Database Implementation -
SQL (Part 3)

SQL for Data Manipulation

Manipulation

- *SQL allows a user or an application program to update the database by adding new data, removing old data, and modifying previously stored data.*

Retrieval

- *SQL allows a user or an application program to retrieve stored data from the database and use it.*

Most Commonly Used Commands

SELECT INSERT

UPDATE DELETE

SQL for Data Manipulation

- High-level Language for data manipulation
- It does not require predefined navigation path
- It does not require knowledge of any key items
- It is uniform language for end-users and programmers
- It operates on one or more tables based on set theory, not on a record at a time.

Command: **SELECT**

- Function:
  - *Retrieves data from one or more rows. Every SELECT statement produces a table of query results containing one or more columns and zero or more rows.*

SELECT {[ALL, DISTINCT]} [(*select-item*,), i]
FROM (*table specification*,)
{WHERE (*search condition*)}
{GROUP BY (*group-column*,)}
{HAVING (*search condition*)}
{ORDER BY (*sort specification*,)}

Employee

E-No	E-Name	D-No
179	Silva	7
857	Perera	4
342	Dias	7

Employee Names

E-No	E-Name
179	Silva
857	Perera
342	Dias

```
SELECT    E-No, E-Name
FROM      Employee ;
```

Employee Names

E-No	E-Name
342	Dias
857	Perera
179	Silva

```
SELECT    E-No, E-Name
FROM      Employee
ORDER BY  E-Name ;
```

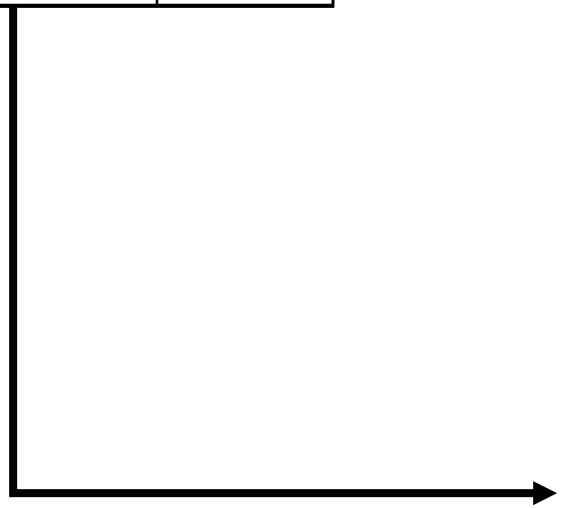
Employee

E-No	E-Name	D-No
179	Silva	7
857	Perera	4
342	Dias	7

Sales Employee

E-No	E-Name	D-No
179	Silva	7
342	Dias	7

```
SELECT    *
FROM      Employee
WHERE     D-No = '7' ;
```



Sales Employee

E-No	E-Name
179	Silva
342	Dias

```
SELECT    E-No, E-Name
FROM      Employee
WHERE     D-No = '7' ;
```

Restrict Rows and Project Columns

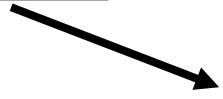
EquiJoin

Employee

E-No	E-Name	D-No
179	Silva	7
857	Perera	4
342	Dias	7

Department

D-No	D-Name	M-No
4	Finance	857
7	Sales	179



Emp-Info

E-No	E-Name	D-No	D-No	D-Name	M-No
179	Silva	7	7	Sales	179
857	Perera	4	4	Finance	857
342	Dias	7	7	Sales	179

```
SELECT    Employee.*, Department.*
FROM      Employee, Department
WHERE     Employee.D-No = Department.D-No ;
```

```
SELECT    E.*, D.*
FROM      Employee E, Department D
WHERE     E.D-No = D.D-No ;
```

Cartesian Product

Employee

E-No	E-Name	D-No
179	Silva	7
857	Perera	4
342	Dias	7

Department

D-No	D-Name	M-No
4	Finance	857
7	Sales	179

Emp-Info

E-No	E-Name	D-No	D-No	D-Name	M-No
179	Silva	7	4	Finance	857
857	Perera	4	4	Finance	857
342	Dias	7	4	Finance	857
179	Silva	7	7	Sales	179
857	Perera	4	7	Sales	179
342	Dias	7	7	Sales	179

```
SELECT    E.*, D.*
FROM      Employee E, Department D
```

