Transactions

---

yaml-language-server: $schema=schemas\page.schema.json

Object type: - Page Backlinks: - sql Creation date: "2026-05-26T09:34:43Z" Created by: - thomas id: bafyreiexau2pcju4y5v62ey66kh2ybaefl4qm3lbt6oppfisir5om2if2i

transatctions

PURPOSE

- data integrity
- Protect against partial updates
- concurrent access so multiple users can work without corrupting data.
- Support complex operations
- Enablesafe rollback

OBJECTIVES

- Define what a transaction is and explain its ACID properties.
- Start, commit, and roll backtransactions using SQL commands.
- Use save points to roll back part of a transaction.

DEFINITION

- A transaction is a sequence of one or more SQL statements that are executed as a single unit of work.
- The transaction must be fully completed(committed) so all changes become permanent, or fully undone(rolled back) so the database returns to its original state, ensuring data integrity.
- ACID describes the four main promises a database transaction makes to keep data safe and reliable.

ACID PROPERTIES

A—Atomicity(“All or Nothing”)

- A transaction’s operations areall completed successfullyornone at all.
- If one part fails, the entire transaction isrolled back.
- Example: In a bank transfer, money isdeducted from one accountandadded to another—never just one of them.

C—Consistency (“Valid State”)

- A transaction brings the database from onevalid state to another valid state, following allrules, constraints, and triggers.
- No transaction can breakdata integrityrules (like unique constraints or foreign keys).
- Example: You can’t insert a row with an invalid foreign key.

I—Isolation(“No Interference”)

- The operations of one transaction areinvisibleto others until the transaction is committed.
- Prevents concurrency problems likedirty readsorlost updates.
- Example: You can’t see a friend’s half-completed edit in an online order system.

D—Durability(“It Sticks”)

- Once a transaction iscommitted, the changes are permanent;evenif the system crashes.

- The database stores committed changes innon-volatile memory(disk, transaction logs).
- Example: After hitting “Place Order,” your order remains recorded even if the server restarts.

COMMIT AND ROLLBACK

- Commit: making a transaction permanent
- Rollback: undoing a transaction oOnce committed, no rollback is possible!
- Autocommit: *Every SQL statement is a transaction Every transaction commits automatically if it succeeds* Errors roll back that single SQL statement Autocommit is the default behaviour in PostgreSQL unless you explicitly start a transaction with START TRANSACTION or BEGIN

TRANSACTION PROCESSING

- Implicit start A new transaction begins automatically after a COMMIT or ROLLBACK when autocommit mode is enabled.
- Explicit start oBEGIN; sql code;COMMIT;—dialect oBEGIN; sql code;ROLLBACK;—dialect oSTART TRANSACTION; sql code;COMMIT; oSTART TRANSACTION; sql code;ROLLBACK; oAlwaysenda transaction explicitly—don’t leave it open.
- When does it make sense ? oIf a particular entry is to be deleted from multiple tables oIf a user made a mistake in adjustments
- Possible exceptions (product restrictions) :Some commands, likecatalogmodifications (e.g., CREATE TABLE, ALTER TABLE), may be handled outside normal transaction control.

SAVEPOINTS

- Purpose: oAllow you toundo part of the current transactionwithout rolling back the entire transaction. oAct asintermediate checkpoint (snapshot)within a transaction.
- You can have multiplesavepointsin a single transaction.
- Rolling back to a save point does not end the transaction.
- Example script template: START TRANSACTION; UPDATE… ; INSERT… ; SAVEPOINTS1;—first checkpoint INSERT… SAVEPOINTS2;—second checkpoint DELETE… ; ROLLBACK TO SAVEPOINTS2;—undo changes after S2 only

---

Revision #1
Created 2026-05-26 16:05:27 UTC by thomas
Updated 2026-05-26 16:05:27 UTC by thomas