Introduction to SQL

Team JDR

Review

Modifying Tables

Constraints

Join Variants

Set Operations

Subqueries and Views

Review

SQL Overview

Clauses

Data Types

Keys

SQL Overview

SQL - Structured Query Language

RDMS - Relational Database Managements System

SQL is used to access, store, and manipulate data that is stored in an RDMS.

SQL Overview - RDMS and Tables

Data in an RDMS is stored in Tables

Tables are composed of entries (rows) and fields (columns)

Example: Table of Students in CS61A

ssid	Major	Age	Name
1	EECS	38	David
2	Geography	39	John
3	EECS	40	Katherine
4	Math	42	Jerry
5	Physics	45	Heather
6	Statistics	47	Sonja
7	EECS	48	George
8	EECS	61	Priscilla

Table

This is the table we will be using for examples:

ssid	Major	Age	Name
1	EECS	38	David
2	Geography	39	John
3	EECS	40	Katherine
4	Math	42	Jerry
5	Physics	45	Heather
6	Statistics	47	Sonja
7	EECS	48	George
8	EECS	61	Priscilla

Clauses - CREATE

```
CREATE TABLE Students(

ssid INT,

Major VARCHAR(25),

Age INT(99),

Name VARCHAR(10),

PRIMARY KEY(ssid));
```

Clauses - SELECT

```
SELECT *

FROM Students
```

ssid	Major	Age	Name
1	EECS	38	David
2	Geography	39	John
3	EECS	40	Katherine
4	Math	42	Jerry
5	Physics	45	Heather
6	Statistics	47	Sonja
7	EECS	48	George
8	EECS	61	Priscilla

Clauses - WHERE

```
SELECT *

FROM Students

WHERE Major = “EECS”
```

ssid	Major	Age	Name
1	EECS	38	David
3	EECS	40	Katherine
7	EECS	48	George
8	EECS	61	Priscilla

Clauses - WHERE

```
SELECT Name, Major

FROM Students

WHERE Age <= 40
```

Name	Major
David	EECS
John	Geography
Katherine	EECS

Clause - ORDER BY

```
SELECT Name, Age

FROM Students

ORDER BY Age
```

Name	Age
David	38
John	39
Katherine	40
Jerry	42
Heather	45
Sonja	47
George	48
Priscilla	61

Clause - GROUP BY

```
SELECT Major, Count(ssid)

FROM Students

GROUP BY Major
```

Major	Count(ssid)
EECS	4
Geography	1
Math	1
Physics	1
Statistics	1

Clause - HAVING

```
SELECT Major, Count(ssid)

FROM Students

GROUP BY Major

HAVING Count(ssid) > 2
```

Major	Count(ssid)
EECS	4

Data Types

CHAR(size)

VARCHAR(size)

INT(size)

BOOL or BOOLEAN

DOUBLE(size, d)

FLOAT(p)

Keys

Primary Keys

Secondary Keys

Modifying Tables

Insert

Update

Delete

Modifying Tables - Insert

Gender	Breed	Age	Name
M	Beagle	3	David
M	Boxer	1	Tyson
F	Corgi	5	Lucky

```
INSERT INTO dogs (Gender, Breed, Age, Name)

VALUES ('F', 'Pomeranian', 8, 'Lucy');
```

Gender	Breed	Age	Name
M	Beagle	3	David
M	Boxer	1	Tyson
F	Corgi	5	Lucky
F	Pomeranian	8	Lucy

Modifying Tables - Update

Gender	Breed	Age	Name
M	Beagle	3	David
M	Boxer	1	Tyson
F	Corgi	5	Lucky
F	Pomeranian	8	Lucy

```
UPDATE dogs
SET Age = 9
WHERE Name='Lucy' AND Breed='Pomeranian';
```

Gender	Breed	Age	Name
M	Beagle	3	David
M	Boxer	1	Tyson
F	Corgi	5	Lucky
F	Pomeranian	9	Lucy

Modifying Tables - Delete

Gender	Breed	Age	Name
M	Beagle	3	David
M	Boxer	1	Tyson
F	Corgi	5	Lucky
F	Pomeranian	9	Lucy

```
DELETE FROM dogs

WHERE Name='Lucy' AND Breed='Pomeranian';
```

Gender	Breed	Age	Name
M	Beagle	3	David
M	Boxer	1	Tyson
F	Corgi	5	Lucky

Constraints

Not NULL

Unique/Distinct

Primary Key

Foreign Key

Check

Default

Constraints - Not NULL

```
CREATE TABLE Politicians(
pol_id INT NOT NULL,
Political_Party VARCHAR(25),
Age INT(99),
Name VARCHAR(10)
);
```

pol_id column will not accept a NULL value

Constraints - Unique/Distinct

```
CREATE TABLE Politicians(

pol_id INT NOT NULL DISTINCT,

Political_Party VARCHAR(25),

Age INT(99),

Name VARCHAR(10)

);
```

pol_id column will not accept a NULL value and will ensure that all values are unique

