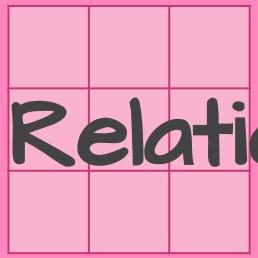
Unit 2



Relational Databases

MSc CS

GAC-CBE

Table of Contents

Relational Model

- Structure of Relational Databases
- Fundamental Relational Algebra Operations
- Additional Relational Algebra Operations.

SQL

- Background
- Data Definition
- Basic Structure of SQL Queries
- Set Operations
- Aggregate Functions
- Null values
- Nested Sub queries
- Views
- Modification of the Database.

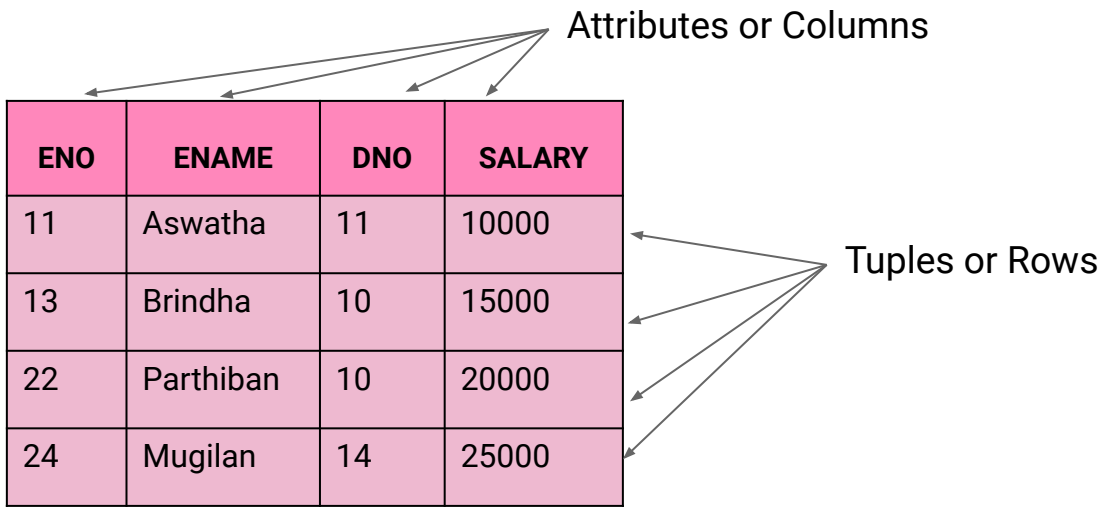
Relational Model

lets user to access and manipulate databases

Structure of Relational Databases

lets user to access and manipulate databases

Relational Model



Relational Schema and Instances

- $A_1, A_2, \ldots, A_n$ are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  Example:

  emp = (*eno, ename, dno, salary*)

- A relation instance r defined over schema R is denoted by r (R).
- The current values a relation are specified by a table
- An element ***t*** of relation ***r*** is called a *tuple* and is represented by a row in a table

Attributes

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain. Indicated that the value is "unknown"
- The null value causes complications in the definition of many operations

Attributes

- Database schema - is the logical structure of the database.
- Database instance - is a snapshot of the data in the database at a given instant in time.
- Eg.schema: emp (eno, ename, dno, salary)
- Schema

Eno	Ename	Dno	Salary
11	Aswatha	11	10000
13	Brindha	10	15000
22	Parthiban	10	20000
24	Mugilan	14	25000

Fundamental Relational Algebra Operations

Select, Project, Union, Set different, Cartesian product and Rename

Relational Algebra

select * from emp;

emp

emp.eno	emp.ename	emp.dno	emp.salary
11	'Aswatha'	11	10000
13	'Brindha'	10	15000
22	'Parthiban'	10	20000
24	'Mugilan'	14	5000

Select Operation

- The **select** operation selects tuples that satisfy a given predicate.
- Notation: $\sigma_{p}(r)$
- p is called the **selection predicate**

Select Operation

- Comparisons are allowed using

  $=, \neq, >, \geq, <, \leq$

  in the selection predicate.
- Several predicates can be combined into a larger predicate by using the connectives:

  $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)

Select Operation

select * from emp where salary > 10000

$\sigma_{\text{salary} > 10000}\ \text{emp}$

emp.eno	emp.ename	emp.dno	emp.salary
13	'Brindha'	10	15000
22	'Parthiban'	10	20000

Project Operation

- A unary operation that returns its argument relation, with certain attributes left out.
- Notation:

$$\Pi_{A1,A2,A3\ldots Ak}(r)$$

where A_1, A_2, ..., A_k are attribute names and r is a relation name.

