ITS SEMANGAT BARU

ITS Institut Teknologi Sepuluh Nopember

Triggers

Triggers

- Triggers are procedural statements executed automatically when a database is modified
    - Usually specified in procedural SQL language, but other languages are frequently supported
- Example: an audit log for bank accounts
    - Every time a balance is changed, a trigger can update an "audit log" table, storing details of the change
        - e.g. old value, new value, who changed the balance, and why
- Why not have applications update the log directly?
    - Could easily forget to update audit log for some updates!
    - Or, a malicious developer might leave a back-door in an application, allowing them to perform unaudited operations

Triggers (2)

- If the database handles audit-log updates automatically and independently:
  - Application code doesn’t become more complex by introducing audit functionality
  - Audit log will be a more trustworthy record of modifications to bank account records
- Triggers are used for many other purposes, such as:
  - Preventing invalid changes to table data
  - Automatically updating timestamp values, derived attributes, etc.
  - Executing business rules when data changes in specific ways
    - e.g. place an order for more parts when current inventory dips below a specific value
  - Replicating changes to another table, or even another database

Trigger Mechanism

- DB trigger mechanism must keep track of two things:
- When is the trigger actually executed?
  - The **e**vent that causes the trigger to be considered
  - The **c**ondition that must be satisfied before the trigger will execute
    - (Not every database requires a condition on triggers…)
- What does the trigger do when it's executed?
  - The **a**ctions performed when the trigger executes
- Called the event-condition-action model for triggers

When Triggers Execute

- Databases usually support triggering on inserts, updates, and deletes
- Can’t trigger on selects
  - Implication: Can’t use triggers to audit or prevent read- accesses to a database (bummer)
- Commercial databases also support triggering on many other operations
  - Data-definition operations (create/alter/drop table, etc.)
  - Login/logout of specific users
  - Database startup, shutdown, errors, etc.

When Triggers Execute

• insert, update, access	structure operation
• the message display is sent to an object of type widget	behavior invocation
• abort, commit, begin-transaction	transaction
• an attempt to access some data without appropriate authorization	exception
• the first day of every month	clock
• the temperature reading goes above 30 degrees	external

When Triggers Execute

- Can typically execute the trigger before or after the triggering DML event
  - Usually, DDL/user/database triggering events only run the trigger after the event (pretty obvious)
  - "Before" triggers can abort the DML operation, if necessary
- Some DBs also support "instead of" triggers
  - Execute trigger instead of performing the triggering operation
- Triggers are row-level triggers or statement-level triggers
  - A row-level trigger is executed for every single row that is modified by the statement
    - (…as long as the row satisfies the trigger condition, if specified…)
  - A statement-level trigger is executed once for the entire statement

Trigger Data

- Row-level triggers can access the old and new version of the row data, when available:
  - Insert triggers only get the new row data
  - Update triggers get both the old and new row data
  - Delete triggers only get the old row data
- Triggers can also access and modify other tables
  - e.g. to look up or record values during execution

Trigger Syntax

- SQL:1999 specifies a syntax for triggers
- Wide variation from vendor to vendor
  - Oracle and DB2 are similar to SQL99, but not identical
    - (triggers always seem to involve vendor-specific features)
  - SQLServer, Postgres, MySQL all have different features
  - Constraints on what triggers can do also vary widely from vendor to vendor
- Will focus on MySQL trigger syntax, functionality

Syntax

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE|AFTER} {INSERT [OR] | UPDATE [OR] | DELETE}
[OF column name] ON table_name
[FOR EACH ROW]
[WHEN condition]
DECLARE
Declaration statements
BEGIN
Executable statements
EXCEPTION
Exception-handling statements
END trigger_name;
```

```
CREATE TRIGGER init_count BEFORE INSERT ON Students
DECLARE
        count INTEGER;
BEGIN
        count:=0;
END
```

event

action

```
CREATE TRIGGER incr_count AFTER INSERT ON Students
WHEN (new.age < 18)
FOR EACH ROW
BEGIN
        count:=count  + 1;
END
```

condition

```
CREATE TRIGGER incr_count AFTER INSERT ON Students
WHEN (new.age < 18)
FOR EACH ROW
BEGIN
        count:=count  + 1;
END
```

Row-Trigger

```
CREATE TRIGGER init_count BEFORE INSERT ON Students
DECLARE
        count INTEGER;
FOR EACH STATEMENT
BEGIN
        count:=0;
END
```

Statement-Trigger

Trigger Example: Bank Overdrafts

- Want to handle overdrafts on bank accounts
- If an update causes a balance to go negative:
  - Create a new loan with same ID as the account number
  - Set the loan balance to the negative account balance
    - (…the account balance went negative…)
  - Need to update borrower table as well!
- Needs to be a row-level trigger, executed before or after updates to the account table
  - If database supports trigger conditions, only trigger on updates when account balance < 0

SQL99/Oracle Trigger Syntax

- Book uses SQL:1999 syntax, similar to Oracle/DB2

