Database Systems I

SQL Basics

Instructor: Ouldooz Baghban Karimi

CMPT 354 - Summer 2019
SQL

✓ Review
  • Create Table
  • Modify Table
  • Delete Table
  • Insert Data Into Table

• New
  • Single-table Queries
    • The SFW query
    • Useful operators: DISTINCT, ORDER BY, LIKE
    • Handle missing values: NULLs

• Multiple-table Queries
  • Foreign key constraints
  • Joins: basics
  • Joins: SQL semantics
  • Set Operations
SQL Basics Review

• **SQL: Structured Query Language**
  - Data Definition Language (DDL)
    The **Data-Definition** sublanguage for declaring database schemas
  - Data Manipulation Language (DML)
    The **Data-Manipulation** sublanguage for querying databases and for modifying the database

• **SQL**
  - Declarative language
  - Supported by all major commercial database systems
  - Standardized: many new features over time

• SQL makes a distinction between
  - **Stored relations** (**Tables**): Exists in DB and can be modified and queried
  - **Views** (Defined by Computation): Not stored but constructed when needed
  - **Temporary tables**: Constructed by SQL processor when it performs its job of executing queries and data modifications
SQL: Data Types

• Primitive Data Types
  • Character string of fixed of varying Length (CHAR(n), VARCHAR(n))
  • Bit Strings (BIT VARYING(n))
  • Boolean (BOOLEAN)
  • Integer (INT, INTEGER, SHORTINT)
  • Floating-point (FLOAT, REAL, DOUBLE PRECISION, DECIMAL, NUMERIC)
  • Date and Time (DATE, TIME)

**NOTE:** SQL keywords (create and table for example) are not case sensitive. Named objects (tables, columns etc.) may be.
SQL: Create

• To create a table use the **CREATE TABLE** statement
  
  • Specify the table name, field names and domains

```sql
CREATE TABLE Customer (  
sin CHAR(11),  
firstName CHAR(20),  
lastName CHAR(20),  
age INTEGER,  
income REAL  
)

CREATE TABLE Movies(  
title CHAR(100),  
year INT,  
length INT,  
genre CHAR(10),  
studioName CHAR(30),  
studioAddress CHAR(50)  
)
```
SQL: Modify Tables

• Modifying Relation Schemas
  • Delete a relation R: \texttt{DROP TABLE R;}
  
  • Modify Schema of relation R: \texttt{ALTER TABLE R}
    • \texttt{ADD} followed by an attribute name and its data type
    • \texttt{DROP} followed by an attribute name

• Example
  \begin{verbatim}
  ALTER TABLE Movie ADD producerCNum INT;
  ALTER TABLE Movie DROP studioAddress;
  \end{verbatim}
SQL: DEFAULT

• Default Values
  • The value used when no other value is known
  • Keyword **DEFAULT**, Value either **NULL** or a Constant
  • Example

    gender CHAR(1) DEFAULT '?',
    birthdate DATE DEFAULT '0000-00-00',

    **ALTER TABLE** MovieStar **ADD** phone CHAR(16) **DEFAULT** 'unlisted';
SQL: Keys

• Declaring Keys
  • Declare key when attribute listed in the relation schema
    • Can be used only when the key is a single attribute
  • Or add to the list of items declared in schema an additional declaration that an attribute or set of attributes from the key
    • This method should be used if the key consists of more than one attribute
  • Two Declarations
    • Either PRIMARY KEY
    • Or UNIQUE
  • Set of attributes $S$ key for relation $R$: Two tuples in $R$ cannot agree on all of the attributes in set $S$, unless one of them is NULL. Violating action rejected.
SQL: Keys

• Primary Key Example

```sql
CREATE TABLE Movies(
    title CHAR(100),
    year INT,
    length INT,
    genre CHAR(10),
    studioName CHAR(30),
    producerCNum INT,
    PRIMARY KEY (title, year)
);
```

```sql
CREATE TABLE Customer (
    sin CHAR(11),
    firstName CHAR(20),
    lastName CHAR(20),
    age INTEGER,
    income REAL,
    PRIMARY KEY (sin)
)
```
To insert a record into an existing table use the **INSERT** statement

