"Database Schema Deployment"

php|tek 2006 in Orlando Florida

Lukas Kahwe Smith
smith@pooteeweet.org

Agenda:

- The Challenge
- Diff Tools
- ER Tools
- Synchronisation Tools
- Logging Changes
- XML Formats
- SCM Tools
- Install Scripting
- Update Scripting
- Alternative Approaches

Terminology:

- DDL
  - Data Definition Language
    - CREATE
    - ALTER
    - DROP
- DML
  - Data Manipulation Language
    - INSERT
    - UPDATE
    - DELETE

The Challenge: Overview

- Getting the DDL and DML
  - SQL diff tools can generate DDL
    - Usually RDBMS specific
  - Data synchronisation very tricky
    - Especially if DDL and DML needs to be mixed
  - Deal with object dependencies
    - Foreign Keys, Views, Stored Routines
  - Column order matters with sloppy code
- No additional steps during development
- Less steps during packaging
- Allow releases to be skipped

The Challenge: Example

- Version 1.0.0
  - User database with a single phone number
- Version 1.1.0
  - Allow infinate phone numbers per user by adding a phone numbers table
    - Add new table phone numbers
      - CREATE TABLE phone_nums (user_id INT REFERENCES user, phone_num CHAR(20))
    - Move all data into the new table
      - INSERT INTO phone_nums SELECT user_id, phone_num FROM users
    - Drop the old phone numbers column
      - ALTER TABLE users DROP COLUMN phone_num

Diff Tools: Overview

- Generate DDL by comparing
  - SQL files
  - Installed schema
- Does not handle DML
- Tend to be RDBMS specific
- Examples
  - SQLYog (Win, MySQL, $)
  - Toad (XXX, different RDBMS, $$)
  - AdeptSQL (Win, MS SQL, $$)
  - Most modeling tools

Diff Tools: Example

Playing with SQLYog

ER Tools: Overview

- ER Modeling tools
  - Visually design schema
  - Synchronize model
  - Reverse engineer model
- Examples
  - DBDesigner (Win, Generic)
  - MySQL Workbench (Win/*nix, MySQL)
  - PowerDesigner (XXX, Generic, $$$)
  - ERWin (XXX, Generic, $$$)
  - Visio (Win, Generic, $$)

ER Tools: Example

Playing with DBDesigner

Synchronisation Tools:

- Find differences in data
- One way synchronisation is easy
- Two way synchronisation is tricky
- Only useable in the rare case where all clients have the same data
  - No way to generate DML to make the same changes on different data

Logging Changes:

- PostGreSQL: log_statement “mod“
- MYSQL: binarylog with mysqlbinlog util
- Oracle: AUDIT command
- DB2: db2audit command
- Alternative approach
  - Cronjob that checks for changes in the information schema
    - Only handles DDL
  - Write all DDL and DML to a log and only execute changes from the log

XML Formats: AXMLS

- RDBMS independent XML format
- Bundled with ADODB
- Supports
  - Tables
    - Columns
    - Autoincrement
    - Constraints (not fully abstracted)
    - Indexes
    - Initialization
  - Queries (not abstracted)
- Create, alter, remove schema
- Execute directly or dump statements

XML Formats: Metabase XML

- Mostly same feature set as AXMLS
  - Adds support for sequences and variables
  - XML format uses no attributes
  - No support for “plain“ queries
  - No support to remove schemas
  - Only support for primary/unique constraints
- Implemented in
  - Metabase
  - PEAR::MDB2_Schema
  - ezc/DatabaseSchema
- DBDesigner exports to Metabase XML

XML Formats: Example

Playing with PEAR::MDB2_Schema and WebBuilder2 Application framework

SCM Tools:

- Standard SCM work line based
  - Needs SQL parser in order to work statement based
  - Few SQL aware solutions available
    - Daversy (Win, SQLite/Oracle)
- Keep one database object per file
  - Watch out for dependencies
    - VIEWs
    - Stored Routines
    - Triggers and Foreign Keys

Install Scripting: Getting Started

- Dump test master for every release
  - Advantage
    - More or less automated
  - Disadvantage
    - No handling for DML
- Initial dump + all DDL and DML
  - Check current schema before applying
    - Can be applied to any version
  - Advantage
    - A singe script for install and upgrades
  - Disadvantage
    - Gets increasingly long