SQL Data Retrieval

Basic Search Conditions:

- Comparison
  - Equal to =
  - Not equal to != or <> or ^=
  - Less than to <
  - Less than or equal to <=
  - Greater than to >
  - Greater than or equal to >=

SQL Data Retrieval

Basic Search Conditions (cont'd) **:**

- Range ([NOT] BETWEEN)
  - *expres-1* [NOT] BETWEEN *expres-2* AND *expres- 3*
  - *Example:* WEIGHT **BETWEEN** 50 AND 60
- Set Membership ([NOT] IN)
  - *Example 1:* WHERE Emp_No **IN** ('E1', 'E2', 'E3')
  - *Example 2:* WHERE Emp_No **IN** (Select Emp_No FROM Employee WHERE Dept_No='7')

Basic Search Conditions (cont'd) **:**

- Pattern Matching ([NOT] LIKE)
  - *expres-1* [NOT] LIKE {*special-register* | *host-variable* | *string-constant*}
  - *Example:* WHERE Proj_Name **LIKE** "INFORM%"
- Null Value (IS [NOT] NULL)
  - *Example:* WHERE Proj_Name **IS** NOT NULL

Compound Search Conditions :

- AND, OR and NOT

Example:
WHERE Proj_Name **LIKE** 'INFORM%' AND Emp_Name = 'DIAS'

SQL Query Features

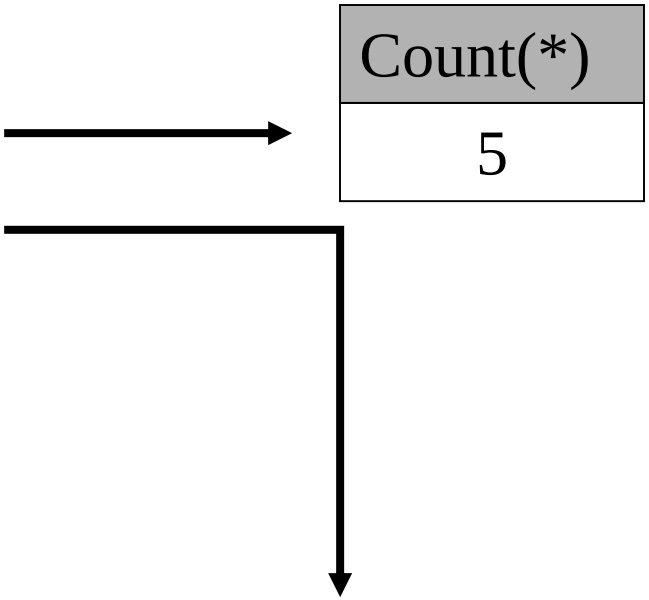
- Summary Queries
  - *Summarize data from the database. In general, summary queries use SQL functions to collapse a column of data values into a single value that summarizes the column. (AVG, MIN, MAX, SUM, COUNT..)*
- Sub-Queries
  - *Use the results of one query to help define another query*

Summarising Data

SELECT COUNT(*)
FROM Employee

Employee

E-No	Job	Salary	D-No
179	Manager	20000	10
857	Clerk	8000	10
342	Clerk	9000	20
477	Manager	15000	30
432	Clerk	10000	30



Count(*)
5

SELECT AVG(Salary)
FROM Employee

AVG(Salary)
12400

SELECT STATEMENT
May also contain
[GROUP BY [HAVING] ORDER BY]

GROUP BY

A result of a previous specified clause is grouped using the group by clause.

e.g.

```
SELECT      d-no, AVG(salary)
FROM        employee
GROUP BY    d-no
```

Employee

E-No	Job	Salary	D-No
179	Manager	20000	10
857	Clerk	8000	10
342	Clerk	9000	20
477	Manager	15000	30
432	Clerk	10000	30

D-No	AVG(Salary)
10	14,000
20	9,000
30	12,500

[GROUP BY [HAVING] ORDER BY]

HAVING

Used for select groups that meet specified conditions.