- The list of column names is optional
  - If omitted, the values must be in the same order as the columns

```sql
INSERT INTO Customer(sin, firstName, lastName, age, income)
VALUES ('111', 'Sam', 'Spade', 23, 65234)
```
SQL: Modify Records

- Use the **UPDATE** statement to modify a record, or records, in a table
  - Note that the **WHERE** statement is evaluated *before* the **SET** statement
- Like **DELETE** the **WHERE** clause specifies which records are to be updated

```
UPDATE Customer
SET age = 37
WHERE sin = '111'
```
To delete a record use the **DELETE** statement

- The **WHERE** clause specifies the record(s) to be deleted

```
DELETE
FROM  Customer
WHERE  sin = '111'
```

Be careful, the following SQL query deletes *all* the records in a table

```
DELETE
FROM  Customer
```
SQL

• Review
  • Create Table
  • Modify Table
  • Delete Table
  • Insert Data Into Table

✓ New
  • Single-table Queries
    • The SFW query
    • Useful operators: DISTINCT, ORDER BY, LIKE
    • Handle missing values: NULLs

  • Multiple-table Queries
    • Foreign key constraints
    • Joins: basics
    • Joins: SQL semantics
    • Set Operations
## Sample Table

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>school</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Adam</td>
<td>SFU</td>
<td>23</td>
<td>3.2</td>
</tr>
<tr>
<td>1002</td>
<td>Aiden</td>
<td>UBC</td>
<td>19</td>
<td>3.5</td>
</tr>
<tr>
<td>1003</td>
<td>Alice</td>
<td>SFU</td>
<td>18</td>
<td>3.7</td>
</tr>
<tr>
<td>1004</td>
<td>Bob</td>
<td>UBC</td>
<td>22</td>
<td>3.1</td>
</tr>
<tr>
<td>1005</td>
<td>David</td>
<td>SFU</td>
<td>20</td>
<td>3.2</td>
</tr>
<tr>
<td>1006</td>
<td>John</td>
<td>SFU</td>
<td>21</td>
<td>3.1</td>
</tr>
<tr>
<td>1007</td>
<td>Mary</td>
<td>UBC</td>
<td>21</td>
<td>3.4</td>
</tr>
<tr>
<td>1008</td>
<td>Mike</td>
<td>SFU</td>
<td>24</td>
<td>3.1</td>
</tr>
<tr>
<td>1009</td>
<td>Sarah</td>
<td>UBC</td>
<td>18</td>
<td>3.0</td>
</tr>
</tbody>
</table>
The SFW Query

- **Select From Where**

- To write the query, ask yourself three questions:
  - Which **table** do you want information from?
  - Which **rows** do you want information from?
  - Which **columns** do you want information from?

```
SELECT <columns>
FROM <table name>
WHERE <conditions>
```
Columns

• Which **columns** do you want information from?

```sql
SELECT * FROM Students
```

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>school</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Adam</td>
<td>SFU</td>
<td>23</td>
<td>3.2</td>
</tr>
<tr>
<td>1002</td>
<td>Aiden</td>
<td>UBC</td>
<td>19</td>
<td>3.5</td>
</tr>
<tr>
<td>1003</td>
<td>Alice</td>
<td>SFU</td>
<td>18</td>
<td>3.7</td>
</tr>
<tr>
<td>1004</td>
<td>Bob</td>
<td>UBC</td>
<td>22</td>
<td>3.1</td>
</tr>
<tr>
<td>1005</td>
<td>David</td>
<td>SFU</td>
<td>20</td>
<td>3.2</td>
</tr>
<tr>
<td>1006</td>
<td>John</td>
<td>SFU</td>
<td>21</td>
<td>3.1</td>
</tr>
<tr>
<td>1007</td>
<td>Mary</td>
<td>UBC</td>
<td>21</td>
<td>3.4</td>
</tr>
<tr>
<td>1008</td>
<td>Mike</td>
<td>SFU</td>
<td>24</td>
<td>3.1</td>
</tr>
<tr>
<td>1009</td>
<td>Sarah</td>
<td>UBC</td>
<td>18</td>
<td>3.0</td>
</tr>
</tbody>
</table>
**Columns**

