

# DDL

CS 341 Database Systems Lab



# Data Types

---

Data Type	Category	Description
NUMBER	Numeric	Used to store numeric values with high precision
INTEGERs	Numeric	Used to store whole number
FLOAT	Numeric	Used to store approximate numeric values
CHAR	Character	Used to store fixed-length character strings
NCHAR	Character	Used to store fixed-length national character set strings
VARCHAR2	Character	Used to store variable-length character strings
NVARCHAR2	Character	Used to store variable-length national character set strings
CLOB	Large Object	Used to store large character data, extensive text such as lengthy documents.
NCLOB	Large Object	Used to store large national character set data
LONG	Large Object	Used to store variable-length character data
BLOB	Large Object	Used to store binary data such as images, audio, video and other binary data upto 4GB.
RAW	Other	Used to store variable-length binary data
DATE	Date and Time	Used to store date and time information
TIMESTAMP	Date and Time	Used to store more precise date and time information such as in milliseconds.

# Data Types

- **CHAR(len)**: Fixed length character data. If len is given then it can store up to len number of characters. Default width is 1
- **Varchar(len)** - characters variable-length (ANSI Standard)
- **Varchar2(len)** - same as varchar (Oracle Standard)
- **Int** - Integer
- **Decimal (Precision, Scale)** - total length of the digits including both side of decimal, length of digit after decimal
- **Number (Precision, Scale)** - Same as decimal but more popular. If you skip the precision and scale, Oracle uses the maximum range and precision for the number.

# Create Table

- Define the structure of a table. Define the table name, variable names and datatypes.

```
CREATE TABLE table_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    ....  
);
```

```
CREATE TABLE Persons (  
    PersonID int,  
    LastName varchar2(255),  
    FirstName varchar2(255),  
    Address varchar2(255),  
    City varchar2(255)  
);
```

# Integer and Number

- INT is recognized as NUMBER(38) on Oracle
- When you specify NUMBER(38), it means that the number can have up to 38 digits of precision, including both the integer and fractional parts.
- If you're looking to store integer values and don't need to accommodate fractional parts, INT or NUMBER(38) is more than sufficient.
- If you want to ensure it's an integer, you might use NUMBER(p,0) where p is the maximum number of digits you need and 0 digits after decimal.

# Create Table If Not Exists

- Creates the table only if it does not exist in the database.

(Not supported by Oracle)

```
CREATE TABLE IF NOT EXISTS Persons  
(  
    PersonID int,  
    LastName varchar2(255),  
    FirstName varchar2(255),  
    Address varchar2(255),  
    City varchar2(255)  
);
```

# Describe Table

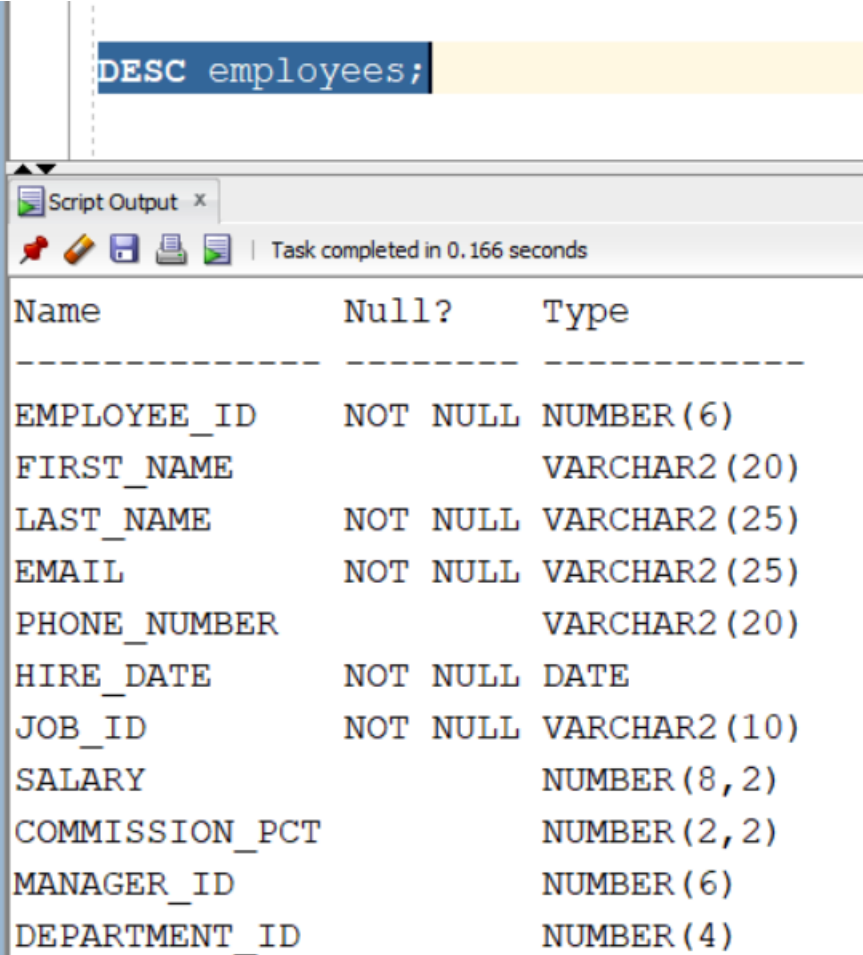
- Describe the structure of the table

Desc Persons;

Describe Persons;

- Helps to see the table structure. You can test it on the employees table

DESC employees;



The screenshot shows a database interface with a command window at the top containing the text `DESC employees;`. Below it is a 'Script Output' window titled 'Script Output x' which displays the result of the command. The output is a table with three columns: 'Name', 'Null?', and 'Type'. The table lists the structure of the 'employees' table, including columns like EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, and DEPARTMENT\_ID, along with their respective data types and nullability constraints.

