Refactoring Databases
Evolutionary Database Design

Scott W. Ambler, IBM
Pramod Sadalage, ThoughtWorks

Questions?

Please ask burning questions during the tutorial

The only stupid question is the one that you don't ask
Tutorial Overview

- Introduction
- Evolutionary Database Development I
- Evolutionary Database Development II
- Technical Foundation
- Conclusion

Introduction

- What You’ll See
- Traditional Databases
- Agile Databases
- Agile Data Method
- Why We Need This
- Everyone Gets a Database
- First Time Deployment
- Configuration Management
- Demo
What You’ll See

- Proven techniques for:
  - Evolutionary database development
  - DBAs to be effective members of agile teams
  - Improving the quality of your data assets
  - Organizations to improve their approach to data

Question

Can you rename a column in your production database and safely deploy the change in a single day?
Answer: Yes you can!

The Traditional Database

- Continuous changes not allowed
- Production changes are rare
- Migration is a project
- Non-existent change management
- Non-existent testing
The Agile Database

- Slowly and safely changing database
- Functionality added in increments
- Change management
- Facilitates automated testing
- Knowledge by developers of the functionality
- Acknowledged importance of team interaction
- DBA = Role != Person

Philosophies of Agile Data Method

www.agiledata.org

1. **Data.** Data is one of several important aspects of software-based systems.
2. **Enterprise issues.** Development teams must consider and act appropriately regarding enterprise issues.
3. **Enterprise Groups.** Enterprise groups exist to nurture enterprise assets and to support other groups, such as development teams, within your organization. These enterprise groups should act in an agile manner that reflects the expectations of their customers and the ways in which their customers work.
4. **Unique situation.** Each development project is unique, requiring a flexible approach tailored to its needs. One software process does not fit all and therefore the relative importance of data varies based on the nature of the problem being addressed.
5. **Work together.** IT professionals must work together effectively, actively striving to overcome the challenges that make it difficult to do so.
6. **Sweet spot.** You should actively strive to find the “sweet spot” for any issue, avoiding the black and white extremes to find the gray that works best for your overall situation.
People

Why pair with the DBA?

- Helps DBA understand the application
- The DBA has a better understanding of other areas of the business application
- Will help you write the change scripts
- Will help you to write database tests
- Data migration code for production data is critical
- Gain knowledge of SQL Tuning etc
- Make the team aware of production data

Why Do We Need This?

Modern Development

- Leading development lifecycles are iterative and incremental
- New applications typically use a combination of object and data technologies – the world isn't pure
- There is growing proof that serial/traditional approaches are ineffective
- Data in only one of many important aspects, and it is rarely the primary one
- There is growing evidence that craftspeople, or “generalizing specialists”, are significantly more effective than specialists
- Traditional IT organizations appear to be vulnerable to offshore outsourcers

Cultural

- There is a serious rift between the data and development communities
- Each community is relatively ignorant of what the other one is doing
- There are few books within either community that do the other justice
- The agile movement is coming from the development community
- Many developers are prepared to ignore data professionals, and do so quite often
- White boards are the most common CASE tool
Everyone Gets a Database

- Database instance = application instance
- Developers work independent of other developers
- Liberty of experimenting with the database
- Use ANT/Maven to do database tasks.

Copyright 2001-2007

Why Everyone Needs a Database

Database schema
Point of contention during development

Copyright 2001-2007
show image with multiple computers and users with db on laptop.

Pramod Sadalage, 8/9/2004
When everyone is truly independent

First Time Deployment

- Should be handled almost exactly the same as deploying code
- Database master script is used to create the production instance of the database.
- Branch if necessary
- The production instance will be either a clean copy of the application database schema or full of converted data
show image with multiple computers and users with db on laptop.

Pramod Sadalage, 8/9/2004
Schema and Setup Data

- Setup data; is non-transactional, enables application functionality
- Is not test data
- Every alteration to the schema and setup data must be chronologically logged and source-controlled with the application code
- A fully-functional database instance can be built at any time for any build in the development timeline
- Old database instances can be rolled forward by applying the logged changes made to schema and setup data

Every Project Has ...

