Multi-table Databases

Dr Peadar Grant

October 15, 2024

Contents

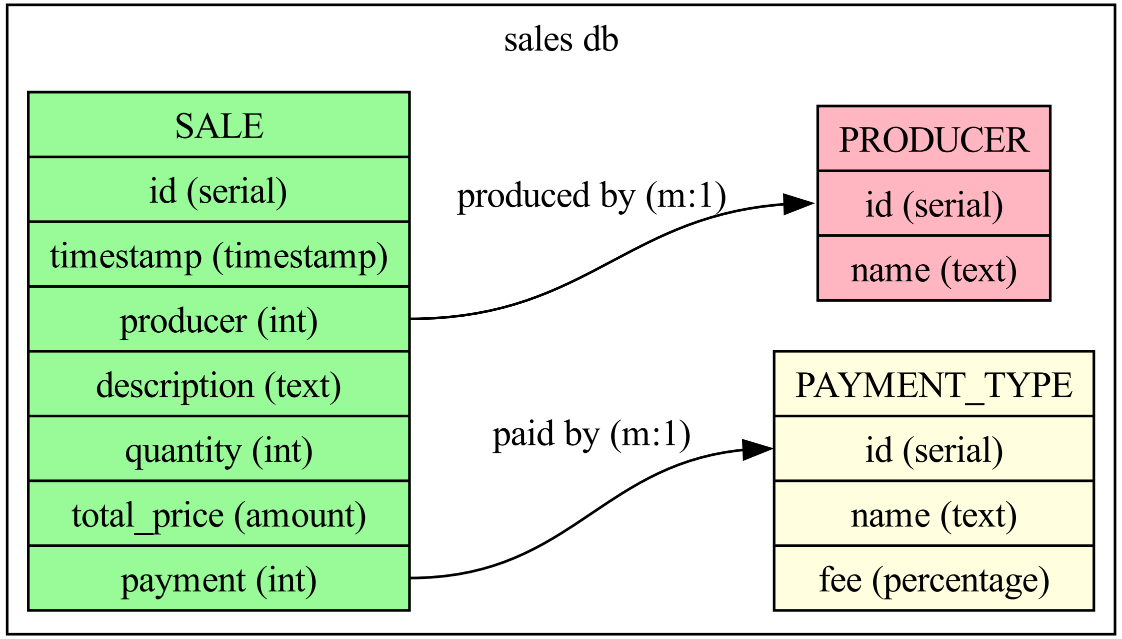
1 Scenario

A group of local artisan producers operate a popup stall for a 3-month period. They contract a local IT expert to build a simple sales tracking and reporting system.

Main requirements

1. Sales must be numbered and timestamped.
2. Every sale is separate (no concept of a "basket" for simplicity!)
3. Stall accepts cash and card tender.
    - Card payments have a 1.68% processor fee (which may vary).
    - May be requirement to accept additional payment types during period.
4. Sale must capture producer, description, quantity sold, amount paid and tender.
5. Everyone will be paid at the end of the 3-month period.
6. Database must be able to produce the following reports:

1.1 Implementation with 3 tables



1.2 Key decisions

1. **Auto-numbered** `id` column using `SERIAL`.
2. **Primary key** will be the auto-numbered `id` column.
3. All fields will be **not null**.
4. Every sale must match a valid:

   producer from the `producer` table.

   tender from the `tender` table.
5. Storage of numerical values:

   Amounts as `NUMERIC(6,2)` typed `amount`.

   Rate as `NUMERIC(6,2)` typed `rate`.

2 Domains

Domains are user-defined types based on underlying type. Defaults to NULL allowed, best to define any NOT NULL conditions on the underlying columns. CHECK constraints can be defined.

```
/* Number to hold 1-10 user rating */
CREATE DOMAIN rating AS integer CHECK ( VALUE >= 1 AND VALUE <=10 );

/* Just use the domain as type when creating table */
CREATE TABLE restaurants (
    id bigserial primary key,
    /* creating two columns using our domain: */
    visitor_rating rating not null,
    reviewer_rating rating,
    /* other columns */
);
```

3 Foreign keys

Foreign keys require that values in a column (or a group of columns) must match the values appearing in some row of another table. This maintains the referential integrity.

```
/* each employee must be in a valid department */

CREATE TABLE department (
id bigserial primary key,
name text not null unique
/* other fields as required */
);

CREATE TABLE employee (
id bigserial primary key,
surname text not null,
firstname text not null,
department bigint not null REFERENCES department
/* other fields as required */
)
```

3.1 DELETE / UPDATE behaviour

Possible behaviours: NO ACTION, CASCADE, SET NULL.

```
CREATE TABLE product (
/* other fields */

department bigint references departments,
/* NO ACTION is the default, prohibits conflicting delete */

supplier bigint references suppliers ON DELETE CASCADE,
/* DELETE in suppliers deletes linked products */

policy bigint references policies ON DELETE SET NULL,
/* SETS product.policy to NULL when row in policies deleted  */

);
```

4 JOIN

The JOIN operation permits queries across more than one table. See both the JOIN tutorial and the Table expressions section from Postgres manual for full details.

Assume R1 to be a row of Table T1. Similarly R2 for T2. Normally should explicitly specify columns required and use table prefix to avoid ambiguity.

4.1 INNER JOIN

For each row R1 of T1, the joined table has a row for each row in T2 that satisfies the join condition with R1.

4.2 LEFT JOIN

Same as INNER JOIN except that output also includes any row in T1 that does not match one or more rows in T2. Null values are substituted for T2 in the output row.

4.3 RIGHT JOIN

Similar to LEFT JOIN. Same as INNER JOIN, except any row in T2 that does not match $\geq$ 1 rows in T1 will be output. Null values are subtituted for T1 columns in the otuput row.

4.4 FULL JOIN

Similar to combination of LEFT and RIGHT JOIN. INNER JOIN performed. Then rows in T1 without corresponding T2 output with nulls for T2. Same again, rows in T2 without corresponding T1 rows output with nulls for T1.

5 Views

Views are defined by @connolly:2015:database as:

> The dynamic result of one or more relational operations operating on the base relations to produce another relation. A view is a virtual relation that does not necessarily exist in the database but can be produced upon request by a particular user, at the time of the request.

```sql
/* Creation syntax: */
CREATE VIEW my_view AS
SELECT ... ;
/* select statement can be any valid select */

/* VIEW can be selected like any other table */
SELECT * FROM my_view ;
```