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER (6)
FIRST_NAME		VARCHAR2 (20)
LAST_NAME	NOT NULL	VARCHAR2 (25)
EMAIL	NOT NULL	VARCHAR2 (25)
PHONE_NUMBER		VARCHAR2 (20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2 (10)
SALARY		NUMBER (8, 2)
COMMISSION_PCT		NUMBER (2, 2)
MANAGER_ID		NUMBER (6)
DEPARTMENT_ID		NUMBER (4)

# Create a copy of a table

## 1. Duplicate structure

(Not supported by Oracle)

```
CREATE TABLE new_table_name LIKE  
existing_table_name;
```

## 2. Duplicate structure and data

```
CREATE TABLE new_table_name AS  
SELECT column1, column2,...  
FROM existing_table_name  
WHERE ....;
```



# Drop or Truncate

Delete the table

```
DROP TABLE table_name;
```

Delete all data in the table

```
TRUNCATE TABLE table_name;
```

# ALTER

Add/delete/modify columns or add/drop constraints

- Add column
- Drop column
- Rename column
- Modify Datatype

```
ALTER TABLE table_name  
ADD column_name datatype;
```

```
ALTER TABLE table_name  
DROP COLUMN column_name;
```

```
ALTER TABLE table_name  
RENAME COLUMN old_name to new_name;
```

```
ALTER TABLE table_name  
MODIFY column_name datatype;
```



# Constraints



# Constraints - Rules for Data

- Limit the type of data that can be in a table.
- If there is any violation between constraint and data action, the action is aborted.
- Constraints:
  - Column-Level
  - Table-level

# Constraints

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different i.e. no duplicates.
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Prevents actions that would destroy links between tables, ensures the references between tables
- **CHECK** - Ensures that the values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column if no value is specified
- **CREATE INDEX** - Used to create and retrieve data from the database very quickly

# Not Null

- Does not accept null values.
- Cannot insert/update without entering a value for this column.

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar2(255) NOT NULL,  
    FirstName varchar2(255) NOT NULL,  
    Age int  
);
```

```
ALTER TABLE Persons  
MODIFY Age int NOT NULL;
```

# Unique

- Unique values – no duplicates

```
CREATE TABLE Persons (  
    ID int NOT NULL UNIQUE,  
    LastName varchar2(255) NOT NULL,  
    FirstName varchar2(255),  
    Age int  
);
```

```
ALTER TABLE Persons  
ADD CONSTRAINT UC_Person UNIQUE (ID, LastName);
```

Naming a  
constraint on the  
table-level allows  
us to access it  
later

```
ALTER TABLE Persons  
DROP CONSTRAINT UC_Person;
```

# Primary Key

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Unique and Not Null

```
CREATE TABLE Persons (  
    ID int NOT NULL PRIMARY KEY,  
    LastName varchar2(255) NOT NULL,  
    FirstName varchar2(255),  
    Age int  
);
```

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar2(255) NOT NULL,  
    FirstName varchar2(255),  
    Age int,  
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)  
);
```



# Foreign Key or Referential Integrity

- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

```
CREATE TABLE Orders (  
    OrderID int NOT NULL PRIMARY KEY,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    FOREIGN KEY (PersonID) REFERENCES Persons(ID));
```

```
CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)  
    REFERENCES Persons(ID)  
);
```

# Check

- The data should meet a condition
- Column level – specific column
- Table level – multiple columns

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar2(255) NOT NULL,  
    FirstName varchar2(255),  
    Age int CHECK (Age>=18)  
);
```

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar2(255) NOT NULL,  
    FirstName varchar2(255),  
    Age int,  
    City varchar2(255),  
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Karachi')  
);
```

# Default

- Set a default value

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar2(255) NOT NULL,  
    FirstName varchar2(255),  
    Age int,  
    City varchar2(255) DEFAULT 'Karachi'  
);
```

```
ALTER TABLE Persons  
MODIFY City DEFAULT 'Karachi';
```

# Auto-increment Primary Key

```
CREATE TABLE Persons (  
    ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    LastName varchar2(255) NOT NULL,  
    FirstName varchar2(255),  
    Age int  
);
```



Oracle also allows you to use sequences to generate the next value based on a starting value and increment intervals.

# Create Data

CRUD

# INSERT

Specify both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

If you are inserting data for all columns of table. Make sure data is entered in the right sequence:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Add multiple records at once (Not supported by Oracle)

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...),
      (value1, value2, value3, ...),
      (value1, value2, value3, ...);
```



You can also import from a file when you have lots of data