Databases

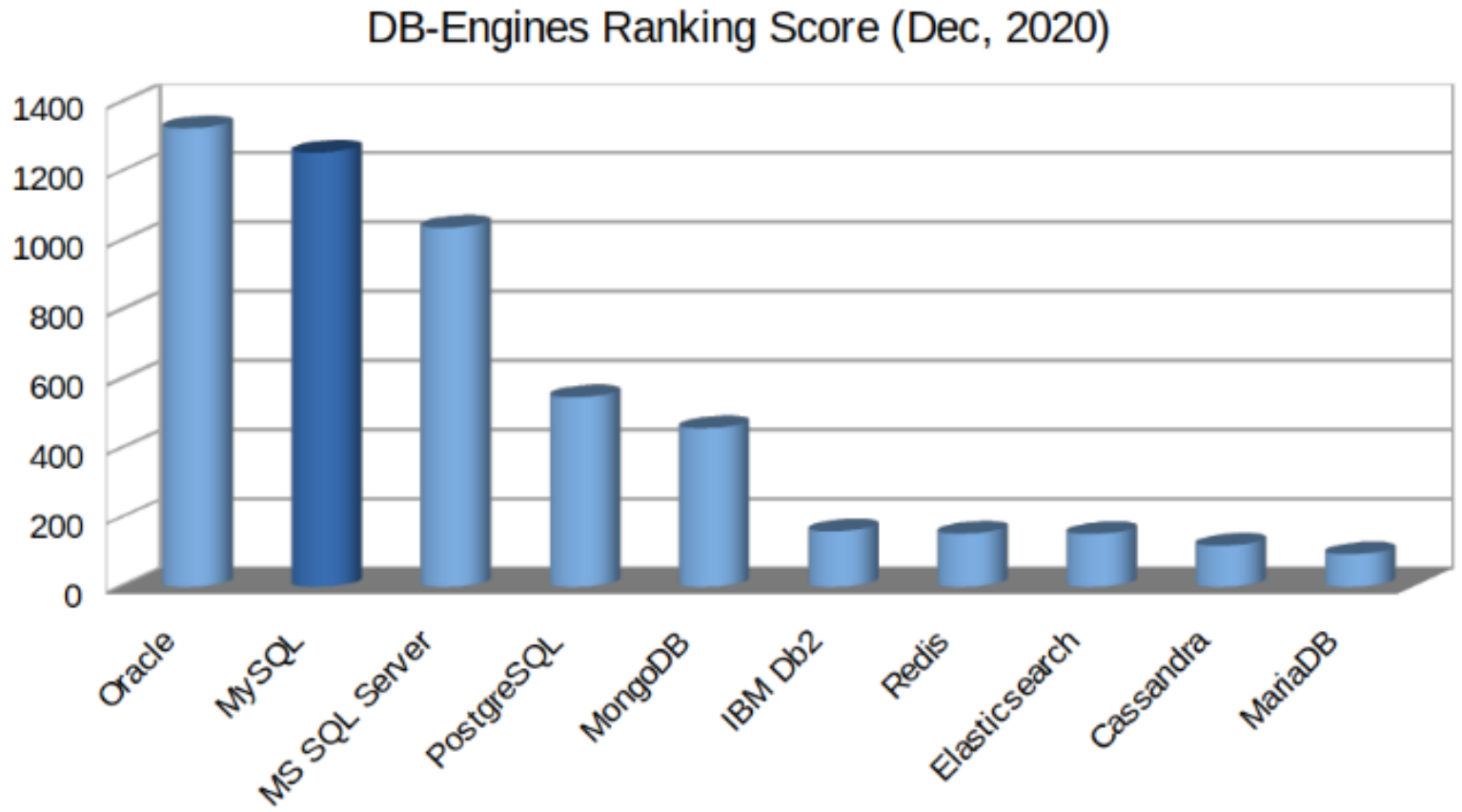
An Advanced Introduction to Unix/C Programming

John Dempsey
COMP-232 Programming Languages
California State University, Channel Islands

What percent of all software written interfaces with a database?

Database Popularity

There are 415 databases ranked at https://db-engines.com/en/ranking in 2023.



