Understanding RDBMS

IT4GIS
Keith T. Weber, GISP
GIS Director
ISU-GIS Training and Research Center

FUNDAMENTALS

RDBMS

- Relational Database Management System
- The "I" in GIS (Information)
BTW

• The Data-to-Wisdom Pathway

Data  Information  Knowledge  Wisdom


Database software...

• Light Duty
• Medium Duty
• Heavy Duty
Relational Databases

• Why are databases important?
  – Drive many business processes
  – Store large amounts of data
  – Retrieve data quickly

Spreadsheets vs. Databases

• Integrity!
• Structure

RDBMS CONCEPTS AND TERMS
Independence

- Physical
- Logical

Logical Consistency Example

- Character based database design
  - FirstName (1-4)
  - LastName (5-10)
  - Address (11-46)

- Record #1: Paul Bunyun, 100 Main Street, Pocatello, ID 83201
  NOTE: Record #2 starts at character #47

What Happens When We Add a New Field?

- New Field = ZIP+4 (47-50)
  - Example, Paul's ZIP+4 = 1234
- Scripts written and referring to the original design will fail
- Record #1: Paul Bunyun, 100 Main Street, Pocatello, ID 83201
- Record #2: 1234 Johnhe, nry 150 Main Street, Pocatello, ID 8
Integrity

- Important for consistency and transaction management.
- Types:
  - **Domain**: all values come from predefined domains or are null
  - **Redundancy**: problems can occur as a result of repetitive storage that is not consistently updated and from stored data that is derived from other stored data. Redundant data must be consistent.

Integrity Types (cont’d)

- **Constraint**: Business integrity. Stored data must not violate business rules.
- **Entity**: Every record must be uniquely identifiable (index field or ObjectID)
- **Referential**: Relationships must not be ambiguous. Two types…
  - Cascading or non-cascading

Enforcing Integrity Rules

- Programmatic
- Systematic
Key Fields versus Index fields

- Unique Identifiers are Index fields
- Relate fields are Key fields
  - Primary key
  - Foreign key

RDBMS STRUCTURE

Database Tables

- Database
  - Table1
  - Table2
  - Table3
Table Structure

<table>
<thead>
<tr>
<th>ROW 1 (RECORD OR ENTITY)</th>
<th>COLUMN 1 (FIELD OR ATTRIBUTE)</th>
<th>COLUMN 2</th>
<th>VALUE</th>
</tr>
</thead>
</table>

Data Value Types

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Storage Occupied, data value</th>
<th>Valid Domain Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Integer</td>
<td>2 bytes</td>
<td>-32768 to 32767</td>
</tr>
<tr>
<td>Long Integer</td>
<td>4 bytes</td>
<td>-2147483648 to 2147483647</td>
</tr>
<tr>
<td>Float</td>
<td>4 bytes</td>
<td>Any number from -3.402823e38 to 3.402823e38</td>
</tr>
<tr>
<td>Double</td>
<td>8 bytes</td>
<td>Any number from -1.7976931348623157e308 to 1.7976931348623157e308</td>
</tr>
<tr>
<td>Text (string)</td>
<td>10 + max. length = bytes</td>
<td>Any alphanumeric characters</td>
</tr>
<tr>
<td>Date</td>
<td>8 bytes</td>
<td>Jan 1, 100 to Dec. 31, 9999</td>
</tr>
<tr>
<td>LOB (variant)</td>
<td>22 + max. length = bytes</td>
<td>Any alphanumeric characters</td>
</tr>
</tbody>
</table>

(BTW) Raster Data Types Worth Knowing

- **1_BIT** — A 1-bit unsigned integer. The values can be 0 or 1.
- **2_BIT** — A 2-bit unsigned integer. The values supported can be from 0 to 3.
- **4_BIT** — A 4-bit unsigned integer. The values supported can be from 0 to 15.
- **8_BIT_UNSIGNED** — An unsigned 8-bit data type. The values supported can be from 0 to 255.
- **8_BIT_SIGNED** — A signed 8-bit data type. The values supported can be from -128 to 127.
- **16_BIT_UNSIGNED** — A 16-bit unsigned data type. The values can range from 0 to 65,535.
- **16_BIT_SIGNED** — A 16-bit signed data type. The values can range from -32,768 to 32,767.
- **32_BIT_UNSIGNED** — A 32-bit unsigned data type. The values can range from 0 to 4,294,967,295.
- **32_BIT_SIGNED** — A 32-bit signed data type. The values can range from -2,147,483,648 to 2,147,483,647.
- **32_BIT_FLOAT** — A 32-bit data type supporting decimals.
- **64_BIT** — A 64-bit data type supporting decimals.
Making Sense of all this...

