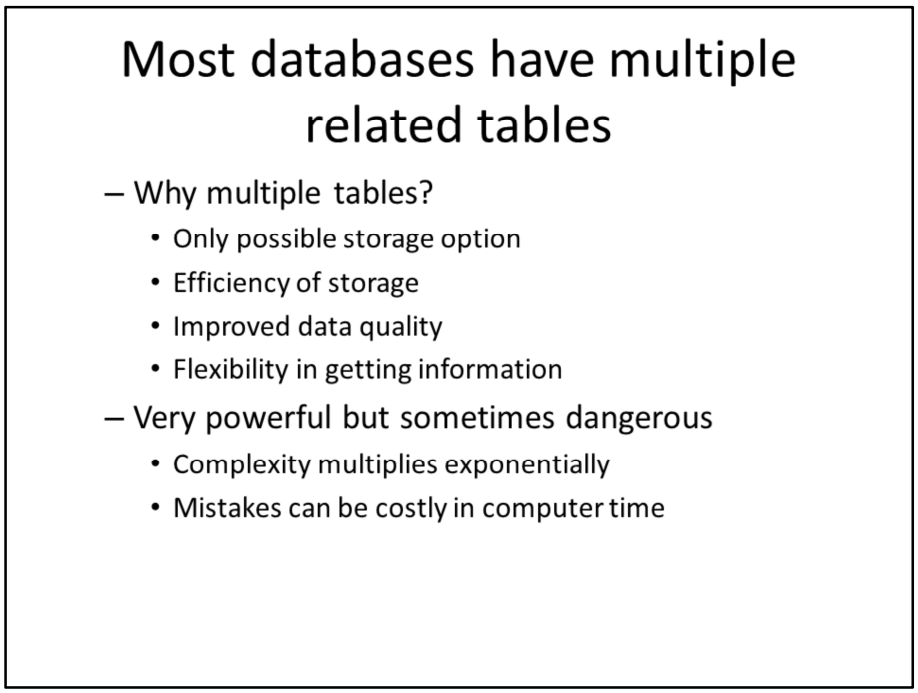
Managing multiple tables in SQL

Suman Sahil, Steve Simon

Creation date: 2017-09-30

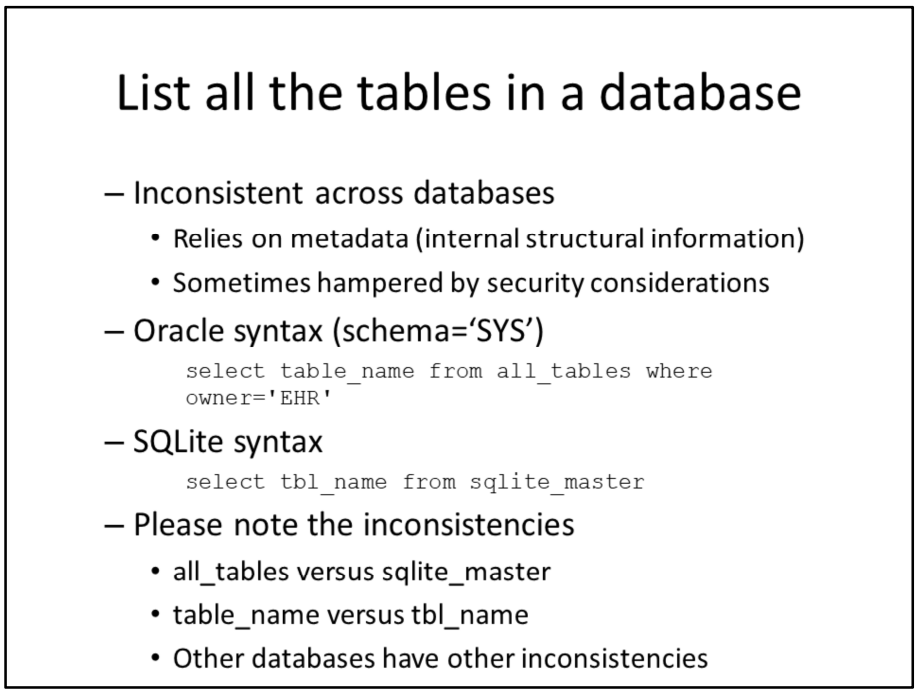


All of the examples you have seen so far are a bit simplistic because they have just a single table. In almost every real-world database that you will encounter, the database will have multiple related tables.

