Lecture 15: Extended Database Logic

15-445/645 Database Systems (Fall 2017)
Carnegie Mellon University
Prof. Andy Pavlo

Embedded Database Logic

1. Until now, we have assumed that all of the logic for an application is located in the application itself.

2. We can think of it as the application has a "conversation" with the DBMS to store or retrieve data.

3. It may be possible to move complex application logic into the DBMS to avoid multiple network round-trips. Doing this can improve efficiency and reusability in the application.

User-Defined Functions

1. A **user defined function** is a function written by the application developer that extends the system's functionality beyond its built-in operations

2. Process:

   (a) It takes in input arguments (scalars).

   (b) Performs computation.

   (c) Returns a result (scalar, table).

3. Return types

   (a) **Scalar Functions:** Return a single data value.

   (b) **Table Functions:** Return a single result table.

4. Computation

   (a) **SQL Functions:** A sql based UDF contains a list of SQL statements that the DBMS executes in order when the UDF is invoked. The UDF returns whatever the result is of the last query.

   (b) **External Programming Language:** Some DBMSs support writing UDFs in languages other than SQL.

Stored Procedures

1. A **stored procedure** is a self-contained function that performs more complex logic inside of the DBMS.

2. Stored procedures can have many input/output parameters and can modify the database table/structures.

3. UDFs are usually meant to be read-only, while stored procedures are expected to not be read-only.

Database Triggers

1. A **trigger** instructs the DBMS to invoke a UDF when some event occurs in the database

2. The developer has to define:

   (a) **Event Type:** Type of modification (INSERT, UPDATE, DELETE, ALTER, etc).

   (b) **Event Scope:** Scope of the modification (TABLE, DATABASE, VIEW, etc).

   (c) **Timing:** When the trigger should be activated based on statement (before, after, instead of, etc).

3. Some examples of trigger usage are constraint checking or auditing

Change Notifications

1. A **change notification** is like a trigger except that the DBMS sends a message to an external entity that something notable has happened in the database.

2. Can be chained with a trigger to pass along whenever a change occurs.

3. SQL standard uses `LISTEN` and `NOTIFY` commands.

Non-Native types

1. What if we want to store data that doesn't match any of the built in types?

2. One potential solution is to store the complex form in a serialized form, but this has problems:

   (a) How do you edit a sub-element?

   (b) How can the optimizer estimate selectivity on predicates that access serialized data?

   (c) How do you execute aggregates and other functions on serialized data?

3. Instead we can use **user-defined type** (UDT). This is a special data type that is defined by the application developer that the DBMS can be stored natively.

4. Each DBMS exposes a different API that allows you to create a UDT.

Views

1. **Views** are "virtual" tables containing the output from a `SELECT` query.
2. Mechanism for hiding data from view of certain users.
3. Under the hood, queries on views are converted into a single query using the original query that generated view.
4. Can be used to simplify complex queries that are executed often, but won't result in a speedup because the original query is still run.
5. Unlike `SELECT...INTO`, a view does not allocate a table to store the result of the view.
6. The SQL-92 standard dictates that views can only contain a single based table, and can't contain aggregations, distinctions, union, or grouping.
7. A **Materialized View** maintains the result of a view internally that is automatically updated when the underlying tables change.

Conclusion

1. Moving application logic into the DBMS has lots of benefits:
   (a) Better efficiency
   (b) Reusable across applications
2. However, it has its problems:
   (a) Not portable
   (b) DBAs don't like constant change
   (c) Potentially need to maintain different versions