SQLite example

Contents

This example showcases hooking up an LLM to answer questions over a database.

This uses the example Chinook database. To set it up follow the instructions on https://database.guide/2-sample-databases-sqlite/, placing the `.db` file in a notebooks folder at the root of this repository.

```
from langchain import OpenAI, SQLDatabase, SQLDatabaseChain
```

```
db = SQLDatabase.from_uri("sqlite:///../../../../notebooks/Chinook.db")
llm = OpenAI(temperature=0)
```

NOTE: For data-sensitive projects, you can specify `return_direct=True` in the `SQLDatabaseChain` initialization to directly return the output of the SQL query without any additional formatting. This prevents the LLM from seeing any contents within the database. Note, however, the LLM still has access to the database scheme (i.e. dialect, table and key names) by default.

```
db_chain = SQLDatabaseChain(llm=llm, database=db, verbose=True)
```

```
db_chain.run("How many employees are there?")
```

Skip to main content

```
> Entering new SQLDatabaseChain chain...
How many employees are there?
SQLQuery:
```

```
/Users/harrisonchase/workplace/langchain/langchain/sql_database.py:120: SAWarning:
Dialect sqlite+pysqlite does *not* support Decimal objects natively, and
SQLAlchemy must convert from floating point - rounding errors and other issues may
occur. Please consider storing Decimal numbers as strings or integers on this
platform for lossless storage.
  sample_rows = connection.execute(command)
```

```
 SELECT COUNT(*) FROM Employee;
SQLResult: [(8,)]
Answer: There are 8 employees.
> Finished chain.
```

```
' There are 8 employees.'
```

Customize Prompt

You can also customize the prompt that is used. Here is an example prompting it to understand that foobar is the same as the Employee table

```
from langchain.prompts.prompt import PromptTemplate

_DEFAULT_TEMPLATE = """Given an input question, first create a syntactically
correct {dialect} query to run, then look at the results of the query and return
the answer.
Use the following format:

Question: "Question here"
SQLQuery: "SQL Query to run"
SQLResult: "Result of the SQLQuery"
Answer: "Final answer here"

Only use the following tables:

{table_info}

If someone asks for the table foobar, they really mean the employee table.

Question: {input}"""
```

Skip to main content

```
    input_variables=["input", "table_info", "dialect"], template=_DEFAULT_TEMPLATE
)
```

```
db_chain = SQLDatabaseChain(llm=llm, database=db, prompt=PROMPT, verbose=True)
```

```
db_chain.run("How many employees are there in the foobar table?")
```

```
> Entering new SQLDatabaseChain chain...
How many employees are there in the foobar table?
SQLQuery: SELECT COUNT(*) FROM Employee;
SQLResult: [(8,)]
Answer: There are 8 employees in the foobar table.
> Finished chain.
```

```
' There are 8 employees in the foobar table.'
```

Return Intermediate Steps

You can also return the intermediate steps of the SQLDatabaseChain. This allows you to access the SQL statement that was generated, as well as the result of running that against the SQL Database.

```
db_chain = SQLDatabaseChain(llm=llm, database=db, prompt=PROMPT, verbose=True,
return_intermediate_steps=True)
```

```
result = db_chain("How many employees are there in the foobar table?")
result["intermediate_steps"]
```

```
> Entering new SQLDatabaseChain chain...
How many employees are there in the foobar table?
SQLQuery: SELECT COUNT(*) FROM Employee;
SQLResult: [(8,)]
Answer: There are 8 employees in the foobar table.
> Finished chain.
```

Skip to main content

```
[' SELECT COUNT(*) FROM Employee;', '[(8,)]']
```

Choosing how to limit the number of rows returned

If you are querying for several rows of a table you can select the maximum number of results you want to get by using the 'top_k' parameter (default is 10). This is useful for avoiding query results that exceed the prompt max length or consume tokens unnecessarily.

```
db_chain = SQLDatabaseChain(llm=llm, database=db, verbose=True, top_k=3)
```

```
db_chain.run("What are some example tracks by composer Johann Sebastian Bach?")
```

```
> Entering new SQLDatabaseChain chain...
What are some example tracks by composer Johann Sebastian Bach?
SQLQuery: SELECT Name, Composer FROM Track WHERE Composer LIKE '%Johann Sebastian Bach%' LIMIT 3;
SQLResult: [('Concerto for 2 Violins in D Minor, BWV 1043: I. Vivace', 'Johann Sebastian Bach'), ('Aria Mit 30 Veränderungen, BWV 988 "Goldberg Variations": Aria', 'Johann Sebastian Bach'), ('Suite for Solo Cello No. 1 in G Major, BWV 1007: I. Prélude', 'Johann Sebastian Bach')]
Answer: Some example tracks by composer Johann Sebastian Bach are 'Concerto for 2 Violins in D Minor, BWV 1043: I. Vivace', 'Aria Mit 30 Veränderungen, BWV 988 "Goldberg Variations": Aria', and 'Suite for Solo Cello No. 1 in G Major, BWV 1007: I. Prélude'.
> Finished chain.
```

```
' Some example tracks by composer Johann Sebastian Bach are \'Concerto for 2 Violins in D Minor, BWV 1043: I. Vivace\', \'Aria Mit 30 Veränderungen, BWV 988 "Goldberg Variations": Aria\', and \'Suite for Solo Cello No. 1 in G Major, BWV 1007: I. Prélude\'.'
```