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets

Project Operation

select eno from emp

π_{eno} emp

emp.eno
11
13
22
24

Project Operation

select ename from emp

$\pi_{\text{ename, salary}}$ emp

emp.ename	emp.salary
'Aswatha'	10000
'Brindha'	15000
'Parthiban'	20000
'Mugilan'	5000

Operation involving Selection and Projection

select ename from emp where salary > 10000

$\pi_{ename}\ \sigma_{salary > 10000}\ emp$

emp.ename
'Brindha'
'Parthiban'

Operation involving Selection and Projection

select ename,dno from emp where salary > 10000

$\pi\ e_{name,\ dno}\ \sigma_{\ salary > 10000}\ emp$

emp.ename	emp.dno
'Brindha'	10
'Parthiban'	10

Operation involving Selection and Projection

select ename from emp where salary > 10000 and dno = 10;

$\pi_{\text{ename}} \sigma_{\text{salary > 10000 and dno = 10}} \text{emp}$

emp.ename
'Brindha'
'Parthiban'

Operation involving Selection and Projection

select ename from emp where salary> 5000 and dno> 10

$\pi_{ename}\ \sigma_{salary > 5000 \text{ and } dno > 10}\ emp$

emp.ename

'Aswatha'

Operation involving Selection and Projection

select ename from emp where salary >10000

$\pi_{ename}\ \sigma_{salary > 10000}\ emp$

emp.ename
'Brindha'
'Parthiban'

Cartesian-product Operation

- The Cartesian-product operation (denoted by X) allows to combine information from any two relations.
- select * from emp,dept;
- Example: The Cartesian product of the relations *emp* and dept is written as:

emp X *dept*

emp.eno	emp.ename	emp.dno	emp.salary	dept.dno	dept.dname	dept.address
11	'Aswatha'	11	10000	10	'Marketing'	'Coimbatore'
11	'Aswatha'	11	10000	11	'Accountant'	'Salem'
11	'Aswatha'	11	10000	12	'Purchase'	'Perundurai'
11	'Aswatha'	11	10000	13	'Sales'	'Tiruppur'
11	'Aswatha'	11	10000	14	'Auditing'	'Erode'
13	‘Brindha'	10	15000	10	'Marketing'	'Coimbatore'
13	'Brindha'	10	15000	11	'Accountant'	'Salem'
13	'Brindha'	10	15000	12	'Purchase'	'Perundurai'
13	'Brindha'	10	15000	13	'Sales'	'Tiruppur'
13	'Brindha'	10	15000	14	'Auditing'	'Erode'
22	'Parthiban'	10	20000	10	'Marketing'	'Coimbatore'
22	'Parthiban'	10	20000	11	'Accountant'	'Salem'
22	'Parthiban'	10	20000	12	'Purchase'	'Perundurai'
22	'Parthiban'	10	20000	13	'Sales'	'Tiruppur'
22	'Parthiban'	10	20000	14	'Auditing'	'Erode'
24	'Mugilan'	14	5000	10	'Marketing'	'Coimbatore'
24	'Mugilan'	14	5000	11	'Accountant'	'Salem'
24	'Mugilan'	14	5000	12	'Purchase'	'Perundurai'
24	'Mugilan'	14	5000	13	'Sales'	'Tiruppur'
24	'Mugilan'	14	5000	14	'Auditing'	'Erode'

emp × dept

Rename Operation

- The results of relational-algebra expressions do not have a name to refer them and the rename operator, ρ, provides that purpose
- The expression: $\rho_x(E)$

  returns the result of expression E under the name x
- Another form of the rename operation: $\rho_{x(A1,A2,\ldots An)}(E)$

select ename as Employee_name from emp

emp.Employee_name

'Aswatha'