- Which **columns** do you want information from?

```sql
SELECT name, age
FROM Students
```

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>school</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Adam</td>
<td>SFU</td>
<td>23</td>
<td>3.2</td>
</tr>
<tr>
<td>1002</td>
<td>Aiden</td>
<td>UBC</td>
<td>19</td>
<td>3.5</td>
</tr>
<tr>
<td>1003</td>
<td>Alice</td>
<td>SFU</td>
<td>18</td>
<td>3.7</td>
</tr>
<tr>
<td>1004</td>
<td>Bob</td>
<td>UBC</td>
<td>22</td>
<td>3.1</td>
</tr>
<tr>
<td>1005</td>
<td>David</td>
<td>SFU</td>
<td>20</td>
<td>3.2</td>
</tr>
<tr>
<td>1006</td>
<td>John</td>
<td>SFU</td>
<td>21</td>
<td>3.1</td>
</tr>
<tr>
<td>1007</td>
<td>Mary</td>
<td>UBC</td>
<td>21</td>
<td>3.4</td>
</tr>
<tr>
<td>1008</td>
<td>Mike</td>
<td>SFU</td>
<td>24</td>
<td>3.1</td>
</tr>
<tr>
<td>1009</td>
<td>Sarah</td>
<td>UBC</td>
<td>18</td>
<td>3.0</td>
</tr>
</tbody>
</table>
Columns

• Which **columns** do you want information from?

```sql
SELECT name as studentName
FROM Students
```
Columns

- Which **columns** do you want information from?

\[
\text{SELECT } \text{age} \times 365 \text{ as ageDay}
\]
\[
\text{FROM Students}
\]
Conditions

• Which **rows** do you want information from?

```sql
WHERE gpa >= 3.5
```
Conditions

• Which **rows** do you want information from?

WHERE school='SFU'
AND gpa >= 3.5
Conditions

- Which **rows** do you want information from?

\[\text{WHERE } (\text{school}='\text{SFU}') \text{ OR } (\text{school}='\text{UBC}') \text{ AND } \text{gpa} \geq 3.5\]

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>school</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Adam</td>
<td>SFU</td>
<td>23</td>
<td>3.2</td>
</tr>
<tr>
<td>1002</td>
<td>Aiden</td>
<td>UBC</td>
<td>19</td>
<td>3.5</td>
</tr>
<tr>
<td>1003</td>
<td>Alice</td>
<td>SFU</td>
<td>18</td>
<td>3.7</td>
</tr>
<tr>
<td>1004</td>
<td>Bob</td>
<td>UBC</td>
<td>22</td>
<td>3.1</td>
</tr>
<tr>
<td>1005</td>
<td>David</td>
<td>SFU</td>
<td>20</td>
<td>3.2</td>
</tr>
<tr>
<td>1006</td>
<td>John</td>
<td>SFU</td>
<td>21</td>
<td>3.1</td>
</tr>
<tr>
<td>1007</td>
<td>Mary</td>
<td>UBC</td>
<td>21</td>
<td>3.4</td>
</tr>
<tr>
<td>1008</td>
<td>Mike</td>
<td>SFU</td>
<td>24</td>
<td>3.1</td>
</tr>
<tr>
<td>1009</td>
<td>Sarah</td>
<td>UBC</td>
<td>18</td>
<td>3.0</td>
</tr>
</tbody>
</table>
**Conditions**

- Which **rows** do you want information from?

```sql
WHERE age * 365 > 7500
```

