Relational Database Design: Database Design Priciples

Thomas H. Grayson
23 January 2002

The Relational Model

All data are represented as tables
- The results of any given query are just another table!

Tables are comprised of rows and columns

Rows and columns are (officially) unordered (i.e., the order in which rows and columns are referenced does not matter).
- Rows are sorted only upon request. Otherwise, their order is arbitrary, and may change for a dynamic database
- Column order is determined by each query

Each table has a **primary key,** a unique identifier constructed from one or more columns

Most primary keys are a single column (e.g., TOWN_ID)

A table is linked (joined) to another by including the other table's primary key. Such an included column is called a **foreign key**

Primary keys and foreign keys are the most important concepts in database design. Take the time to understand what they are!

Qualities of a Good Database Design

Reflects real-world structure of the problem

Can represent all expected data over time

Avoids redundant storage of data items

Provides efficient access to data

Supports the maintenance of data integrity over time

Clean, consistent, and easy to understand

Note: These objectives are sometimes contradictory!

Introduction to Entity-Relationship Modeling

Entity-Relationship (E-R) Modeling: A method for designing databases

A simplified version is presented here

Represents the data by **entities** that have **attributes**.

An entity is a class of distinct identifiable objects or concepts

Entities have **relationships** with one another

Result of the process is a **normalized** database that facilitates access and avoids duplicate data
*Note: Much of formal database design is focused on **normalizing** the database and ensuring that design conforms to a **level of normalization** (e.g., first normal form, second normal form, etc.). This level of formality is beyond this discussion, but one should know that such formalizations exist.*

E-R Modeling Process

Identify the entities that your database must represent

Determine the **cardinality** relationships among the entities and classify them as one of

- **One-to-one** (e.g., a parcel has one address)
- **One-to-many** (e.g., a parcel may be involved in many fires)
- **Many-to-many** (e.g., parcel sales: a parcel may be sold by many owners, and an individual owner may sell many parcels)

Draw the entity-relationship diagram

Determine the attributes of each entity

Define the (unique) primary key of each entity

Database Issues with ArcView®

ArcView® only allows joins using a **single** column

- Forces users to resort to elaborate procedures to build a single-column key

ArcView® cannot work with expressions "on the fly"

- Percentages and ratios need to be stored in extra columns
- These values can get out of data and require significant extra work to create

ArcView® does not handle many-to-many relationships well

- A one-to-many join yields an arbitrary value in the joined table
- A "link" rather than a join provides some many-to-one capability, but it is limited

Limited tools for aggregation and "grouping" (Field | Statistics and Field | Summarize)

Awkward tools for dealing with external database management systems (e.g., Oracle)

From E-R Model to Database Design

Entities with **one-to-one** relationships should be merged into a single entity

Each remaining entity is modeled by a table with a primary key and attributes, some of which may be foreign keys

One-to-many relationships are modeled by a foreign key attribute in the table representing the entity on the "many" side of the relationship

Many-to-many relationships among two entities are modeled by a third table that has foreign keys that refer to the entities. These foreign keys should be included in the relationship table's primary key, if appropriate

Commercially available tools can automate the process of converting a E-R model to a database schema