Eliminating Database Design Mistakes

Carol Keeney
Database Goddess
BKeeney Software, Inc.

Today's Focus

- Logical Database Design
- Implementing the Database
- Databases and Xojo

Database Design

Principle 1:
Begin with the End in Mind

What is a good database design?

- Reduced redundant data
- Helps ensure data integrity
- Supports business processes and reporting needs

How do I create a good database design?

- Focus on your purpose
- Find the data you need
- Organize the data
- Normalize to make it better

Define your purpose

- What do you already know?
- Document the purpose, who will use it and how will they use it
- Use the document throughout

Find the data you need

- Look at existing information
- Collect data on major subject areas
- Think about how the user will want to see the information

Database Design Example

Pizza Orders

Last Name	First Name	Delivery Address	Phone	Delivery?	Requested Time	Order	CC
Smith	John	123 Main	(913) 555-1212 - cell	x	12:30 pm	4 Large Pepperoni	Visa 1344
Smith	John	1020 Elm Street	(217) 999-2919	X	7:30 pm	1 Large Pepperoni, 1 Small Cheese	AmEx 1234
Jones	Mary	111 Oak Street	313-222-1231 x. 111		6:00 pm	2 Medium Pepperoni	MC 1112
Jones	Bob	111 Oak Street	404-358-2121 Ask for Bob	X	6:30 pm	2 Large Pepperoni	MC 1112

Organize Your Data
Subject Areas / Tables

Customers: Name Address Phone	Orders: Order Items Payment Delivery
Food Items: Item Type Quantity Available	Equipment: Location Status

Think about the Details - Tables/Entities

Customer
Name: First Name, Last Name
Contact Info: Address, Phone, Email
Status

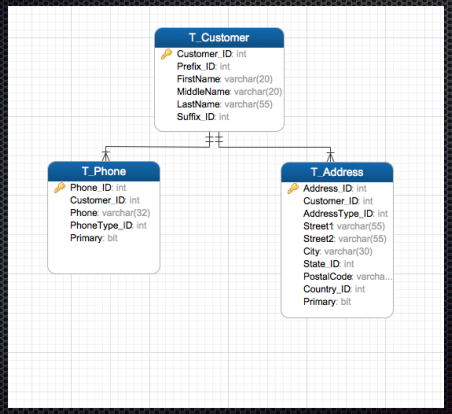
Database Design
Principle 2:
A Picture Is Worth 1000 Words

What's a Data Model?

Graphical representation of tables, columns and relationships. It includes:

- Entities: Anything of interest about which data can be stored. ("Table")
- Attributes: Single piece of data about the entities. ("Column")
- Relationships between entities ("Foreign Keys").

Sample Data Model



Database Design
Principle 3:
When In Doubt, Split It Out!

"Normalize" the Data

From Wikipedia:

Database normalization is the process of organizing the fields and tables of a relational database to *minimize redundancy* and dependency. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database using the defined relationships.

Normalization - Why?

- Over time, the data design tends to remain stable even when the processes manipulating the data are in flux.
- Isolating the data design from the code via views (data independence) provides stability.
- The goal is to have all of the fields (columns) in the table be dependent on the key of the table.
- There will be less issues when data is updated.
- Makes general purpose reporting/querying easier

Normalization

- Steps:
  - List the entities (tables) & attributes (columns).
  - Define the logical key.
  - Assign the attributes to the appropriate entity.
  - Normalize: Make sure there are no repeating groups and the attributes are all dependent on the logical key.
- Logical design can be different from the physical design implemented.

Normalization

- First Normal Form: No repeating groups
  - phone_1, phone_2 is bad!
- Second Normal Form: No key combos
  - employee, project, department
- Third Normal Form: The key & nothing but the key
  - employee name, department and department name

Normalization Examples

Pizza Orders

Last Name	First Name	Delivery Address	Phone	Delivery?	Requested Time	Order	CC
Smith	John	123 Main	(913) 555-1212 - cell	x	12:30 pm	4 Large Pepperoni	Visa 1344
Smith	John	1020 Elm Street	(217) 999-2919	X	7:30 pm	1 Large Pepperoni, 1 Small Cheese	AmEx 1234
Jones	Mary	111 Oak Street	313-222-1231 x. 111		6:00 pm	2 Medium Pepperoni	MC 1112
Jones	Bob	111 Oak Street	404-358-2121 Ask for Bob	X	6:30 pm	2 Large Pepperoni	MC 1112

