The Preference SQL JDBC Driver

Markus Endres
endres@informatik.uni-augsburg.de

May 21, 2012

This is a short introduction how to use the Preference SQL JDBC Driver to evaluate preference queries on your own database.

1 Preference SQL

Preference SQL [HK05] extends the `SELECT` statement of SQL by an optional `PREFERRING` clause. This `PREFERRING` clause selects all interesting tuples, i.e., tuples that are not dominated by other tuples. Preference SQL currently supports most of the SQL92 standard as well as all base preferences together with the constructors for Pareto, Prioritization and Rank preferences as listed in [Kie02, Kie05].

A Preference SQL query block has the following schematic design:

```
SELECT       ...  <selection>
FROM         ...  <table_reference>
WHERE        ...  <hard_conditions>
 PREFERRING  ...  <soft_conditions>
 GROUPING    ...  <attribute_list>
 BUT ONLY    ...  <but_only_condition>
 TOP         ...  <number>
GROUP BY     ...  <attribute_list>
HAVING       ...  <hard_conditions>
ORDER BY     ...  <attribute_list>
LIMIT        ...  <number>
```

Table 1: Preference SQL query block.

A preference is evaluated in the `PREFERRING` clause on the result of the hard constraint stated in the `WHERE` clause. Empty result sets can only occur in cases where all tuples have been filtered out by the hard conditions.

The syntax of the preference extensions is straightforward, cp. Table 2.

Preference constructor	Preference SQL expression
$BETWEEN_d(A, low, up)$	`A BETWEEN low AND up, d REGULAR`
$AROUND_d(A, z)$	`A AROUND z, d REGULAR`
$HIGHEST_d(A)$	`A HIGHEST sup, d REGULAR`
$LOWEST_d(A)$	`A LOWEST inf, d REGULAR`
$LAYERED_m(A, L_1, \ldots, L_{m+1})$	`A LAYERED (`L_1`, ..., `L_{m+1}`) REGULAR`
$POS/POS(A, S_1, S_2)$	`A IN` S_1 `ELSE` S_2 `REGULAR`
$POS(A, S)$	`A IN` S `REGULAR`
$POS/NEG(A, S_1, S_2)$	`A IN` S_1 `NOT IN` S_2 `REGULAR`
$NEG(A, S)$	`A NOT IN` S `REGULAR`
$ANTICHAIN(A)$	`A ANTICHAIN REGULAR`
$PARETO(P_1, \ldots, P_m)$	P_1 `AND ...` `AND` P_m
$PRIORITIZATION(P_1, \ldots, P_m)$	P_1 `PRIOR TO ... PRIOR TO` P_m
$B^{\leftrightarrow}\&P$	P `GROUPING` B

Table 2: Preference SQL Syntax

Note that base preferences with trivial SV-semantics can be expressed by omitting the keyword `REGULAR` at the end of their definition, cp. [Kie05]. The sets needed for a LAYERED$_m$ preference and its sub-constructors (POS, NEG, etc.) are entered as comma-separated values in brackets: `('a', 'b', 'c', ..., 'z')`.

More information on Preference SQL can be found in [Kie02, Kie05, HK05, KEW11] and in the www:

- Preference SQL in general:

  `http://ursaminor.informatik.uni-augsburg.de/trac/wiki/PSQL`

- Preference SQL syntax with many examples:

  `http://ursaminor.informatik.uni-augsburg.de/trac/wiki/Preference%20SQL`

2 The Preference SQL JDBC Driver

JDBC (Java Database Connectivity) is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases.

Preference SQL is implemented in Java 1.6 and works on the top of a database. Figure 1 shows the architecture of our Preference SQL JDBC driver and the communication between client and server.

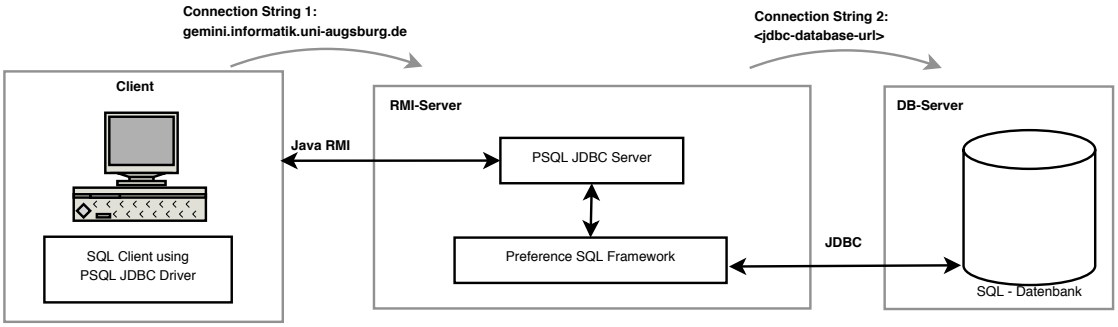

Figure 1: Preference SQL JDBC Driver – Server architecture.

