SQL: Data Definition Transactions and Access

Week 12

Semester 1, CITS1402, 2024

Transactions

A Transaction is a

- logical unit of work
- consisting of one or more SQL statements
- that is guaranteed to be atomic with respect to recovery

SQL defines transaction model based on COMMIT and ROLLBACK.

Transactions

- **Transactions are ACID:**
  - **Atomicity:** ensures that all operations within the work unit are completed successfully; otherwise, the transaction is aborted at the point of failure and previous operations are rolled back to their former state.
  - **Consistency:** ensures that the database properly changes states upon a successfully committed transaction.

Transactions

- **Transactions are ACID:**
  - **Isolation:** enables transactions to operate independently of and transparent to each other.
  - **Durability:** ensures that the result or effect of a committed transaction persists in case of a system failure.

Transactions

- An SQL transaction automatically begins with a *transaction-initiating* SQL statement
  - e.g. SELECT, INSERT

- Changes made by transaction are not visible to other concurrently executing transactions until transaction completes.

Transactions

- Transaction can complete in one of four ways:

  - COMMIT ends transaction successfully, making changes permanent.

  - ROLLBACK aborts transaction, backing out any changes made by transaction.

  - For programmatic SQL, successful program termination ends final transaction successfully, even if COMMIT has not been executed.

  - For programmatic SQL, abnormal program end aborts transaction.

Transactions

- New transaction starts with next transaction-initiating statement. (select, insert, update, …)
- SQL transactions cannot be nested.
- SET TRANSACTION allows users to configure transaction properties:

```
SET TRANSACTION
   [READ ONLY | READ WRITE] |
   [ISOLATION LEVEL READ UNCOMMITTED |
   READ COMMITTED | REPEATABLE READ | SERIALIZABLE ]
```

Only safe level

SQLite Transactions

- **SQLite is in "auto-commit" mode by default**
  - **commits after every statement**
- begin [transaction]; -- begin a transaction (turn-off auto)
- commit; -- ends and saves the transaction
- rollback; -- undo changes back to the begin;
- **Transactions can be**
  - **deferred: lock acquired on database access**
  - **immediate: lock acquired at begin**
  - **exclusive: no other process can access database**
- **Transactions in SQLite are SERIALIZABLE**

Immediate and Deferred Integrity Constraints

- **Do not always want constraints to be checked immediately, but instead at transaction commit.**
- **Constraint may be defined as INITIALLY IMMEDIATE or INITIALLY DEFERRED, indicating mode the constraint assumes at start of each transaction.**
- **In former case, also possible to specify whether mode can be changed subsequently using qualifier [NOT] DEFERRABLE.**
- **Default mode is INITIALLY IMMEDIATE.**

Not in SQLite

Immediate and Deferred Integrity Constraints

- SET CONSTRAINTS statement used to set mode for specified constraints for current transaction:

```
SET CONSTRAINTS
    {ALL | constraintName [, . . . ]}
      {DEFERRED ¦ IMMEDIATE}
```

e.g. constraintName → PRIMARY KEY, REFERENCES

Chapter 7 - Objectives

- Data definition
- CREATE table statements
- Data types supported by SQL standard
- ALTER table statements
- Purpose of integrity enhancement feature of SQL
- Purpose of Views
- ISO transaction model
- **Access Control**

Not in SQLite

Access Control - Authorization Identifiers and Ownership

- Authorization identifier is normal SQL identifier used to establish identity of a user. Usually has an associated password.
- Used to determine which objects user may reference and what operations may be performed on those objects.
- Each object created in SQL has an owner, as defined in AUTHORIZATION clause of schema to which object belongs.
- Owner is only person who may know about it.

Privileges

- **Actions user permitted to carry out on given base table or view:**

SELECT	Retrieve data from a table.
INSERT	Insert new rows into a table.
UPDATE	Modify rows of data in a table.
DELETE	Delete rows of data from a table.
REFERENCES	Reference columns of named table in integrity constraints.
USAGE	Use domains, collations, character sets, and translations.

Privileges

- **Can restrict INSERT/UPDATE/REFERENCES to named columns.**
- **Owner of table must grant other users the necessary privileges using GRANT statement.**
- **To create view, user must have SELECT privilege on all tables that make up view and REFERENCES privilege on the named columns.**

GRANT

```
GRANT    {PrivilegeList | ALL PRIVILEGES}
ON       ObjectName
TO {AuthorizationIdList | PUBLIC}
[WITH GRANT OPTION]
```

- *PrivilegeList* **consists of one or more privileges separated by commas.**
- **ALL PRIVILEGES grants all privileges to a user.**

GRANT

```
GRANT   {PrivilegeList | ALL PRIVILEGES}
ON      ObjectName
TO      {AuthorizationIdList | PUBLIC}
[WITH GRANT OPTION]
```

- **PUBLIC allows access to be granted to all present and future authorized users.**
- ***ObjectName* can be a base table, view, domain, character set, collation or translation.**
- **WITH GRANT OPTION allows privileges to be passed on.**

Not in SQLite

Example 7.7/8 - GRANT

Give Manager full privileges to Staff table.

```
GRANT ALL PRIVILEGES
ON Staff
TO Manager WITH GRANT OPTION;
```

Give users Personnel and Director SELECT and UPDATE on column salary of Staff.

```
GRANT SELECT, UPDATE (salary)
ON Staff
TO Personnel, Director;
```

Not in SQLite

Example 7.9 - GRANT Specific Privileges to PUBLIC

Give all users SELECT on Branch table.

```
GRANT SELECT
ON Branch
TO PUBLIC;
```

Not in SQLite

REVOKE

- REVOKE takes away privileges granted with GRANT.

```
REVOKE [GRANT OPTION FOR]
  {PrivilegeList | ALL PRIVILEGES}
ON ObjectName
FROM {AuthorizationIdList | PUBLIC}
    [RESTRICT | CASCADE]
```

- ALL PRIVILEGES refers to all privileges granted to a user by user revoking privileges.

