Managing Metadata Changes

Introduction

About myself

Been using Firebird since
first beta versions came out.

Manager of FlameRobin project and
author of FBExport and FBCopy tools

Metadata Management Problem

- Development activities:
  - Changing Database Schema
  - Updating the “helper” data in database
    - “support” data not entered by user, but needed
  - Changing the Application Code
    - easy to manage with VCS
    - distribute executables or patches to customers

Points of change

- Development databases
  - sandboxes where developers play
  - developers in the field using notebooks
- Master development database
  - only the “final” changes go into
- Releasing new versions
  - updating customer's database
  - multiple client sites

Possible solutions

- Compare source and target and run the change script
  - IBExpert, IB DB Comparer, etc.
- Create blank database and pump the data
  - isql -x, gbak -m, DELETE FROM *
- Both neglect changes in “support” data

The Solution

- To track changes: record them
- Replay them on target databases
- Simple textual .sql files (or database table)
- Single vs Multiple statements per file
  - Break changes into atomic units: a single statement
  - Each change is a new file and new database version

Tracking the database version

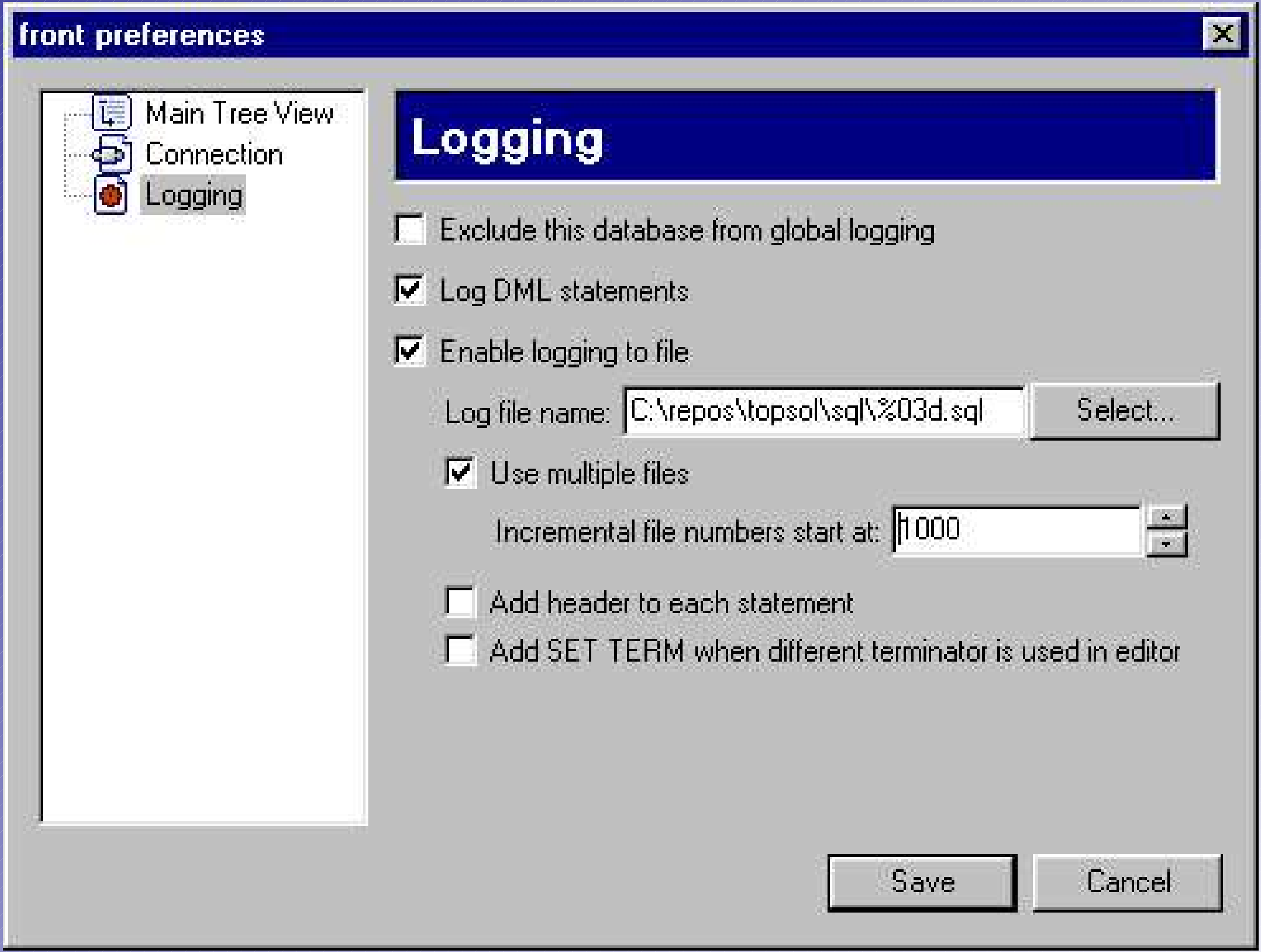
- Keep version number in database
- CREATE TABLE database_version
- Start at any time, give db version one
- For each change:
  - Increment version
  - Save .sql script
- To update target database, run scripts

Logging changes

- Manual copy/paste

OR

- Automatic logging by admin tool


front preferences

- Main Tree View
- Connection
- Logging

Logging

- [ ] Exclude this database from global logging
- [✓] Log DML statements
- [✓] Enable logging to file

Log file name: C:\repos\topsol\sql\%03d.sql [Select...]

- [✓] Use multiple files

Incremental file numbers start at: 1000

- [ ] Add header to each statement
- [ ] Add SET TERM when different terminator is used in editor

[Save] [Cancel]

Set of tools

- Handle the script files: movedb.php
  - Re-number files
  - Fill the gaps (changes that shouldn't propagate)
- Update Databases: updatedb.php
  - Check the current version
  - Change the current version for sandbox db
  - Update the target database

Demonstration

- Flamerobin logging of changes
- updatedb.php
- movedb.php

Customer's database

- Run updatedb manually
- Add updatedb equivalent to your
  - Installer
  - Application
- Deny users if database version is not compatible with application version
- Replication: check structure before start

Tips and tricks

- Commit after each change (FK, etc.)
  - some changes fail only at commit
- Reconnect after each change
  - to aviod famous OBJECT IN USE error
- Name your constraints
  - autogenerated constraint names change after backup/restore

Using version control software

- Version control systems manage files
- Databases changes can be represented by .sql files
- Keep change scripts close to application source code
- Commit changes to application and database together
- Track WHO changed WHAT and WHY
- Conflict control (movedb.php to fix)

Master databases

- Q: To use or not to use?
- A: It isn't absolutely required but helps
- Master database rules:
  - Nobody changes database directly
  - Only committed changes are applied
- Think of master database as copy of customer's database at your office

Going back a version

- Easy with source code
- Database changes are one-way
- Approaches:
  - Write “reverse” statements
  - Store database file in repository
    - only master database should be stored
  - Roll forward from initial version (demo)

Questions

?????