`Connection String 1` defines the RMI Server, i.e. the counterpart of our Preference SQL JDBC Client `psqlClient.jar`. The Preference SQL JDBC Server communicates with our Preference SQL Framework for preference evaluation. Furthermore, `Connection String 2` defines the underlying database URL. This database can be at any place in the world. Keep this schema in mind when specifying the URL for the JDBC driver later.

3 Using Preference SQL in a SQL Client

This section describes how to use the Preference SQL JDBC driver in combination with any JDBC SQL Client. Note that you need Sun Java 1.6 and the SQL Client must support Java 1.6.

3.1 General Approach

1) Register the Preference SQL JDBC Driver (psqlClient.jar) in your SQL Client.

2) Create a database connection with the following connection parameters:

- URL:
  `jdbc:psql://gemini.informatik.uni-augsburg.de@`
  `<jdbc-database-driver>::<jdbc-database-url>`

- Driver class:
  `psql.connector.client.PSQLDriver`

3.2 DbVisualizer and Preference SQL JDBC

We will demonstrate the registration of the driver and the connection to a database using the DbVisualizer SQL Client from `http://www.dbvis.com/`. You can download the *free version* of DbVisualizer at

`http://www.dbvis.com/products/dbvis/download/`

1) Install and run DbVisualizer. When the `Connection Wizard` appears, click `Cancel`.

2) Register the Preference SQL JDBC Driver (psqlClient.jar).

   a) In the menu of DbVisualizer click `Tools -> Driver Manager...`

   b) In the menu of DbVisualizer click `Driver -> Create Driver`. The window for JDBC driver registration appears.

   c) In the `User Specified` tab of the `Driver File Paths` section navigate to the Preference SQL JDBC driver `psqlClient.jar` by clicking the `Open` icon.

   d) Fill the remaining fields with the following data, also cp. Figure 2. Close the window.

Name:	**Preference SQL** (or whatever you want)
URL Format:	**jdbc:psql://gemini.informatik.uni-augsburg.de@ <jdbc-database-driver>::<jdbc-database-url>**
Driver Class:	**psql.connector.client.PSQLDriver** (automatically set after the import of psqlClient.jar)

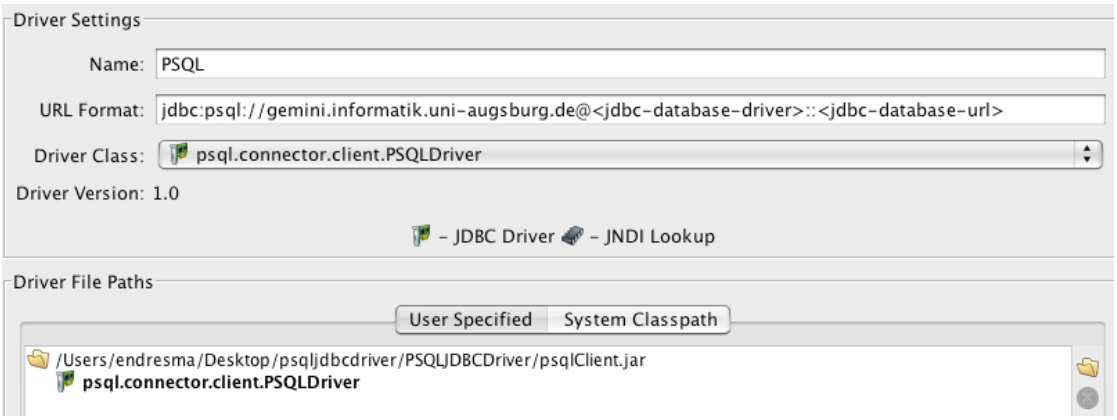

Figure 2: Preference SQL JDBC Driver registration in DbVisualizer.

3) Create a database connection.

   a) Right-click `Connections -> Create Database Connection`. Don't use the wizard. The new `Database Connection` window appears.

   b) Fill the fields as depicted in Figure 3 and connect to the database.