Constraints - Primary Key

```
CREATE TABLE Politicians(
pol_id INT,
Political_Party VARCHAR(25),
Age INT(99),
Name VARCHAR(10),
PRIMARY KEY (pol_id)
);
```

Primary Keys cannot contain NULL values and they must be unique

Primary Keys can also consist of multiple columns, but only one Primary Key per table

```
CONSTRAINT PK_pol PRIMARY KEY (pol_id, name)
```

Constraints - Foreign Key

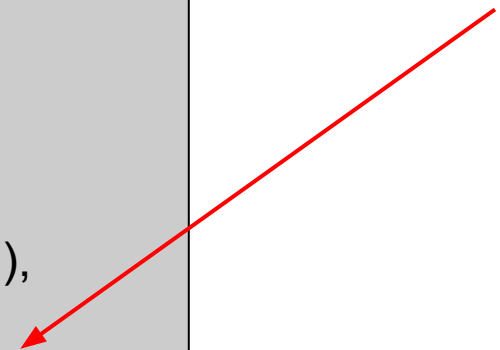
```
CREATE TABLE Politicians(

pol_id INT PRIMARY KEY,

Political_Party VARCHAR(25),

Age INT(99),

Name VARCHAR(10)

);
```

```
CREATE TABLE Bills(

bill_id INT PRIMARY KEY,

Title VARCHAR(25),

pol_id INT,

FOREIGN KEY (pol_id)
REFERENCES Politicians(pol_id)

);
```

Foreign Key prevents invalid entries into Bills and prevents actions that would destroy links between the tables

Constraints - Check

```
CREATE TABLE Politicians(
pol_id INT,
Political_Party VARCHAR(25),
Age INT(99),
Name VARCHAR(10),
PRIMARY KEY (pol_id),
CHECK (Age>=25)
);
```



Every value entered for Age must be greater than or equal to 25

Constraints - Default

```
CREATE TABLE Politicians(
pol_id INT,
Political_Party VARCHAR(25) DEFAULT 'Independent',
Age INT(99),
Name VARCHAR(10),
PRIMARY KEY (pol_id),
CHECK (Age>=25)
);
```

The default value for the political party column is Independent

Join Variants

Inner

Natural

Left vs Right

Join Variants - Inner

Office Table

emp_id	First_Name	Age	cuffed
1	Michael	42	Y
2	Dwight	39	Y
3	Jim	37	Y
4	Pam	35	Y
5	Angela	45	Y
6	Toby	47	N
7	Creed	77	N
8	Kelly	35	N
9	Jan	44	N
10	Meredith	50	N
11	Karen	38	N
12	Josh	36	U

Managers Table

mgr_id	branch
1	Scranton
9	All branches
11	Utica
12	Stamford

mgr_id references emp_id as a foreign key

Join Variants - Inner

emp_id	First_Name	Age	cuffed
1	Michael	42	Y
2	Dwight	39	Y
3	Jim	37	Y
4	Pam	35	Y
5	Angela	45	Y
6	Toby	47	N
7	Creed	77	N
8	Kelly	35	N
9	Jan	44	N
10	Meredith	50	N
11	Karen	38	N
12	Josh	36	U

```
SELECT * FROM Office

INNER JOIN Managers

ON (emp_id = mgr_id);
```

mgr_id	branch
1	Scranton
9	All branches
11	Utica
12	Stamford

Join Variants - Inner

emp_id	First_Name	Age	cuffed
1	Michael	42	Y
2	Dwight	30	Y
3	Jim	37	Y
4	Pam	35	Y
5	Angela	45	Y
6	Toby	47	N
7	Creed	77	N
8	Kelly	35	N
9	Jan	44	N
10	Meredith	50	N
11	Karen	38	N
12	Josh	36	U

mgr_id	branch
1	Scranton
9	All branches
11	Utica
12	Stamford

Join Variants - Inner

```
SELECT * FROM Office
INNER JOIN Managers
ON (emp_id = mgr_id);
```

emp_id	First_Name	Age	cuffed	mgr_id	branch
1	Michael	42	Y	1	Scranton
9	Jan	44	N	9	All branches
11	Karen	38	N	11	Utica
12	Josh	36	U	12	Stamford

Join Variants - Natural

Office Table

emp_id	First_Name	Age	cuffed
1	Michael	42	Y
2	Dwight	39	Y
3	Jim	37	Y
4	Pam	35	Y
5	Angela	45	Y
6	Toby	47	N
7	Creed	77	N
8	Kelly	35	N
9	Jan	44	N
10	Meredith	50	N
11	Karen	38	N
12	Josh	36	U

