PAYAP UNIVERSITY

Advanced SQL

Vaidė Narváez

Computer Information Systems

January 6th, 2011

Views and indices

PAYAP UNIVERSITY

Views

An expression that describes a table without creating it. A relation that is defined by a query over other relations.

Two kinds:

1. *Virtual:* not stored in the database; just a query for constructing the relation.
2. *Materialized:* actually constructed and stored.

PAYAP UNIVERSITY

Views: example

```
CREATE VIEW name AS query;
```

The view `CanDrink` is the set of drinker-beer pairs such that the drinker frequents at least one bar that serves the beer.

```
CREATE VIEW CanDrink AS
SELECT drinker, beer
FROM Frequents, Sells
WHERE Frequents.bar = Sells.bar;
```

PAYAP UNIVERSITY

Accessing a view

Querying Views: Treat the view as if it was a materialized relation

```
SELECT beer FROM CanDrink WHERE drinker = 'Sally'
```

Indices

An index is a data structure used to speed access to tuples of a relation, given values of one or more attributes

```
CREATE INDEX BeerInd ON Beers(manufacturer);
```

```
CREATE INDEX SellInd ON Sells(bar, beer);
```

PAYAP UNIVERSITY

Use of Indices

- *Pros*: The existence of an index on an attribute may speed up the execution of those queries which uses that attribute
- *Con*: An index slows down all modifications(insertions, deletions, and updates) on its relation because the index must be modified too.

Transactions in SQL

PAYAP UNIVERSITY

Transactions

A database transaction comprises a unit of work performed within a database management system against a database.

“all-or-nothing” proposition:
each work-unit performed in a database must either complete in its entirety or have no effect whatsoever.

PAYAP UNIVERSITY

Purpose

1. To provide consistency after recovery and system failures

2. To provide isolation between programs accessing a database concurrently

PAYAP UNIVERSITY

The ACID Properties

- **Atomicity:** A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
- **Consistency:** A transaction is consistency preserving if its complete execution take the database from one consistent state to another.
- **Isolation:** A transaction should appear as though it is being executed in isolation from other transactions. That is, the execution of a transaction should not be interfered with by any other transaction execution concurrently.
- **Durability:** The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure.

PAYAP UNIVERSITY

Concurrency control

1. Pessimistic concurrency control (Locking)
2. Optimistic concurrency control (Multiversioning)

PAYAP UNIVERSITY

Concurrency control

1. Pessimistic concurrency control (Locking)
2. Optimistic concurrency control (Multiversioning)

	Read	Write
Read	S	X
Write	X	X

Compatibility of locks

PAYAP UNIVERSITY

Commit/Rollback

- The SQL statement COMMIT causes a transaction to complete.
  - Its database modifications are now permanent in the database.
- The SQL statement ROLLBACK also causes the transaction to end, but by aborting.
  - No effects on the database.

PAYAP UNIVERSITY

Isolation Levels

SQL defines four *isolation levels*: choices about what interactions are allowed by transactions that execute concurrently.

SET TRANSACTION ISOLATION LEVEL X
where X is

- SERIALIZABLE
- REPEATABLE READ
- READ COMMITTED
- READ UNCOMMITTED

Only serializable level achieves complete ACID transactions.

PAYAP UNIVERSITY

Transactions in MySQL

Use a transactional storage engine: *InnoDB*

SET autocommit = 0

START TRANSACTION

...

sql statements

...

