Oracle Data Definition Language (DDL)

*Data types

* Constraints

Oracle

- Learn about Data Definition Language (DDL) statements to work with the structure of an Oracle database table.
  - Various data types used in defining columns in a database table.
  - Integrity and value constraints
  - Viewing, modifying, and removing a table structure.

NAMING RULES AND CONVENTIONS

- A table is an object that can store data in an Oracle database.

- When you create a table, you must specify

  1. the table name,
  2. the name of each column,
  3. the data type of each column,
  4. and the size of each column.

NAMING RULES AND CONVENTIONS

- Oracle provides you with different constraints
  - to specify a primary or a composite key for the table,
  - to define a foreign key in a table that references a primary key in another table,
  - to set data validation rules for each column,
  - to specify whether a column allows NULL values,
  - and to specify if a column should have unique values only.

Data Types

- When a table is created, each column in the table is assigned a data type.

- Some important data types:
  - Varchar2
  - Char
  - Number

Varchar2

- The VARCHAR2 type is a character data type to store **variable-length** alphanumeric data in a column.
- The size is specified within parentheses, for example, VARCHAR2(20).
- If the data are smaller than the specified size, only the data value is stored, and trailing spaces are not added to the value.
- VARCHAR2 is the most appropriate type for a column whose values do not have a fixed length.

Char

- The CHAR type is a character data type to store **fixed-length** alphanumeric data in a column.
- The CHAR data type uses the storage more efficiently and processes data faster than the VARCHAR2 type.

Number

- The NUMBER data type is used to store negative, positive, integer, fixed-decimal, and floating-point numbers.

- When a number type is used for a column, its **precision** and **scale** can be specified.

  - Precision is the total number of significant digits in the number, both to the left and to the right of the decimal point.
  - Scale is the total number of digits to the right of the decimal point.

Number -- integer

- An **integer** is a whole number without any decimal part.
- The data type for it would be defined as NUMBER(3), where 3 represents the maximum number of digits.