Standard Designs in Database Design

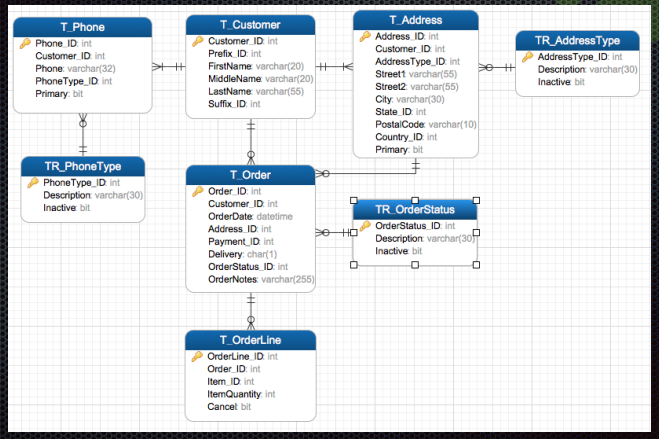
Header Detail Example

- This Occurs In:
- Orders / Order Lines
- Department / Employees
- One to Many

Order #	Customer #	Date	Order Status
1	3	1/1/14	In Process
2	32	1/3/14	Complete
3	6	2/2/14	Cancelled
4	7	2/7/14	In Process

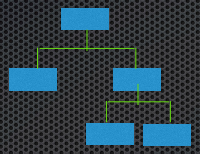
Order #	Line #	Item	Quantity
1	1	Pepperoni	5
1	2	Breadsticks	2
1	3	Cola	3
2	1	Salad	2

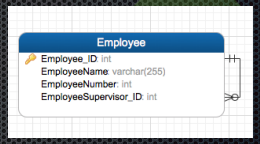
Header Detail Example



Hierarchy Example

- Employee Org Chart
- Table points to itself
- Big Boss
  - Mr. VP
    - Worker
    - Worker





Employee ID	Name	Employee #	Supervisor
1	Big Boss	1	
2	Mr. VP	2	1
3	VP 2	5	1
4	Joe Worker	3	2
5	Jill Worker2	4	2

Summary: Good DB Design

- Document your purpose and use that throughout the project
- Organize the data
- Normalize to make it better

Database Design Tips

- Design Tips
- Naming Standards
- Design Tools

Design Tips

- Keys & Indexes
  - Use non-meaningful primary keys.
  - Let the database do the work
- Don't store calculated data
- When in doubt, split it out

Design Tips

- Views are your friend…..

Design Tips

- And Triggers are not…..

Naming Standards

Table & Column Standards

- Use non-plural names
- Don't use "data" or "info" in the name
- Don't use "ment","tion","ance", or "ing"
- Table Name is prefix + subject area + details
  - TR_CUSTOMER_TYPE
  - T_ORDER_LINEITEM
- Column Name is attribute + classification
  - Customer_ID, OrderDate, OrderAmount

Cross Database Standards

- Prefix: T_ (table), V_ (view), S_ (stored procedure), IDX_ (index), PK_ (primary key), FK_ (foreign key)
- Split by table type:
  - TR_ (reference or "lookup" tale)
  - TX_ (cross reference)
  - T_ (standard table)

Naming Standards - Other Considerations

- Don't use special characters or spaces
- Don't use keywords
- Case? CamelCase; include underscores?
- Standard abbreviations to be used in table and column names:
  - Subject Area: Cust vs. Customer, Emp vs. Employee
  - Classifications (prefix or suffix?): num vs no vs number, date vs. dt, description, amount, id vs. key
  - Other acceptable abbreviations (app specific): PR (payroll)

Data Design Tools?

- True data base design tools:
  - Toad, ERwin (Windows)
  - Navicat (Mac, LINUX, Windows)
- Less expensive options - draw pictures:
  - MS-Access, Visio (Windows)
  - OmniGraffle (Mac)
  - Presentation tools - Keynote, PowerPoint

Databases & Coding In Xojo

ActiveRecord

- Maps tables to classes
- Maps columns to properties
- Generic save/delete methods
- Design time help
  - IDE auto complete of tables/fields
  - Compiler catches data type mismatches
  - Throws exception for missing columns

ActiveRecord (2)

- Uses prepared statements
- Has events with transaction support
  - Before/After Create
  - Before/After Update
  - Before/After Delete
- Design database first *before* coding
- Business logic in one place

ARGen

- Uses database to create ActiveRecord project
- AR data classes in their own name space
- Options on how to handle property names
- Version 3.0 released last year:
  - Allow UUIDs for primary keys
  - Support SQLite in iOS
  - Generates UI for desktop and web
  - Support audit trails

Summary

- Begin with the End in Mind
- A Picture is Worth 1000 Words
- When in Doubt, Split it Out
- ActiveRecord is an option for making things easier; ARGen makes ActiveRecord easier

Questions?

- Carol Keeney
- carolk@bkeeney.com

Please be sure to give us feedback in th XDC app!

BKeeney Software, Inc.