COMMIT or ROLLBACK

Triggers

PAYAP UNIVERSITY

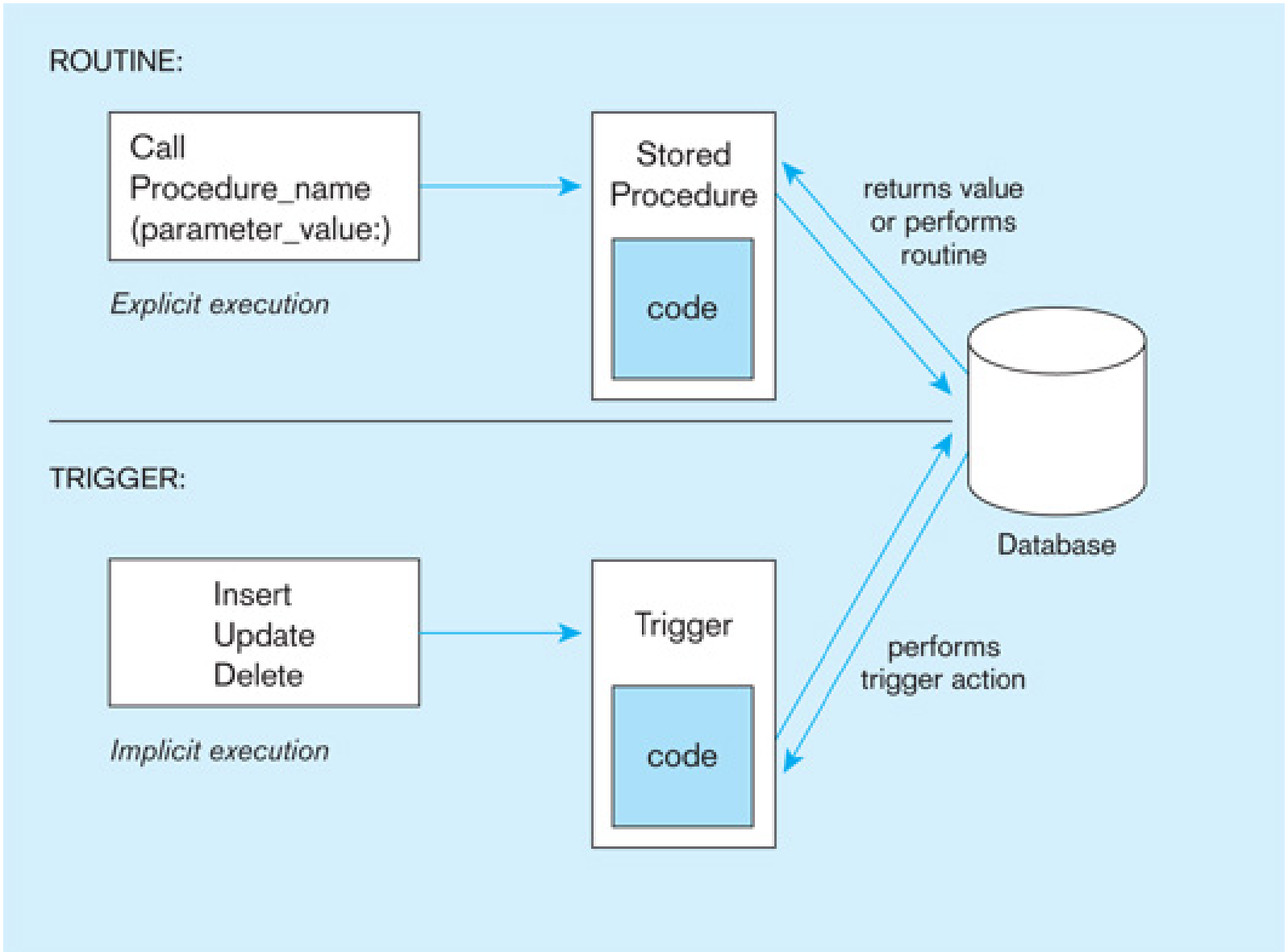
Triggers

Trigger is a block of procedural code which is stored in the database and runs automatically on triggering events.

Routines are stored blocks of code that need to be called to operate

PAYAP UNIVERSITY

Triggers vs. routines



PAYAP UNIVERSITY

Event-Condition-Action

- **Event** : typically a type of database modification, e.g.,*"insert on Sells"*
- **Condition**: Any SQL boolean-valued expression.
- **Action**: Any SQL statements.

PAYAP UNIVERSITY

Triggers: examples

```
CREATE TRIGGER trigger_name
        {BEFORE | AFTER | INSTEAD OF} {INSERT | DELETE | UPDATE} ON
        table_name
        [FOR EACH {ROW | STATEMENT}] [WHEN (search condition)]
        <triggered SQL statement here>;
```

```
CREATE TRIGGER BeerTrig
AFTER INSERT ON Sells REFERENCING NEW ROW AS NewTuple
FOR EACH ROW
WHEN (NewTuple.beer NOT IN (SELECT name FROM Beers))
INSERT INTO Beers(name) VALUES(NewTuple.beer);
```

PAYAP UNIVERSITY

Triggers:examples

```
CREATE TRIGGER PriceTrig AFTER UPDATE OF price ON Sells
REFERENCING OLD ROW AS ooo NEW ROW AS nnn
FOR EACH ROW
WHEN(nnn.price > ooo.price + 1.00)
INSERT INTO RipoffBars
VALUES(nnn.bar);
```