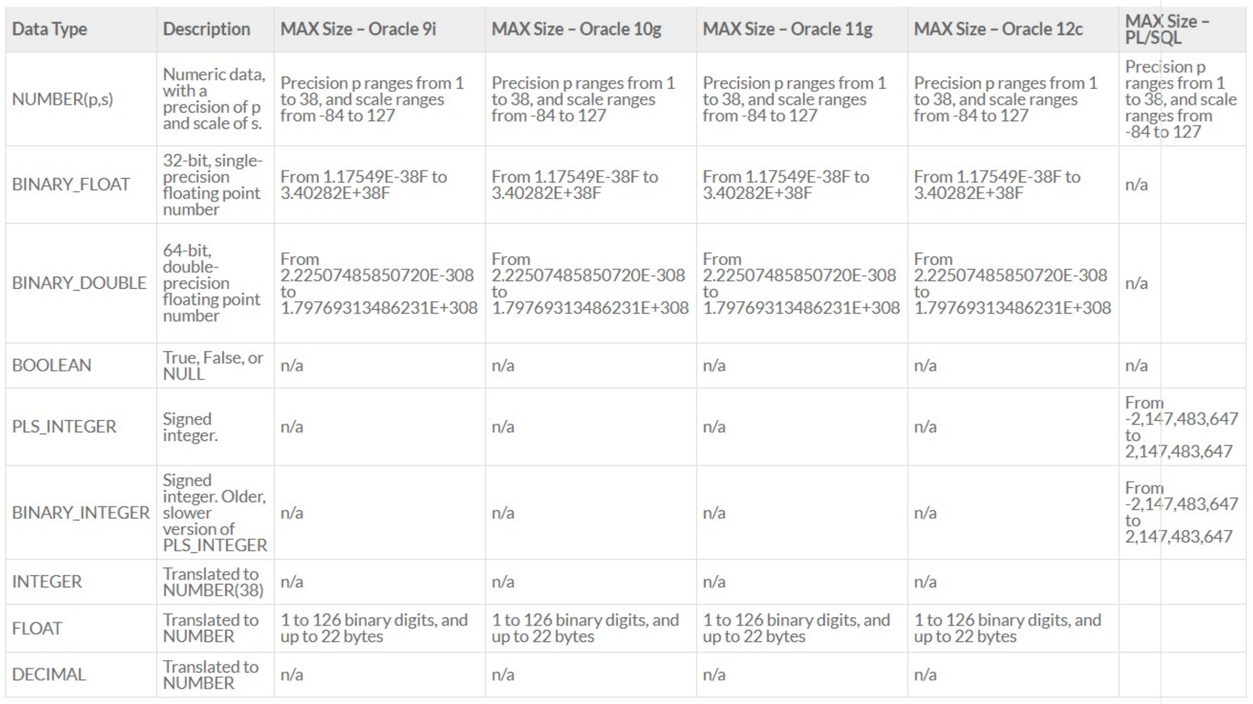
Number – fixed-point

- decimal number has a specific number of digits to the right of the decimal point.
- The PRICE column has values in dollars and cents, which requires two decimal places - for example, values like 2.95, 3.99, 24.99, and so on.
- If it is defined as NUMBER(4,2), the first number specifies the **precision** and the second number the **scale**.

Number – floating-point

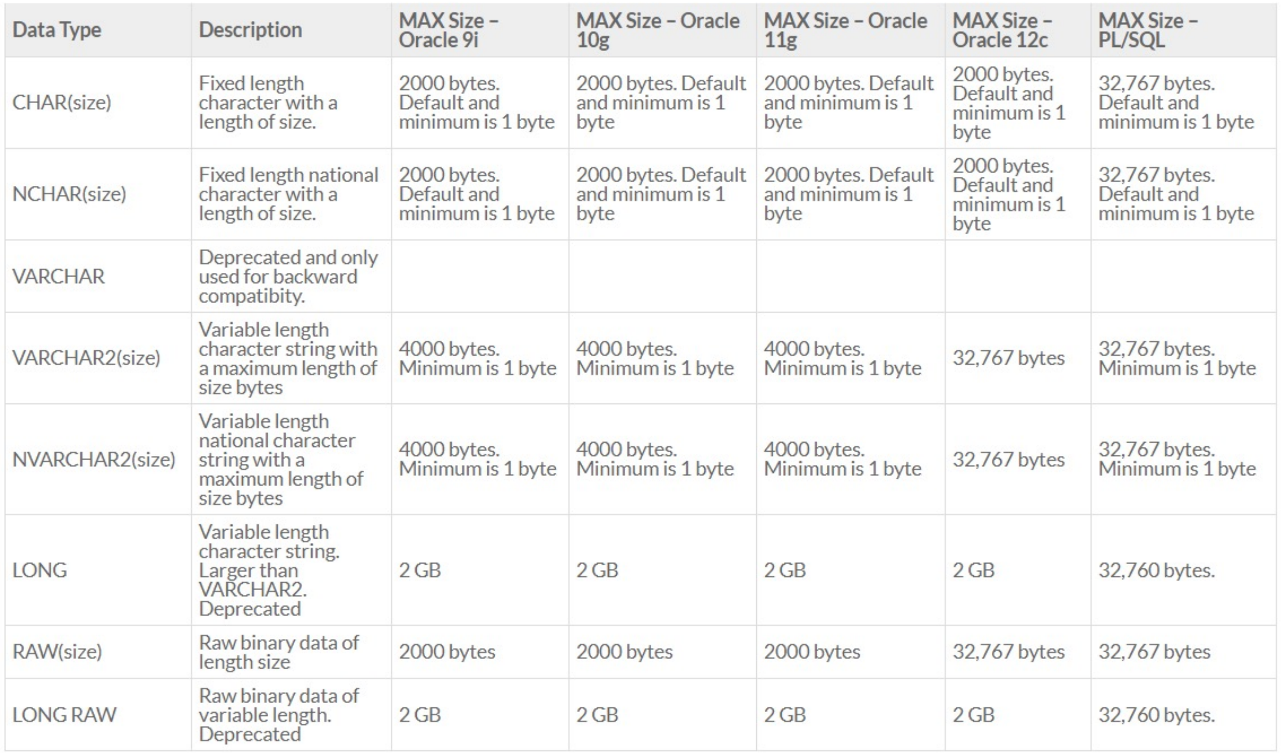
- A **floating-point** decimal number has a variable number of decimal places
- To define such a column, do not specify the scale or precision along with the NUMBER type.
- By defining a column as a floating-point number, a value can be stored in it with very high precision