Always used with GROUP BY clause.

```
SELECT      d-no, AVG(salary)
FORM        employee
GROUP BY    d-no
HAVING      AVG(salary)>12000
```

Employee

E-No	Job	Salary	D-No
179	Manager	20000	10
857	Clerk	8000	10
342	Clerk	9000	20
477	Manager	15000	30
432	Clerk	10000	30

D-No	AVG(Salary)
10	14,000
30	12,500

Nested Queries

A sub query is SELECT statement that nest inside the WHERE clause of another SELECT statement. The results are need in solving the main query.

Get a list of all suppliers supplying part P2.

```
SELECT sname
FROM supplier
WHERE sno IN
  (SELECT sno FROM supply WHERE pno = ‘P2’);
```

```
SELECT sname
FROM supplier, supply
WHERE supplier.sno = supply.sno and pno = ‘P2’;
```

Nested Queries

```
SELECT ename , salary
FROM employee
WHERE salary = (SELECT MIN (salary) FROM employee)
```

Sub queries with EXISTS

e.g. find all publishers who publish business books

```
SELECT    DISTINCT    pub_name
FROM publishers
WHERE EXISTS
(SELECT * FROM title
 WHERE pub_id = publishers.pub_id and type = “business”)
```

DISTINCT – will remove multiple occurrences

Command: **INSERT**

- Function:
  - *Places data one or more rows into a table*
  - *Data can also be downloaded from another computer system or collected from other sites.*

INSERT Command

INSERT INTO *table-name* (*column-name*,),
| VALUES ([*constant*, NULL],)
or
| SELECT *retrieval condition*

Command: **INSERT** (cont'd)

- Single-Row Insert

```
INSERT INTO Employee (Emp_No, Emp_Name, Age, Dept)
    VALUES   ('E1', 'Dias', 26, 'PER')
```

- Multi-Row Insert

```
INSERT INTO Manager (Emp_No, Emp_Name, Age, Dept)
    SELECT Emp_No, Emp_Name, Age, Dept
    FROM    Employee
    WHERE  Job = 'Manager'
```

RESTRICT INSERT

Insert with referential integrity

In Employee Table

CONSTRAINT Emp_Dep_FK
FOREIGN KEY (Dept) **REFERENCES**
Department(Dept_Code)

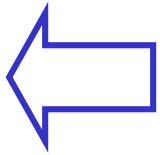
INSERT INTO Employee
VALUES (342, ‘Dias, 26, ‘Sale’);

An employee can only be inserted if its department is found in department table

RESTRICT INSERT

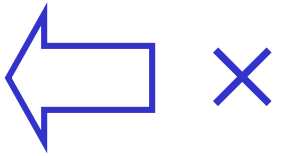
Department

Dept_Code	Dep_Name	Manager
SAL	Sales	179
FIN	Finance	857



Employee

Emp_No	Emp_Name	Age	Dept_Code
179	Silva	27	SAL
857	Perera	34	FIN
342	Dias	26	Sale



Command: **UPDATE**

- Function: *Changes data in one or more rows of a table*

UPDATE *table-name*
SET (*column-name = expression,*),
WHERE *search-condition*

Example:

```
UPDATE STUDCLASS
SET FEES = 1200
WHERE STUDNO = 1234
```

Selective Update

```
UPDATE STUDCLASS
SET FEES = 1200
```

Update All Rows

Command: **UPDATE** (cont'd)

Example:

Update with Subquery

```
UPDATE Works_On
SET Hours = 12
WHERE Proj_No IN(SELECT Proj_No FROM Project
    WHERE Proj_Name = 'INFORMATION TECHNOLOGY')
```

```
UPDATE Employee
SET         Age = Age+1
```

RESTRICT UPDATE

Update with referential integrity

In Employee Table

CONSTRAINT Emp_Dep_FK

FOREIGN KEY (Dept) **REFERENCES**

Department(Dept_Code) **ON UPDATE RESTRICT**

UPDATE Department **SET** Dept_Code = ‘Sale’

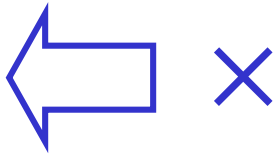
WHERE Dept_Code = ‘SAL’