'Brindha'

'Parthiban'

'Mugilan'

$\rho_{\text{Employee\_name} \leftarrow \text{ename}}\ \pi_{\text{ename}}\ \text{emp}$

Union Operation

- Allows to combine two relations
- Notation: r U s
- For r U s to be valid.
  1. r, s must have the *same* **arity** (same number of attributes)
  2. The attribute domains must be **compatible** (example: 2nd column of r deals with the same type of values as does the 2nd column of s)

select * from dept union select * from dept1

dept.dno	dept.dname	dept.address
10	'Marketing'	'Coimbatore'
11	'Accounts'	'Salem'
12	'Purchase'	'Perundurai'
13	'Sales'	'Tiruppur'
14	'Auditing'	'Erode'
15	'HR'	'Chennai'
16	'Research'	'Bangalore'

dept U dept1

Set-Intersection Operation

- The set-intersection operation allows to find tuples that are in both the input relations.
- Notation: r ∩ s
- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible

select * from dept intersect select * from dept1

dept.dno	dept.dname	dept.address
10	'Marketing'	'Coimbatore'
11	'Accounts'	'Salem'
12	'Purchase'	'Perundurai

dept ∩ dept1

Set-difference Operation

- Allows us to find tuples that are in one relation but are not in another.
- Notation r - s

```
select * from dept except select * from dept1
```

dept.dno	dept.dname	dept.address
13	'Sales'	'Tiruppur'
14	'Auditing'	'Erode'

dept - dept1

Set-difference Operation

- Set differences must be taken between **compatible** relations.
  - r and s must have the same arity
  - attribute domains of r and s must be compatible

select * from dept1 except select * from dept

dept1.dno	dept1.dname	dept1.address
15	'HR'	'Chennai'
16	'Research'	'Bangalore'

dept1 - dept

Additional Relational Algebra Operations

lets user to access and manipulate databases

Join Operation

- The join operation allows to combine a select operation and a Cartesian=Product operation into a single operation
- consider relations r(R) and s(S).
- Let "theta" be a predicate on attributes in the schema R "union" S. The join operation $r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$

JOIN OPERATION

select eno,ename,emp.dno,salary,dept.dno AS d_dno,dname,address from emp JOIN dept on emp.eno=dept.dno

$$\rho_{d\_dno \leftarrow dept.dno}\ \pi_{eno,\ ename,\ emp.dno,\ salary,\ dept.dno,\ dname,\ address}\ (\ emp \bowtie_{emp.eno\ =\ dept.dno} dept\ )$$

emp.eno	emp.ename	emp.dno	emp.salary	dept.d_dno	dept.dname	dept.address
11	'Aswatha'	11	10000	11	'Accounts'	'Salem'
13	'Brindha'	10	15000	13	'Sales'	'Tiruppur'

JOIN OPERATION without Rename Operation

select eno,ename,emp.dno,salary,dept.dno from emp JOIN dept on emp.eno=dept.dno

$$\pi_{\text{eno, ename, emp.dno, salary, dept.dno}} (\text{emp} \bowtie_{\text{emp.eno = dept.dno}} \text{dept})$$

SQL Background

lets user to access and manipulate databases

SQL Parts

- **DML** - provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.
- **Integrity** - the DDL includes commands for specifying integrity constraints.
- **View definition** - The DDL includes commands for defining views.
- **Transaction control** - includes commands for specifying the beginning and ending of transactions.
- **Embedded SQL and dynamic SQL** - define how SQL statements can be embedded within general-purpose programming languages.
- **Authorization** - includes commands for specifying access rights to relations and views.

Data Definition

Used to summarize the data

Data Definition Language

The SQL data-definition language (DDL) allows the specification of information about relations, including:

- The schema for each relation.
- The type of values associated with each attribute.
- The Integrity constraints
- The set of indices to be maintained for each relation.
- Security and authorization information for each relation.
- The physical storage structure of each relation on disk.