You might see multiple tables for the simple reason that your data is too complex to fit inside a single table. Even if it could fit inside a single table, that option would often be very inefficient in terms of both speed and storage capacity. Separate tables can also improve the quality of the data by removing redundancies that can become potential sources of error. The multiple table format also gives you a lot of flexibility in how you get information out of the database.

Multiple tables carry some risks. The complexity of your SQL queries increases exponentially with the number of tables that you use. If you make a mistake when querying multiple tables, it can sometimes be very costly in terms of computer time.

I don't want to scare anyone away with these cautionary statements. Do proceed carefully, however.



If you have multiple tables in a database, you need some practical way to get the names of all these tables. If you are fortunate, someone will provide those names in the documentation of the database. Sometimes, though, you have to explore on your own.

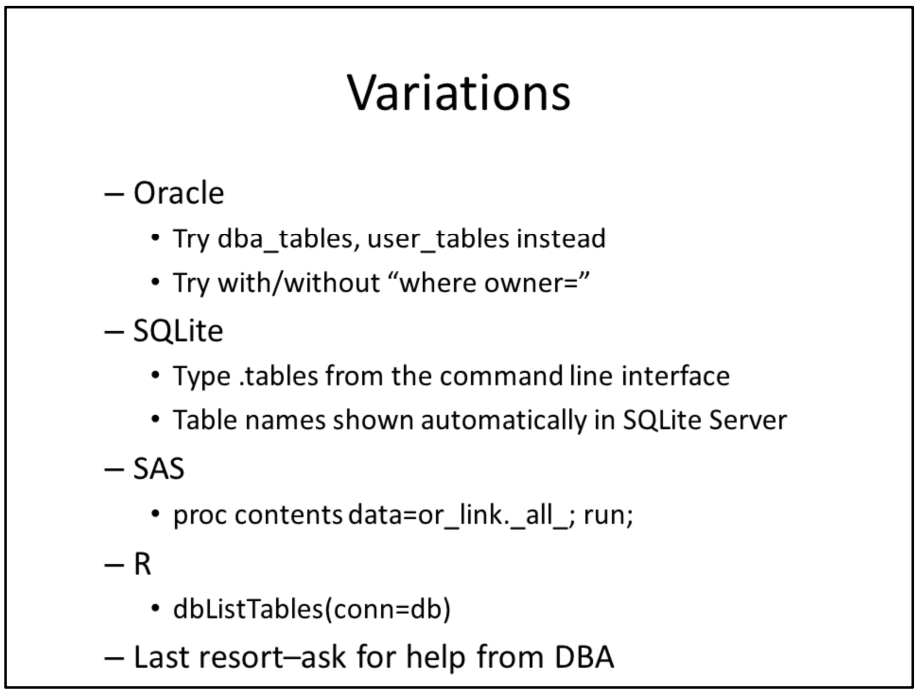
The syntax for listing the names of all the tables in your database varies markedly from one database system to another. This is unfortunate, but inevitable. The names of all your tables appears in the metadata of a database. Metadata means data about data. Every SQL database stores metadata, information about itself, in a proprietary format. SQL itself is an inexact standard, but the internal structure of a SQL database is not even a standard and there is no commonality across any of the databases. The other thing is that you may not have access to some of the internal structure of a database because of security and privacy concerns.

In Oracle, you can find the names of all the tables in several places, but the table named (quite logically) all_tables is as good a place as any. It's probably a smart idea to limit the search by the owner of the table, as there are lots and lots of internal tables that you would not be interested in. Every table created by Suman Sahil has an owner name of EHR (note here that capitalization is important). Tables created by Steve Simon have an owner name of SIMONS.

The all_tables table is not owned by Suman Sahil or Steve Simon and in SAS this is important. You need to specify schema='SYS' if you are querying Oracle from SAS. It turns out that this is not an issue with R.

In SQLite, the listing of the names of all the tables appears in the sqlite_master table. There are no owners in SQLite. If you have possession of a SQLite file, then you own every table inside that file.

Notice also that not only is there inconsistency between all_tables and sqlite_master, there is inconsistency between table_name and tbl_name. It gets worse when you look at other databases like MySQL and SQL Server. There is almost no consistency in how you access table names when you switch from one database to another.