```
CREATE TRIGGER trg_overdraft AFTER UPDATE ON account
REFERENCING NEW ROW AS nrow
FOR EACH ROW WHEN nrow.balance < 0
BEGIN ATOMIC
   INSERT INTO loan VALUES (nrow.account_number, nrow.branch_name,
                                   -nrow.balance);
   INSERT INTO borrower
         (SELECT customer_name, account_number
          FROM depositor AS d
          WHERE nrow.account_number = d.account_number);
   UPDATE account AS a SET balance = 0
         WHERE a.account_number = nrow.account_number;
END
```

MySQL Trigger Syntax

- MySQL has more limited trigger capabilities
  - Trigger execution is only governed by events, not conditions
    - Workaround: Enforce the condition within the trigger body
  - Old and new rows have fixed names: OLD, NEW
- Change the overdraft example slightly:
  - Also apply an overdraft fee! “Kick ‘em while they’re down!”
- What if the account is already overdrawn?
  - Loan table will already have a record for overdrawn account…
  - Borrower table will already have a record for the loan, too!
  - Previous version of trigger would cause duplicate key error!

MySQL `INSERT` Enhancements

- MySQL has several enhancement to the `INSERT` command
  - (Most databases provide similar capabilities)
- Try to insert a row, but if key attributes are same as another row, simply don't perform the insert:

```
INSERT IGNORE INTO tbl ...;
```

- Try to insert a row, but if key attributes are same as another row, update the existing row:

```
INSERT INTO tbl ... ON DUPLICATE KEY
    UPDATE attr1 = value1, ...;
```

- Try to insert a row, but if key attributes are same as another row, replace the old row with the new row
  - If key is not same as another row, perform a normal `INSERT`

```
REPLACE INTO tbl ...;
```

MySQL Trigger Syntax (2)

```
CREATE TRIGGER trg_overdraft BEFORE UPDATE ON account FOR EACH ROW
BEGIN
   DECLARE overdraft_fee NUMERIC(12, 2) DEFAULT 30;
   DECLARE overdraft_amt NUMERIC(12, 2);
   -- If an overdraft occurred then handle by creating/updating a loan.
   IF NEW.balance < 0 THEN
      -- Remember that NEW.balance is negative.
      SET overdraft_amt = overdraft_fee - NEW.balance;
      INSERT INTO loan (loan_number, branch_name, amount)
          VALUES (NEW.account_number, NEW.branch_name, overdraft_amt)
      ON DUPLICATE KEY UPDATE amount = amount + overdraft_amt;
      INSERT IGNORE INTO borrower (customer_name, loan_number)
          SELECT customer_name, account_number FROM depositor
          WHERE depositor.account_number = NEW.account_number;
      SET NEW.balance = 0;
   END IF;
END;
```

Trigger Pitfalls

- Triggers may or may not execute when you expect…
  - e.g. MySQL insert-triggers fire when data is bulk-loaded into the DB from a backup file
    - Databases usually allow you to temporarily disable triggers
  - e.g. truncating a table usually does not fire delete-triggers
- If a trigger for a commonly performed task runs slowly, it will kill DB performance
- If a trigger has a bug in it, it may abort changes to tables at unexpected times
  - The actual cause of the issue may be difficult to discern
- Triggers can write to other tables, which may also have triggers on them…
  - Not hard to create an infinite chain of triggering events

Alternatives to Triggers

- Triggers can be used to implement many complex tasks
- Example: Can implement referential integrity with triggers!
  - On all inserts and updates to referencing table, ensure that foreign-key column value appears in referenced table
    - If not, abort the operation!
  - On all updates and deletes to referenced table, ensure that value doesn't appear in referencing table
    - If it does, can abort the operation, or cascade changes to the referencing relation, etc.
- This is definitely slower than the standard mechanism

Alternatives to Triggers (2)

- Can you use stored procedures instead?
  - Stored procedures usually have fewer limitations than triggers
    - Stored procs can take more detailed arguments, return values to indicate success/failure, have out-params, etc.
    - Can perform more sophisticated transaction processing
  - Trigger support is also very vendor-specific, so either implementation choice will have this limitation
- Typically, triggers are used in very limited ways
  - Update "row version" or "last modified timestamp" values in modified rows
  - Simple operations that don't require a great deal of logic
  - Database replication (sometimes)

Triggers and Summary Tables

- Triggers are sometimes used to compute summary results when detail records are changed
- Example: a table of branch summary values
  - e.g. (branch_name, total_balances, total_loans)
- Motivation:
  - If these values are used frequently in queries, want to avoid overhead of recomputing them all the time
- Idea: update this summary table with triggers
  - Anytime changes are made to account or loan, update the summary table based on the changes