Ebury Design Document Trade ID Generation

Author: Adam Abidi

Overview

This project solves the problem of generating unique, human-readable trade IDs from a fixed set of characters.
Each ID is 7 characters long and is guaranteed to be unique.

Approach

- Implemented **Base-N encoding** to transform an integer into a fixed-length code.
- Here N = 34, using characters: `0123456789ABCDEFGHJKLMNPQRSTUVWXYZ`.
- Each ID represents an integer in base-34 padded to 7 characters.

Database

- SQLite is used for persistence.
- Database file: `identity.db` (created in the project root).
- Schema:

```sql
CREATE TABLE IF NOT EXISTS trade_id_counter (
  code CHAR(7) NOT NULL UNIQUE
);
```

- The table always contains exactly one row: “code” which stores the latest allocated ID.

- Behavior:
If the table is empty and a generation is requested, the sequence starts at 0000000.
On each generation, the stored code is decoded, incremented, encoded and updated.
On bulk generation, only the last generated code is persisted.

Only one column is required:
The code itself is the state; no need to store the numeric index.
Previous IDs are not stored (no business use, avoids wasted space).

Concurrency & Reliability

- Each generator call opens its own SQLite connection (check_same_thread=False).

- **BEGIN IMMEDIATE** transactions ensure only one writer updates at a time.

- WAL mode + synchronous=NORMAL improve concurrency and performance:

  - Readers are not blocked while a writer is active.
  - Writes remain durable enough for production use.
  - The DB guarantees crash safety: if the server restarts, the last allocated ID is preserved.

Fault Tolerance

- Restarting the process does not duplicate IDs.

- State is persisted in the DB, not memory or environment variables.

- If two processes attempt the first insert at the same time, the UNIQUE constraint ensures only one succeeds.

Performance

- Bulk generation is implemented in a single transaction:

- Decode the last code, generate n new codes, update the DB with the last.

- Tests confirm millions of IDs can be generated in a few seconds.

Overflow Protection

- If a request exceeds the maximum (34^7), an OverflowError("Space Limit Reached") is raised.

- A dedicated unit test validates this behavior.

Code Quality

- Followed **DRY** principle.

- A single **create_ids(n)** function implements both single and bulk generation.

- Clear separation of logic:

    - Encoding/decoding logic.

    - Database persistence.

    - generate & generate_bulk.

Test Adjustments

- The provided test `test_restarting_process_does_not_duplicate_ids` assumed a Unix-like environment (it used `/usr/bin/env` and relative paths).
- Since I worked on Windows, I updated the test to use `sys.executable` and absolute paths so that the subprocess could correctly locate `scripts/generate.py`.

- This change is environment-specific. On Linux/macOS the original test will work without modification.
- The core logic of the test ensures no duplicate IDs after a restart is preserved.