How to Improve Database Connectivity With the Data Tools Platform

John Graham (Sybase Data Tooling)
Brian Payton (IBM Information Management)
Agenda

• DTP Overview
• Creating a Driver Template
• Creating a Connection Profile
• Customizing a Database Definition
• Extending the SQL Query Parser
• Conclusion
DTP Structure

• Top level project at Eclipse
• Contains four subprojects
  – Model Base
  – Connectivity
  – SQL Dev Tools
  – Enablement
Model Base

• Domain models for data centric applications using EMF
• Currently
  – SQL
  – SQL Query
  – SQL XML Query
  – Database Definition
Connectivity

• Create and manage connections to data sources
• Frameworks
  – Driver Management
  – Connection Management
  – ODA
• Data Source Explorer
SQL Dev Tools

• Developer centric SQL tools

• Frameworks
  – Editor
  – Results view
  – SQL Query Parser

• Tools
  – SQL Editor
  – Script history
  – Results
Enablement

• For specialization of DTP to specific data sources
• Examples
  – Specific databases
  – XML data source
• Other data sources?
Support for PostgreSQL 8.x

• Can define generic driver template
• Can connect using JDBC connection profile
• Can use SQL Dev with this connection profile
Demonstration: Connecting to PostgreSQL
Limitations

• Not intuitive for driver and connection profile definitions
• Not all data types correct
• Not all database objects appear in DSE
✓ We will address the first two today
Driver Template

• Specify reasonable defaults for PostgreSQL 8.x
• Reuse generic driver template
• Make the choice obvious in the driver template list
Demonstration: Creating a Driver Template
Connection Profile

• Simple reuse of the generic JDBC connection profile (for now)

• Responsibilities
  – Manage connection on request
  – Provide meta-data about connection
Demonstration:
Creating a Connection Profile
Further (future) specializations

- Specialized class loading
- Specialized content for DSE
- Contribute to SQL Dev extension points
Data Type Problems

• Why is this happening?
• Specialize database definition
  – Modify XMI vendor file
  – Based on generic version
  – Add new names for data types
Demonstration:
Specialized DB Definition
Using the DB Customization

✓ Specialized types added
• (Future) Support all data types in a similar manner
Beyond Connecting

• Now that you have a (customized) connection to your database, what can you do with it?
  – All kinds of things, but here are some important ones:
    • Retrieve data
    • Update data
  – Retrieving and updating data typically requires creating SQL queries
SQL Facts of Life

• Each database type has its own flavor of SQL
  – No database really implements the ISO SQL standard
  – Each has missing features, added non-standard features, and some features present but just different

• This creates a problem for database tools!
  – Need a way to create tools that can handle the differences
DTP SQL Query Parser

• DTP includes a SQL Query parser framework and example implementation
• Parses SQL DML statements:
  – SELECT, INSERT, UPDATE, DELETE
• Based on IBM LALR Parser Generator (LPG)
• Output of parser is instance of SQL Query Model
• Both parser and model are designed to be customized and extended
Example: PostgreSQL

• PostgreSQL has lots of interesting features and differences from standard SQL. Here are two we will use for our example extension:
  – Additional table join types
  – Dollar-quoted string constants
PostgreSQL: Additional table join types

- PostgreSQL supports many table join types, including some unusual ones:
  - CROSS JOIN
  - NATURAL JOIN
- These are in ISO SQL but are not in many other DB’s, and are not included in the base SQL Query model and parser
- Will illustrate how to extend the SQL Query model and parser
PostgreSQL: Dollar-quoted string literals

• PostgreSQL has an unusual alternative syntax for specifying string literals in SQL
  – $$string literal here$$
  – Can include any characters inside the literal (avoids the need to double single quotes)

• Typically used to “quote” SQL code in procedure and function definitions

• Will illustrate how to extend the SQL lexer
Task Overview: Extending the Model

1. Create plugin for model extension
2. Define the model extension
3. Generate model code (EMF)
4. Extend the source formatter
Demonstration: Extending the SQL Query Model
Task Overview: Extending the Parser

1. Create plugin for parser extension
2. Copy and modify LPG templates
3. Define LPG command runner
4. Add and override parser rules
5. Generate parser code
Demonstration: Extending the SQL Query Parser
Task Overview: Extending the Lexer

• In LPG, the lexer is a specialized parser, so steps are similar to the parser steps
  1. Copy LPG lexer template
  2. Modify grammar rules in lexer
  3. Generate lexer code
Demonstration: Extending the SQL Lexer
Putting it all together

• Now have a SQL parser and model that work together. Can be used for:
  – Query analysis
  – Query building
  – Query syntax checking