Install Scripting: Dependency Hell

- Native dump tools handle dependencies
- Create dependency graph
  - Figure out dependencies
  - Order statements accordingly
- Use dummies
  - Create dummy implementations of all referenceing database objects
    - VIEWs
    - Stored Routines
  - Replace dummies with actual implementation

Update Scripting: Get DDL and DML

- Log every DDL and DML
- Diff between current and last release
- Compare diff against DDL and DML log

- LOG
  - CREATE TABLE phone_nums (user_id INT REFERENCES users, phone_num INT)
  - ALTER TABLE phone_nums MODIFY phone_num CHAR(20)
  - (2) INSERT INTO phone_nums SELECT user_id, phone_num FROM users
  - (3) ALTER TABLE users DROP COLUMN phone_num

- DIFF
  - ALTER TABLE users DROP COLUMN phone_num
  - (1) CREATE TABLE phone_nums (user_id INT REFERENCES users, phone_num CHAR(20))

Update Scripting: Organize DDL and DML

- Ordered list of DDL and DML changes
  - Dependency order follows from log
  - Every change has
    - unique name per release
    - code to detect if the change is required
    - potentially a rollback script
- Ordered list of data unrelated objects
  - Views and summary tables
  - Stored routines

Update Scripting: Code Flow

- Determine version and integrity of current installed database
- Load all necessary changes
  - Enclose in transaction
    - not supported in MySQL
  - Load previous changes if necessary
    - Hard code deviations from previous releases by referencing the unique change name
      - Skip buggy irrelevant/changes
      - Fold multiple changes into single change
  - Reload data unrelated objects
  - Update table stats

Update Scripting: Notes

- Always explicitly hard code columns
  - ~~INSERT INTO foo VALUES (..); SELECT * ..~~
- Grants are a major PITA
  - Store grants with object definitions
- Old RDBMS versions might not support
  - New DDL: emulate with copy, drop, create
  - New optional features
    - MySQL only syntax: /*!50100 PARTITION .. */
  - backwards compatibility
- Optionally show list of statements before execution for additional security

Alternative Approaches: Some more ideas

- Plan ahead to minimize changes ;-)
- Keep old schema unchanged
  - Create a new schema for all new features
    - Use VIEWs to handle changes to existing tables
    - And/or copy old data to new schema as needed
  - Disadvantages
    - Schema becomes messy
    - Performance overhead

References:

- These slides
  - http://pooteeweet.org/files/phptek06/database_schema_deployment.pdf
- SQLYog
  - http://www.webyog.com/
- Toad
  - http://www.oracle.com/technology/products/designer
- DBDesigner
  - http://fabforce.net/dbdesigner4/
- MySQL Workbench
  - http://forums.mysql.com/list.php?113
- AdeptSQL
  - http://www.adeptsql.com/

References: More ..

- Sybase Powerdesigner
  - http://www.sybase.com/products/developmentintegration/powerdesigner
- ERWin
  - http://www3.ca.com/Solutions/Product.asp?ID=260
- Visio
  - http://office.microsoft.com/visio/
- Daversy
  - http://www.svn-hosting.com/trac/Daversy
  - http://www.svn-hosting.com/trac/Daversy/wiki/Dependencies

References: Still more

- PostGreSQL logging
  - http://www.postgresql.org/docs/8.1/interactive/runtime-config-logging.html
- MySQL logging
  - http://dev.mysql.com/doc/refman/5.0/en/binary-log.html
- Oracle logging
  - http://www.securityfocus.com/infocus/1689
- DB2 logging
  - http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/core/r0002051.htm

References: Yet More ..

- ADODB xml-schema
  - http://adodb-xmlschema.sourceforge.net/docs/index.html
- ezc/DatabaseSchema
  - http://ez.no/doc/components/view/latest/(file)/classtrees_DatabaseSchema.html
- PEAR::MDB2_Schema
  - http://pear.php.net/package/MDB2_Schema/
- WebBuilder2 Schema Manager
  - http://svn.oss.backendmedia.com/modules/schema_manager/schema_manager.phps
- SCM for databases?
  - http://blogs.ittoolbox.com/database/soup/archives/007666.asp

Thank you for listening ..
Comments? Questions?

smith@pooteeweet.org