Create Table

- An SQL relation is defined using the create table command:

  create table r (A_1 D_1, A_2 D_2, ..., A_n D_n (integrity-constraint$_1$), ..., (integrity-constraint$_k$))

  - r is the name of the relation
  - each A_i is an attribute name in the schema of relation r
  - D_i is the data type of values in the domain of attribute A_i
- Example:

```
create table dept(dno number(8) primary key,dname
varchar2(20),address varchar2(20));
```

Table created.

```
create table emp(eno number(8) primary key,ename
varchar2(20),dno number(8) references dept(dno),salary
number(8);
```

Table created

Create Table

DESC

```
desc dept;
```

TABLE DEPT

Column	Null?	Type
DNO	NOT NULL	NUMBER(8,0)
DNAME	-	VARCHAR2(20)
ADDRESS	-	VARCHAR2(20)

DESC

```
desc emp;
```

TABLE EMP

Column	Null?	Type
ENO	NOT NULL	NUMBER(8,0)
ENAME	-	VARCHAR2(20)
DNO	-	NUMBER(8,0)
SALARY	-	NUMBER(8,0)

Integrity Constraints in Create Table

Types of integrity constraints

- primary key (A_1, ..., A_n)

  The eno in emp and dno in dept
- foreign key (A_m, ..., A_n) references r

  The emp table references dept table
- not null

  In the emp table the salary is modified to add constraint not null

SQL prevents any update to the database that violates an integrity constraint.

Create Table

The emp table is altered and the salary is modified to have not null constraint.

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	ENO	NUMBER	22	8	0	No		
2	ENAME	VARCHAR2	20			Yes	Byte	
3	DNO	NUMBER	22	8	0	Yes		
4	SALARY	NUMBER	22	8	0	No		

Create Table

DESC

```
desc dept;
```

DESC

```
desc emp;
```

TABLE DEPT

Column	Null?	Type
DNO	NOT NULL	NUMBER(8,0)
DNAME	-	VARCHAR2(20)
ADDRESS	-	VARCHAR2(20)

TABLE EMP

Column	Null?	Type
ENO	NOT NULL	NUMBER(8,0)
ENAME	-	VARCHAR2(20)
DNO	-	NUMBER(8,0)
SALARY	-	NUMBER(8,0)

Alter Table

ADD

```
alter table emp add phoneno number(12)
not null;
```

Table altered.

DROP

```
alter table emp drop column phoneno;
```

Table altered.

TABLE EMP

Column	Null?	Type
ENO	NOT NULL	NUMBER(8,0)
ENAME	-	VARCHAR2(20)
DNO	-	NUMBER(8,0)
SALARY	-	NUMBER(8,0)
PHONENO	NOT NULL	NUMBER(12,0

TABLE EMP

Column	Null?	Type
ENO	NOT NULL	NUMBER(8,0)
ENAME	-	VARCHAR2(20)
DNO	-	NUMBER(8,0)
SALARY	-	NUMBER(8,0)

Truncate and Drop table

```
truncate table emp;
```

Table truncated.

```
select * from emp;
```

no data found

```
drop table emp;
```

Table dropped.

```
desc emp
```

ORA-20001: object EMP does not exist

TABLE EMP

ENO	ENAME	DNO	SALARY
11	Aswatha	11	10000
13	Brindha	10	15000
22	Parthiban	10	20000
24	Mugilan	14	25000

Basic Structure of SQL Queries

SQL Query consists of 3
clauses select, from and where

The structure of SQL Query

- A typical SQL query has the form:
  select $A_1, A_2, \ldots, A_n$
  from $r_1, r_2, \ldots, r_m$
  where P
  - A_i represents an attribute
  - R_i represents a relation
  - P is a predicate.
- The result of an SQL query is a relation.

The Select Clause

- Lists the attributes desired in the result of a query
- Corresponds to the projection operation of the relational algebra
- Example: find the names of all employees
  select ename from emp;

The Select Clause

- An asterisk in the select clause denotes "all attributes"
  select * from emp
- An attribute can be a literal with no from clause
  select '200'
  Results is a table with one column and a single row with value "200"
- A column name can be:
  select '200' as FOO
- An attribute can be a literal with from clause
  select 'A' from emp
  Result is a table with one column and N rows (number of tuples in the emp table), each row with value "A"

The Select Clause(cont.)