Not in SQLite

Example 7.10/11 - REVOKE Specific Privileges

Revoke privilege SELECT on Branch table from all users.

```
REVOKE SELECT
ON Branch
FROM PUBLIC;
```

Revoke all privileges given to Director on Staff table.

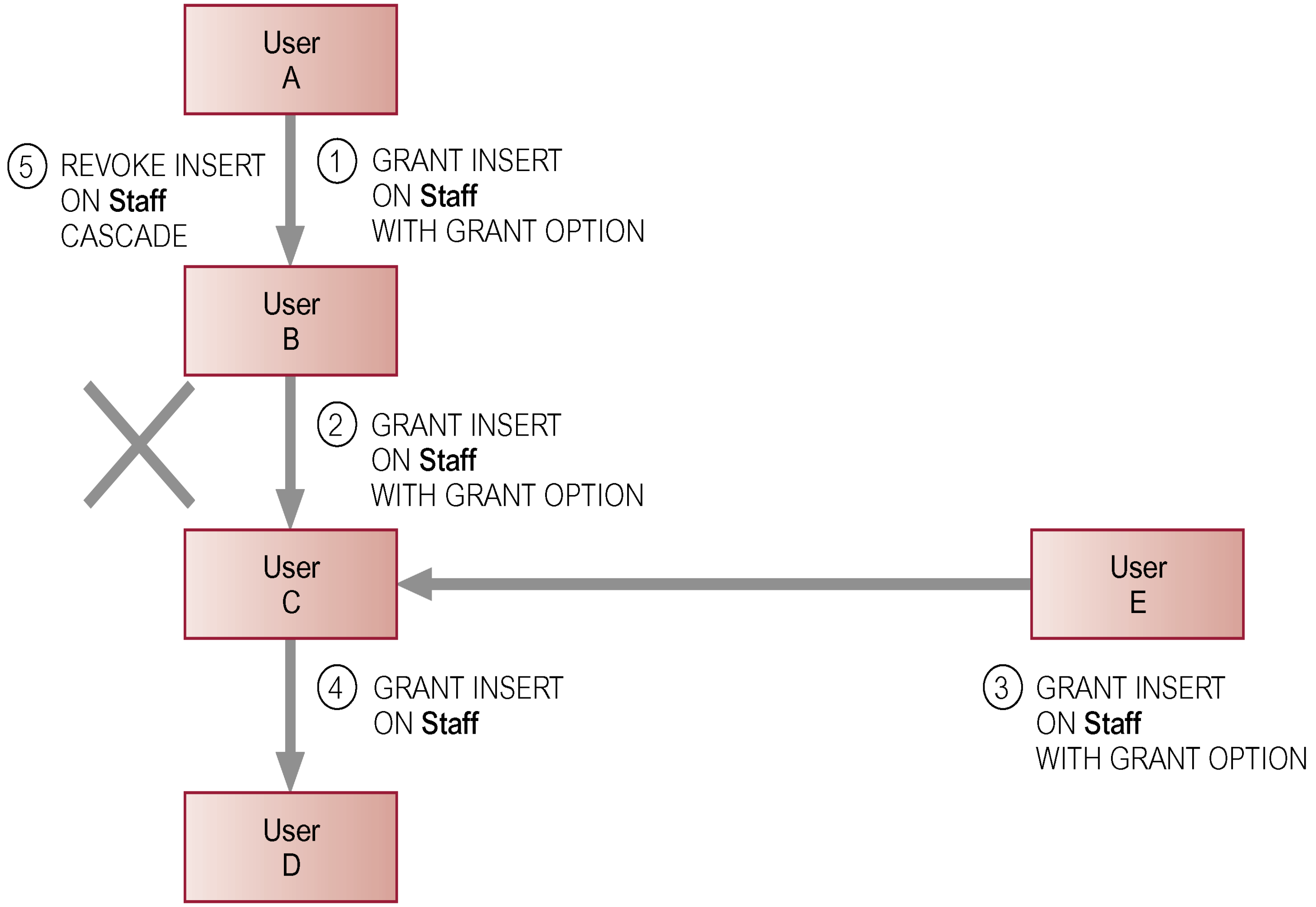
```
REVOKE ALL PRIVILEGES
ON Staff
FROM Director;
```

Not in SQLite

REVOKE

- GRANT OPTION FOR allows privileges passed on via WITH GRANT OPTION of GRANT to be revoked separately from the privileges themselves.
- REVOKE fails if it results in an abandoned object, such as a view, unless the CASCADE keyword has been specified.
- Privileges granted to this user by other users are not affected.

REVOKE


User A
5 REVOKE INSERT ON Staff CASCADE
1 GRANT INSERT ON Staff WITH GRANT OPTION
User B
2 GRANT INSERT ON Staff WITH GRANT OPTION
User C
User E
4 GRANT INSERT ON Staff
3 GRANT INSERT ON Staff WITH GRANT OPTION
User D