

Database Schema Deployment

Lukas Smith - lukas@liip.ch
CodeWorks 2009 - PHP on the ROAD

Agenda

The Challenge

ER Tools

Diff Tools

Data synchronisation Tools

Logging Changes

XML Formats

SCM Tools

Manual Scripting

Terminology

DDL - Data Definition Language:
CREATE, ALTER, DROP

DML - Data Manipulation Language:
SELECT, INSERT, UPDATE, DELETE

The Challenge

Getting the DDL and DML to update/downgrade

SQL diff tools can generate DDL

Data synchronisation very tricky, especially of DDL and DML need to be mixed in a particular order

Do not deal well with object dependencies like Foreign Keys, Views, Stored Routines

Sloppy code can break even if theoretically the database schema is the same

Example

Version 1.0.0: User table with a single phone number

Version 1.1.0: Allow infinite phone numbers per user by adding a phone numbers table

Add new table: `CREATE TABLE phone_nums (user_id INT REFERENCES user, phone_num CHAR(20))`

Move all data: `INSERT INTO phone_nums SELECT user_id, phone_num FROM users`

Drop the old column: `ALTER TABLE users DROP COLUMN phone_num`

ER Tools

ER Modeling tools to visually design schema

Usually support schema reverse engineering and synchronizing to a file or live database

MySQL Workbench (Win/*nix,)

Visio (Win only, expensive)

PowerDesigner (very powerful and very expensive)

ERWin (very powerful and very expensive)

Diff Tools

Generate DDL by comparing SQL files and/or installed database schema

Do not handle DML

Tend to be RDBMS specific

SQLYog (Win only, MySQL only, fairly cheap *)

Toad (bit more expensive)

AdeptSQL (Win only, MS SQL only, bit more expensive)

Data synchronisation

Find and fix differences in data

One way synchronisation is easy

Two way synchronisation is tricky

Inefficient as in most cases schema changes requires only a few DML statements that can affect a lot of data

Only really useable for lookup tables that are fixed between development and production

XML Schema definition

Define schema in an RDBMS independent XML format

Store which version of the XML schema is used, to be able to compare two versions of the XML to compute the necessary changes

Do not handle DML

Limited support for various standard and non standard SQL features

One XML schema file covers all supported RDBMS

SQL version control

Standard VCS work line based to determine diffs

Would need an SQL parser in order to work SQL statement based

Daversy (Win only, SQLite/Oracle only, limited and buggy and a dead project)

Manual scripting

Update script with all necessary changes

Watch out when using auto generated DDL, often very convoluted SQL that tries to create temp tables

Keep a table in the DB to store which change scripts have been applied already

Question: Also maintain an up to date schema file for new installations?

Dependency Hell

Native dump tools often handle dependencies

Create dependency graph to figure out dependencies and order statements accordingly

Create dummy (non referencing) implementations of all referencing database objects and in a second pass replace dummies with actual implementation

Logging DDL and DML

PostgreSQL: log_statement "mod"

MYSQL: binarylog with mysqlbinlog util

Oracle/DB2: AUDIT/db2audit command

Drawback: Large change scripts that need to be manually optimized to reduce the required statements

Alternative approach: Write all DDL and DML to a log and only execute changes from the log

Use MySQL Proxy to write log automatically based on a comment in the SQL, but what happens if its forgotten?

Organize DDL and DML

Ordered list of DDL and DML changes

Dependency order follows from log

Necessary DML follows from log

Every change has a unique name per release, code to detect if the change is required and potentially a rollback script

Logging DDL and DML

LOG

```
CREATE TABLE phone_nums  
(user_id INT REFERENCES  
users, phone_num INT)
```

```
ALTER TABLE phone_nums  
MODIFY phone_num  
CHAR(20)
```

```
INSERT INTO phone_nums  
SELECT user_id, phone_num  
FROM users
```

```
ALTER TABLE users DROP  
COLUMN phone_num
```

DIFF

```
ALTER TABLE users DROP  
COLUMN phone_num
```

```
CREATE TABLE phone_nums  
(user_id INT REFERENCES  
users, phone_num CHAR(20))
```

Logging DDL and DML

Final Change Script

```
CREATE TABLE phone_nums  
(user_id INT REFERENCES  
users, phone_num CHAR(20))
```

```
INSERT INTO phone_nums  
SELECT user_id, phone_num  
FROM users
```

```
ALTER TABLE users DROP  
COLUMN phone_num
```

Migrations

RDBMS independent API to help in schema changes

More readable than full blown SQL statements

For many changes, rollback scripts can automatically be provided

Fallback to native SQL when necessary

Supports both DDL and DML

Doctrine Migrations

Generates PHP files with migration script code comparing the current model classes with the changed schema file

```
./doctrine generate-migrations-diff
```

Migrate the database to the current version

```
./doctrine migrate
```

Automatic up/down handling

```
class MigrationTest extends Doctrine_Migration_Base
{
    public function migrate($direction)
    {
        $this->column($direction, 'table_name', 'column_name', 'string', '255');
    }
}
```

Multiple live versions

Leave current schema unchanged

Create a new schema with all the new tables, columns

Create (materialized) VIEW's to pull in the data from the old schema

Alternatively, migrate all data to the new schema and recreate the old tables via VIEW's

This gets messy quickly and not all operations work on VIEWS

Some tips

Plan ahead to minimize changes ;-)

Explicitly hard code columns, do not use `SELECT * ..`

Store grants statements with object definitions

MySQL only syntax for version dependent new optional features: `/*!50100 PARTITION .. */`

Many RDBMS (Oracle, MySQL ..) do an implicit commit after every DDL, some do not (MS SQL, PostgreSQL ..)

Show list of statements before execution via some magic frontend for additional comfort

References

SOLYog

Toad

MySQL Workbench

AdeptSQL

Sybase Powerdesigner

ERWin

Visio

References

[PostgreSQL logging](#)

[MySQL logging](#)

[MySQL Proxy](#)

[Oracle logging](#)

[DB2 logging](#)

References

[ADODB xml-schema](#)

[ezc/DatabaseSchema](#)

[PEAR::MDB2 Schema](#)

[DBDeploy](#)

[LiquiBase](#)

[SCM for databases?](#)

[Doctrine Migrations](#)

Thank you!
Questions? Comments?

Lukas Smith - lukas@liip.ch
CodeWorks 2009 - PHP on the ROAD