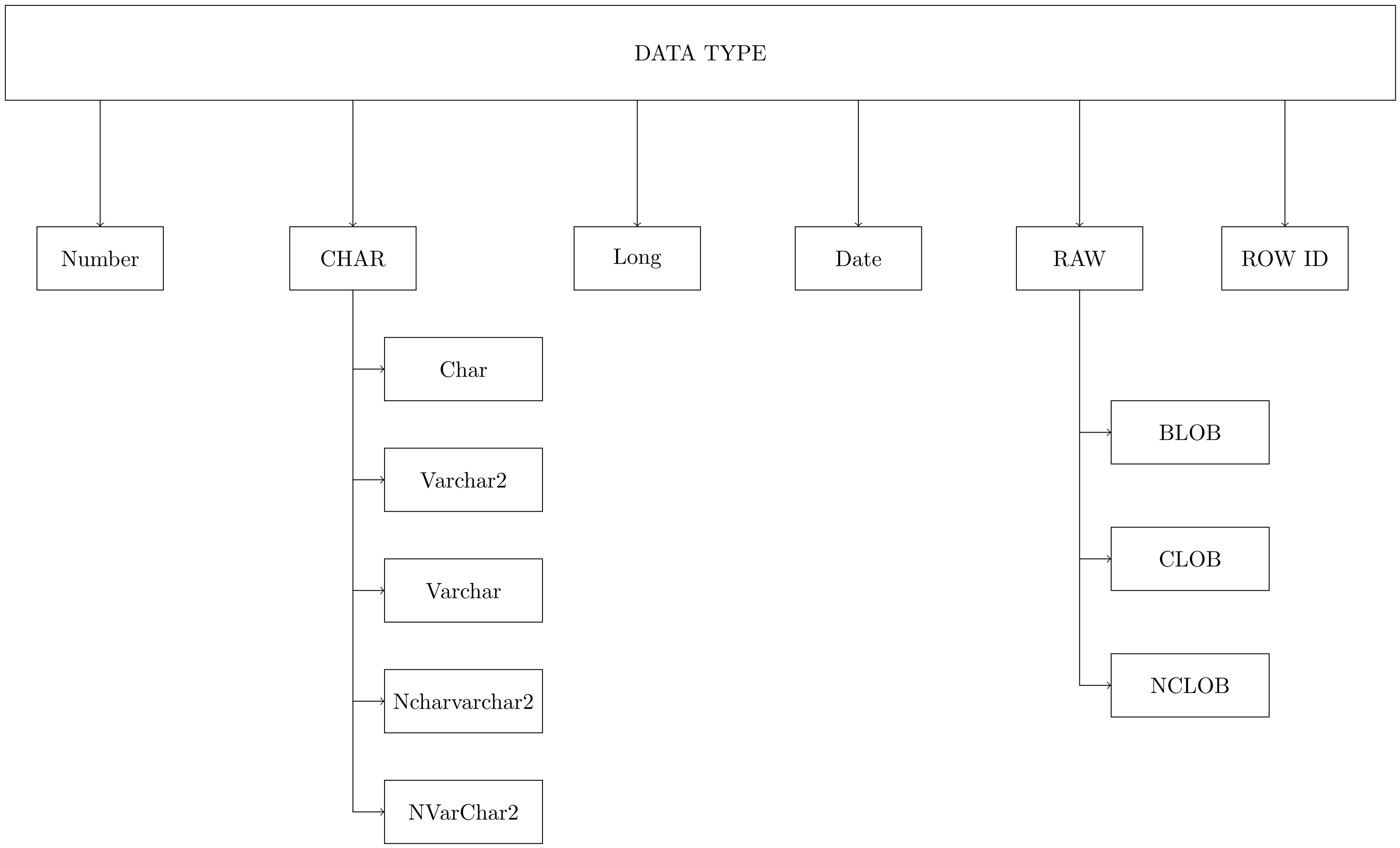
1 Data Types

- Number :
- Char :
  - Char
  - Varchar
  - Varchar2
  -
  -
- Date :
- Long :
- Raw :
  - Blob :
  - Clob :
  - Nclob :
- Row ID :



2 Classification Of SQL Statement

- DDL :
  - Create :
  - Alter :
  - Drop :
  - Truncate :
  - Rename :
- DRL/SQL :
- DML :
  - Insert :
  - Update :
  - Delete :

- DCL :
  - Grant :
  - Revoke :
- TCL :
  - Commit :
  - Rollback :
  - SavePoint :