- The **select** clause can contain arithmetic expressions involving the operation, +, -, *, and /, and operating on constants or attributes of tuples.
- The query:

  select eno, ename,dno, salary/10 **from** emp;

  would return a relation that is the same as the emp relation, except that the value of the attribute salary is divided by "salary/10" can be renamed using
- **select** eno,ename, salary/10 **as** monthly_salary

The Where Clause(cont.)

- The **where** clause specifies conditions that the result must satisfy
  - Corresponds to the selection predicate of the relational algebra.
- To find all employees in dept

```
select ename from emp where dno =10;
```

ENAME
Brindha
Parthiban

The Where Clause(cont.)

- SQL allows the use of the logical connectives **and**, **or**, and **not**
- The operands of the logical connectives can be expressions involving the comparison operators <, <=, >, >=, =, and <>.
- Comparisons can be applied to results of arithmetic expressions
- To find all instructors in Marketing dept (dno 10) with salary > 10000

```
select ename from emp where dno =10 and
salary>10000;
```

ENAME
Brindha
Parthiban

The From Clause(cont.)

- The **from** clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation of the relational algebra.
- Find the Cartesian product dept X emp

```
select * from dept,emp;
```

  - generates every possible dept - emp pair, with all attributes from both relations.
  - Cartesian product are not very useful directly. Useful when combined with where-clause condition (selection operation in relational algebra).

Cartesian Product

For common attributes (e.g., DNO), the attributes in the resulting table has to be renamed using the relation name (e.g., emp.dno)

DNO	DNAME	ADDRESS	ENO	ENAME	DNO	SALARY
10	Marketing	Coimbatore	13	Brindha	10	15000
11	Accountant	Salem	13	Brindha	10	15000
12	Purchase	Perundurai	13	Brindha	10	15000
13	Sales	Tiruppur	13	Brindha	10	15000
14	Auditing	Erode	13	Brindha	10	15000
10	Marketing	Coimbatore	22	Parthiban	10	20000
11	Accountant	Salem	22	Parthiban	10	20000
12	Purchase	Perundurai	22	Parthiban	10	20000
13	Sales	Tiruppur	22	Parthiban	10	20000
14	Auditing	Erode	22	Parthiban	10	20000
10	Marketing	Coimbatore	24	Mugilan	14	25000
11	Accountant	Salem	24	Mugilan	14	25000
12	Purchase	Perundurai	24	Mugilan	14	25000
13	Sales	Tiruppur	24	Mugilan	14	25000
14	Auditing	Erode	24	Mugilan	14	25000
10	Marketing	Coimbatore	11	Aswatha	11	10000
11	Accountant	Salem	11	Aswatha	11	10000
12	Purchase	Perundurai	11	Aswatha	11	10000
13	Sales	Tiruppur	11	Aswatha	11	10000
14	Auditing	Erode	11	Aswatha	11	10000

Set Operations

Used to combine the two or more SQL SELECT statements.

SET OPERATIONS

UNION

- Used to combine the result of two or more SQL SELECT queries.
- The number of data type and columns must be same in both the tables
- Eliminates the duplicate rows from its resultset

UNION ALL

- All operation is equal to the Union operation.
- It returns the set without removing duplication and sorting the data.

SET OPERATIONS

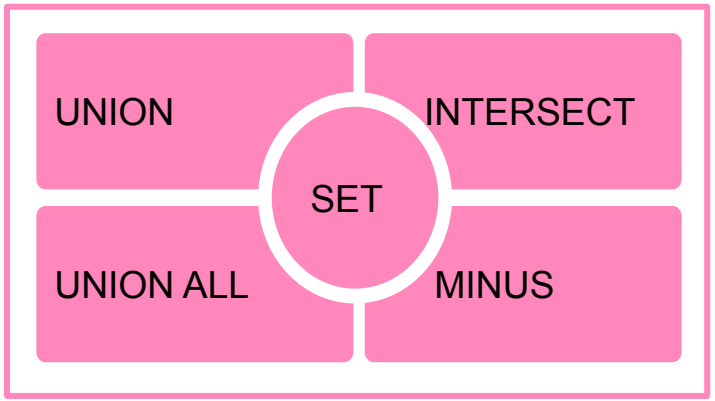
INTERSECT

- Used to combine two SELECT statements
- Returns the common rows from both the SELECT statements
- The number of data type and columns must be the same.
- Has no duplicates
- Arranges the data in ascending order by default.

