Transactions for web developers

Aymeric Augustin - @aymericaugustin

DjangoCong - Belfort - September 28th, 2013
Today's talk

- **Framework**
  - `django`
  - `sqlite3`
  - `psycopg2`

- **Interface**
- **Database**
Transactions in SQL
“An **SQL-transaction** (sometimes simply called a "transaction") is a sequence of executions of SQL-statements that is **atomic** with respect to recovery.”

Source: [http://www.sqlite.org/atomiccommit.html](http://www.sqlite.org/atomiccommit.html)
“An **SQL-transaction** (sometimes simply called a "transaction") is a sequence of executions of SQL-statements that is **atomic** with respect to recovery.”

Source: [http://www.sqlite.org/atomiccommit.html](http://www.sqlite.org/atomiccommit.html)
BEGIN;

UPDATE cars
SET status = 'available'
WHERE id = 1;

UPDATE rentals
SET end = CURRENT_TIMESTAMP
WHERE car_id = 1 AND end IS NULL;

COMMIT;          -- or ROLLBACK;
Lifecycle of a transaction

Transaction initiating statement

Commit
Explicit rollback
Implicit rollback
Autocommit

- Commit implicitly after each statement.
- Wrap each statement in its own transaction.
- Just execute my query KTHXBYE!

- SQLite, PostgreSQL, MySQL have autocommit turned on by default.
Lifecycle under autocommit

START TRANSACTION
BEGIN (alias)

COMMIT
ROLLBACK

Auto-commit ON
Auto-commit OFF
Auto-commit ON

Transaction
Savepoints

becomes either

or

or

after

SAVEPOINT S3;

RELEASE SAVEPOINT S2;

ROLLBACK TO SAVEPOINT S2;
Python client libraries
PEP 249

• “If the database supports an auto-commit feature, this must be initially off.”

• “An interface method may be provided to turn it back on.”

• “Closing a connection without committing the changes first will cause an implicit rollback to be performed.”
• Tracks the transaction state.
• Inserts a BEGIN before each statement, unless there’s already a transaction in progress.
• Even before SELECT statements.
• “Idle in transaction”.
• `cnx.autocommit = True` disables this behavior.
• Tracks the transaction state.
• Parses statements to insert BEGIN or COMMIT:
  • SELECT: —
  • INSERT, UPDATE, DELETE, REPLACE: BEGIN
  • Any other statement: COMMIT
• cnx.isolation_level = None disables this behavior.
• This is totally unrelated to the isolation level!
sqlite3

• Tracks the transaction state.
• Parses statements to insert BEGIN or COMMIT:
  • SELECT:  —
  • INSERT, UPDATE, DELETE, REPLACE:  BEGIN
  • Any other statement:  COMMIT
  • Broken by design.
  • “SAVEPOINT foo” triggers a commit!
Django

≥ 1.6
Database-level autocommit

Database connection opened

SELECT

UPDATE

DELETE

INSERT

SELECT

SELECT

SELECT

Database connection closed
One HTTP request = one transaction. Commit on success, roll back on exception.

Per database.

Only for the view function.
High-level API: atomic

• Usable as a decorator or as a context manager.
• Commits on success, rolls back on exceptions.
• Can be nested without any restrictions.
  • Concept stolen from xact by Christophe Pettus.
• Guarantees atomicity.
  • “Don’t Give Users Guns Aimed At Feet” principle.
Low-level APIs

• Still there in case you’re sufficiently masochistic to implement your own transaction management.

• There aren’t many sane ways to combine them.
Thank you!

Questions?

https://docs.djangoproject.com/en/dev/topics/db/transactions/