1 Customizing SQLSmith for SQL Query Generation

1.1 SQLSmith

SQLSmith is an open-source tool for randomized generation of SQL queries, designed originally for database fuzz testing.

SQLSmith is implemented as a grammar-driven random generator. It relies on three main components:

1. **`grammar`**: grammar production, defines the syntactic structure of queries (SELECT, DELETE, MERGE, FROM, JOIN, etc.);
2. **`expr`**: value expressions production, generates expression including values, predicates, comparisons, functions, and aggregations;
3. **`relmodel`**: models relations, columns, and aliases such as `ref_0`, `subq_0`, and `c0` which SQLSmith uses internally during generation.

1.2 Motivation for Customization

The goal is to adapt SQLSmith to generate a dataset of *realistic and structurally controlled* SQL queries, suitable for being used in ProvSQL and LLM fine-tuning. This required constraining or removing the following behaviours:

- automatically generated aliases (`ref_0`, `c12`, etc.);
- nonstandard system functions (distributed functions, probability functions etc.);
- `TABLESAMPLE`, `LATERAL`, and CTEs;
- nested aggregates (e.g., `SUM (MIN (x))`);

Three source files were modified: `grammar.cc`, `expr.cc`, `postgres.cc` and `relmodel.hh`.

1.3 Modifications to `grammar.cc`

The file `grammar.cc` controls the high-level structure of SQL queries. The following customizations were applied:

- disabling constructs that introduce uncontrolled complexity:
  - `table_subquery`;
  - `lateral_subquery`;
  - `table_sample`;
  - subqueries generated via `WITH` (CTEs).

- rewriting the `table_ref::factory` method to produce only real base tables or joins between base tables;
- simplifying the `SELECT` list by removing automatically generated derived column aliases (`c0`, `ref0`, ...);
- enforcing the use of only `SELECT` statements (disabling `DELETE`, `UPDATE`, `MERGE`, etc.);
- allowed only inner joins by modifying the `join_cond::factory` method to always set the join type to `inner`.

1.4 Modifications to `expr.cc`

The file `expr.cc` defines how expressions, predicates, and function calls are generated. Some modifications were applied, as:

- **Disabling subqueries in expressions** (`atomic_subselect`);
- **Disabling window functions** (`window_function`);
- **Avoiding nested aggregate functions** by modifying the `funcall` constructor:

```
if (agg && dynamic_cast<funcall*>(p->pprod)) {
    fail("nested aggregate not allowed");
}
```

- **Removing complex or system-defined functions** from the generative process.
- **Removing exists predicates** from the generative process because it is not supported in ProvSQL.
- **Modifying the set of available operators** to use only basic operators in comparisons.

```
bool name_ok =
name == "="  ||
name == "<"  ||
name == ">"  ||
name == "<=" ||
name == ">=";
```

1.5 Modifications to `postgres.cc`

The file `postgres.cc` defines PostgreSQL-specific functions and types. The following changes were applied:

- removing PostgreSQL aggregates and routines from the generative process, such as `pg_catalog.random()`, `pg_catalog.setseed()`.

1.6 Modifications to `relmodel.hh`

The file `relmodel.hh` defines the representation of tables, relations, and aliasing. SQLSmith generates internal aliases such as `ref_0`, `ref_1`, and derived column names such as `c0`, `c1`.

The following changes were applied:

- aliases in the table naming and routine naming are disabled, using the table name directly.

This transforms outputs like:

```
FROM public.partsupp
FROM system.partsupp
SELECT pg_catalog.min()
```

into:

```
FROM partsupp
FROM partsupp
SELECT MIN()
```

1.7 Resulting Behaviour

After applying these modifications, SQLSmith produces:

- only `SELECT` queries (no CTEs, no MERGE/UPDATE/DELETE);
- joins between real base tables only;
- simple and readable `SELECT` lists without synthetic aliases;
- aggregate functions without nesting;