Top 25 Databases July 2024

421 systems in ranking, July 2024

Rank			DBMS	Database Model	Score		
Jul 2024	Jun 2024	Jul 2023			Jul 2024	Jun 2024	Jul 2023
1.	1.	1.	Oracle	Relational, Multi-model	1240.37	-3.72	-15.64
2.	2.	2.	MySQL	Relational, Multi-model	1039.46	-21.89	-110.89
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	807.65	-13.91	-113.95
4.	4.	4.	PostgreSQL	Relational, Multi-model	638.91	+2.66	+21.08
5.	5.	5.	MongoDB	Document, Multi-model	429.83	+8.75	-5.67
6.	6.	6.	Redis	Key-value, Multi-model	156.77	+0.82	-7.00
7.	↑ 8.	↑ 11.	Snowflake	Relational	136.53	+6.17	+18.84
8.	↓ 7.	8.	Elasticsearch	Search engine, Multi-model	130.82	-2.01	-8.77
9.	9.	↓ 7.	IBM Db2	Relational, Multi-model	124.40	-1.50	-15.41
10.	10.	10.	SQLite	Relational	109.95	-1.46	-20.25
11.	11.	↓ 9.	Microsoft Access	Relational	100.63	-0.53	-30.09
12.	12.	12.	Apache Cassandra	Wide column, Multi-model	99.13	+0.30	-7.40
13.	↑ 14.	↑ 14.	Splunk	Search engine	92.92	+3.82	+5.80
14.	↓ 13.	↓ 13.	MariaDB	Relational, Multi-model	90.58	-0.45	-5.52
15.	15.	↑ 18.	Databricks	Multi-model	83.29	+2.21	+14.83
16.	16.	↓ 15.	Microsoft Azure SQL Database	Relational, Multi-model	76.75	-0.03	-2.21
17.	17.	↓ 16.	Amazon DynamoDB	Multi-model	70.95	-3.50	-7.86
18.	↑ 19.	↑ 20.	Google BigQuery	Relational	57.82	-0.28	+2.40
19.	↓ 18.	↓ 17.	Apache Hive	Relational	57.29	-2.46	-15.58
20.	20.	↑ 21.	FileMaker	Relational	48.59	+0.68	-4.73
21.	21.	↑ 22.	Neo4j	Graph	45.75	+0.86	-6.31
22.	22.	↓ 19.	Teradata	Relational, Multi-model	44.52	-0.35	-15.73
23.	23.	23.	SAP HANA	Relational, Multi-model	44.12	-0.15	-6.60
24.	24.	24.	Apache Solr	Search engine, Multi-model	38.88	-2.15	-9.68
25.	25.	25.	SAP Adaptive Server	Relational, Multi-model	34.74	-0.34	-8.13

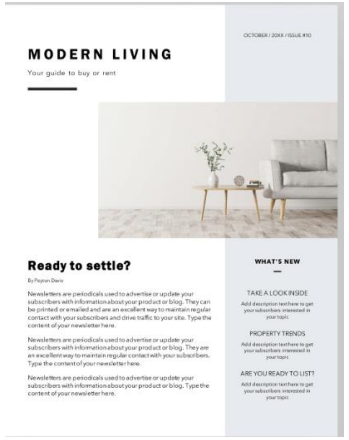
Two Types of Data: Structured and Unstructured Data

Structured data can be found in fixed or variable length fields in a table or spreadsheet.

First Name	Last Name	Student Id	Sex	Date of Birth
Brian	Becerra	123456	M	01/22/2004
John	Chavez	444444	M	09/05/2003
Jasmine	Torrez	738372	F	07/20/2004

Unstructured data can be any data like images, newsletters, emails, files, videos, text messages.

```
{
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]
}
```