- Schema creation script known as master DB creation script
- Source controlled database code
- Setup data creation script
- Collection of change scripts that are maintained by Build/Day/Iteration etc
Typical Environment

Developer — Source Control — Cruise Control — DB Server

Check in — Check out — Check out

Uses

Demo

When Andy joined our project.
Evolutionary Database Development I

Agile Model Driven Development
Agile Data Modeling
Database Refactoring (Intro)
Why Database Refactoring is Hard
Demo

Copyright 2001-2007
Survey Says: Agilists are Modeling


- Whiteboard Sketching: 92.7%
- Init. Agile Req. Modeling: 77.7%
- Init. Agile Arch. Modeling: 77.2%
- Paper Modeling: 65.9%
- CASE Tool Modeling: 47%

Agile Data Modeling

- Data modeling is the act of exploring data-oriented structures
- Evolutionary data modeling is data modeling performed in an iterative and incremental manner
- Agile data modeling is evolutionary data modeling done in a collaborative manner
- New requirements motivate changes to your database schema, just as they motivate changes to your application code
A database refactoring is a simple change to a database schema that improves its design while retaining both its behavioral and informational semantics.

A database schema includes both structural aspects such as table and view definitions as well as functional aspects such as stored procedures and triggers.

Important: Database refactoring is a subset of schema transformations, but they do not add functionality.

www.databaserefactoring.com
www.agiledata.org/essays/databaseRefactoring.html

Why DB Refactoring is Hard

Persistence Frameworks
Other Applications You Know About
Your Application
Other Applications You Don’t Know About
Other Databases
Your Database
Data File
Data Imports
Data Exports
Test Code

Replace Column

Original Schema

Customer
CustomerPOID <<PK>>
CustomerNumber: integer
FirstName
LastName

Transition Period

Customer
CustomerPOID <<PK>>
CustomerNumber: integer { drop date = June 14 2009 }
CustomerID: char(12)
FirstName
LastName

SynchronizeCustomerIDNumber
{ event = update | insert, drop date = June 14 2009 }

Resulting Schema

Customer
CustomerPOID <<PK>>
CustomerID: char(12)
FirstName
LastName

Demo

Replace Column

http://www.databaserefactoring.com/catalog/replaceColumn.html
Evolutionary Database Development II

Schema Scripts
Database Refactoring Examples
Deployment
Demo

Copyright 2001-2007

Merge Columns

Copyright 2001-2007
Demo

Replace One-To-Many With Associative Table
http://www.databaserefactoring.com/ReplaceOneToMany.html

Technical Fundamentals

Continuous Integration
Ant Targets for the Database
Allow Developers to Experiment
Database Regression Testing
Encapsulating DB Access
Clean Tests
Infrastructure
Demo
Refactoring is Continuous, Deployment is Controlled

Continuous Integration

Copyright 2001-2007
Demo

The Schema Scripts (example)
ANT Targets for the database

build.xml

```xml
<target name="dbinit" depends="init">
  <sql password="\$\{password\}" userid="\$\{username\}" url="\$\{db.url\}" driver="\$\{driver.name\}" classpath="\$\{driver.classpath\}"/>

  <fileset includes="drop_all_objects.sql" dir="db"/>
  <fileset includes="create_sequences.sql" dir="db"/>
  <fileset includes="create_tables.sql" dir="db"/>
  <fileset includes="create_setup_data.sql" dir="db"/>
  <fileset includes="create_indexes.sql" dir="db"/>
  <fileset includes="create_views.sql" dir="db"/>
  <fileset includes="create_comments.sql" dir="db"/>
  <fileset includes="create_constraints.sql" dir="db"/>
</sql>
</target>
```
Allow Developers To Experiment

- Developers can change the model in their local schema.
- The DBA should be involved in the design of the functionality being developed.
- The DBA should help the developers to design the tables, Indexes, tune queries for performance.
- DBA and Developers should pair all the time.

