7CCSMDDW

Triggers

1 Using Triggers to Simulate Function-Based Indexes

Some types of information are more easily analysed using not the original values, but an expression computed from them. For example, if data values lie along an exponential curve, applying a logarithmic transform to them yields a more linear scale. Queries against a table that stores exponential values might therefore typically use expressions that refer to the log values:

```
SELECT * FROM expdata WHERE LOG10(value) < 2;
```

A disadvantage of such expressions is that referring to the value column within a function call prevents the optimizer from using any index on it. A DBMS must retrieve the values to apply the function to them, and the function values are not indexed. The result is diminished performance.
Usually, DBMSs do not support this capability, but there is a workaround:

- Define a secondary column to store the function values and index that column.
- Define triggers that keep the secondary column up to date when the original column is initialized or modified.
- Refer directly to the secondary column in queries so that the optimizer can use the index on it for efficient lookups.

In the following example you will apply this technique, using a table designed to store values that lie along an exponential curve:

1. Create a table to store decimal values as follows:

```
CREATE TABLE expdata(
 id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
 value FLOAT, # original values
 INDEX (value) # index on original values
);
```

2. Insert some data:

```
INSERT INTO expdata (value) VALUES (.01),(.1),(1),(10),(100);
```

3. Add a new column (named `log10_value`) for the transformed values. (TIP: Use and ALTER TABLE STATEMENT)
4. Add an index on these new column (named `log10_value`)

```
ALTER TABLE expdata ADD INDEX transformedValues (log10_value);
```

5. Update the table with the transformed values

```
UPDATE expdata SET log10_value=LOG10(value);
```

6. Create an `INSERT` trigger to initialize the `log10_value` value from value for new rows, and an `UPDATE` trigger to keep `log10_value` up to date when value changes.
7. To test the implementation, insert and modify some data and check the result of each operation

This same technique can not only be applied to numeric values.

2 Using Triggers to Preprocess or Reject Data

There are situations in which you want to check/preprocess the data entered into a table. You can use triggers to perform several types of input checks:

- Reject bad data by raising a signal.
- Preprocess values and modify them, if you won't want to reject them outright. For example, map out-of-range values to be in range or sanitize values to put them in canonical form, if you permit entry of close variants.

In this case we are going to implement a trigger to reject bad and preprocess bad data:

- Create a table as follows:

```
CREATE TABLE contact_info(
 id INT NOT NULL AUTO_INCREMENT,
 name VARCHAR(30), # state of residence
 email VARCHAR(50), # email address
 url VARCHAR(255), # web address
 PRIMARY KEY (id)
);
```

- For entry of new rows, you want to enforce constraints or perform preprocessing as follows:
  - Email address values must contain an @ character to be valid.
  - For website URLs, strip any leading 'http://' to save space
- To handle these requirements, create triggers. TIPS:
  - The function `LOCATE` can be used to check if the email is correct.
  - The function `TRIM` can be used to remove the 'http://' prefix of URLs.
- Test the trigger by executing some INSERT statements to verify that it accepts valid values, rejects bad ones, and trims URLs:

```
mysql> INSERT INTO contact_info (name,email,url)
-> VALUES('Jen','jen@example.com','http://www.example.com');
```

```
mysql> INSERT INTO contact_info (name,email,url)
-> VALUES('Jen','jen','http://www.example.com');
ERROR 45000: invalid email address
```

```
mysql> SELECT * FROM contact_info;
+----+------+-------+-----------------+-----------------+
| id | name | state | email           | url             |
+----+------+-------+-----------------+-----------------+
| 1  | Jen  | NY    | jen@example.com | www.example.com |
+----+------+-------+-----------------+-----------------+
```

3 Personal Study

1. Read the information about how to define triggers in MySQL here:

   `http://dev.mysql.com/doc/refman/5.7/en/trigger-syntax.html`