3 DDL Commands

3.1 Create Table

Definition

To create a table in oracle sql we just have to give the table a name and define each column known as attribut by giving each of them a name , a dataType and an optional constraint that can be added in same line of attribut definition or on its own line , we will see contraint in details in the next section

Syntax :

Table Creation

create table <table_name>(
<attribute$_1$> <DataType$_1$> [Constraint$_1$],
<attribute$_2$> <DataType$_2$> [Constraint$_2$],
....
<attribute$_n$> <DataType$_n$> [Constraint$_n$],
[table contraints]
);

Example :

let's create student table

```
create table student (
id number,
firstname varchar2(50),
lastname varchar2(50),
grade number(2,2)
);
```

3.2 Table Constraints

Definition

Constraints are conditions set on the columns (attributes) of a table to ensure data integrity and consistency. Constraints can be defined:

- During table creation, either on the same line as the attribute definition or on a separate line
- After table creation using the ALTER TABLE command

There are two types of constraints: static and dynamic.

Static Constraints

Static constraints are fixed conditions that do not change based on data input.

- **NOT NULL**: Ensures that the attribute must have a value when inserting into the table.
- **UNIQUE**: Ensures that each value in the attribute is distinct. Unlike PRIMARY KEY, it allows null values.
- **PRIMARY KEY**: Combines UNIQUE and NOT NULL properties to ensure each value is unique and not null. Used to identify rows uniquely.
- **FOREIGN KEY**: References a primary key from another table to establish a relationship between tables.
- **DELETE ON CASCADE**: When deleting a row from the referenced (parent) table, all rows in the child table that contain the matching foreign key are also deleted.

Dynamic Constraints

Dynamic constraints apply conditions that can change based on specified criteria.

- **CHECK**: Validates a specified condition before allowing data to be inserted or updated.
- **DEFAULT**: Sets a default value for the attribute if no value is provided during insertion.

Syntax

Inline Constraint

```
<attribute_i> <DataType_i> not null
<attribute_i> <DataType_i> unique
<attribute_i> <DataType_i> primary key
<attribute_i> <DataType_i> references  referenced_table(references_attribute)
<attribute_i> <DataType_i> default (value)
```

Outline Constraint

constraint <contraint_name> <attribute$_i$> not null
<attribute$_i$> not null

constraint <contraint_name> <attribute$_i$> unique
<attribute$_i$> unique

constraint <contraint_name> <attribute$_i$> primary key
primary key (attribute$_1$,..., attribute$_n$)

constraint <contraint_name> foreign key <attribute$_i$> references referenced_table(references_attribute)
foreign key <attribute$_i$> references referenced_table(references_attribute)

constraint <contraint_name> <attribute$_i$> default (value)
<attribute$_i$> default (value)

Example :

let's create a new table section and recreate the student table with constraints

Creating Section Table

Inline Method

```
    create table section (
    id_section number primary key,
    name varchar2(5) not null
    );
```

Outline Method

```
    create table section (
    id_section number,
    name varchar2(5),
    name not null,
    constraint pk_sec primary key (id_section)
    );
```

Create Student Table
Inline Method

```
    create table student (
    id number primary key,
    lastname varchar2(50) not null,
    firstname varchar2(50) not null,
    id_section number references section(id_section) on delete cascade,
    grade number(4,2) default 00.00 check (grade between 0 and 20),
    dob date not null check (dob<= add_months(sysdate,-18*12))
    );
```

Outline Mehtod

```
    create table student (
    id number,
    constraint pk_student primary key(id),
    lastname varchar2(50),
    firstname varchar2(50),
    constraint nn_student_lastname lastname not null,
    constraint nn_student_firstname firstname not null,
    id_section number,
    constraint fr_student foreign key (id_section) references section(id_section) on delete cascade,
    grade number(4,2),
    grade default 00.00,
    constraint chk_student_grade check (grade between 0 and 20),
    dob date not null,
    constraint chk_student_dob check (dob<= add_months(sysdate,-18*12))
    );
```

Note

Name Convention Of Constraint

- Primary Key : PK_<tableName>
- Foreign Key : FK_<tableName>
- Unique : UQ_<tableName>_<columnName>
- Check : CHK_<tableName>_<columnName>
- Default : DF_<tableName>_<columnName>
- Not Null : NN_<tableName>_<columnName>

Constraint Name Must Be Unique
Tables inside the same PDB (pluggable data base) can't share the same constraints name
Multiple Constraints
It is possible to define multiple constraints on a single attribute using the inline method. However, with the outline method, each constraint needs to be specified individually.