MINUS

- Combines the result of two SELECT statements.
- Used to display the rows which are present in the first query but absent in the second query.
- Has no duplicates
- Data arranged in ascending order by default.

Type of Set Operations



STUDENT1 TABLE

ROLLNO	NAME	DEPT
1	Iswarya	CS
2	Jeevika	CA
3	Kalaiyarasi	IT
4	Preethi	CS

STUDENT2 TABLE

ROLLNO	NAME	DEPT
2	Jeevika	CA
3	Kalaiyarasi	IT
5	Priyanka	CS
6	Nagomi	CA

Set Operations

UNION

```
select * from student1 UNION select * from
student2
```

ROLLNO	NAME	DEPT
1	Iswarya	CS
2	Jeevika	CA
3	Kalaiyarasi	IT
4	Preethi	CS
5	Priyanka	CS
6	Nagomi	CA

MINUS

```
select * from student1 MINUS  select * from
student2
```

ROLLNO	NAME	DEPT
1	Iswarya	CS
4	Preethi	CS

Set Operations

MINUS

```
select * from student2 MINUS select * from
student1
```

ROLLNO	NAME	DEPT
5	Priyanka	CS
6	Nagomi	CA

INTERSECT

```
select * from student1 INTERSECT select *
from student2
```

ROLLNO	NAME	DEPT
2	Jeevika	CA
3	Kalaiyarasi	IT

Set Operations

UNION ALL

```
select * from student2 UNION ALL select *
from student1
```

ROLLNO	NAME	DEPT
1	Iswarya	CS
3	Kalaiyarasi	IT
4	Preethi	CS
2	Jeevika	CA
2	Jeevika	CA
3	Kalaiyarasi	IT
5	Priyanka	CS
6	Nagomi	CA

Aggregate Functions

Used to summarize the data

Aggregate Functions

- Used to perform the calculations on multiple rows of a single column of a table.
- It returns a single value.

Product Table

PID	PNAME	PRICE	QTY	MFGDATE
100	mysore sandal soap	38.5	5	21-NOV-20
101	amul ghee	475.65	4	17-NOV-20
102	pathanjali soap	35.75	10	16-OCT-20
103	neem paste	47.25	4	30-SEP-20
104	ayur beauty powder	203.54	1	15-AUG-20

round()

round()

```
select round(price) from
product;
```

ROUND(PRICE)
39
476
36
47
204

upper()

upper()

```
select upper(pname) from
product;
```

UPPER(PNAME)
MYSORE SANDAL SOAP
AMUL GHEE
PATHANJALI SOAP
NEEM PASTE
AYUR BEAUTY POWDER

max()

```
select max(price)from
product;
```

max()

MAX(PRICE)
475.65

min()

```
select
min(price)MINIMUM_PRICE
from product;
```

min()

MINIMUM_PRICE
35.75

avg()

```
select
avg(price)AVERAGE_PRICE
from product;
```

avg()

AVERAGE_PRICE
160.138

to_char()

to_char()

```
select mfgdate
MANUFACTURING_DATE,
to_char(mfgdate,'month/dd/yyyy')
DISPLAY_FORMAT from product;
```

MANUFACTURING_DATE	DISPLAY_FORMAT
21-NOV-20	november /21/2020
17-NOV-20	november /17/2020
16-OCT-20	october /16/2020
30-SEP-20	september/30/2020
15-AUG-20	august /15/2020

count(*)

```
select qty,count(*) from
product group by qty;
```

count(*)

QTY	COUNT(*)
1	1
5	1
4	2
10	1

sum()

```
select
sum(price) from
product;
```

sum()

SUM(PRICE)
800.69

sum()

sum() with group by

```
select sum(price)
from product where
qty>1 group by qty;
```

SUM(PRICE)
38.5
522.9
35.75

Null Values

A field that has been left blank during record creation!