3 Types of Databases

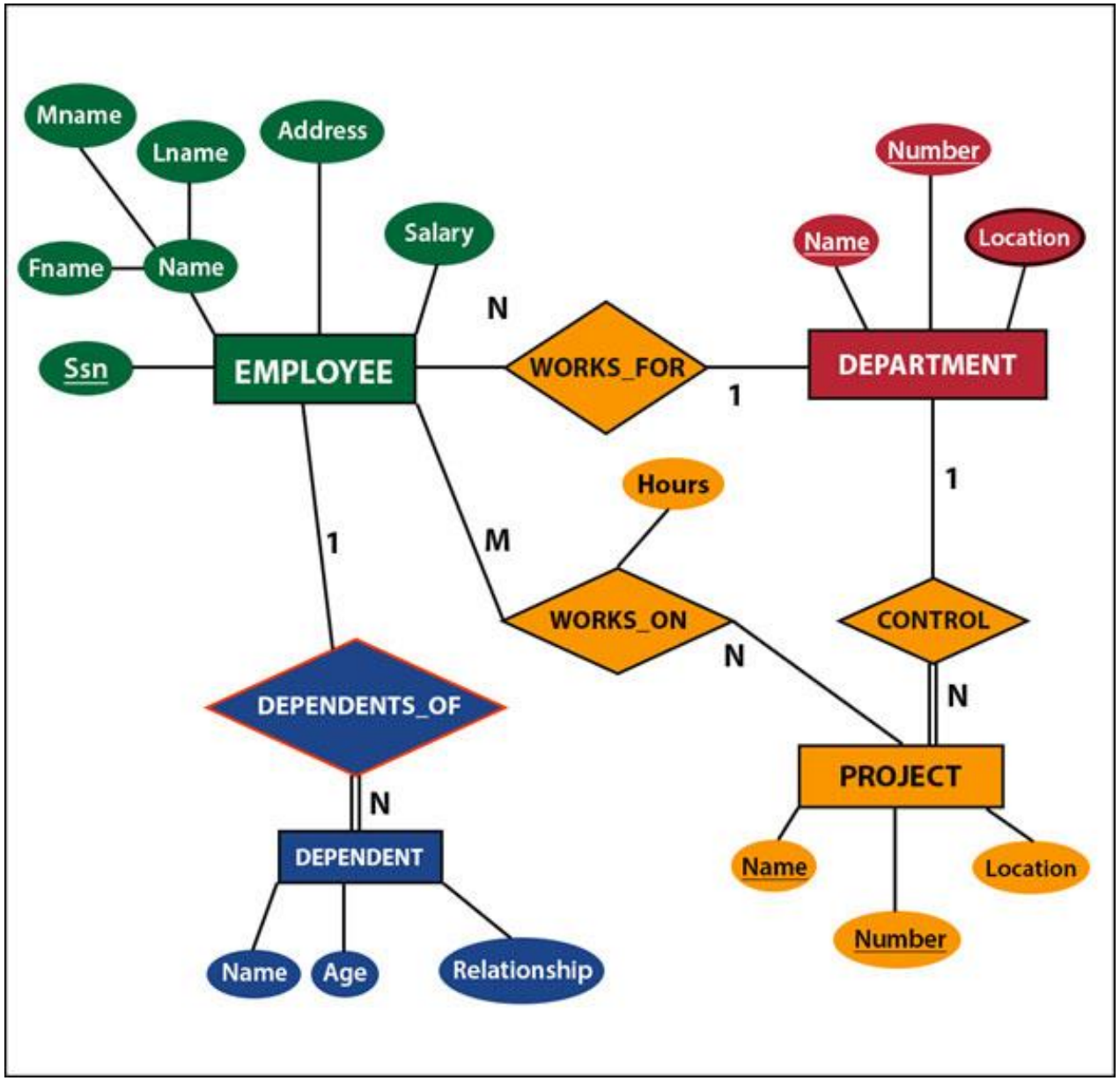
- Database
- Data Warehouse
- Data Lake

Database

- Online Transaction Processing (OLTP)
- Typical transactions are completed in less than a second.
- Database is used to read and write data for real-time applications.
- Data is stored in tables.
- Tables are defined by rows and columns.
- Schemas can be drawn to help design and understand the database relationships.
- ER (Entity/Relationship) Diagrams can be used to help define schemas.