- Recall, there are 8 bits in 1 byte
- Cross-reference
  - 8-bit is byte data
  - 16-bit is short integer (2 bytes)
  - 32-bit (signed or unsigned) is long integer (4 bytes)
  - 32-bit (float) is single-precision floating point (4 bytes)
  - 64-bit is double-precision floating point (8 bytes)

DATABASE DESIGN

- Basic Steps in Database Design
  - Understand and document the business' needs.
  - Problem statement
  - Business object types
  - Business relationships
  - Business constraints
  - Create an ERM
  - Data and process inventory
  - Develop tuple types
  - Tuple types to tables
  - Integrity
  - Populate the database
READING A BUSINESS STATEMENT

Identify Candidate Classes

- A candidate class may or may not remain a class throughout the design process
  - A candidate class may or may not become a table
- Try not to think about tables when reading the business statement at this point

Think Object-Oriented

- Classes are nouns
- A noun is a “person, places, and things”
Methods

• Identifying *candidate* methods allows us to better understand how the business operates and how the Enterprise uses GIS data.
• A method is a behavior...a relationship between classes
• Ultimately, a connection between two tables
• The candidate methods will describe an inheritance, aggregation, or dependency relationship

And now...Verbs

• *Candidate* methods are verbs
  – They show action
  – They are behaviors

Create an Entity Relationship Model (ERM)

• Symbolized.
  – Standard Representation
  – Attribute Representation
  – Entity Instance Representation

<table>
<thead>
<tr>
<th>BUILDINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>K BldgNum: 126</td>
</tr>
<tr>
<td>Name: Graveley Hall</td>
</tr>
<tr>
<td>Type: Education</td>
</tr>
</tbody>
</table>
Relationships

• Determine the Relationship between Entity Types.
• Add these to the ERM

(more about database relationship classes later in the semester)

Data and Process Inventory

• Database Dictionary
  – BldgName
    • The name of the building
  – Type
    • Primary use of the building (e.g., 0 = Unknown or n/a; 1 = Education, 2 = Offices, etc.)
  – Floors
    • The total number of floors

Develop Tuple Types

• Use your ERM with relationships
• Perform a “Walk-through” exercise
  – Simulate data is being added/retrieved in your database.
• Create another, more mature ERM using Attribute Representation
Normalization

- First-Fifth Form Normal (1FN, 2FN, ..., 5FN)
- Academic
- Applied

1FN

- “All values are atomic”
  - Each cell in the table contains only a single data value
- Eliminate repeating groups
  - Puppy_Trick1, Puppy_Trick2, ...
Check this (1FN)...

2FN

- Satisfy 1FN and...
- Redundant data must be eliminated
  - How?
  - Example: Puppy_ID, Trick_ID, Trick_Name

Check this (2FN)...

---
3FN

- Satisfy 1NF and 2NF and...
- No non-key attributes are dependent on other non-key attributes.
  - Example: Appointment_ID, Name, Date, Time, Species

After Normalization

- New **tuple types** will be created.
- New **tables** will be planned.
- Many-many relationships will be handled using associative tables (bridge tables).

De-Normalization

- What? Is this heresy?
Designing the Actual RDBMS

• Visual modeling based upon your ERM and Tuple type model.
• Implementation of integrity rules based upon your business constraints.

Populate the Database

• Questions and concerns to revisit
  – Null data
  – Reporting discrepancies and variations
  – Measuring or estimating methods
  – Client utility/efficiency

The Last Step?

Validation!
Professional Hints and Tips

- Using Google drive, sharing files/folders and communicating this in email

Questions?