NuLL

INSERTING NULL BY DEFAULT

```
insert into emp(eno,ename,dno)
values(25,'Madhavan',14);
insert into emp(eno,ename,dno)
values(26,'Kesavan',11);
```

When values are inserted only to eno,ename and dno then salary will be appended with a null value by default.

TABLE EMP

ENO	ENAME	DNO	SALARY
13	Brindha	10	15000
22	Parthiban	10	20000
24	Mugilan	14	25000
11	Aswatha	11	10000
25	Madhavan	14	-
26	Kesavan	11	-

NULL

IS NULL

```
select ename from emp where
salary is null;
```

2 rows selected.

TABLE EMP

ENAME
Madhavan
Kesavan

IS NOT NULL

```
select ename from emp where
salary is  not null
```

4 rows selected.

TABLE EMP

ENAME
Brindha
Parthiban
Mugilan
Aswatha

NULL

ADDING NULL TO SALARY

```
update emp set salary = salary +
null;
```

6 row(s) updated.

TABLE EMP

ENO	ENAME	DNO	SALARY
13	Brindha	10	-
22	Parthiban	10	-
24	Mugilan	14	-
11	Aswatha	11	-
25	Madhavan	14	-
26	Kesavan	11	-

Nested Subqueries

A subquery nested within another subquery

Subquery

- A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.
- Used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.
- Used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

Subqueries

Subqueries with the SELECT Statement

SELECT * FROM EMP WHERE ENO IN (SELECT ENO FROM EMP WHERE SALARY > 10000) ;

ENO	ENAME	DNO	SALARY
13	Brindha	10	15000
22	Parthiban	10	20000
24	Mugilan	14	25000

Subqueries with the INSERT Statement

INSERT INTO EMP1 SELECT * FROM EMP WHERE ENO IN (SELECT ENO FROM EMP) ;

ENO	ENAME	DNO	SALARY
11	Aswatha	11	10000
13	Brindha	10	15000
22	Parthiban	10	20000
24	Mugilan	14	25000

Subqueries

Subqueries with the UPDATE Statement

UPDATE EMP SET SALARY = SALARY * 0.25 WHERE DNO IN (SELECT DNO FROM EMP1 WHERE DNO >= 11);

ENO	ENAME	DNO	SALARY
11	Aswatha	11	2500
13	Brindha	10	15000
22	Parthiban	10	20000
24	Mugilan	14	6250

Subqueries with the DELETE Statement

DELETE FROM EMP WHERE DNO IN (SELECT DNO FROM EMP1 WHERE DNO > 10);

ENO	ENAME	DNO	SALARY
13	Brindha	10	15000
22	Parthiban	10	20000

Nested Subqueries

```
SELECT DNAME FROM DEPT WHERE
DEPT.DNO IN (SELECT EMP.ENO FROM
EMP WHERE DNO IN( SELECT DEPT.DNO
FROM DEPT WHERE DEPT.DNO> 10) );
```

TABLE EMP , DEPT

DNAME
Accountant

Nested Sub Queries

```
Step 1:SELECT DEPT.DNO FROM DEPT
WHERE DEPT.DNO> 10;
The Inner Query is executed
first.
Step 2:SELECT EMP.ENO FROM EMP
WHERE DNO IN( SELECT DEPT.DNO
FROM DEPT WHERE DEPT.DNO> 10);
```

TABLE EMP,

DNO
11
12
13
14

TABLE Emp,Dept

ENO
11
24

Views

Virtual Table

Views

- A view also has rows and columns as they are in a real table in the database.
- We can create a view by selecting fields from one or more tables present in the database.
- A View can either have all the rows of a table or specific rows based on certain condition.

