Integrity and security

Jan Chomicki

University at Buffalo
Outline

1 Integrity

2 Security
Transactions

Transaction Execution of a user program in a DBMS.

Transaction properties

- **Atomicity**: all-or-nothing execution
- **Consistency**: database consistency is preserved
- **Isolation**: concurrently executing transactions have no effect on one another
- **Durability**: results survive failures.

Transaction outcome

- **COMMIT**: success, effects made permanent
- **ROLLBACK**: failure, effects removed.

Jan Chomicki (University at Buffalo)

Integrity and security
Transactions

Transaction

Execution of a user program in a DBMS.
Transactions

Transaction

Execution of a user program in a DBMS.

Transaction properties

- **Atomicity**: all-or-nothing execution
- **Consistency**: database consistency is preserved
- **Isolation**: concurrently executing transactions have no effect on one another
- **Durability**: results survive failures.
Transactions

Transaction

Execution of a user program in a DBMS.

Transaction properties

- Atomicity: all-or-nothing execution
- Consistency: database consistency is preserved
- Isolation: concurrently executing transactions have no effect on one another
- Durability: results survive failures.

Transaction outcome

- COMMIT: success, effects made permanent
- ROLLBACK: failure, effects removed.
Integrity constraints and triggers

Integrity constraints
Logical conditions that database instances must satisfy maintained by the DBMS.

Triggers
Rules for enforcing integrity executed by the DBMS. Flexible reaction to integrity violations.
Integrity constraints and triggers

Maintaining the **logical integrity** of the database.
Integrity constraints and triggers

Maintaining the **logical integrity** of the database.

**Integrity constraints**
- logical conditions that database instances must satisfy
- maintained by the DBMS
Integrity constraints and triggers

Maintaining the **logical integrity** of the database.

### Integrity constraints
- logical conditions that database instances must satisfy
- maintained by the DBMS

### Triggers
- rules for enforcing integrity
- executed by the DBMS
- flexible reaction to integrity violations
Integrity constraints
Integrity constraints

Column and table constraints

- CHECK constraints
- key constraints
- foreign keys
- associated with a table but can refer to multiple tables
- violated only by insertion/update to the same table.
Integrity constraints

Column and table constraints

- CHECK constraints
- key constraints
- foreign keys
- associated with a table but can refer to multiple tables
- violated only by insertion/update to the same table.

Assertions

- maintained across multiple tables
Integrity constraints

Column and table constraints
- CHECK constraints
- key constraints
- foreign keys
- associated with a table but can refer to multiple tables
- violated only by insertion/update to the same table.

Assertions
- maintained across multiple tables

Checking mode
- immediate: after an operation
- deferred: after transaction ends
Referential integrity actions (SQL:1999)

- Modifications violating a foreign key constraint
  - Referenced table:
  - Referencing table:
  - Actions:
    - ON UPDATE
    - ON DELETE
    - SET DEFAULT
    - SET NULL
    - CASCADE
    - NO ACTION (default: change not made if constraint ultimately violated)
    - RESTRICT (no temporary violations)
Referential integrity actions (SQL:1999)

Modifications violating a foreign key constraint

- referencing table: *disallowed*
- referenced table:
  - events:
    - ON UPDATE
    - ON DELETE
  - actions:
    - SET DEFAULT
    - SET NULL
    - CASCADE
    - NO ACTION (default: change not made if constraint ultimately violated)
    - RESTRICT (no temporary violations)
Active databases

Database become active when augmented with active rules (triggers).

Basic format (ECA rules)

- Event
- if Condition
- then Action

Compare with integrity constraints
referential integrity actions
Active databases

Database become active when augmented with active rules (triggers).
Active databases

Database become active when augmented with active rules (triggers).

Basic format (ECA rules)

on Event if Condition then Action
Active databases

Database become active when augmented with active rules (triggers).

Basic format (ECA rules)

\text{on Event if Condition then Action}

Compare with

- integrity constraints
- referential integrity actions
Trigger execution

while there are triggered rules do
  find a triggered rule R
  evaluate the condition of R
  if the condition is true
    then execute the action of R

Execution granularity
smallest
database operation
data manipulation command
at the end of a transaction
Trigger execution

**Execution cycle**

while there are triggered rules do
  find a triggered rule R
  evaluate the condition of R
  if the condition is true
    then execute the action of R
Trigger execution

**Execution cycle**

while there are triggered rules do
    find a triggered rule R
    evaluate the condition of R
    if the condition is true
        then execute the action of R

**Execution granularity**

- *smallest* database operation
- data manipulation command
- at the end of a transaction
Triggers in SQL:1999

Defined using CREATE TRIGGER, associated with tables.
Triggers in SQL:1999

Defined using CREATE TRIGGER, associated with tables.