A department code can only be changed if it is not found in employee table (i.e. no employees working for them)

RESTRICT UPDATE

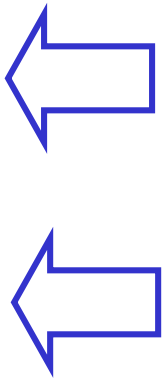
Department

Dept_Code	Dep_Name	Manager
SAL	Sales	179
FIN	Finance	857



Employee

Emp_No	Emp_Name	Age	Dept
179	Silva	27	SAL
857	Perera	34	FIN
342	Dias	26	SAL



CASCADE UPDATE

Update with referential integrity

In Employee Table

CONSTRAINT Emp_Dep_FK

FOREIGN KEY (Dept) **REFERENCES**

Department(Dept_Code) **ON UPDATE CASCADE**

UPDATE Department **SET** Dept_Code = ‘Sale’

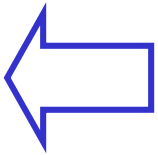
WHERE Dept_Code = ‘SAL’

Updating a department code will result in changing it in the employee table (update with new department code for the employees working for them)

CASCADE UPDATE

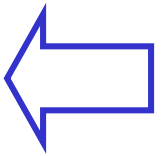
Department

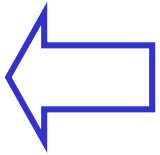
Dept_Code	Dep_Name	Manager
Sale	Sales	179
FIN	Finance	857



Employee

Emp_No	Emp_Name	Age	Dept
179	Silva	27	Sale
857	Perera	34	FIN
342	Dias	26	Sale





SET NULL UPDATE

Update with referential integrity

In Employee Table

CONSTRAINT Emp_Dep_FK

FOREIGN KEY (Dept) **REFERENCES**

Department(Dept_Code) **ON UPDATE SET NULL**

UPDATE Department **SET** Dept_Code = ‘Sale’

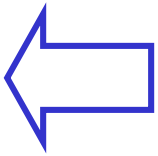
WHERE Dept_Code = ‘SAL’

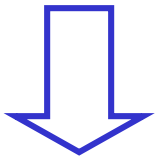
Updating a department code will result in changing the department code of their employees to NULL (only if NULL values are allowed)

SET NULL UPDATE

Department

Dept_Code	Dep_Name	Manager
Sale	Sales	179
FIN	Finance	857





Employee

Emp_No	Emp_Name	Age	Dept
179	Silva	27	NULL
857	Perera	34	FIN
342	Dias	26	NULL

SET DEFAULT UPDATE

Update with referential integrity

In Employee Table

CONSTRAINT Emp_Dep_FK

FOREIGN KEY (Dept) **REFERENCES**

Department(Dept_Code) **ON UPDATE**

SET DEFAULT 'AAA'

UPDATE Department **SET** Dept_Code = 'Sale'

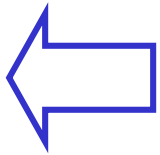
WHERE Dept_Code = 'SAL'

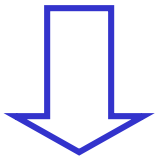
Updating a department code will result in changing the department code of their employees to a default value

SET DEFAULT UPDATE

Department

Dept_Code	Dep_Name	Manager
Sale	Sales	179
FIN	Finance	857





Employee

Emp_No	Emp_Name	Age	Dept
179	Silva	27	AAA
857	Perera	34	FIN
342	Dias	26	AAA

Command: **DELETE**

- Function: *Removes one or more rows from a table*

DELETE FROM *table-name*
{**WHERE** *search-condition*}

Example:

DELETE FROM Employee
WHERE Emp_No = ‘E1’

Select Delete

DELETE FROM Employee

Delete All Rows

Delete with Subquery

DELETE FROM Dependent
WHERE Emp_No = (SELECT Emp_No FROM Employee
WHERE Emp_Name = ‘Dias’)

RESTRICT DELETE

Delete with referential integrity

In Employee Table

CONSTRAINT Emp_Dep_FK

FOREIGN KEY (Dept) **REFERENCES**

Department(Dept_Code) **ON DELETE RESTRICT**

DELETE FROM Department

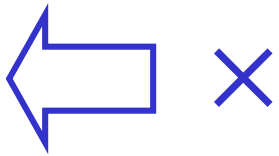
WHERE Dept_Code = ‘SAL’

