Oracle

Oracle??

- Relational Database Management System
- SQL – Structured Query Language
- PLSQL - Procedural Language extensions to SQL

Basic Data Types

Character Datatype

char(n)

- Fixed-length character data (string), n characters long
- The maximum size for n is 2000 bytes
- string of type char is always padded on right with blanks to full length of n
- **can be memory consuming**

Example: char(40)

varchar2(n)

- Variable-length character string
- The maximum size for n is 4000
- Only the bytes used for a string require storage

Example: varchar2(80)

Numeric Datatype

number(o , d)

Numeric data type for integers and reals
o = overall number of digits
d= number of digits to the right of the decimal point.

Maximum values: o =38, d= - 84 to +127. Examples: number(8), number(5,2)

Eg:number(5,2) cannot contain anything larger than 999.99 without resulting in an error
Data types derived from number are int[eger], dec[imal], smallint , real.

Long

Character data up to a length of 2GB. Only one long column is allowed per table.

Date Datatype

date

Date data type for storing date and time.

The default format for a date is: DD-MMM-YY. Examples: '13-OCT-94', '07-JAN-98'

Large Object Datatype

Data Type	Details	Explanation
bfile	Maximum file size of 2^{64}-1 bytes.	File locators that point to a binary file on the server file system (outside the database).
blob	Store up to (4 gigabytes -1) * (the value of the CHUNK parameter of LOB storage).	Stores unstructured binary large objects.
clob	Store up to (4 gigabytes -1) * (the value of the CHUNK parameter of LOB storage) of character data.	Stores single-byte and multi-byte character data.
nclob	Store up to (4 gigabytes -1) * (the value of the CHUNK parameter of LOB storage) of character text data.	Stores unicode data.

Query Categories

Categories

- DDL
  - Deals with the structure /schema
- DML
  - For data processing
- DCL
  - Deals with Security of data
- DQL
  - For data retrieval

Data Definition Language

Create

```
CREATE TABLE table_name
(
  column1 datatype [ NULL | NOT NULL ],
  column2 datatype [ NULL | NOT NULL ],
  ...
  column_n datatype [ NULL | NOT NULL ]
);
```

```
CREATE TABLE customers
( customer_id number(10) NOT NULL,
  customer_name varchar2(50) NOT NULL,
  city varchar2(50)
);
```

Practice Exercise # 1

Create an Oracle table called suppliers that stores supplier ID, name, and address information

Solution - Exercise # 1

```
CREATE TABLE suppliers
( supplier_id number(10) NOT NULL,
  supplier_name varchar2(50) NOT NULL,
  address varchar2(50),
  city varchar2(50),
  state varchar2(25),
  zip_code varchar2(10)
);
```

Problems??

```
CREATE TABLE suppliers
( supplier_id number(10) NOT NULL,
  supplier_name varchar2(50) NOT NULL,
  address varchar2(50),
  city varchar2(50),
  state varchar2(25),
  zip_code varchar2(10)
);
```

Primary Key

- **Primary key** is a single field or combination of fields that uniquely defines a record
- None of the fields that are part of the primary key can contain a null value
- A table can have only one primary key
- Cannot contain more than 32 columns
- Can be defined in either
  - CREATE TABLE statement
  - ALTER TABLE statement.