Views

View using Single Table

```
create view EmpView as select eno,ename,
salary from emp where salary > 10000;
```

View created.

```
create view EnameView as select
eno,ename,dno,salary
from emp order by ename;
```

View created

EMPView

ENO	ENAME	SALARY
13	Brindha	15000
22	Parthiban	20000
24	Mugilan	25000

EnameView

ENO	ENAME	DNO	SALARY
11	Aswatha	11	10000
13	Brindha	10	15000
24	Mugilan	14	25000
22	Parthiban	10	20000

Views

View using Multiple Tables

```
create view EmpDeptView as select eno, ename, salary, dept.dno, dname,
address from emp,dept where emp.dno=dept.dno;
```

View created

EmpDeptView

ENO	ENAME	SALARY	DNO	DNAME	ADDRESS
13	Brindha	15000	10	Marketing	Coimbatore
22	Parthiban	20000	10	Marketing	Coimbatore
11	Aswatha	10000	11	Accountant	Salem
24	Mugilan	25000	14	Auditing	Erode

Views

Update View

```
create or replace view EmpDeptView as select ename, salary,
dept.dno,dname,address from emp,dept where emp.dno=dept.dno;
```

View created

EmpDeptView

ENAME	SALARY	DNO	DNAME	ADDRESS
Brindha	15000	10	Marketing	Coimbatore
Parthiban	20000	10	Marketing	Coimbatore
Aswatha	10000	11	Accountant	Salem
Mugilan	25000	14	Auditing	Erode

Views

Dropping Views

```
drop view EmpView;
```

View dropped.

```
select * from EmpView;
```

ORA-00942: table or view does not exist

Modification of the Database

Insertion, Deletion and Updation

INSERTION

```
insert into emp
values(11,'Aswatha',11,10000)

insert into dept
values(10,'Marketing','Coimbatore');
```

TABLE EMP

ENO	ENAME	DNO	SALARY
11	Aswatha	11	10000
13	Brindha	10	15000
22	Parthiban	10	20000
24	Mugilan	14	25000

TABLE DEPT

DNO	DNAME	ADDRESS
10	Marketing	Coimbatore
11	Accountant	Salem
12	Purchase	Perundurai
13	Sales	Tiruppur
14	Auditing	Erode

Updation

Update with Condition

```
update emp set salary = 20000 where eno=11;
```

1 row(s) updated.

TABLE EMP

ENO	ENAME	DNO	SALARY
11	Aswatha	11	20000
13	Brindha	10	15000
22	Parthiban	10	20000
24	Mugilan	14	25000

Update with Condition

```
update emp set salary = 50000;
```

4 row(s) updated

TABLE EMP

ENO	ENAME	DNO	SALARY
11	Aswatha	11	50000
13	Brindha	10	50000
22	Parthiban	10	50000
24	Mugilan	14	50000

Deletion

Delete with Condition

```
delete from emp where dno=10;
```

2 row(s) deleted.

Deletion

```
delete from emp;
```

2 row(s) deleted.

```
select * from emp;
```

no data found

TABLE EMP

ENO	ENAME	DNO	SALARY
11	Aswatha	11	50000
24	Mugilan	14	50000

Truncate and drop table

```
truncate table emp;
```

Table truncated.

```
select * from emp;
```

no data found

```
drop table emp;
```

Table dropped.

```
desc emp
```

ORA-20001: object EMP does not exist

TABLE EMP

ENO	ENAME	DNO	SALARY
11	Aswatha	11	10000
13	Brindha	10	15000
22	Parthiban	10	20000
24	Mugilan	14	25000

Thanks!

Links

- JOIN TUTORIAL : https://livesql.oracle.com/apex/livesql/file/tutorial_K1PL8IWJPLMM4ZBZSX2K1LRQU.html
- DDL and DML : https://livesql.oracle.com/apex/livesql/s/k3bh424e7vkt1qful22s4pb4i
- Set Operations : https://livesql.oracle.com/apex/livesql/s/k0c2f5da4l35d91jsajoz678p
- Views : https://livesql.oracle.com/apex/livesql/s/k3n2uojej6xf66q2eenlcylua
- Aggregate fun : https://livesql.oracle.com/apex/livesql/s/ky6p8f7m2fz6yjhxc2iseu2w5

Credits

- Abraham Silberschatz ,Henry F. Korth and S. Sudarshan,"Database System Concepts", Seventh Edition, McGRAW Hill Education
- http://slidesgo.com/
- SQL Examples worked out using : https://livesql.oracle.com/
- Relational Algebra worked out using : https://dbis-uibk.github.io/relax/landing