CGS 3066: Spring 2015
SQL Reference

Can also be used as a study guide.
Only covers topics discussed in class. This is by no means a complete guide to SQL.

1 Databases

A database is a repository of information, where data is stored according to a particular model. Databases make the storage and retrieval of data easy and safe. Without databases, we'd have to store data in memory, which is impractical due to volatility of memory and the ever growing amount of data; or we'd have to store data in text files, which, while being persistent storage, makes the retrieval of data onerous.

A database is an organized collection of data. The data is typically organized to model aspects of reality in a way that supports the processes requiring the information. Databases are supported by Database Management Systems.

2 Database Management System (DBMS)

Database management systems are computer software applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. different DBMS' satisfy different properties and requirements. DBMS' are classified in different ways, including relational and object-oriented, SQL and NoSQL, etc. Some of the popular DBMS are

- Oracle: Object-relational database management system produced and marketed by Oracle Corporation.
- DB2: Family of database server products developed by IBM. These products all support the relational model. Some products have been extended to support object-relational features and non-relational structures.
- SQL Server: Relational database management system developed by Microsoft. There are at least a dozen different editions of Microsoft SQL Server aimed at different audiences.
- PostgreSQL: object-relational database management system (ORDBMS) with an emphasis on extensibility and standards-compliance.
- MySQL: The world's second most widely used relational database management system and most widely used open-source RDBMS.

3 Structured Query Language

SQL is the standard language for querying and manipulating data. Queries are commands given to the database to carry out certain actions. Most of the time, queries are questions about the data that SQL generates responses to after looking at the database.

- Initially developed at IBM byDonald ChamberlinandRaymond Boycein the early 1970s.
- A very-high-level language, in which programmers are able to avoid specifying a lot of data-manipulation details that would be necessary in languages like C++.
- Its queries are optimized quite well, yielding efficient query executions.
- Many standards out there: SQL92, SQL3, SQL99, SQL 2011 ...
- Vendors support various subsets of these standards.

3.1 Sub-Languages

SQL has 3 sub-languages.

- Data Definition Language (DDL): Used to define and modify the database schema. Schema refers to the structure of the database.
- Data Manipulation Language (DML): Used to insert, update, delete and query the data in the tables.
- Data Control Language (DCL): Used to control access to the data. Mainly for maintaining users.

4 Some Terminology

This is some common SQL terminology.

- Table: A database most often contains one or more tables. Each table is identified by a name. Tables contain records (rows) with data.
- Field: A column in a table.
- Record/Tuple: A row in a table.
- Key: A group of field names that can uniquely identify a row in a table.
- Primary Key: A single field key that's chosen to be the key for a table.
- Foreign Key: A field in one table that acts as a primary key for another table.

5 Database Setup

This is the sample database setup we'll use for the rest of this document.

Database: Bank

Table: Accounts

Name	Social	Address	AccountNo	DateOfCreation	Type

Table: Transactions

TransID	DateofTransaction	AccountNo	Type	Amount	Balance

6 DDL Queries

This section outlines the queries used to set up the schema (or structure) of the database. Words in uppercase are SQL keywords.

6.1 Create Database

Used to create a database. A database contains all the tables that are used in a particular application. Very large applications might have several databases, each serving a particular purpose. For creating the database "Bank",

```
CREATE DATABASE Bank;
```

6.2 Create Table

Used to create a table in a database. Tables are organized into rows and columns; and each table must have a name. Columns must have a name and a datatype, with optional size and constraints. Constraints include uniqueness, primary key or foreign key constraints, etc. To set up the 2 tables above, we use the two queries below.

```
CREATE TABLE Accounts
(
        Name                  VARCHAR(50),
        Social                  INTEGER(10) UNIQUE,
        Address                  VARCHAR(200),
        AccountNo          INTEGER(14) PRIMARY KEY,
        DateOfCreation DATE,
        Type                  VARCHAR(10)
);

CREATE TABLE Transactions
(
        TransID                 INTEGER(20) PRIMARY KEY,
        DateOfTransaction        DATE,
        AccountNo        INTEGER(14) FOREIGN KEY REFERENCES Accounts(AccountNo),
        Type                VARCHAR(10),
        Amount                FLOAT,
        Balance                FLOAT
);
```

6.3 Alter Table

The alter table query is used to add or remove columns from existing tables. While adding columns, we specify the column name and its datatype. If we want to add a column "Balance" to the Accounts tables, we use the following query.

```
ALTER TABLE Accounts ADD Balance FLOAT;
```

If we want to remove columns from the table, we just ask for the column to be dropped. If we decide that having "Balance" in the Accounts table doesn't make sense, we use the query,

```
ALTER TABLE Accounts DROP COLUMN Balance;
```

6.4 Truncate Table

This query is used to delete all the rows in a table. To remove all data from the Transactions table,

```
TRUNCATE TABLE Transactions;
```

6.5 Drop Table

This query is used to delete a table. A table can be deleted only if it is empty.

```
DROP TABLE Transactions;
```

7 DML Queries

This section describes te queries used to manipulate the data in the tables. Ideally, DDL queries are executed only once, during the initial set up of the application. DML queries are used in everyday interactions with the database. Words in uppercase are SQL keywords. SQL is not case sensitive, but it is convention to write SQL keywords in uppercase.