Primary Key

During Table Creation stage

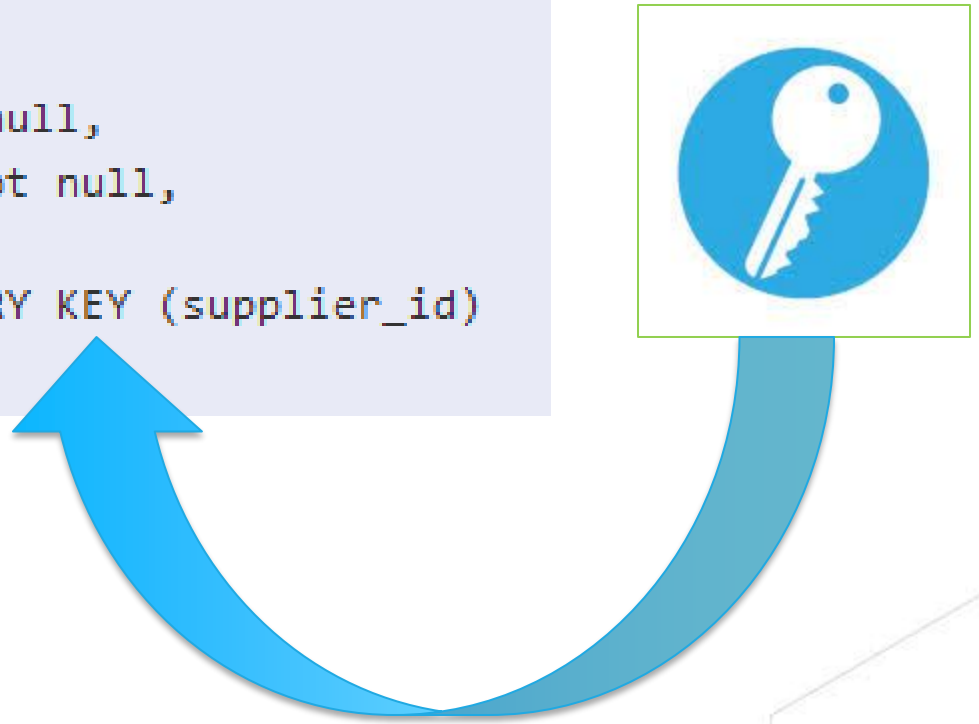
```
CREATE TABLE table_name
(
  column1 datatype null/not null,
  column2 datatype null/not null,
  ...

  CONSTRAINT constraint_name PRIMARY KEY (column1, column2, ... column_n)
);
```

Primary Key??

```
CREATE TABLE suppliers
( supplier_id number(10) NOT NULL,
  supplier_name varchar2(50) NOT NULL,
  address varchar2(50),
  city varchar2(50),
  state varchar2(25),
  zip_code varchar2(10)
);
```

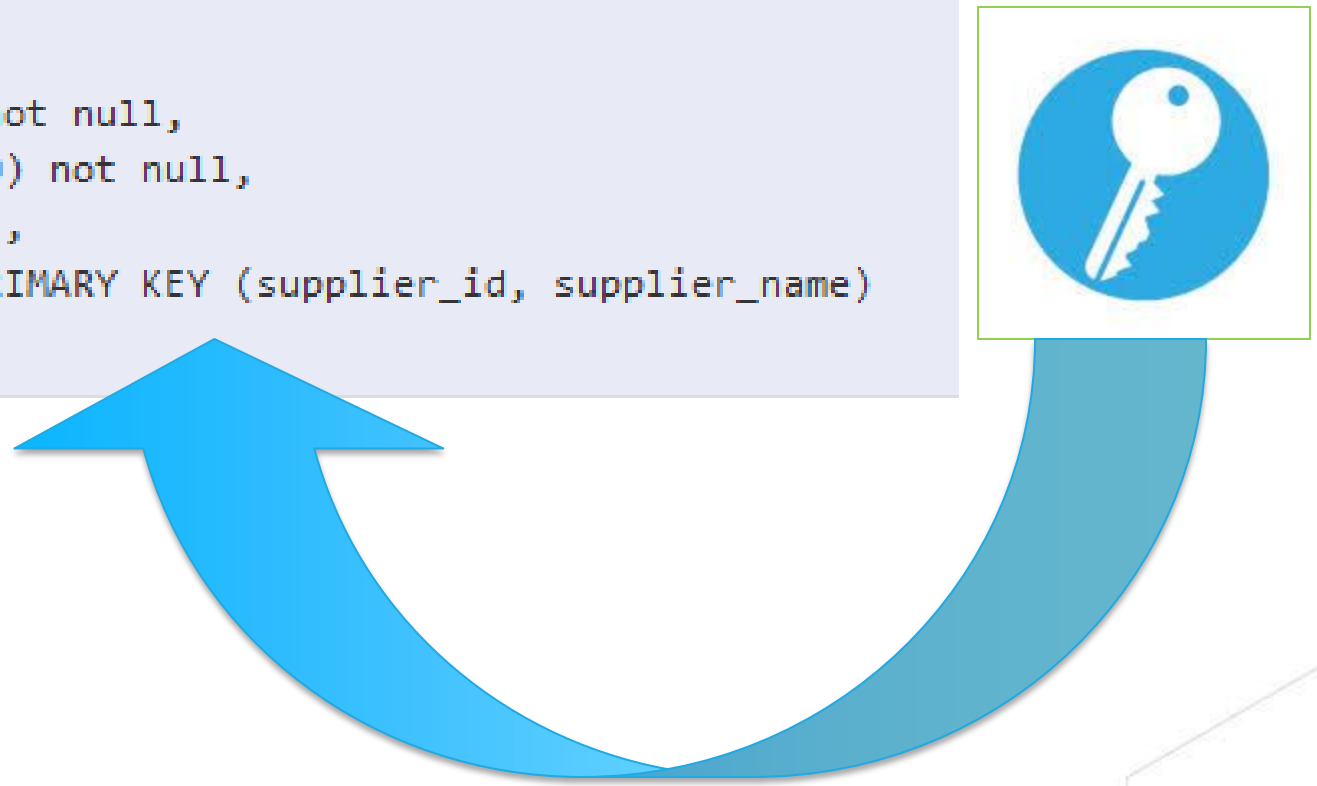
Solution # 1

```
CREATE TABLE supplier
(
  supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
);
```