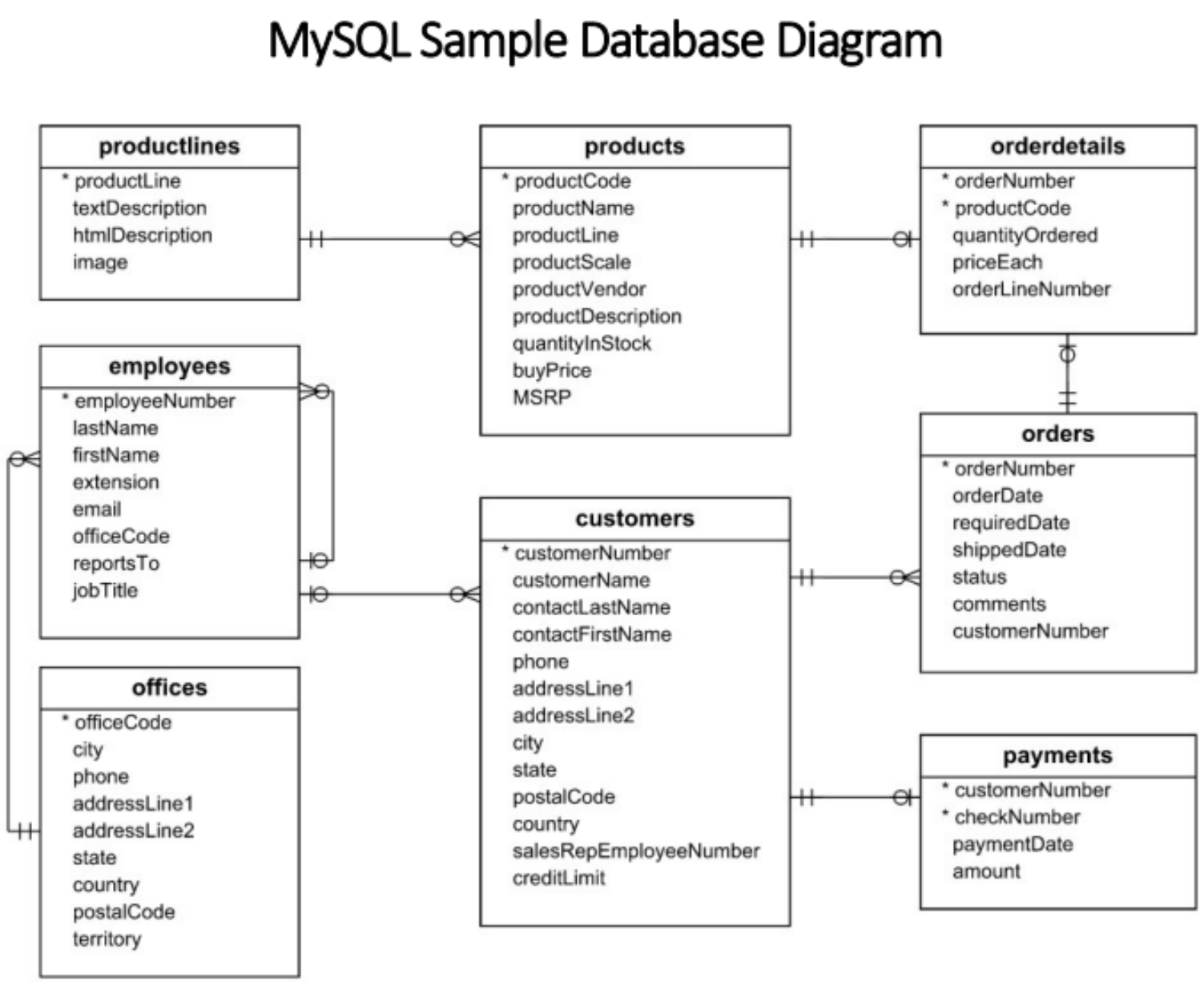
ER Diagrams

ER Diagrams help identify entities, attributes, and relationships for a given application. "Ok."



Schema

Based on the entities, attributes, and relationships in an ER Diagram, you can create a schema showing tables and relationships.



Data Warehouses

- Online Analytical Processing (OLAP)
- Can store years of historical data depending on contract requirements.
- Data can be partitioned and stored by date in the database, e.g., October 2023 partition.
- Data is loaded using ETL (Extract, Translate, and Load) tools, from multiple databases, and/or from a mainframe.
- Data Warehousing databases can run long running queries and used for data analytics and data visualization.

Data Lake

- Centralized storage of structured and unstructured raw data, like images, text files, text messages, videos, newsletters.
- Storing data in a data lake is similar to storing data on a disk drive.
- Schema can be defined when reading the data.
- Used for long term storage.

Database Block Size

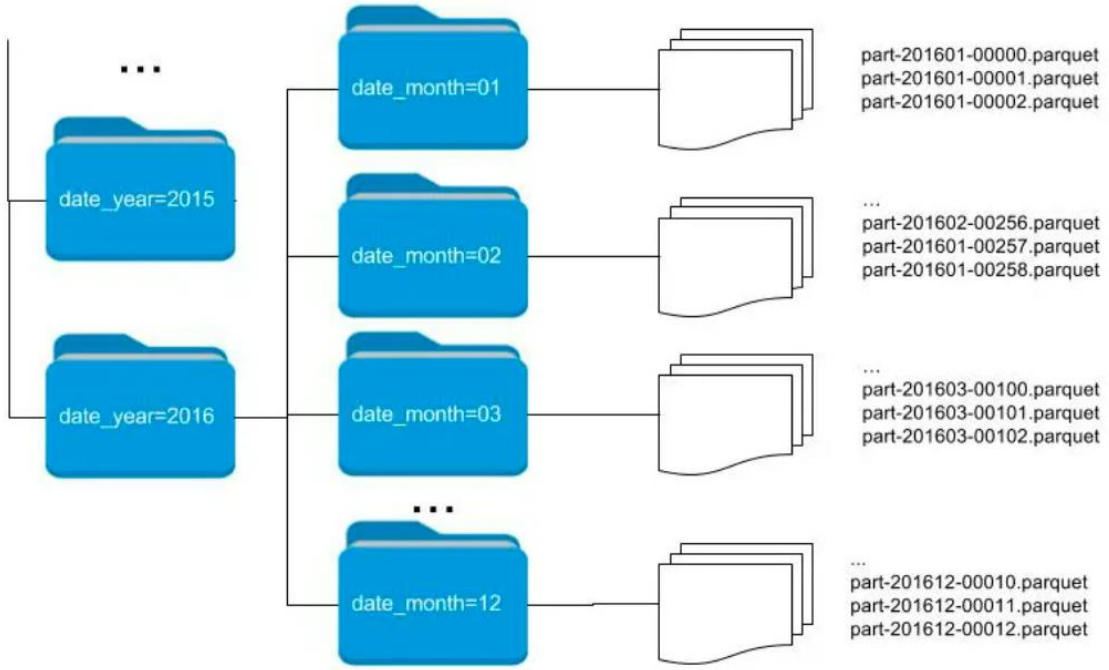
For real-time databases, the block size is usually 8,192 bytes to match the block size of the file system.