Alias:	**Preference SQL** (or whatever you want)
Database Type :	**Generic**
Driver (JDBC):	**Preference SQL**
Database URL:	e.g. **jdbc:psql://gemini.informatik.uni-augsburg.de@org.postgresql.Driver::jdbc:postgresql://localhost:5432/psqldb** (if you want to connect to our sample database)
Userid:	**psqldbuser** (if you want to connect to our sample database)
Password:	**psqldbpwd** (if you want to connect to our sample database)

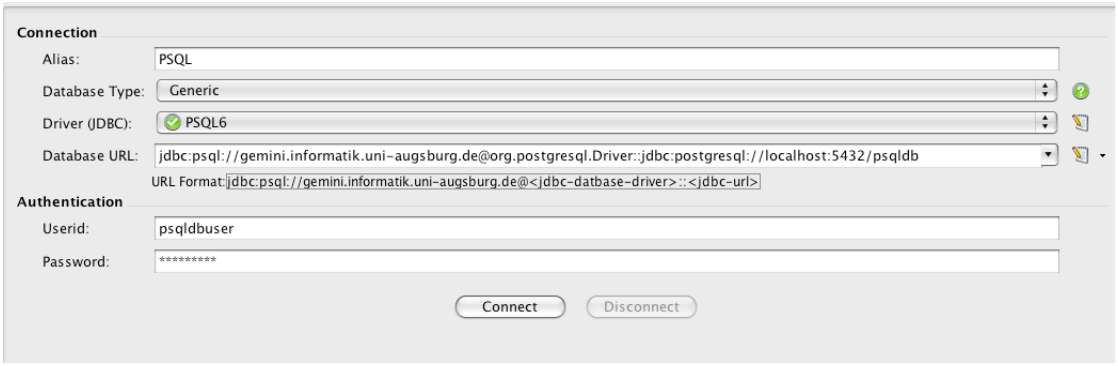

Figure 3: Create a database connection in DbVisualizer.

3.3 Sample Database

Our sample database holds a relation `Cars`, which contains cars with different attributes. Figure 4 shows the schema. Just query `SELECT * FROM Car` to get a more detailed overview of the table's content. Note that the table is read-only.

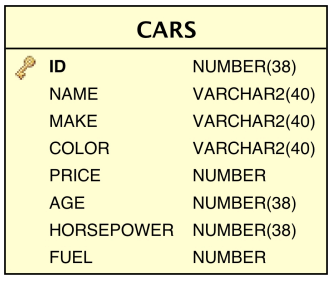

Figure 4: Sample database relation.

As an example one may prefer `VW` or `Audi`[1]. The `Horsepower` should be between 50 and 80, but a difference of 10 does not matter. Both are more important than `Age` around 5 years. A `Price` less than 5.000 Euro is a hard constraint. In Preference SQL we write:

```
SELECT * FROM Car
WHERE PRICE < 5000
PREFERRING MAKE IN ('vw','audi') REGULAR AND
           Horsepower BETWEEN 50 AND 80, 10 REGULAR PRIOR TO
           Age AROUND 5;
```

Type it up[2] in the DbVisualizer and be surprised by the result.

[1] Note that our preference constructors are case-sensitive and our relation only contains lower-case.
[2] Do not copy & paste due to the wrong selection of quotes in a PDF document.

4 Future Work

Although Preference SQL supports all kind of preference queries and the most part of SQL92, there are still some open issues. For example, (correlated) subqueries or `create user` do not work until now. Furthermore, only Oracle, PostgreSQL and My-SQL are supported. We work on further databases, e.g. Microsoft SQL Server or SQLite.

Of course, Preference SQL is not bug free. Therefore, if you determine a bug, or you think there is a bug, contact

bugs@PreferenceSQL.com

References

[HK05] B. Hafenrichter and W. Kießling, *Optimization of Relational Preference Queries*, ADC '05: Proceedings of the 16th Australasian database conference (Darlinghurst, Australia), Australian Computer Society, Inc., 2005, pp. 175–184.

[KEW11] W. Kießling, M. Endres, and F. Wenzel, *The Preference SQL System - An Overview*, Bulletin of the Technical Commitee on Data Engineering, IEEE Computer Society **34** (2011), no. 2, 11–18.

[Kie02] W. Kießling, *Foundations of Preferences in Database Systems*, VLDB '02: Proceedings of the 28th International Conference on Very Large Data Bases (Hong Kong, China), VLDB Endowment, 2002, pp. 311–322.

[Kie05] ———, *Preference Queries with SV-Semantics*, COMAD '05: Advances in Data Management 2005, Proceedings of the 11th International Conference on Management of Data (Goa, India) (Jayant R. Haritsa and T. M. Vijayaraman, eds.), Computer Society of India, 2005, pp. 15–26.