A department can only be deleted if it is not found in employee table (i.e. no employees working for them)

RESTRICT DELETE

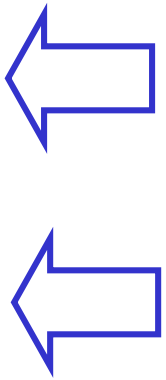
Department

Dept_Code	Dep_Name	Manager
SAL	Sales	179
FIN	Finance	857



Employee

Emp_No	Emp_Name	Age	Dept
179	Silva	27	SAL
857	Perera	34	FIN
342	Dias	26	SAL



CASCADE DELETE

Delete with referential integrity

In Employee Table

CONSTRAINT Emp_Dep_FK

FOREIGN KEY (Dept) **REFERENCES**

Department(Dept_Code) **ON DELETE CASCADE**

DELETE FROM Department

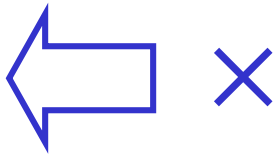
WHERE Dept_Code = ‘SAL’

Deleting a department will result in deleting it from the employee table (delete employees working for them)

CASCADE DELETE

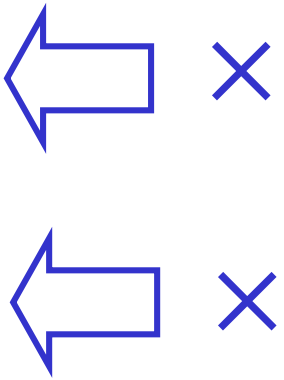
Department

Dept_Code	Dep_Name	Manager
SAL	Sales	179
FIN	Finance	857



Employee

Emp_No	Emp_Name	Age	Dept
179	Silva	27	SAL
857	Perera	34	FIN
342	Dias	26	SAL



SET NULL DELETE

Delete with referential integrity

In Employee Table

CONSTRAINT Emp_Dep_FK

FOREIGN KEY (Dept) **REFERENCES**

Department(Dept_Code) **ON DELETE SET NULL**

DELETE FROM Department

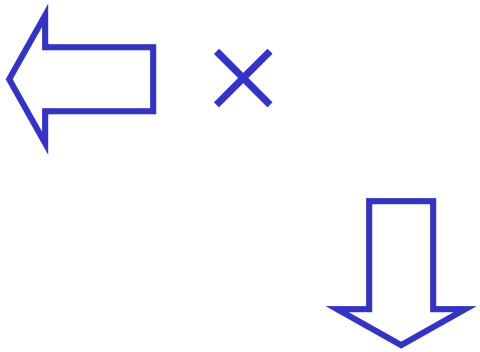
WHERE Dept_Code = 'SAL'

Deleting a department will result in changing the department of their employees in the employee table to NULL (only if NULL values are allowed)

SET NULL DELETE

Department

Dept_Code	Dep_Name	Manager
SAL	Sales	179
FIN	Finance	857



Employee

Emp_No	Emp_Name	Age	Dept
179	Silva	27	NULL
857	Perera	34	FIN
342	Dias	26	NULL

SET DEFAULT DELETE

Delete with referential integrity

In Employee Table

CONSTRAINT Emp_Dep_FK
FOREIGN KEY (Dept) **REFERENCES**
Department(Dept_Code) **ON DELETE**
SET DEFAULT 'AAA'

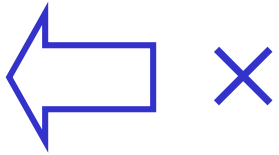
DELETE FROM Department
WHERE Dept_Code = 'SAL'

Deleting a department will result in changing the department of their employees in the employee table to a specified default value

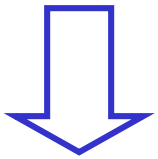
SET DEFAULT DELETE

Department

Dept_Code	Dep_Name	Manager
SAL	Sales	179
FIN	Finance	857



Employee



Emp_No	Emp_Name	Age	Dept
179	Silva	27	AAA
857	Perera	34	FIN
342	Dias	26	AAA