Triggering operation

- INSERT, DELETE, UPDATE
- execution mode: BEFORE or AFTER the triggering statement.
Triggers in SQL:1999

Defined using CREATE TRIGGER, associated with tables.

**Triggering operation**

- INSERT, DELETE, UPDATE
- execution mode: BEFORE or AFTER the triggering statement.

**Condition**

- arbitrary SQL predicate
- can reference new/old versions of affected rows or tables.
Triggers in SQL:1999

Defined using CREATE TRIGGER, associated with tables.

Triggering operation

- INSERT, DELETE, UPDATE
- execution mode: BEFORE or AFTER the triggering statement.

Condition

- arbitrary SQL predicate
- can reference new/old versions of affected rows or tables.

Granularity

- row-level (executed once for each modified row)
- statement-level (executed once for each statement)
Triggers in SQL:1999

Defined using `CREATE TRIGGER`, associated with tables.

**Triggering operation**
- INSERT, DELETE, UPDATE
- execution mode: BEFORE or AFTER the triggering statement.

**Condition**
- arbitrary SQL predicate
- can reference new/old versions of affected rows or tables.

**Granularity**
- row-level (executed once for each modified row)
- statement-level (executed once for each statement)

**Action**
- one or more SQL statements
Execution

Trigger execution order determined by their definition order.
Execution

Trigger execution order determined by their *definition order*.

**BEFORE triggers**

- executed immediately
- cannot modify the database.
Execution

Trigger execution order determined by their *definition order.*

<table>
<thead>
<tr>
<th>BEFORE triggers</th>
</tr>
</thead>
<tbody>
<tr>
<td>• executed immediately</td>
</tr>
<tr>
<td>• cannot modify the database.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AFTER triggers</th>
</tr>
</thead>
<tbody>
<tr>
<td>• queued</td>
</tr>
<tr>
<td>• fire after integrity checks and the execution of referential integrity actions.</td>
</tr>
</tbody>
</table>
Execution
Trigger execution order determined by their definition order.

BEFORE triggers
- executed immediately
- cannot modify the database.

AFTER triggers
- queued
- fire after integrity checks and the execution of referential integrity actions.

Sequencing
1. execution of BEFORE triggers
2. execution of transaction
3. execution of referential integrity actions
4. constraint evaluation
5. execution of AFTER triggers
Views

Updatable views in SQL

A single SELECT from some relation R cannot appear in subqueries.

The SELECT list has to contain enough attributes that every tuple inserted into the view can be filled with nulls or default values (this implies that the list contains the primary key).

View maintenance

INSTEAD triggers
Updatable views in SQL

- a single SELECT from some relation $R$
- $R$ cannot appear in subqueries
- SELECT list has to contain enough attributes that every tuple inserted into the view can be filled with nulls or default values (this implies that the list contains the primary key)
Views

Updatable views in SQL

- a single SELECT from some relation R
- R cannot appear in subqueries
- SELECT list has to contain enough attributes that every tuple inserted into the view can be filled with nulls or default values (this implies that the list contains the primary key)

View maintenance

- INSTEAD triggers
Authorization in SQL:1999

Privileges for accessing/modifying data
- reading data from a relation/view
- inserting/updating/deleting data in a relation/view
- creating/dropping relations
- creating/dropping views
- adding/dropping columns
- referencing a relation (foreign keys)

Roles

Integrity and security
Authorization in SQL:1999

Privileges for accessing/modifying data

- reading data from a relation/view
- inserting/updating/deleting data in a relation/view
- creating/dropping relations
- creating/dropping views
- adding/dropping columns
- referencing a relation (foreign keys)
- roles
Granting and revoking privileges

Grant

grant

privilege list

on

relation or view name

to

user/role list

[with grant option]

Revoke

revoke

privilege list

on

relation or view name

from

user/role list

[restrict | cascade]
Granting and revoking privileges

Grant

grant privilege list on relation or view name
to user/role list [with grant option]
Granting and revoking privileges

**Grant**

grant privilege list on relation or view name
to user/role list [with grant option]

**Revoke**

revoke privilege list on relation or view name
from user/role list [restrict | cascade]
Checking authorization

Authorization graph
nodes: users + privileges
edges: authorizations granted (and not revoked)

$U$ has authorization to do $A$ iff there is a path authorizing $A$ because of the database element in question to $U$. 
Checking authorization

Authorization graph

- nodes: users + privileges
- edges: authorizations granted (and not revoked)
Checking authorization

Authorization graph

- **nodes**: users + privileges
- **edges**: authorizations granted (and not revoked)

\[ U \text{ has authorization to do } A \text{ iff there is a path authorizing } A \text{ from the node that has } A \text{ because of the database element in question to } U. \]
Authorization on views
Authorization on views

To compute view contents

- read privileges on the underlying relations
Authorization on views

To compute view contents
- read privileges on the underlying relations

To modify a view
- appropriate modification privileges on the underlying relations