Adding example rows from each table

Sometimes, the format of the data is not obvious and it is optimal to include a sample of rows from the tables in the prompt to allow the LLM to understand the data before providing a final

Skip to main content

query. Here we will use this feature to let the LLM know that artists are saved with their full names by providing two rows from the `Track` table.

```
db = SQLDatabase.from_uri(
    "sqlite:///../../../../notebooks/Chinook.db",
    include_tables=['Track'], # we include only one table to save tokens in the prompt :)
    sample_rows_in_table_info=2)
```

The sample rows are added to the prompt after each corresponding table's column information:

```
print(db.table_info)
```

```
CREATE TABLE "Track" (
        "TrackId" INTEGER NOT NULL,
        "Name" NVARCHAR(200) NOT NULL,
        "AlbumId" INTEGER,
        "MediaTypeId" INTEGER NOT NULL,
        "GenreId" INTEGER,
        "Composer" NVARCHAR(220),
        "Milliseconds" INTEGER NOT NULL,
        "Bytes" INTEGER,
        "UnitPrice" NUMERIC(10, 2) NOT NULL,
        PRIMARY KEY ("TrackId"),
        FOREIGN KEY("MediaTypeId") REFERENCES "MediaType" ("MediaTypeId"),
        FOREIGN KEY("GenreId") REFERENCES "Genre" ("GenreId"),
        FOREIGN KEY("AlbumId") REFERENCES "Album" ("AlbumId")
)
/*
2 rows from Track table:
TrackId Name    AlbumId MediaTypeId     GenreId Composer        Milliseconds
Bytes   UnitPrice
1       For Those About To Rock (We Salute You) 1       1       1       Angus
Young, Malcolm Young, Brian Johnson     343719  11170334        0.99
2       Balls to the Wall       2       2       1       None    342562  5510424
0.99
*/
```

```
/home/jon/projects/langchain/langchain/sql_database.py:135: SAWarning: Dialect
sqlite+pysqlite does *not* support Decimal objects natively, and SQLAlchemy must
convert from floating point - rounding errors and other issues may occur. Please
consider storing Decimal numbers as strings or integers on this platform for
lossless storage.
  sample_rows = connection.execute(command)
```

Skip to main content

```
db_chain = SQLDatabaseChain(llm=llm, database=db, verbose=True)
```

```
db_chain.run("What are some example tracks by Bach?")
```

```
> Entering new SQLDatabaseChain chain...
What are some example tracks by Bach?
SQLQuery: SELECT Name FROM Track WHERE Composer LIKE '%Bach%' LIMIT 5;
SQLResult: [('American Woman',), ('Concerto for 2 Violins in D Minor, BWV 1043: I. Vivace',), ('Aria Mit 30 Veränderungen, BWV 988 "Goldberg Variations": Aria',), ('Suite for Solo Cello No. 1 in G Major, BWV 1007: I. Prélude',), ('Toccata and Fugue in D Minor, BWV 565: I. Toccata',)]
Answer: Some example tracks by Bach are 'American Woman', 'Concerto for 2 Violins in D Minor, BWV 1043: I. Vivace', 'Aria Mit 30 Veränderungen, BWV 988 "Goldberg Variations": Aria', 'Suite for Solo Cello No. 1 in G Major, BWV 1007: I. Prélude', and 'Toccata and Fugue in D Minor, BWV 565: I. Toccata'.
> Finished chain.
```

```
' Some example tracks by Bach are \'American Woman\', \'Concerto for 2 Violins in D Minor, BWV 1043: I. Vivace\', \'Aria Mit 30 Veränderungen, BWV 988 "Goldberg Variations": Aria\', \'Suite for Solo Cello No. 1 in G Major, BWV 1007: I. Prélude\', and \'Toccata and Fugue in D Minor, BWV 565: I. Toccata\'.'
```

Custom Table Info

In some cases, it can be useful to provide custom table information instead of using the automatically generated table definitions and the first `sample_rows_in_table_info` sample rows. For example, if you know that the first few rows of a table are uninformative, it could help to manually provide example rows that are more diverse or provide more information to the model. It is also possible to limit the columns that will be visible to the model if there are unnecessary columns.