For data warehouses, the database block size is usually 16k, 32k, or 64k.

Why is the block size so important?

Database Partitioning Based On Date

transaction_id	user	credit_number	amount	t_date	purchase_type	description
34100220501	Brad Pitt	1111-2222-3333-4444	54.95	01-02-2016	Food	Restaurant
34100220502	Angelina Jolie	1111-1111-1111-0000	200	01-03-2016	Services	Amazon
...	...	...	...	...	...	...
5000220100	Kim Kardashian	5566-2222-3333-4444	60	02-04-2016	Sports	Basketball match



Case 1. Creating a table defining the partitioning keys, year and month.

```
CREATE TABLE MYSCHEMA.TRANSACTIONS_PARTITIONED (
TRANSACTION_ID BIGINT , USER_ID STRING, CREDIT_NUMBER STRING,
T_DATE DATE, PURCHASE_TYPE STRING, DESCRIPTION STRING)
PARTITIONED BY (DATE_YEAR INT, DATE_MONTH INT);
```

Case 2. Loading data into table from an existing one.

```
INSERT OVERWRITE TABLE MYSCHEMA.TRANSACTIONS_PARTITIONED
PARTITION(DATE_YEAR, DATE_MONTH)
SELECT *,YEAR(TR.T_DATE) AS DATE_YEAR, MONTH(TR.T_DATE) AS DATE_MONTH
FROM MYSCHEMA.TRANSACTIONS TR;
```

https://www.datio.com/iaas/understanding-the-data-partitioning-technique/

SQL Is Standardized

How do you access the database?

The **Structured Query Language (SQL)** is used to query a database from a command line and is standardized.

But sometimes vendors will “enhance” the SQL they use, but in doing so they are no longer standardized. In turn, they typically lock the customer into their database.

Four Main Database Operations

- **SELECT**

```
select last_name, city, state from recipient where id = '123456';
```

- **INSERT**

```
insert into income_levels(income, ranking)
    values ('100000', 'high');
```

- **DELETE**

```
delete from transactions where date_rcvd < '2020-OCT-01';
```

- **UPDATE**

```
update agent_tbl set dre='123456789' where agent_id = '1000';
```

Four Main Database Operations

sql> **insert into customer(customer_id, first_name, last_name, city, state)**
values('123456', 'John', 'Smith', 'Camarillo', 'CA');

sql> **update customer set city = 'Los Angeles', state='CA', zip = '90001'**
where customer_id = 123456;

sql> **select * from customer where city = 'Oxnard' and state = 'CA';**

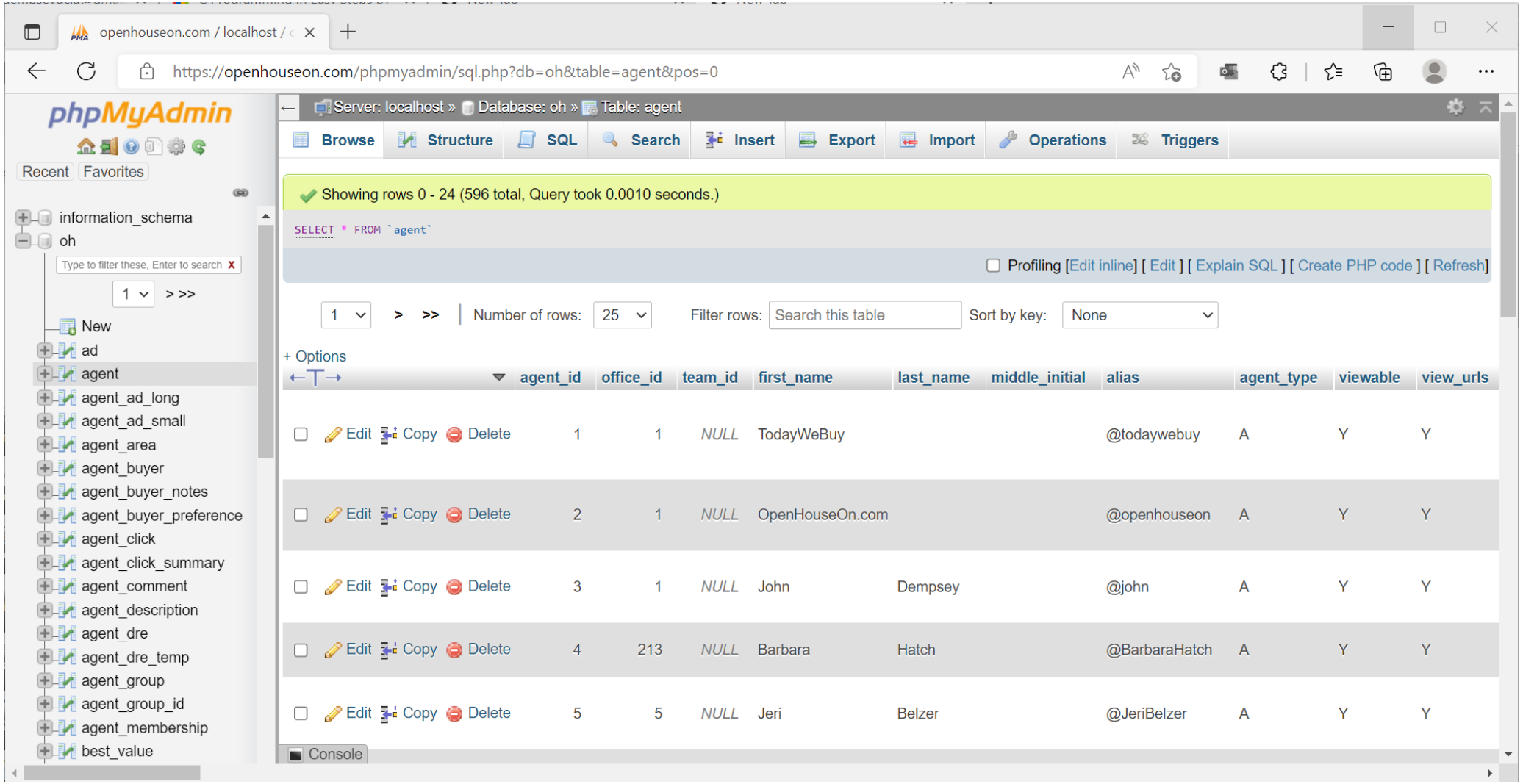
sql> **delete from customer where customer_id = '123456';**

