Structured Query Language (SQL)

- SQL is a database language for:
  - creating database and table structures
  - performing data manipulation and administration
  - querying the database to extract useful information

- It is a nonprocedural language
  - The user specifies what must be done, but not how
  - Where the data is actually stored isn't important
  - only the specified relations and relationships matter

Structured Query Language (SQL)

- SQL is an international standard
  - American National Standards Institute (ANSI)
  - International Organization for Standardization (ISO)
    • Consortium of 150 countries

- All relational DBMS software thus supports SQL
  - Oracle, MySQL, SQL Server, DB2, MS Access
  - Many database vendors have developed extensions
    • Basic, simple vocabulary of < 100 "words"
    • Different "dialects" with minor differences
History of SQL

• 1974: SEQUEL, a “skunkworks” language used by IBM programmers while trying to develop a RDBMS prototype
• 1976: SEQUEL/2
• 1979: Oracle introduces the first database that utilizes relational concepts and has an SQL implementation
• 1981: SQL/DS – IBM’s first relational database product, later to become DB2
• 1982: ANSI and ISO propose a standard
• 1992: SQL-92 standard in place
  - Most database vendors largely have it in place
• 1999: SQL-99 ratified
• 2011: SQL-2011
• 2016: SQL-2016 – most recent version (ANSI/ISO standard)

Categories of SQL commands

• Data Definition Language (DDL)
  - Commands that define a database, including creating, altering, and dropping tables and stored procedures, and establishing constraints
    • CREATE TABLE
    • set PRIMARY KEY

• Data Manipulation Language (DML)
  - Commands that are used to manipulate data and extract information
    • SELECT, UPDATE, INSERT, DELETE

Data Definition Language (DDL)

• Primary commands:
  • CREATE TABLE / CREATE PROCEDURE
  • ALTER TABLE
    • Impacts attributes or constraints
      • ADD (new attribute, primary key, NULL, etc.)
      • MODIFY (data type, etc.)
      • DROP (column)
  • DROP TABLE  (DROP PROCEDURE)
Data Definition Language (DDL)

- **Example of syntax:**

```
CREATE TABLE tablename (column1 data type [constraint],
column2 data type [constraint]],
PRIMARY KEY (column1 [, column2]),
FOREIGN KEY (column1 [, column2]) REFERENCES tablename],
CONSTRAINT constraint);
```

SQL command keywords
- Users-provided parameters
  - anything inside square brackets is optional

Data Definition Language (DDL)

- **Example of command:**

```
CREATE TABLE PRODUCT (PROD_ID VARCHAR(10) NOT NULL UNIQUE,
PROD_DESC VARCHAR(35) NOT NULL,
PROD_PRICE DECIMAL(8,2) NOT NULL,
VEND_CODE INTEGER,
PRIMARY KEY (PROD_ID),
FOREIGN KEY (VEND_CODE) REFERENCES VENDOR
ON UPDATE CASCADE);
```

Entity integrity: not needed in MS Access if attribute is PK

Changes to VEND_CODE automatically applied: not supported in Oracle

Data types

- **ANSI/ISO SQL data types:**
  - INTEGER / SMALLINT
  - DECIMAL(precision, scale)
  - CHAR(n) - fixed length character data
  - VARCHAR(n) - variable length character data
  - DATE - Julian date format
  - plus several more…

- Other DBMS add additional data types, for example….
Data types (cont.)

MySQL data types (cont.)

• MySQL data types
    – Primary numeric types:
      • TINYINT -128 to 127 (or 0 to 255) = 1 byte
      • SMALLINT -32768 to 32767 (or 0 to ...) = 2 bytes
      • MEDIUMINT -8.39 × 10^6 to 8.39 × 10^6 = 3 bytes
      • INT -2.15 × 10^9 to 2.15 × 10^9 = 4 bytes
      • BIGINT -9.22 × 10^18 to 9.22 × 10^18 = 8 bytes
      • DECIMAL(M, D) M total digits / D digits after decimal

MySQL data types (cont.)

– Primary date and time types:
  • DATE "YYYY-MM-DD" format
    range: ‘1000-01-01’ to ‘9999-12-31’
  • DATETIME "YYYY-MM-DD HH:MM:SS" format
    range: ‘… 00:00:00’ to ‘… 23:59:59’

Invalid dates and times are converted to zero values: ‘0000-00-00’

Some built-in functions:
  NOW( ), CURDATE( ), DATEDIFF( ), INTERVAL, DATE( ), TIME( ), DAY( ), YEAR( ), MONTH( ), etc.

MySQL data types (cont.)

– Primary string types:
  • CHAR(n) - always allocates n bytes of storage
  • VARCHAR(n) - only allocates used space (plus 1 byte)
    Values that exceed n characters in length are truncated
  • BLOB - used for large binary strings of data
  • TEXT - used for large character strings of data
  • ENUM - string object with a value chosen from a list of permitted values that is specified at table creation time
In practice…

- Although they use SQL behind the scenes, database creation, alteration, etc. are typically handled using a user interface within a DBMS, rather than directly through SQL.