Managers Table

emp_id	branch
1	Scranton
9	All branches
11	Utica
12	Stamford

*** mgr_id has been changed to emp_id for this section ***

Join Variants - Natural

```
SELECT * FROM Office
NATURAL JOIN Managers;
```

emp_id	First_Name	Age	cuffed	branch
1	Michael	42	Y	Scranton
9	Jan	44	N	All branches
11	Karen	38	N	Utica
12	Josh	36	U	Stamford

Note: The resulting table is the exact same from Inner Join

Join Variants - Left vs Right Outer

Salary Table

Player_ID	Salary
1	999000
2	842000
3	630000
4	1250000
5	700000
6	873000
7	800000
8	450000

Legends Table

Player_ID	Name	Position
2	Roger Clemens	Pitcher
3	Stan Musial	Outfield
5	Honus Wagner	(null)
7	Barry Bonds	Outfield

Join Variants - Left vs Right

```
SELECT Legends.Player_ID, Name, Salary
FROM Legends
LEFT JOIN Salary
ON Salary.Player_ID = Legends.Player_ID;
```

vs

```
SELECT Legends.Player_ID, Name, Salary
FROM Salary
RIGHT JOIN Legends
ON Salary.Player_ID = Legends.Player_ID;
```

Player_ID	Name	Salary
2	Roger Clemens	842000
3	Stan Musial	630000
5	Honus Wagner	700000
7	Barry Bonds	800000

Both of these queries have the same resulting table

Join Variants - Left vs Right

What happens when we change RIGHT JOIN to LEFT JOIN

```
SELECT Legends.Player_ID, Name, Salary
FROM Salary
LEFT JOIN Legends
ON Salary.Player_ID = Legends.Player_ID;
```

Player_ID	Name	Salary
1	(null)	999000
2	Roger Clemens	842000
3	Stan Mulsia	630000
4	(null)	1250000
5	Honus Wagner	700000
6	(null)	800000
7	Barry Bonds	450000

Notice the changes between the two resulting tables

Join Variants - Left vs Right

Player_ID	Name	Salary
2	Roger Clemens	842000
3	Stan Musial	630000
5	Honus Wagner	700000
7	Barry Bonds	800000

vs

Player_ID	Name	Salary
1	(null)	999000
2	Roger Clemens	842000
3	Stan Mulsia	630000
4	(null)	1250000
5	Honus Wagner	700000
6	(null)	800000
7	Barry Bonds	450000

Notice the changes between the two resulting tables

Set Operations

Union

Union All

Intersect

Minus

Set Operations

Students Table

SSID	Name
1	John Smith
2	Tony Stark
56	Din Djarin
104	Frodo Baggins

Employees Table

SSID	Name
1	John Smith
2	Tony Stark
7	John Wick
23	Johnny Appleseed

Set Operations - Union

```
SELECT SSID, Name FROM Students

UNION

SELECT SSID, Name FROM Employees;
```

SSID	Name
1	John Smith
2	Tony Stark
56	Din Djarin
104	Frodo Baggins
7	John Wick
23	Johnny Appleseed

Set Operations - Union All

```
SELECT SSID, Name FROM Students

UNION ALL

SELECT SSID, Name FROM Employees;
```

SSID	Name
1	John Smith
2	Tony Stark
56	Din Djarin
104	Frodo Baggins
1	John Smith
2	Tony Stark
7	John Wick
23	Johnny Appleseed

Set Operations - Intersect

```
SELECT SSID, Name FROM Students

UNION

SELECT SSID, Name FROM Teachers;
```

SSID	Name
1	John Smith
2	Tony Stark

Set Operations - Minus

```
SELECT SSID, Name FROM Students

UNION

SELECT SSID, Name FROM Teachers;
```

SSID	Name
56	Din Djarin
104	Frodo Baggins

Subqueries and Views

Subqueries

Views

Subqueries and Views

Students Table

SSID	Name
1	John Smith
2	Tony Stark
56	Din Djarin
104	Frodo Baggins

Employees Table

SSID	Name
1	John Smith
2	Tony Stark
7	John Wick
23	Johnny Appleseed

Set Operations - Subqueries

```
SELECT joined_table.SSID, joined_table.Name
FROM (SELECT SSID, Name FROM Students
      WHERE Name != "John Smith"
      UNION
      SELECT SSID, Name FROM Teachers
      WHERE Name != "John Smith"
      ) AS joined_table
WHERE joined_table.SSID < 8;
```

SSID	Name
2	Tony Stark
7	John Wick

Set Operations - Subqueries

```
CREATE VIEW joined_table AS
  SELECT SSID, Name FROM Students
  WHERE Name != "John Smith"
  UNION
  SELECT SSID, Name FROM Teachers
  WHERE Name != "John Smith";
```

```
SELECT SSID, Name
FROM joined_table
WHERE SSID < 8;
```

SSID	Name
2	Tony Stark
7	John Wick