3.3 Delete Table

Definition

We can delete table using the drop command

3.3.1 Syntax

Table Deletion

drop table <tableName>;

3.3.2 Example

lets delete the section table we created

```
drop table section;
```

3.4 Rename Table

Definition

We can rename tables by using the rename command

Syntax

Renaming Table

rename <old_tableName>to <new_tableName>;

Example

```
rename section to mama;
```

3.5 Alter Table

Definition

The ‘ALTER‘ command is a versatile command that allows us to change various aspects of a table:

- Columns
  - **Renaming Column**: Rename the column.
  - **Modify Column**: Change the constraint and data type.
  - **Add Column**: Add a new column.
  - **Remove Column**: Remove a column.
- Constraints
  - **Add Constraint**: Add a new constraint.
  - **Remove Constraint**: Remove a constraint.
  - **Enable Constraint**: Enable an already existing constraint.
  - **Disable Constraint**: Disable an already existing constraint without deleting it.

Syntax

Columns Modification

Renaming Column
alter table <tableName>rename column <old_columnName>to <new_columnName>;
Modify Column
alter table <tableName>modify (columnName [new column definition & constraints]);
Add Column
alter table <tableName>add (columnName [column definition & constraints]);
Remove Column
alter table <tableName>drop column <columnName>;

Constraints

Rename Constraint
alter table <tableName>rename constraint <old_constraintName>to <new_constraintName>;
Add Constraint
alter table <tableName>add constraint <constraintName>[Constraint];
Remove Constraint
alter table <tableName>drop constraint <constraintName>;
Enable Constraint
alter table <tableName>enable constraint <constraintName>;
Disable Constraint
alter table <tableName>disable constraint <constraintName>;

Example

3.6 Truncate Table

Definition

To remove all rows from a table efficiently we use the truncate command

Syntax

Truncing Table

truncate table <tableName>;

Example

lets delete all records from student table

```
    truncate table student;
```

4 DRL Commands

4.1 Select

Definition

To display the contents of one or more tables at once, we use the SELECT command. We can choose specific columns and tables to display, and if a table name is too long, we can assign a shorter alias to columns and tables name using the AS keyword. When selecting from multiple tables without a join condition, a Cartesian product occurs, meaning each row from one table is paired with each row from the other.

Syntax

Table Selection

SELECT * FROM table$_1$;
SELECT column$_1$,..., column$_n$ FROM table$_1$;
SELECT t$_1$.col$_1$,...,t$_1$.col$_n$, t$_2$.col$_1$,...,t$_2$.col$_n$ FROM table$_1$ AS t$_1$, table$_2$ AS t$_2$;

4.2 Where

Definition

The WHERE clause is used to filter rows in a table when displaying data with the SELECT command. Only rows that meet the specified condition(s) are shown in the result.

Syntax

Where Clause

SELECT column$_1$, column$_2$, ... FROM table WHERE [Conditions];

4.3 Order By

Definition

We can sort the results of a query in either ascending or descending order using ORDER BY. This can be applied to one or multiple columns. The order of the columns specified is important; the database first sorts by the first column, and if there are rows with identical values in that column, it then sorts those rows by the next column, and so on. This allows for a prioritized sorting strategy.

Syntax

Order By Clause

SELECT column$_1$, column$_2$, ... FROM table WHERE [Conditions] ORDER BY column$_1$ DESC ,..., column$_n$ ASC;

4.4 Group By

Definition

To group rows that have the same value in a specified column, we use the GROUP BY command. We can group by multiple columns; the order is important because it will first group by the first column. If there are rows that have the same value in the first column but differ in the second column, those rows will appear in separate groups in the output. This allows us to apply aggregate functions to summarize data for each group.

Syntax

Group By Clause

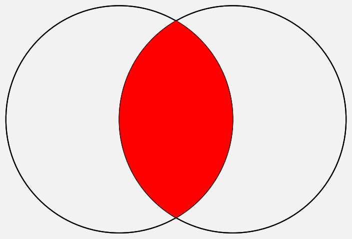
SELECT column$_1$, column$_2$, ... FROM table WHERE [Conditions] GROUP BY column$_1$, ... ,column$_n$
ORDER BY column$_1$ DESC ,..., column$_n$ ASC;

4.5 Joins

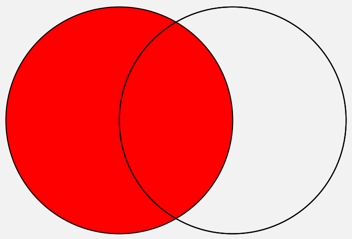
Definition

joins allow you to combine rows from two or more tables based on related columns (referenced key)