This information can be provided as a dictionary with table names as the keys and table information as the values. For example, let's provide a custom definition and sample rows for the Track table with only a few columns:

```
custom_table_info = {
    "Track": """CREATE TABLE Track (
        "TrackId" INTEGER NOT NULL,
```

Skip to main content

```
        PRIMARY KEY ("TrackId")
)
/*
3 rows from Track table:
TrackId Name    Composer
1       For Those About To Rock (We Salute You) Angus Young, Malcolm Young, Brian
Johnson
2       Balls to the Wall       None
3       My favorite song ever   The coolest composer of all time
*/"""
}
```

```
db = SQLDatabase.from_uri(
    "sqlite:///../../../../notebooks/Chinook.db",
    include_tables=['Track', 'Playlist'],
    sample_rows_in_table_info=2,
    custom_table_info=custom_table_info)

print(db.table_info)
```

```
CREATE TABLE "Playlist" (
        "PlaylistId" INTEGER NOT NULL,
        "Name" NVARCHAR(120),
        PRIMARY KEY ("PlaylistId")
)
/*
2 rows from Playlist table:
PlaylistId      Name
1       Music
2       Movies
*/

CREATE TABLE Track (
        "TrackId" INTEGER NOT NULL,
        "Name" NVARCHAR(200) NOT NULL,
        "Composer" NVARCHAR(220),
        PRIMARY KEY ("TrackId")
)
/*
3 rows from Track table:
TrackId Name    Composer
1       For Those About To Rock (We Salute You) Angus Young, Malcolm Young, Brian
Johnson
2       Balls to the Wall       None
3       My favorite song ever   The coolest composer of all time
*/
```

Note how our custom table definition and sample rows for `Track` overrides the

Skip to main content

in this example `Playlist`, will have their table info gathered automatically as usual.

```
db_chain = SQLDatabaseChain(llm=llm, database=db, verbose=True)
db_chain.run("What are some example tracks by Bach?")
```

```
> Entering new SQLDatabaseChain chain...
What are some example tracks by Bach?
SQLQuery: SELECT Name, Composer FROM Track WHERE Composer LIKE '%Bach%' LIMIT 5;
SQLResult: [('American Woman', 'B. Cummings/G. Peterson/M.J. Kale/R. Bachman'),
('Concerto for 2 Violins in D Minor, BWV 1043: I. Vivace', 'Johann Sebastian
Bach'), ('Aria Mit 30 Veränderungen, BWV 988 "Goldberg Variations": Aria', 'Johann
Sebastian Bach'), ('Suite for Solo Cello No. 1 in G Major, BWV 1007: I. Prélude',
'Johann Sebastian Bach'), ('Toccata and Fugue in D Minor, BWV 565: I. Toccata',
'Johann Sebastian Bach')]
Answer: Some example tracks by Bach are 'American Woman', 'Concerto for 2 Violins
in D Minor, BWV 1043: I. Vivace', 'Aria Mit 30 Veränderungen, BWV 988 "Goldberg
Variations": Aria', 'Suite for Solo Cello No. 1 in G Major, BWV 1007: I. Prélude',
and 'Toccata and Fugue in D Minor, BWV 565: I. Toccata'.
> Finished chain.
```

```
' Some example tracks by Bach are \'American Woman\', \'Concerto for 2 Violins in
D Minor, BWV 1043: I. Vivace\', \'Aria Mit 30 Veränderungen, BWV 988 "Goldberg
Variations": Aria\', \'Suite for Solo Cello No. 1 in G Major, BWV 1007: I.
Prélude\', and \'Toccata and Fugue in D Minor, BWV 565: I. Toccata\'.'
```

SQLDatabaseSequentialChain

Chain for querying SQL database that is a sequential chain.

The chain is as follows:

```
1. Based on the query, determine which tables to use.
2. Based on those tables, call the normal SQL database chain.
```

This is useful in cases where the number of tables in the database is large.

```
from langchain.chains import SQLDatabaseSequentialChain
db = SQLDatabase.from_uri("sqlite:///../../../../notebooks/Chinook.db")
```

Skip to main content

```
chain = SQLDatabaseSequentialChain.from_llm(llm, db, verbose=True)
```

```
chain.run("How many employees are also customers?")
```

```
> Entering new SQLDatabaseSequentialChain chain...
Table names to use:
['Customer', 'Employee']

> Entering new SQLDatabaseChain chain...
How many employees are also customers?
SQLQuery: SELECT COUNT(*) FROM Employee INNER JOIN Customer ON Employee.EmployeeId
= Customer.SupportRepId;
SQLResult: [(59,)]
Answer: 59 employees are also customers.
> Finished chain.

> Finished chain.
```

```
' 59 employees are also customers.'
```