solution #2

```
CREATE TABLE supplier
(
  supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id, supplier_name)
);
```



Using Alter Query

Primary Key

In existing table

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name PRIMARY KEY (column1, column2, ... column_n);
```

Primary Key

Drop

```
ALTER TABLE supplier
DROP CONSTRAINT supplier_pk;
```

Disable

```
ALTER TABLE supplier
DISABLE CONSTRAINT supplier_pk;
```

Enable

```
ALTER TABLE supplier
ENABLE CONSTRAINT supplier_pk;
```

Solution # 1

```
CREATE TABLE suppliers
( supplier_id number(10) NOT NULL,
  supplier_name varchar2(50) NOT NULL,
  address varchar2(50),
  city varchar2(50),
  state varchar2(25),
  zip_code varchar2(10)
);
```

```
ALTER TABLE supplier
ADD CONSTRAINT supplier_pk PRIMARY KEY (supplier_id);
```

Problem??

```
CREATE TABLE suppliers
( supplier_id number(10) NOT NULL,
  supplier_name varchar2(50) NOT NULL,
  address varchar2(50),
  city varchar2(50),
  state varchar2(25),
  zip_code varchar2(10)
);
```

```
ALTER TABLE supplier
ADD CONSTRAINT supplier_pk PRIMARY KEY (supplier_id);
```

How to refer the supplier id in another table ????

Foreign Key

- A way to enforce referential integrity within your Oracle database
- A foreign key means that values in one table must also appear in another table
- The referenced table is called the ***parent table*** while the table with the foreign key is called the ***child table***
- The foreign key in the child table will generally reference a primary key in the parent table
- A foreign key can be defined in either a CREATE TABLE statement or an ALTER TABLE statement

Foreign Key

```
CREATE TABLE table_name
(
  column1 datatype null/not null,
  column2 datatype null/not null,
  ...

  CONSTRAINT fk_column
    FOREIGN KEY (column1, column2, ... column_n)
    REFERENCES parent_table (column1, column2, ... column_n)
);
```

Foreign Key

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
);

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10) not null,
  CONSTRAINT fk_supplier
    FOREIGN KEY (supplier_id)
    REFERENCES supplier(supplier_id)
);
```

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id, supplier_name)
);

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  CONSTRAINT fk_supplier_comp
    FOREIGN KEY (supplier_id, supplier_name)
    REFERENCES supplier(supplier_id, supplier_name)
);
```

Foreign Key

```
ALTER TABLE products
ADD CONSTRAINT fk_supplier
  FOREIGN KEY (supplier_id)
  REFERENCES supplier(supplier_id);
```

```
ALTER TABLE products
ADD CONSTRAINT fk_supplier
  FOREIGN KEY (supplier_id, supplier_name)
  REFERENCES supplier(supplier_id, supplier_name);
```

Foreign Key with Cascade Delete

A foreign key with cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted

```
CREATE TABLE table_name
(
  column1 datatype null/not null,
  column2 datatype null/not null,
  ...

  CONSTRAINT fk_column
    FOREIGN KEY (column1, column2, ... column_n)
    REFERENCES parent_table (column1, column2, ... column_n)
    ON DELETE CASCADE
);
```

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
);

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10) not null,
  CONSTRAINT fk_supplier
    FOREIGN KEY (supplier_id)
    REFERENCES supplier(supplier_id)
    ON DELETE CASCADE
);
```

Foreign Keys with Set Null on Delete

- If a record in the parent table is deleted, then the corresponding records in the child table will have the foreign key fields set to null
- The records in the child table will **not** be deleted

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
);

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10),
  CONSTRAINT fk_supplier
    FOREIGN KEY (supplier_id)
    REFERENCES supplier(supplier_id)
    ON DELETE SET NULL
);
```

Foreign Keys with Set Null on Delete

```
ALTER TABLE products
ADD CONSTRAINT fk_supplier
  FOREIGN KEY (supplier_id)
  REFERENCES supplier(supplier_id)
  ON DELETE SET NULL;
```

Drop Foreign Key

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;
```