7.1 Insert

This statement is used to add records into the table. There are 2 modes for the insert statement. If we know the order of the columns in the table, we need not specify column names in the insert statement.

```
INSERT INTO Accounts VALUES ('John Smith', 1234567890, '1, Blue Road,
Greenville, MA 12345', 2340000000057, '2015-03-25', 'Savings');
```

If we don't know the order, or if we don't have values for every field, we can specify the colums where we want to insert data.

```
INSERT INTO Accounts (Name, Social, AccountNo, DateOfCreation, Type)
VALUES ('John Smith', 1234567890, 2340000000057, '2015-03-25', 'Savings');
```

7.2 Update

This query is used to change data in particular fields. Basically, it's used to update existing data. When update is used without the WHERE clause, it will change the value for that field for ALL rows in the table. The query

```
UPDATE Transactions SET Type='Deposit';
```

will change the transaction type of every transaction ever made to 'Deposit'. If we don't want this to happen, we should use the WHERE clause. If we want to affect only one row, we would use the primary key in the WHERE clause. It is also possible to use the WHERE clause with a range to affect more than one row.

```
UPDATE Transactions SET Type='Deposit' WHERE TransID=10003451890000000072;
```

7.3 Delete

We can also delete rows from a table. Delete queries are usually used with WHERE clauses. If we don't specify a WHERE clause, we'll end up deleting ALL the rows in the table. To remove all checking accounts from the Accounts table,

```
DELETE FROM Accounts WHERE Type='Checking';
```

All the queries encountered so far would return only success or failure. If the query was executed successfully, it will also mention how many rows were affected. The SELECT query, on the other hand returns a table upon successful execution.

7.4 Select

The Select statement is used to select data from a database. The result is stored in a result table, called the result-set. The order of rows in the result-set is To Select ALL rows and columns from a table,

```
SELECT * FROM Accounts;
```

If we want all the rows, but only a few columns, we can specify the columns we want.

```
SELECT Name, AccountNo, Type FROM Accounts;
```

The WHERE clause is used to extract only those records that fulfill a specified criterion.

```
SELECT AccountNo, Amount, Type, Balance FROM Transactions
WHERE DateOfTransaction = '2015-03-26';
```

The AND & OR operators are used to filter records based on more than one condition.

```
SELECT AccountNo, Amount, Type, Balance FROM Transactions
WHERE DateOfTransaction = '2015-03-26' AND Amount>100 OR Type='Withdrawal';
```

This query retrieves rows where the transaction took place on March 26 2015 with the amount being grater than $100 or if it was a withdrawal (that happened since the beginning of time for any amount).

The ORDER BY keyword is used to sort the result-set according to a particular field. By default, it's presented in ascending order. To sort the rows in descending order, we use the keyword DESC.

```
SELECT AccountNo, Amount, Type, Balance FROM Transactions
WHERE DateOfTransaction = '2015-03-26' AND Amount>100 OR Type='Withdrawal'
ORDER BY AccountNo DESC;
```

7.5 Nested Queries

It is possible to have Select queries inside Select queries. The inner query is executed first, and the outer query is executed on the result of the inner query. Nested queries can be used as an alternative to joins. The inner query and the outer query must be matched by a foreign key - primary key pair. Also, instead of using relational operators, we use the IN keyword. For example, if we wanted to print the names of account holders who made a transaction on March 30 2015, we would use the following query.

```
SELECT Name from Accounts WHERE AccountNo IN (
SELECT AccountNo from Transactions WHERE DateOfTransaction = '2015-03-30');
```

7.6 Aggregate Function

Aggregate functions are used to obtain certain statistics on the data. SQL aggregate functions return a single value, calculated from values in a column. For example,

```
SELECT Sum(Amount) FROM Transactions WHERE Type='Deposit'
AND DateOfTransaction = '2015-03-26';
```

Some of the commonly used aggregate functions are:

- AVG: Calculates average.
- SUM: Calculates sum.

- MAX: Calculates maximum value in a column.
- MIN: Calculates minimum value in a column.
- COUNT: Calculates number of rows.
- COUNT DISTINCT: Calculates number of distinct values in a column.

7.6.1 GROUP BY and HAVING

The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns. If we want to have all the deposit transactions grouped together, followed by all the withdrawal, we can use the GROUP BY clause.

```
SELECT TransID, AccountNo, Amount, Type FROM Transactions WHERE Balance >= 2500
GROUP BY Type;
```

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions. For example, if we want to display all the account numbers whose average transaction is $500 or higher, we use the following query.

```
SELECT  DISTINCT AccountNo FROM Transactions GROUP By AccountNo
HAVING AVG(Amount) > 500;
```