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>school</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Adam</td>
<td>SFU</td>
<td>23</td>
<td>3.2</td>
</tr>
<tr>
<td>1002</td>
<td>Aiden</td>
<td>UBC</td>
<td>19</td>
<td>3.5</td>
</tr>
<tr>
<td>1003</td>
<td>Alice</td>
<td>SFU</td>
<td>18</td>
<td>3.7</td>
</tr>
<tr>
<td>1004</td>
<td>Bob</td>
<td>UBC</td>
<td>22</td>
<td>3.1</td>
</tr>
<tr>
<td>1005</td>
<td>David</td>
<td>SFU</td>
<td>20</td>
<td>3.2</td>
</tr>
<tr>
<td>1006</td>
<td>John</td>
<td>SFU</td>
<td>21</td>
<td>3.1</td>
</tr>
<tr>
<td>1007</td>
<td>Mary</td>
<td>UBC</td>
<td>21</td>
<td>3.4</td>
</tr>
<tr>
<td>1008</td>
<td>Mike</td>
<td>SFU</td>
<td>24</td>
<td>3.1</td>
</tr>
<tr>
<td>1009</td>
<td>Sarah</td>
<td>UBC</td>
<td>18</td>
<td>3.0</td>
</tr>
</tbody>
</table>
Values

• **SQL commands** are **CASE INSENSITIVE**
  - SELECT = Select = select
  - Student = student
  - gpa = GPA

• **Values** are **CASE SENSITIVE**
  - 'SFU' ≠ 'sfu'

• **Quotation**
  - SQL strings in **single quotes**
    - e.g. name = 'Mike'
    - Single quotes in a string can be specified using an initial single quote character as an escape
      - author = 'Shaq O''Neal'
  - Strings can be compared **alphabetically** with the comparison operators
    - e.g. 'fodder' < 'foo' is TRUE
Eliminating Repetitive Results

• **DISTINCT**: eliminating duplicates in the query result

\[
\text{SELECT school FROM Students}
\]

Versus

\[
\text{SELECT DISTINCT school FROM Students}
\]
Sorting the Results

• The output of an SQL query can be ordered using **ORDER BY**
  • By any number of attributes
  • In either ascending or descending order

• Default: **Ascending** order

• The keywords **ASC** and **DESC**, following the column name, set the order
Sorting the Results

```
SELECT name, gpa, age
FROM Students
WHERE school = 'SFU'
ORDER BY gpa DESC, age ASC
```
Simple String Pattern Matching

• SQL provides pattern matching support with the `LIKE` operator and two symbols
  • The `%` symbol stands for zero or more arbitrary characters
  • The `_` symbol stands for exactly one arbitrary character
  • The `%` and `_` characters can be escaped with `\`
    • E.g., name `LIKE 'Michael\_Jordan'`

```
SELECT * FROM Students WHERE name LIKE 'A_d%';
```
Simple String Pattern Matching

• Which names will be returned?

```
SELECT * 
FROM Students 
WHERE name LIKE 'Sm_t%'
```

1. Smit
2. SMIT
3. Smart
4. Smith
5. Smythe
6. Smut
7. Smeath
8. Smt
NULL

• Whenever we don’t have a value, we can put a NULL

• Can mean many things:
  • Value does not exist
  • Value exists but is unknown
  • Value not applicable
  • Etc.

• NULL constraints

```sql
CREATE TABLE Students (  
  name CHAR(20) NOT NULL,  
  age CHAR(20) NOT NULL,  
  gpa FLOAT
)
```
NULL: What will happen?