Other Common Database Commands

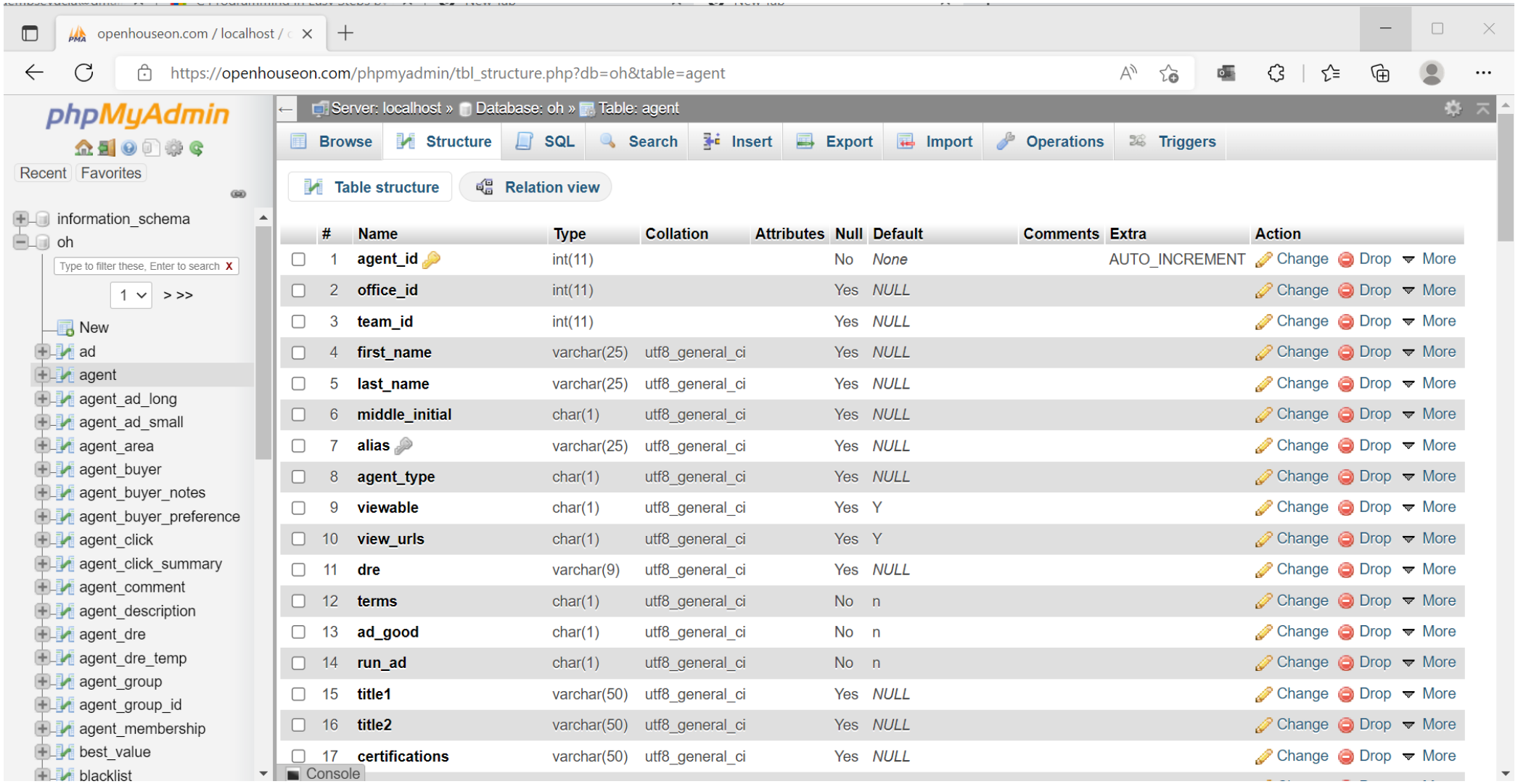
- CREATE TABLE
- CREATE INDEX
- TRUNCATE TABLE
- DROP TABLE

