntax: name-of-table (primary key, field-name)

- bridge-names (bridge-id, bridge-name)
- bridge-designers (bridge-id, designer-id)
- designers (designer-id, designer-name, birthday)
- designer-education (designer-id, degree, institution, year)
- bridge-owners (bridge-id, owner-id)
- owners (owner-id, owner-name)
- bridge-construction (bridge-id, year-built, year-demolished)
- bridge-references (bridge-id, reference)
- span-description (bridge-id, span-id, type, material, clearance, length)
- lane-widths (bridge-id, span-id, lane-id, lane-width)
- walk-widths (bridge-id, span-id, walk-id, walk-width)
Common Mistakes

- Guessing the design, not following the process
- Storing what you can compute (when the value will change)
- Assuming the order of rows and columns is known
- Represent multivalued dependencies in fixed size sets