Disable / Enable Foreign Key

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
);

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10) not null,
  CONSTRAINT fk_supplier
    FOREIGN KEY (supplier_id)
    REFERENCES supplier(supplier_id)
);
```

```
ALTER TABLE products
DISABLE CONSTRAINT fk_supplier;
```

```
ALTER TABLE products
ENABLE CONSTRAINT fk_supplier;
```

Alter Commands

```
ALTER TABLE customers
  ADD (customer_name varchar2(45),
       city varchar2(40));
```

```
ALTER TABLE customers
  MODIFY (customer_name varchar2(100) not null,
          city varchar2(75));
```

```
ALTER TABLE customers
  MODIFY customer_name varchar2(100) not null;
```

Alter Commands

```
ALTER TABLE customers
  DROP COLUMN customer_name;
```

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

```
ALTER TABLE customers
  RENAME TO contacts;
```

Drop Commands

```
DROP TABLE [schema_name].table_name
[ CASCADE CONSTRAINTS ]
[ PURGE ];
```

```
DROP TABLE customers;
```

```
DROP TABLE customers PURGE;
```

drop table products;
select object_name, original_name, type from recyclebin;

OBJECT_NAME	ORIGINAL_NAME	TYPE
BIN$W0BwD0k5M/bgU5Pe5wqdQg==$0	PRODUCTS	TABLE

Unique

```
CREATE TABLE supplier
( supplier_id numeric(10) NOT NULL,
  supplier_name varchar2(50) NOT NULL,
  contact_name varchar2(50),
  CONSTRAINT supplier_unique UNIQUE (supplier_id)
);
```

```
CREATE TABLE supplier
( supplier_id numeric(10) NOT NULL,
  supplier_name varchar2(50) NOT NULL,
  contact_name varchar2(50),
  CONSTRAINT supplier_unique UNIQUE (supplier_id, supplier_name)
);
```

Alter - Unique

```
ALTER TABLE supplier
ADD CONSTRAINT supplier_unique UNIQUE (supplier_id);
```

```
ALTER TABLE supplier
ADD CONSTRAINT supplier_name_unique UNIQUE (supplier_id, supplier_name);
```

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;
```

```
ALTER TABLE supplier
ENABLE CONSTRAINT supplier_unique;
```

```
ALTER TABLE supplier
DISABLE CONSTRAINT supplier_unique;
```

Check

- A check constraint allows you to specify a condition on each row in a table
- A check constraint can NOT be defined on a SQL View
- The check constraint defined on a table must refer to only columns in that table. It can not refer to columns in other tables.
- A check constraint can NOT include a SQL Subquery.

```
CREATE TABLE table_name
(
  column1 datatype null/not null,
  column2 datatype null/not null,

  ...

  CONSTRAINT constraint_name CHECK (column_name condition) [DISABLE]

);
```

Check

```
CREATE TABLE suppliers
(
  supplier_id numeric(4),
  supplier_name varchar2(50),
  CONSTRAINT check_supplier_id
  CHECK (supplier_id BETWEEN 100 and 9999)
);
```

```
CREATE TABLE suppliers
(
  supplier_id numeric(4),
  supplier_name varchar2(50),
  CONSTRAINT check_supplier_name
  CHECK (supplier_name = upper(supplier_name))
);
```

Alter - Check

```
ALTER TABLE suppliers
ADD CONSTRAINT check_supplier_name
  CHECK (supplier_name IN ('IBM', 'Microsoft', 'NVIDIA'));
```

```
ALTER TABLE suppliers
DROP CONSTRAINT check_supplier_id;
```

Create As

```
CREATE TABLE new_table
  AS (SELECT * FROM old_table);
```

```
CREATE TABLE suppliers
AS (SELECT *
    FROM companies
    WHERE company_id < 5000);
```

Create As

```
CREATE TABLE new_table
  AS (SELECT column_1, column2, ... column_n
      FROM old_table);
```

```
CREATE TABLE suppliers
  AS (SELECT company_id, address, city, state, zip
      FROM companies
      WHERE company_id < 5000);
```

Create As

```
CREATE TABLE new_table
  AS (SELECT column_1, column2, ... column_n
      FROM old_table_1, old_table_2, ... old_table_n);
```

```
CREATE TABLE suppliers
  AS (SELECT companies.company_id, companies.address, categories.category_type
      FROM companies, categories
      WHERE companies.company_id = categories.category_id
      AND companies.company_id < 5000);
```