**SELECT** gpa*100 **FROM** students

**SELECT** name **FROM** students **WHERE** gpa > 3.5

**SELECT** name **FROM** students **WHERE** age > 19 **OR** gpa > 3.5

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>school</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Adam</td>
<td>SFU</td>
<td>23</td>
<td>3.2</td>
</tr>
<tr>
<td>1002</td>
<td>Aiden</td>
<td>UBC</td>
<td>19</td>
<td>3.5</td>
</tr>
<tr>
<td>1003</td>
<td>Alice</td>
<td>SFU</td>
<td>18</td>
<td>3.7</td>
</tr>
<tr>
<td>1004</td>
<td>Bob</td>
<td>UBC</td>
<td>22</td>
<td>3.1</td>
</tr>
<tr>
<td>1005</td>
<td>David</td>
<td>SFU</td>
<td>20</td>
<td>NULL</td>
</tr>
<tr>
<td>1006</td>
<td>John</td>
<td>SFU</td>
<td>21</td>
<td>3.1</td>
</tr>
<tr>
<td>1007</td>
<td>Mary</td>
<td>UBC</td>
<td>21</td>
<td>3.4</td>
</tr>
<tr>
<td>1008</td>
<td>Mike</td>
<td>SFU</td>
<td>24</td>
<td>3.1</td>
</tr>
<tr>
<td>1009</td>
<td>Sarah</td>
<td>UBC</td>
<td>18</td>
<td>3.0</td>
</tr>
</tbody>
</table>
NULL: What will happen?

• Arithmetic operations (+, -, *, /) on nulls return **NULL**
  
  ```
  NULL * 100 = NULL
  NULL * 0 = NULL
  ```

• Comparisons with nulls evaluate to **UNKNOWN**
  
  ```
  NULL > 3.5 → UNKNOWN
  NULL = NULL → UNKNOWN
  ```
NULL: What will happen?

```sql
SELECT gpa*100 FROM students
SELECT gpa*0 FROM students
SELECT name FROM students WHERE gpa > 3.5
SELECT name FROM students WHERE gpa = NULL
```
Combinations of true, false, unknown

• Truth values for unknown results

\[
\begin{align*}
\text{TRUE} \ or \ \text{UNKNOWN} &= \text{TRUE} \\
\text{FALSE} \ or \ \text{UNKNOWN} &= \text{UNKNOWN} \\
\text{UNKNOWN} \ or \ \text{UNKNOWN} &= \text{UNKNOWN} \\
\text{TRUE} \ and \ \text{UNKNOWN} &= \text{UNKNOWN} \\
\text{FALSE} \ and \ \text{UNKNOWN} &= \text{FALSE} \\
\text{UNKNOWN} \ and \ \text{UNKNOWN} &= \text{UNKNOWN}
\end{align*}
\]
Combinations of true, false, unknown

\[
\text{SELECT } * \text{ FROM students WHERE age > 15 OR gpa > 3.5}
\]

\[
\text{SELECT } * \text{ FROM students WHERE age > 15 AND gpa > 3.5}
\]

• The result of a \textit{WHERE} clause is treated as \textbf{FALSE} if it evaluates to \texttt{UNKNOWN}
  • \textbf{WHERE UNKNOWN} \rightarrow \textbf{FALSE}
Checking NULL Values

\[
\text{SELECT} \quad * \quad \text{FROM} \quad \text{Students} \quad \text{WHERE} \quad \text{age} < 25 \quad \text{OR} \quad \text{age} \geq 25
\]

\[
\text{SELECT} \quad * \quad \text{FROM} \quad \text{Students} \\
\text{WHERE} \quad \text{age} < 25 \quad \text{OR} \quad \text{age} \geq 25 \quad \text{OR} \quad \text{age} \text{ IS NULL}
\]

• There are special operators to test for null values
  - **IS NULL** tests for the presence of nulls and
  - **IS NOT NULL** tests for the absence of nulls
Acknowledgements

I have used materials from the following resources in preparation of this course:

• Database Systems: The Complete Book
• Database Systems (Kifer, Bernstein, Lewis)
• Database System Concepts: https://www.db-book.com
• Course offerings
  • W 4111 (Eugene Wu - Columbia): https://w4111.github.io/
  • CS 245 (Matei Zaharia - Stanford): http://web.stanford.edu/class/cs245/
  • CS 186 (Joe Hellerstein - Berkeley): https://sites.google.com/site/cs186fall17/