Introduction to SQL

By

Anand Joshi

Structured Query Language (SQL)

• SQL is a **declarative** query language , not a procedural or an imperative language.

BookID	Title	PubDate	ListPrice
1145	Designing Databases	3/1/2012	$45.00
1146	SQLite Made Simple	4/11/2012	$39.95
1147	Pocket Guide to SQL	5/21/2012	$19.95
1148	DB Best Practices	5/22/2012	$48.00
1149	NoSQL Databases	5/26/2012	$35.00
1150	Fun with SQL	5/27/2012	$42.00
...	...	...	...

• A procedural or an imperative language would retrieve data in this fashion.

• Pseudo code for finding books having price greater than 40$ from the table.

```
for each b in Books
  if price > 40
    add b to expensive_books_array
  else
    ignore
  end
end

return expensive_books_array
```

• A declarative language would retrieve data in this fashion.

• SQL code for finding books having price greater than 40$ from the table.

```
SELECT * FROM Books WHERE ListPrice > 40
```

SQL can not only be used to make queries but it can also be used to-

• Create

• Read

• Update

• Delete

Commonly called CRUD

Event

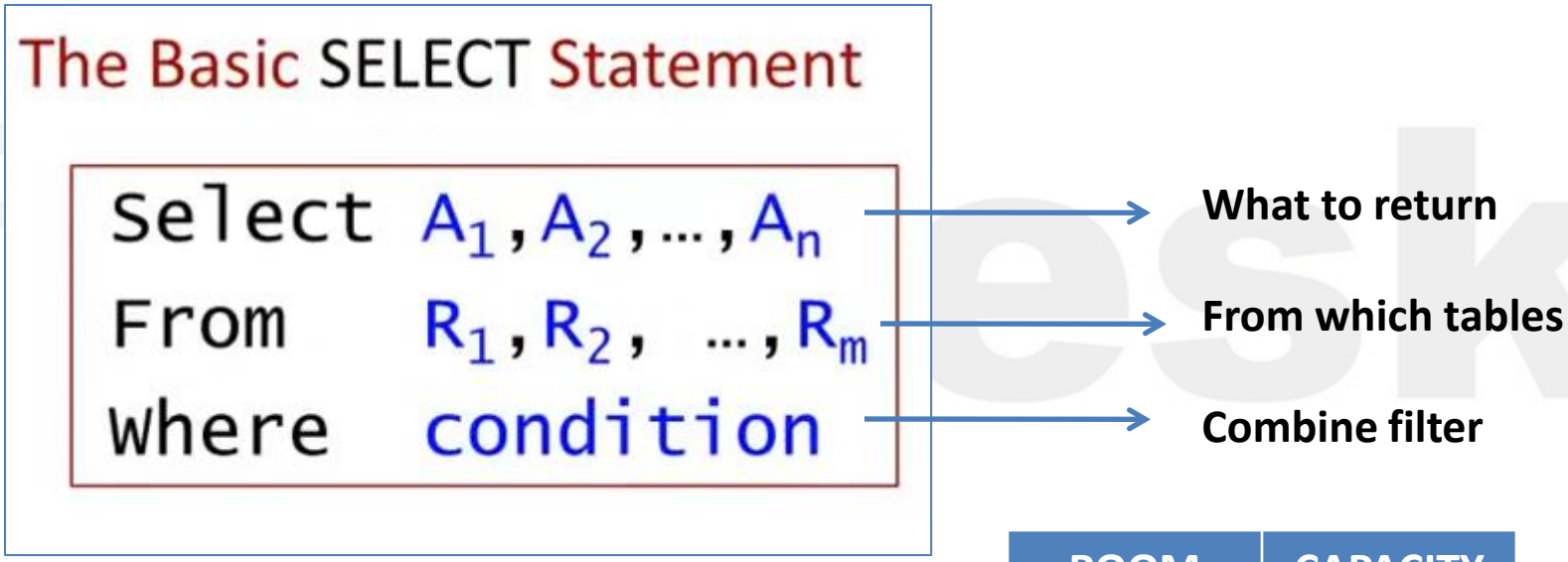
COURSE	DATE	ROOM	CAPACITY	AVAILABLE
SQL101	3/1/2014	4A	12	4
DB202	3/1/2014	7B	14	7
SQL101	10/2/2014	7B	14	10
SQL101	12/3/2014	12A	8	8
CS200	6/3/2014	4A	12	11