Numbers



Data Type	Description	MAX Size – Oracle 9i	MAX Size – Oracle 10g	MAX Size – Oracle 11g	MAX Size – Oracle 12c	MAX Size – PL/SQL
NUMBER(p,s)	Numeric data, with a precision of p and scale of s.	Precision p ranges from 1 to 38, and scale ranges from -84 to 127	Precision p ranges from 1 to 38, and scale ranges from -84 to 127	Precision p ranges from 1 to 38, and scale ranges from -84 to 127	Precision p ranges from 1 to 38, and scale ranges from -84 to 127	Precision p ranges from 1 to 38, and scale ranges from -84 to 127
BINARY_FLOAT	32-bit, single-precision floating point number	From 1.17549E-38F to 3.40282E+38F	From 1.17549E-38F to 3.40282E+38F	From 1.17549E-38F to 3.40282E+38F	From 1.17549E-38F to 3.40282E+38F	n/a
BINARY_DOUBLE	64-bit, double-precision floating point number	From 2.22507485850720E-308 to 1.79769313486231E+308	From 2.22507485850720E-308 to 1.79769313486231E+308	From 2.22507485850720E-308 to 1.79769313486231E+308	From 2.22507485850720E-308 to 1.79769313486231E+308	n/a
BOOLEAN	True, False, or NULL	n/a	n/a	n/a	n/a	n/a
PLS_INTEGER	Signed integer.	n/a	n/a	n/a	n/a	From -2,147,483,647 to 2,147,483,647
BINARY_INTEGER	Signed integer. Older, slower version of PLS_INTEGER	n/a	n/a	n/a	n/a	From -2,147,483,647 to 2,147,483,647
INTEGER	Translated to NUMBER(38)	n/a	n/a	n/a	n/a	
FLOAT	Translated to NUMBER	1 to 126 binary digits, and up to 22 bytes	1 to 126 binary digits, and up to 22 bytes	1 to 126 binary digits, and up to 22 bytes	1 to 126 binary digits, and up to 22 bytes	
DECIMAL	Translated to NUMBER	n/a	n/a	n/a	n/a	

Characters/Text



Data Type	Description	MAX Size – Oracle 9i	MAX Size – Oracle 10g	MAX Size – Oracle 11g	MAX Size – Oracle 12c	MAX Size – PL/SQL
CHAR(size)	Fixed length character with a length of size.	2000 bytes. Default and minimum is 1 byte	2000 bytes. Default and minimum is 1 byte	2000 bytes. Default and minimum is 1 byte	2000 bytes. Default and minimum is 1 byte	32,767 bytes. Default and minimum is 1 byte
NCHAR(size)	Fixed length national character with a length of size.	2000 bytes. Default and minimum is 1 byte	2000 bytes. Default and minimum is 1 byte	2000 bytes. Default and minimum is 1 byte	2000 bytes. Default and minimum is 1 byte	32,767 bytes. Default and minimum is 1 byte
VARCHAR	Deprecated and only used for backward compatibity.					
VARCHAR2(size)	Variable length character string with a maximum length of size bytes	4000 bytes. Minimum is 1 byte	4000 bytes. Minimum is 1 byte	4000 bytes. Minimum is 1 byte	32,767 bytes	32,767 bytes. Minimum is 1 byte
NVARCHAR2(size)	Variable length national character string with a maximum length of size bytes	4000 bytes. Minimum is 1 byte	4000 bytes. Minimum is 1 byte	4000 bytes. Minimum is 1 byte	32,767 bytes	32,767 bytes. Minimum is 1 byte
LONG	Variable length character string. Larger than VARCHAR2. Deprecated	2 GB	2 GB	2 GB	2 GB	32,760 bytes.
RAW(size)	Raw binary data of length size	2000 bytes	2000 bytes	2000 bytes	32,767 bytes	32,767 bytes
LONG RAW	Raw binary data of variable length. Deprecated	2 GB	2 GB	2 GB	2 GB	32,760 bytes.