There are a couple of other tables that might tell you the names of all the tables. Try dba_tables or user_tables. Try running your query without the “where owner=” restriction. It will produce way too many tables, of course, but better too many than too few.

In SQLite, there are some internal commands that you can use. These are not SQL commands, and they don’t work from inside SAS or R. The .table command, run on the command line interface, should work. If you use SQLite Server, that will show you the table names as part of the graphical user interface.

SAS has a special procedure, proc contents, that works with SQL databases. R has a built-in function, dbListTables, that does the same. Both SAS and R will give you every table and you might get lost looking for your tables in the sea of internal tables.

If you have trouble with our databases, come and talk to us, of course. Out in the real world, see if you can get help from the database administrator.

List information about a specific table

– Oracle syntax (use schema='SYS')

```
select column_name
  from all_tab_columns
  where table_name='your-table-name'
```

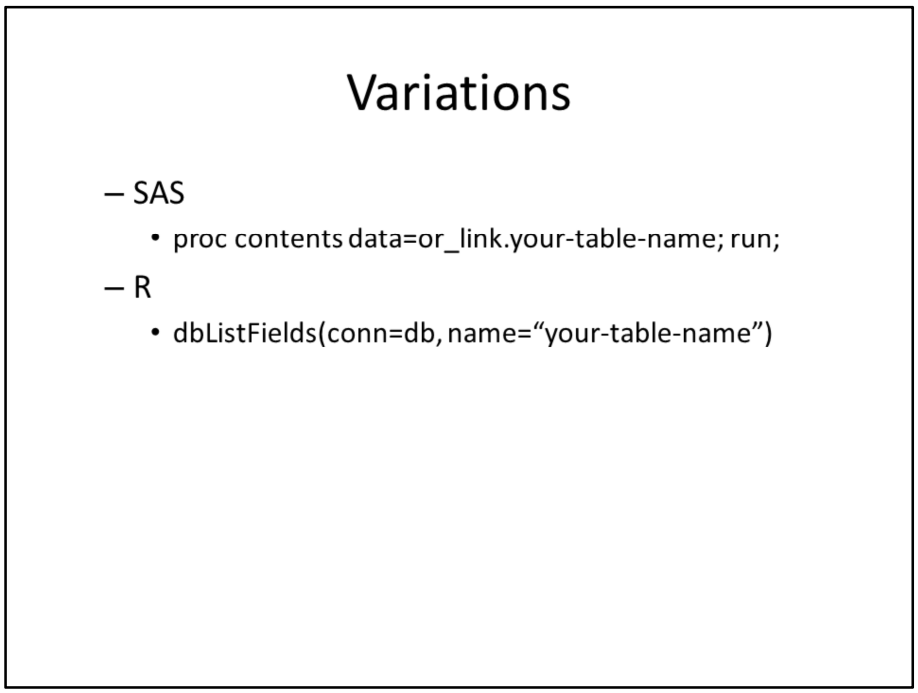
– SQLite syntax

```
pragma table_info(your-table-name)
```

Once you know the name of a table, you might need information about all the fields (variables) in that table. I typically get this by printing out the first five rows, but there is an official way to do this.

In Oracle, you find information in the all_tab_columns table. Remember that in SAS, you won't find this table unless you specify schema='SYS' earlier.

In SQLite, you need to use the pragma command.



SAS and R have special commands that will list field names as well. In SAS, use proc contents and specify the SQL table using the data= option.

In R, use the dbListFields function.

Summary

- Most real-world databases have multiple tables.
- Get the table names using
  - all_tables (Oracle)
  - sqlite_master (SQLite)
- Get information about an individual table using
  - all_tab_columns (Oracle)
  - pragma table_info (SQLite)

Here's a quick review of what you just learned. Most databases have multiple tables and getting information about these tables varies quite a bit from one version of SQL to another.

Your homework

- Pick a database (any database)
  - Use one of the approaches shown above to list all the table names.
- Pick a table (any table)
  - Use one of the approaches shown above to list all the field names.
- Do an Internet search on a database other than Oracle and SQLite.
  - Document how you get a list of all the table names in that database.
- Submit your results in a single PDF file.

This homework is a bit vague, but I just want you to try one or more of the approaches shown above. If one approach does not work, feel free to try another. I just want to get you comfortable with exploring the metadata inside a database.