COURSE_ID	TITLE
SQL101	SQL
DB202	DATABASE DESIGN
CS200	JAVA

Course

Queries in SQL

Q) How to get columns (attributes) from a table ?



E.g. 1) **SELECT ROOM, CAPACITY**
FROM Event
WHERE CAPACITY > 10

ROOM	CAPACITY
4A	12
7B	14
7B	14
4A	12

E.g. 2) **SELECT ***
FROM Event → *** Indicates all columns**

E.g. 3) **SELECT DISTINCT Event.COURSE**
FROM Event →

COURSE
SQL101
DB202
CS200

E.g. 4) **SELECT Course.COURSE_ID, Course.TITLE**
FROM Course
WHERE Course.TITLE = 'JAVA' →

COURSE_ID	TITLE
CS200	JAVA

More Queries

customers

Id	name	address	city	state	zip
...	...	...	...	...	...

1) SELECT city, state, zip
FROM customers

2) SELECT *
FROM customers

3) SELECT id, name
FROM customers
LIMIT 5 , 3

Displays id , name of customer 6, 7, 8

4) SELECT name
FROM customers
ORDER BY name

By default sorting is in ASCENDING ORDER

5) SELECT state, city, name
FROM customers
ORDER BY state, name

6) SELECT name, zip
FROM customers
ORDER BY zip DESC

7) SELECT name, id
FROM customers
ORDER BY id DESC
LIMIT 0, 1

Gets customer with highest id

Filtering Queries

8) SELECT id, name
FROM customers
WHERE id = 54

9) SELECT id, name
FROM customers
WHERE id = 60 OR state = ‘CA’

10) SELECT id, name
FROM customers
WHERE id BETWEEN 25 AND 30

11) SELECT name, state, city
FROM customers
WHERE state = ‘CA’ AND city = ‘Hollywood’

12) SELECT name, state
FROM customers
WHERE state IN('CA', 'NC', 'NY')

13) SELECT name, state
FROM customers
WHERE state NOT IN('CA', 'NC', 'NY')

items

Id	name	cost	seller_id	bids
...	...	...	...	...

Search Queries

14) SELECT id, name
FROM items
WHERE name LIKE 'new%'

15) SELECT id, name
FROM items
WHERE name LIKE '_ boxes of candies'

16) SELECT zip AS POSTAL_CODE
FROM customers

Temporary name of zip will be POSTAL_CODE

17) SELECT CONCAT(city, ' , ' , state) AS address
FROM customers

18) SELECT name, cost, floor(cost*0.8) AS selling_price
FROM items

19) SELECT AVG(cost)
FROM items

20) SELECT MAX(cost)
FROM items

Some more functions : MIN() , COUNT() , SQRT() , SUM() , UPPER() , LOWER()

21) SELECT id, name
FROM items
WHERE bids IS NULL

22) SELECT seller_id, sum(bids) AS Anand's bids
FROM items
WHERE seller_id = 10

23) SELECT COUNT(*) AS item_count ,
MAX(cost) AS max ,
AVG(cost) AS avg
FROM items
WHERE seller_id = 12

item_count	max	avg
3	30	20

24) Write a query to find the total items sold by each seller.

seller_id	total_items
1	6
2	4
…	…

sol) SELECT seller_id, COUNT(*) AS total_items
FROM items
WHERE seller_id = 1 , 2 , 3 ……. And so on

To do this use GROUP BY clause in SQL

```
SELECT  seller_id, COUNT(*) AS total_items
        FROM items
        GROUP BY seller_id
```

```
25 ) SELECT  seller_id, COUNT(*) AS total_items
             FROM items
             GROUP BY seller_id
             HAVING COUNT(*) > 5
```

- Having is just like WHERE but it is used with GROUP BY clause only.