There are a lot of database commands each supporting a lot of options.

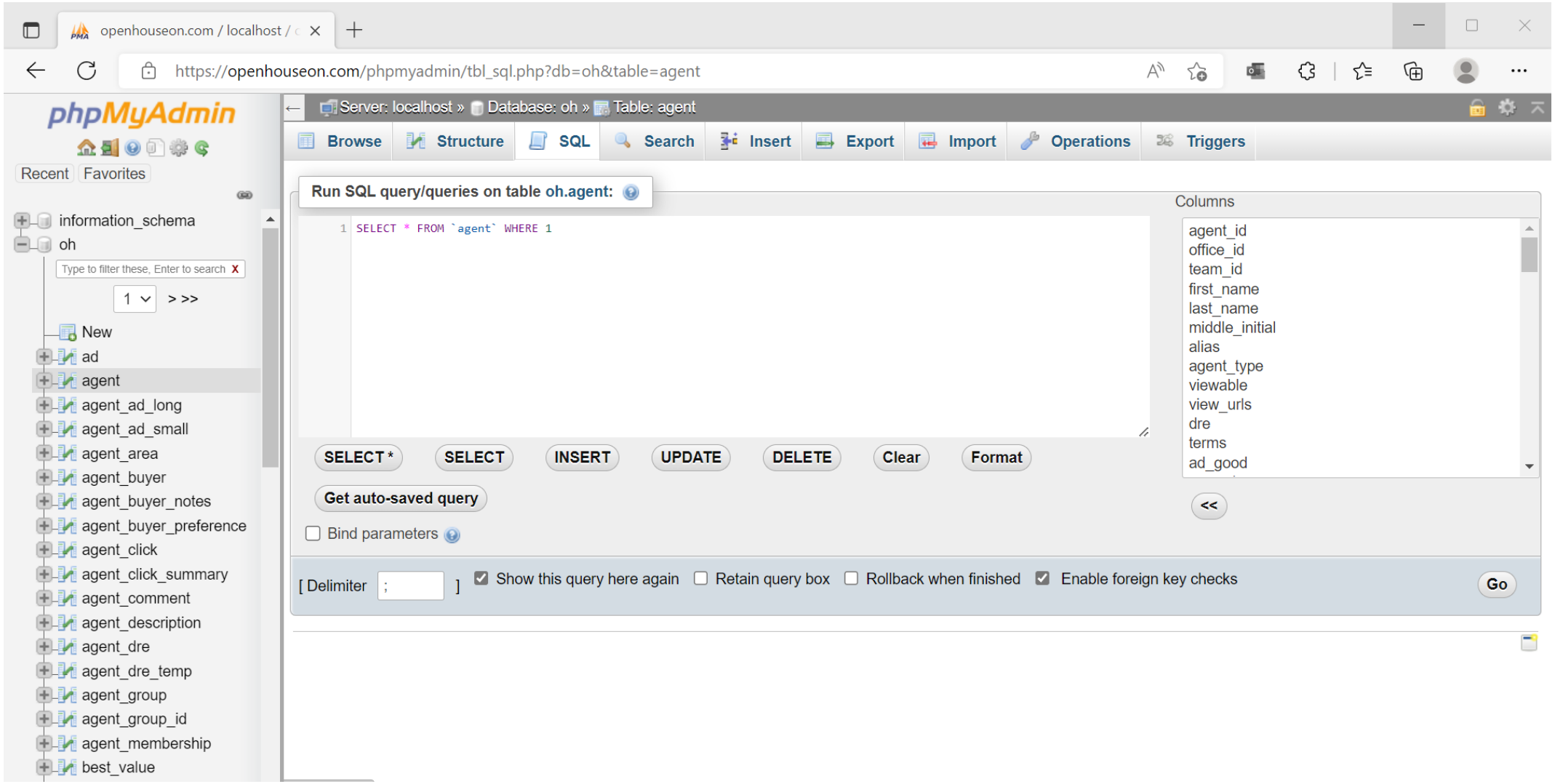
phpMyAdmin Database Interface



phpMyAdmin Database Interface



phpMyAdmin Database Interface



Database Calls

SQL is standardized, but how programs interface with a database is not!!!

