Database Metadata

Chapter 10

Michael Tsiang

Stats 167: Introduction to Databases

UCLA

Motivation

It is often helpful to be able to get an overview of the overall structure (or **schema**) of a relational database from within SQL without having to resort to a schema diagram (or ERD) or other outside information.

The **database metadata** is the information (or data) about the structure, organization, and properties of the database, not about the data itself.

We want to introduce a few tools for viewing database metadata to help get a sense of the data when writing queries.

What is Metadata?

Database metadata includes many different components of the database:

Schema

- Tables: Names, definitions, and relationships between tables.
- Columns: Column names, datatypes, constraints (e.g., NOT NULL, UNIQUE), default values.
- Indexes: Information on indexes used to speed up queries.
- Keys: Primary keys, foreign keys, and their relationships.

What is Metadata? (Continued)

Programs:

- Views: Definitions of saved queries (i.e., virtual tables).
- Stored procedures/functions: Definitions of any saved logic or routines.
- Triggers: Logic that runs in response to database events (e.g., insert, update).

Security:

- User: Who can access the data
- Permissions: What users can access and with what level of permission.

Information Schema: Dialect and DBMS Dependence

One of the challenges of viewing database metadata is that metadata is stored and accessed differently depending on the DBMS.

The **information schema** (`information_schema`) is the ANSI standard set of read-only views that show database metadata in an accessible way. However, not all DBMSs support `information_schema`, and different SQL dialects use different commands to access the information within it.

A list of the DBMSs that support and do not support information schema can be found on Wikipedia: https://en.wikipedia.org/wiki/Information_schema

In particular, **SQLite does not support information schema**.

Examples of Information Schema

For example, in PostgreSQL:

```
SELECT table_name
FROM information_schema.tables
WHERE table_type = 'BASE TABLE'
   AND table_schema = 'public';
```

In MySQL/MariaDB:

```
SELECT table_name
FROM information_schema.tables
WHERE table_type = 'BASE TABLE'
  AND table_schema = DATABASE();
```

Specifically in MySQL/MariaDB, we also have the convenient command:

```
SHOW TABLES;
```

SQLite: sqlite_master

Instead of `information_schema`, SQLite uses a special read-only table called **`sqlite_master`** to store schema information.

The `sqlite_master` table can be queried like a normal table, but it cannot be modified directly.

Example: List of Tables (sqlite_master)

We can use `sqlite_master` to output a list of the tables in the database:

```sql
SELECT name
FROM sqlite_master
WHERE type = 'table';
```

name
Customers
OrderItems
Orders
Products
Vendors
OrdersBackup
OrderItemsBackup

Example: SQL Commands (sqlite_master)

One powerful way to use `sqlite_master` is to see the SQL statements used to create tables. This helps us understand the purpose, keys, and datatypes of each column within the tables.

```sql
SELECT sql
FROM sqlite_master
WHERE tbl_name = 'Orders';
```

SQLite: PRAGMA

In SQLite, **`PRAGMA`** is a special command used to query or modify the internal operational parameters of the SQLite engine. It can also be used to inspect and manage metadata and settings.

The **`PRAGMA`** command is specific to SQLite and is not compatible with any other DBMS.

While `PRAGMA` has many uses, a common one is to output the schema of individual tables in the database:

```
PRAGMA table_info('Orders');
```

cid	name	type	notnull	dflt_value	pk
0	order_num	INT	1	NA	1
1	order_date	datetime	1	NA	0
2	cust_id	char(10)	1	NA	0

Resources for PRAGMA

There is much more that you can do with `PRAGMA` to affect the internal behavior of a SQLite database.

For further resources:

- https://www.sqlite.org/pragma.html
- https://www.tutorialspoint.com/sqlite/sqlite_pragma.htm