Subqueries → **Inception !!**

26) Find the items whose cost is above the average price of all the items.

```
sol )  SELECT  name, cost
       FROM items
       WHERE  cost >  ( SELECT AVG(cost) FROM items )
```

27) Find the item having second highest number of bids.

```
sol )  SELECT  name, MAX(bids)
       FROM items
       WHERE  bids NOT IN  (
                              SELECT MAX(bids)
                              FROM items
                            )
```

SQL JOINS

- JOINS can be used to combine tables.
- There are many types of JOINS.
    - CROSS JOIN
    - INNER JOIN
    - NATURAL JOIN
    - OUTER JOIN

Cross Join

- **A CROSS JOIN B** returns all pairs of rows from A and B.

Student

ID	Name
123	John
124	Mary
125	Mark
126	Jane

Enrolment

ID	Code
123	DBS
124	PRG
124	DBS
126	PRG

```
SELECT * FROM
  Student CROSS JOIN
  Enrolment
```

ID	Name	ID	Code
123	John	123	DBS
124	Mary	123	DBS
125	Mark	123	DBS
126	Jane	123	DBS
123	John	124	PRG
124	Mary	124	PRG
125	Mark	124	PRG
126	Jane	124	PRG
123	John	124	DBS
124	Mar		DBS

Natural Join

• **A NATURAL JOIN B** returns pairs of rows with common values for identically named columns and without duplicating rows

Student

ID	Name
123	John
124	Mary
125	Mark
126	Jane

Enrolment

ID	Code
123	DBS
124	PRG
124	DBS
126	PRG

```
SELECT * FROM
   Student NATURAL JOIN
   Enrolment
```

ID	Name	Code
123	John	DBS
124	Mary	PRG
124	Mary	DBS
126	Jane	PRG

Inner Join

- **A INNER JOIN B** returns pairs of rows satisfying a condition.

id	name	address	city	state	zip
…	…	…	…	…	…

customers

id	name	cost	seller_id	bids
…	…	…	…	…

items

E.g.) **SELECT customers.id, customers.name , items.name**
FROM customers INNER JOIN items
ON customers.id = items.seller_id

Outer Join

- Inner Joins join two tables only on matching attribute.
- Outer JOIN = matching values + non matching values (NULL values).
- Consider the following schemas.

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

• The LEFT OUTER JOIN of both the tables will be.

```
SELECT Product.name, Purchase.store
FROM   Product LEFT OUTER JOIN Purchase ON
                  Product.name = Purchase.prodName
```

All the values from the **LEFT** table are present

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	NULL

Non matching values in the **RIGHT** table are made NULL

• Similarly we also have RIGHT OUTER JOIN.

- The SQL commands learned so far can help us retrieve data of our interest.
- They help us manipulate data.
- This subset of SQL language is called DML - Data Manipulation Language.
- The other set of SQL language is called DDL – Data Definition Language.
- DDL helps us to create, delete or update our database.

DDL examples

1) **CREATE TABLE users (**
id int ,
username VARCHAR(30) ,
password VARCHAR(30),
PRIMARY KEY(id)
)

2) **DROP table users**

- There are many more like **ALTER**, **RENAME**, **UPDATE**, **INSERT INTO** etc

Summary

- SELECT
- WHERE
- FROM
- AND / OR
- IN
- NOT IN
- ORDER BY DESC / ASC
- GROUP BY
- HAVING
- MAX() MIN(), AVG(), COUNT()
- CROSS JOIN
- INNER JOIN
- NATURAL JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- CREATE
- DROP

Contact Info

- trainers@finaldesk.com
- rishabh@finaldesk.com
- nilesh@finaldesk.com
- jignesh@finaldesk.com
- yash@finaldesk.com
- anand@finaldesk.com