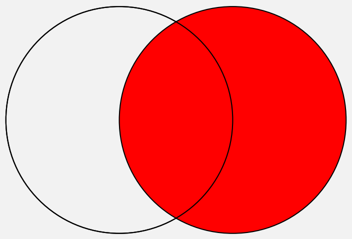
Inner Join An Inner Join returns only the common rows between tables



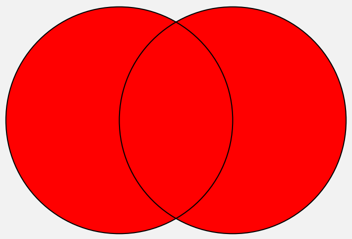
Left Join A Left Join returns all rows from the left table and the matched rows from the right table. If there's no match, NULL values are returned for columns from the right table.



Right Join A Right Join returns all rows from the right table and the matched rows from the left table. If there's no match, NULL values are returned for columns from the left table.



Full Join A Full Join returns all rows when there is a match in either left or right table. If there is no match, NULL values are returned for unmatched columns.



Syntax

Joins

Inner Join
SELECT column_1,...,column_n FROM table1 INNER JOIN table2 ON table1.common_column = table2.common_column;
Left Join
SELECT column_1,...,column_n FROM table1 LEFT JOIN table2 ON table1.common_column = table2.common_column;
Right Join
SELECT column_1,...,column_n FROM table1 RIGHT JOIN table2 ON table1.common_column = table2.common_column;
Full Join
SELECT column_1,...,column_n FROM table1 FULL OUTER JOIN table2 ON table1.common_column = table2.common_column;

4.6 Aggregation Functions

Definition

Aggregation functions perform calculations on a set of values and return a single result. They are commonly used in conjunction with the GROUP BY clause to summarize data.

- **Avg()** : Calculates the average (mean) of numeric values in a specified column.
- **Min()** : Returns the smallest (minimum) value in a specified column.
- **Max()** : Returns the largest (maximum) value in a specified column.
- **Count()** : Counts the number of non-null entries in a specified column (or all rows if * is used).
  - **count(*)** : Counts All rows
  - **count(column$_i$)** : counts number of rows where column$_i$ is not null
  - **count(distinct column$_i$)** : counts number of rows where column$_i$ is not null without repetition
- **Sum()** : Adds up all values in a specified numeric column.

4.7 Operators

Definition

Operators are symbols that specify operations to be performed on operands. They can be categorized as follows:

- **Logical Operators**: Used to combine conditions.
  - Logical And : AND
  - Logical Or : OR
  - Logical Not : NOT
- **Comparison Operators**: Used to compare values.
  - Equal : =
  - Not Equal : !=
  - Greater : >
  - Greater Or Equal : >=
  - Less : <
  - Less Or Equal : <=
  - Between : BETWEEN value_1 AND value_2
  - In : IN (set of values)
- **Arithmetic Operators**: Used for mathematical calculations.
  - Multiplication : *
  - Division : /
  - Sum : +
  - Subtraction : -

5 DML Commands

5.1 Insert

Definition

To insert rows into a table, we use the INSERT command. We can insert one row at a time or multiple rows at once from the same or different tables using the ALL keyword.

Syntax

Insert

Insert One Row At A Time
INSERT INTO tableName (column$_1$,...,column$_n$) VALUES (value$_1$,...,value$_n$);
Insert Multiple Rows
INSERT ALL
INTO tableName$_1$ (column$_1$,...,column$_n$) VALUES (value$_1$,...,value$_n$);
INTO tableName$_2$ (column$_1$,...,column$_n$) VALUES (value$_1$,...,value$_n$);
....
INTO tableName$_n$ (column$_1$,...,column$_n$) VALUES (value$_1$,...,value$_n$);
SELECT * FROM dual;

5.2 Update

Definition

To change the values of some rows in a table, we use the UPDATE command, accompanied by the WHERE clause to update only specific rows.

Syntax

Update

UPDATE tableName SET column$_1$ = value$_1$,...,column$_n$ = value$_n$
WHERE [condition];

5.3 Delete

Definition

To delete rows from a table, we use the DELETE command, accompanied by the WHERE clause to delete specific rows. Although it is possible to delete all rows using DELETE, it is better to use TRUNCATE for that purpose due to performance considerations.

Syntax

Delete

DELETE SPECIFIC ROWS
DELETE FROM tableName WHERE [condition];
DELETE ALL ROWS
DELETE FROM tableName;