What does this mean?

Once an application is written using a database's API (Application Programming Interface), the company for the most part is locked into the database vendor for the life of the software.

Oracle Database – Pro*C

Oracle uses a pre-processor called **proc** to convert a .pc file into a .c file. The size of the .c file is significantly larger.

Oracle C Code Example

```
#include <stdio.h>
EXEC SQL BEGIN DECLARE SECTION;
varchar   first_name_db[20];
varchar   last_name_db[30];
varchar   ssn_db[10];
EXEC SQL END DECLARE SECTION;

#define SQLCA_STORAGE_CLASS extern;

EXEC SQL INCLUDE sqlca.h;

strcpy(first_name_db, “Brent”);
strcpy(last_name_db, “Lin”);
strcpy(ssn_db, “123-45-6789”);

EXEC SQL
     INSERT INTO RECIPIENT(first_name, last_name, ssn_db) VALUES(:first_name, :last_name_db, :ssn_db);

EXEC SQL COMMIT WORK;
```

Oracle C Code SELECT Example

```
#include <stdio.h>
void select_recipient(char *ssn)
{
EXEC SQL BEGIN DECLARE SECTION;
varchar   db_first_name[20];
varchar   db_last_name[30];
varchar   db_ssn[10];
EXEC SQL END DECLARE SECTION;
#define SQLCA_STORAGE_CLASS extern;
EXEC SQL INCLUDE sqlca.h;
EXEC SQL
            SELECT first_name, last_name, ssn
            INTO :db_first_name, :db_last_name, :db_ssn
            FROM RECIPIENT WHERE ssn = :ssn;

if (sqlca.sqlcode == NO_DATA_FOUND)
    printf("No data found for %s.\n", ssn);

printf("First name = %s\n", db_first_name);
printf("Last name = %s\n", db_last_name);
}
```

SQL Server Database Example

```
SQLServerConnection Conn;
try
{
Conn = new SQLServerConnection("host=nc-star;port=1433;
User ID=test01;Password=test01; Database Name=Test");
Conn.Open();
Console.WriteLine ("Connection successful!");
}
catch (Exception ex)
{
// Connection failed
Console.WriteLine(ex.Message);
return;
}
try
{
// Create a SQL command
string strSQL = "SELECT ename FROM emp
                 WHERE sal>50000";
```

```
SQLServerCommand DBCmd
= new SQLServerCommand(strSQL, Conn);

SQLServerDataReader myDataReader;
myDataReader = DBCmd.ExecuteReader();
while (myDataReader.Read())
{
    Console.WriteLine("High salaries: " +
                      myDataReader["ename"].ToString());
}
myDataReader.Close();
// Close the connection
Conn.Close();
}
catch (Exception ex)
{
Console.WriteLine(ex.Message);
return;
}
```

MySQL C Code Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mysql/mysql.h>

int main(int argc, char **argv)
{
      MYSQL             *con;
      int                count;
      int                number_of_rows;
      char               query[1000];
      MYSQL_RES         *result;
      MYSQL_ROW          row;

      con = mysql_init(NULL);
      if (con == NULL) {
         fprintf(stderr, "mysql_init() failed.\n");
        exit(1);
      }
```

```
  if (mysql_real_connect(con, "comp232.com",
          "john", "password", "classicmodels", 0
          NULL, 0) == NULL) {
       fprintf(stderr, "%s\n", mysql_error(con));
       mysql_close(con);
       exit(1);
     }
     printf("The connection is open.\n");

// Query

      strcpy(query,
        "SELECT officeCode, city, state FROM offices");

      if (mysql_query(con, query) != 0)       {
        printf("Query failed.\n");
        printf("%s\n", mysql_error(con));
        exit(1);
      }

      result = mysql_store_result(con);

      printf("Number of rows: %ld\n",
              (long) mysql_num_rows(result));
```

```
      count = 1;
      while((row = mysql_fetch_row(result)) != NULL) {
         printf("%2d. ", count);
         printf("officeCode = %s, ",    row[0]);
         printf("city = %-15s, ",       row[1]);
         printf("state = %s\n",         row[2]);
         count = count + 1;
       }
       printf("\n\n");

       mysql_free_result(result);

       mysql_close(con);

       printf("The connection is closed.\n");
}
```