Types of Constraints

There are two types of constraints:

1. *Integrity constraints*: define both the primary key and the foreign key with the table and primary key it references.

2. *Value constraints*: define if NULL values are disallowed, if UNIQUE values are required, and if only certain set of values are allowed in a column.

Naming a Constraint

- The general convention used for naming constraints is

  <table name>_<column name>_<constraint type>

  - *table name* is the name of the table where the constraint is being defined,
  - *column name* is the name of the column to which the constraint applies,
  - and *constraint type* is an abbreviation used to identify the constraint’s type.

Naming a Constraint

For example, a constraint name *emp_deptno_fk* refers to:

- a constraint in table EMP on column DeptNo of type foreign key. A constraint name *dept_deptno_pk* is for a primary key constraint in table DEPT on column DeptNo.

Popular Constraint abbreviations

Primary Key	pk
Foreign Key	fk
Unique	uk
Check	ck
Not Null	nn

Defining a Constraint

- A constraint can be created at the same time the table is created, or it can be added to the table afterward. There are two levels where a constraint is defined:
  - Column level.
  - Table level.

Column level

- A column-level constraint references a single column and is defined along with the definition of the column.
- Any constraint can be defined at the column level except for a FOREIGN KEY and COMPOSITE primary key constraints.

Column datatype [CONSTRAINT constraint_name] constraint_type

Example:

```
    Building VARCHAR2(7) CONSTRAINT location_building_nn NOT NULL
```

Table level

- A table-level constraint references one or more columns and is defined separately from the definitions of the columns.
- Normally, it is written after all columns are defined.
- All constraints can be defined at the table level except for the NOT NULL constraint.

[CONSTRAINT constraint_name] constraint_typ (Column, . . .),

Example:

CONSTRAIN location_roomid_pk PRIMARY KEY(Roomid)

The Primary Key Constrain

- The PRIMARY KEY constraint is also known as the **entity integrity constraint**
- It creates a primary key for the table. A table can have only one primary key constraint.
- If a table uses more than one column as its primary key (i.e., a composite key), the key can only be declared at the table level.

The Primary Key Constrain

- At the column level, the constraints is defined by

  DeptId NUMBER (2) CONSTRAINT dept_deptid_pk PRIMARY KEY,

- At the table level, the constraint is defined by

  CONSTRAINT dept_deptid_pk PRIMARY KEY(DeptId),

The FOREIGN KEY Constraint

- The FOREIGN KEY constraint is also known as the **referential integrity constraint.**

- It uses a column or columns as a foreign key, and it establishes a relationship with the primary key of the same or another table.

The FOREIGN KEY Constraint

- To establish a foreign key in a table, the other referenced table and its primary key must already exist.
- Foreign key and referenced primary key columns <u>need not have the same name</u>, but a foreign key value **must match** the value in the parent table's primary key value or be NULL

The FOREIGN KEY Constraint

- At the table level **ONLY**

  CONSTRAINT student_facultyid_fk FOREIGN KEY(FacultyId)

  REFERENCES faculty (FacultyId),

The NOT NULL Constraint

- The NOT NULL constraint ensures that the column has a value and the value is not a null value
- A space or a numeric zero is not a null value
- At the column level **ONLY**, the constraint is defined by:

```
Name VARCHAR2(15) CONSTRAINT faculty_name_nn NOT NULL,
```

The UNIQUE Constraint

- The UNIQUE constraint requires that every value in a column or set of columns be unique.