Examples:

Sample Database

- Posted on website: (2 versions – same set of relations)
  - *class.accdb* (MS Access)
  - *class.sql* (MySQL)
  - We will be running examples of queries on this database as we cover the material in class, to help illustrate how the different SQL statements work.
  - You may wish to have your laptop with you in class in order to actively participate and better understand these examples.

Sample Database

- 6 Tables:
Data Manipulation Language (DML)

- Primary commands:
  - Data retrieval ("standard" queries):
    - SELECT
  - Action queries:
    - INSERT
    - UPDATE
    - DELETE

SELECT statement

- General format:
  ```
  SELECT [DISTINCT | ALL] {* | column1 [AS new_name] [*, ...]}
  FROM table_name [alias] [, ...]
  [WHERE condition(s)]
  [GROUP BY column_list] [HAVING condition]
  [ORDER BY column_list]
  ```

SELECT: Topics to be covered

- SELECT (basic syntax) (for a single relation / table)
- Wildcard (*)
- WHERE clause (conditional filter)
  - Arithmetic operators (=, <, >, <=, >=)
  - Logical operators (AND, OR, NOT)
  - Clauses (BETWEEN, LIKE, IN)
- NULL values
SELECT: Topics to be covered (cont.)

- Ordering / Sorting
- Column aliases
- Date syntax
- Avoiding duplicates (DISTINCT)
- Aggregate functions

SELECT (basic syntax)

\[
\text{SELECT column\_name(s)} \\
\text{FROM table\_name}
\]

\textbf{Ex:} List the first name and last name (in that order) of each employee

SELECT with *

\texttt{*} can replace list of column names

\textbf{Ex:} List the full details of every employee (i.e., the entire table)
SELECT with WHERE clause

• Specify a filtering criteria:

```
SELECT column_name(s) | *
FROM table_name
WHERE criteria (is true)
```

**Ex:** List the first and last name of every employee with a credit limit less than $25

WHERE clause: operators

• Expressions in the WHERE clause can include:
  – Arithmetic operators (=, <, >, <=, =>)
  – Logical operators (AND, OR, NOT)
  – Clauses (BETWEEN, LIKE, IN)

WHERE clause: matching data types

• To make a comparison between a field (attribute) and a particular value, you have to use the correct syntax to indicate the data type

  Numeric (& currency): WHERE Emp_CreditLimit < 25
  Text: WHERE Emp_Lastname = 'Brown' (or... = "Brown")
  Date: WHERE Emp_HireDate >= '2007-01-01' (or... >= "2007-01-01")
WHERE clause: logical operators

- Combine instances of arithmetic comparisons into logical statements:
  …WHERE (W >= Y) AND (X < Z)
  …WHERE (X < 5) OR (X > 15)
  …WHERE NOT (X = 'Thomas')

**Ex:** List the credit limit of every employee hired after 1/5/1996 who has a department code of 'Mkt' or 'Sal'

WHERE clause: BETWEEN

- Used with logical operator AND to specify a range of values

…WHERE X BETWEEN Y AND Z

**Ex:** List the last name of every employee hired in calendar year 1998

WHERE clause: LIKE (with wildcards)

- Gives ability to do simple pattern matching
- Pattern always expressed as a string
- MS Access uses different characters than standard SQL
  MySQL: % matches zero or more characters (* in Access)
  MySQL: _ matches exactly one character (# in Access)

…WHERE Emp_LastName LIKE "Z%e"
…WHERE Emp_Phone LIKE "__9"

**Ex:** List the ID of every employee whose last name is four letters long and ends in 'e'
WHERE clause: IN

- Can be used to combine logical statements separated by ORs
  
  \[ \text{WHERE } (X = 'first') \text{ OR } (X = 'last') \]

\[ \text{WHERE } X \text{ IN ('first', 'last')} \]

WHERE clause: NULL values

- Identifies NULL values

\[ \text{WHERE } X \text{ IS NULL} \]

\[ \text{WHERE } X \text{ IS NOT NULL} \]

Ex: List all employees who do not have a manager

Ordering / sorting

- ORDER BY column1 [ASC | DESC] [, column2 [ ]][[, ...]]

- Can order by column names or by column indices
- Columns must have been SELECT-ed to be displayed

Ex: List all employee last names and dept codes from departments whose codes begin with 'S'. Sort the list in descending order of departmental code and ascending order of employee last name.
Column aliases

- Allows columns to be renamed
  - Fine-tuned output
  - Often used to assign name to computed column

SELECT X as 'new name', X * Y as product FROM...

**Ex:** List all employee last names together with twice their credit limit, where this second column is named "double credit"

---

SELECT DISTINCT

- Removes duplicate values

SELECT DISTINCT X FROM....

**Ex:** List all unique managers' IDs