Database Regression Testing

www.agiledata.org/essays/databaseTesting.html
Encapsulate the Database

- Coupling is enemy #1
- Encapsulation is your ally
- Encapsulation strategies:
  - Brute force (embedded SQL)
  - Data access objects
  - Persistence frameworks
  - Services

Infrastructure

- Start converting data when replacing legacy app; day 1 of the project.
- If feasible use light weight DBMS, while in development e.g. Hypersonic, MySQL etc.
- Write a data generator, when needed.
- Follow enterprise naming conventions, ... e.g. Apply Modeling Standards
Production

Handling the application in production is surprisingly simple. Because:

- The database is being developed, tracked and sourced with the same rigor as code
- The Master script is your initial deployment script
- The Change Scripts are your migration scripts, for new releases or patches.

Demo

How to make an initial release
All of the change scripts should be applied following the last production release in chronological order. (Change Logs are stored by Build/Release/Day)

Since data migration scripts are part of change logs, data is also migrated.

It is wise to test the migration on a test database.
The Change Script

- A stored collection of all changes (Schema and Data) made to the master script.
- Will likely be written in SQL, if feasible be automatically generated by cruisecontrol.
- Should be source controlled
- Is the tool by which instances will be rolled forward (backward if needed)
- Should be named by Build/Release/Day etc
- Make every change such that rolling them back is possible, especially in production situations.

Integrate more than once a day.
http://cruisecontrol.sourceforge.net

The Change Script example

PROMPT Upgrade to build 184
BEGIN
  IF upgradecontroller.canupgradefromversion(183) THEN
    BEGIN
      EXECUTE IMMEDIATE 'ALTER TABLE customer ADD verified NUMBER(1)';
      INSERT INTO (movie_id,movie_description,movie_rental_rate,movie_title)
      VALUES (1124,'Pixar Movie',3,'Finding Nemo');
      upgradecontroller.upgradetoversion(184);
      COMMIT;
    END;
  END IF;
END;
/

Oracle pl/sql example
Demo

How to make next release

Thinking Forward

- Production database instances can be migrated to a future state of deployment.
- New releases can be acceptance tested against the most recent data currently in production.
- Migration is a snap, so why not deploy weekly or frequently.
Conclusion

- What You Saw
- Why We Need This
- Additional Resources
- Keep In Touch

---

What You Saw

- Proven techniques for:
  - Evolutionary database development
  - DBAs to be effective members of agile teams
  - Improving the quality of your data assets
  - Organizations to improve their approach to data
Why We Need This

- Leading development lifecycles are iterative and incremental
- It leads to greater levels of data quality than traditional approaches
- New applications typically use a combination of object and data technologies – the world isn't pure
- There is growing proof that serial/traditional approaches are ineffective

Additional Resources

- groups.yahoo.com/group/agileDatabases/
- www.agiledata.org
- www.databaserefactoring.com
- www.martinfowler.com
Keep In Touch

scott_ambler@ca.ibm.com
pramod@thoughtworks.com

Backup Slides
The Database Instance

- The database analog to the application instance
- A data space containing, at the very least, the schema and setup data necessary to support an application instance
- Independent of other data spaces
- Not able to affect other data spaces

Database Instance Examples

Oracle Database

Microsoft Access

Sybase Server

MySQL Server
Why does the DBA need it?

- Easier to maintain specialized datasets
  - for development, automated testing, demos, load testing, production copies, etc.
- Ability to migrate older datasets
- Allows the database to be another component in the system, with a build number and a context

Why Would the Team Want It?

- To allow every developer the ability to code and test with his or her own, independent database
  - Would you have an entire team develop all using the same single application instance?
- To make the database portable and restorable
  - A developer can code and test against a specific set of data. If the developer breaks the dataset, no one else is affected and the developer simply loads up a fresh copy.
- Promotes better team velocity
- Allows to fail faster
Is State full and Locatable

- E.g. iteration 2, regression test database
- E.g. build 1956

The Database as File System

Think of the database instance as if it were a file. Commands such as initialize, create and delete can be made applicable to the database instance.