- At the table level, the constraint is defined by

  CONSTRAINT dept_deptname_uk UNIQUE(DeptName),

- At the column level, the constraint is defined by:

  DeptName VARCHAR2(12) CONSTRAINT dept_deptname_uk UNIQUE,

The CHECK Constraint

- The CHECK constraint defines a condition that every row must satisfy
- At the column level, the constraint is defined by
  DeptId NUMBER(2) CONSTRAINT dept_deptid_cc
  CHECK((DeptId >= 10) and (DeptId <= 99)),
- At the table level, the constraint is defined by:
  CONSTRAINT dept_deptid_cc
  CHECK((DeptId >= 10) and (DeptId <= 99)),

CREATING AN ORACLE TABLE

A table is created as soon as the CREATE statement is successfully executed by the Oracle server. The general syntax of CREATE TABLE statement is

CREATE TABLE [schema.] tablename

(column1 datatype [CONSTRAINT constraint_name] constraint_type . . .,

(column2 datatype [CONSTRAINT constraint_name] constraint_type,

[CONSTRAINT constraint_name] constraint_type (column, . . .), . . .);

Create Table example

```
CREATE TABLE student
     (StudentId   CHAR (5),
   Last       VARCHAR2 (15) CONSTRAINT student_last_nn NOT NULL,
   First      VARCHAR2 (15) CONSTRAINT student_first_nn NOT NULL,
   Street           VARCHAR2 (25),
   City       VARCHAR2 (15),
   State      CHAR (2) DEFAULT 'NJ',
   Zip        CHAR (5),
   StartTerm        CHAR (4),
   BirthDate        DATE,
   FacultyId        NUMBER (3),
   MajorId          NUMBER (3),
   Phone            CHAR (10),
   CONSTRAINT student_studentid_pk PRIMARY KEY (StudentID));
```

Viewing a Table’s Structure

- The SQL*Plus command to view a table’s structure is **DESCRIBE**, which does not need a semicolon at the end because it is not a SQL statement.

```
SQL>  DESCRIBE student
```

Adding a New Column to an Existing Table

- The general syntax to add a column to an existing table is

ALTER TABLE tablename

ADD columnname datatype;

```
SQL> ALTER TABLE student
  2  ADD SocialSecurity CHAR(9);
Table altered.
SQL>
```

Modifying an Existing Column

- *The general syntax to modify an existing column is*

  ALTER TABLE tablename

  MODIFY columnname newdatatype;

where newdatatype is the new data type or the new size for the column.

```
SQL> ALTER TABLE student
  2  MODIFY SocialSecurity VARCHAR2(11);
Table altered.
SQL>
```

Adding a Constraint

- To add a constraint using ALTER TABLE, the syntax for table level constraint is used. The general syntax of ALTER TABLE is

  ALTER TABLE tablename

  ADD [CONSTRAINT constraint_name] constraint_type (column, …),

```
SQL> ALTER TABLE COURSE
  2  ADD CONSTRAINT COURSE_PREREQ_FK FOREIGN KEY (PREREQ)
  3     REFERENCES COURSE(COURSEID);
Table altered.
SQL>
```

Displaying Table Information

- When a user creates a table or many tables in the database, Oracle tracks them using its own data dictionary
- Viewing a User's Table Names

```
SELECT TABLE_NAME FROM USER_TABLES;
```

- To display all information:

```
SELECT * FROM USER_TABLES;
```

Dropping a Column

- The general syntax is

  ALTER TABLE tablename DROP COLUMN columnname;

Dropping a Table

- The general syntax is

  DROP TABLE tablename [CASCADE CONSTRAINTS];

- For example,

  DROP TABLE sample;

- Oracle displays a "Table dropped" message when a table is successfully dropped.
- If you add optional CASCADE CONSTRAINTS clause, it removes foreign key references to the table also.