Practice 4: DDL2

Jesús Manuel Juárez Pasillas

September 07 2021

1 Introduction

The correct use of the DDL sentences allows us to create, manage and consult the schemas that are required and with these we can do the same with the tables that are inside it as well as the data stored in the tables. For this, it is necessary to use all the tools that Oracle provides us for the good management of the schemes, these tools are included in Oracle as objects, which are the indexes, sequences and synonyms, with which it will be facilitated and the database can be optimized of data.

2 Developing

2.1 Activity 1:

Read all the choices carefully because there might be more than one correct answer. Choose all the correct answers for each question.
(The underlined answers are the chosen answers).

1. What are distinguishing characteristics of a public synonym rather than a private synonym?

   A. Public synonyms are always visible to all users.
   B. Public synonyms can be accessed by name without a schema name qualifier.
   C. Public synonyms can be selected from without needing any permission.
   D. Public synonyms can have the same names as tables or views.

   Answer: A public synonym is used to refer to an object without having to prepend its schema, and it can also be seen by all users.

2. Consider these three statements:
   create synonym s1 for staff;
   create public synonym s1 for warehouse;
   select * from s1;
   Which of the following statements is correct?

A. The second statement will fail because an object S1 already exists.

B. The third statement will show the contents of warehouse.

C. The third statement will show the contents of staff.

D. The third statement will show the contents of the table S1, if such a table exists in the current schema.

You cannot overwrite the ready-made synonym.

3. A view and a synonym are created as follows:
create view dept_v as select * from dept;
create synonym dept_s for dept_v;
Subsequently the table DEPT is dropped. What will happen if you query the synonym DEPT_S?

A. There will not be an error because the synonym addresses the view, which still exists, but there will be no rows returned.

B. There will not be an error if you first recompile the view with the command ALTER VIEW DEPT_V COMPILE FORCE;

C. There will be an error because the synonym will be invalid.

D. There will be an error because the view will be invalid.

E. There will be an error because the view will have been dropped implicitly when the table was dropped.

Since the only thing was deleted was the table and not the view, the view will give an error because it is not deleted yet.

4. A sequence is created as follows:
create sequence seq1 maxvalue 100;
If the current value is already 100, when you attempt to select SEQ1.NEXTVAL what will happen?

A. The sequence will cycle and issue 0.

B. The sequence will cycle and issue 1.

C. The sequence will reissue 100.

D. There will be an error.

Since it is not indicated if it is cycle or nocycle, the default value will be nocycle and when the maximum value is reached there will no longer be another value that can be obtained.

5. You create a sequence as follows:
create sequence seq1 start with 5;
After selecting from it a few times, you want to reinitialize it to reissue the numbers already generated. How can you do this?

A. You must drop and re-create the sequence.

B. You can't. Under no circumstances can numbers from a sequence be reissued once they have been used.

C. Use the command ALTER SEQUENCE SEQ1 START WITH 5; to reset the nex value to 5.

D. Use the command ALTER SEQUENCE SEQ1 CYCLE; to reset the sequence to its starting value.

To restart the sequence it is necessary to redo the sequence.

6. Study the following exhibit: Assuming that the sequence SEQ1 was cre-

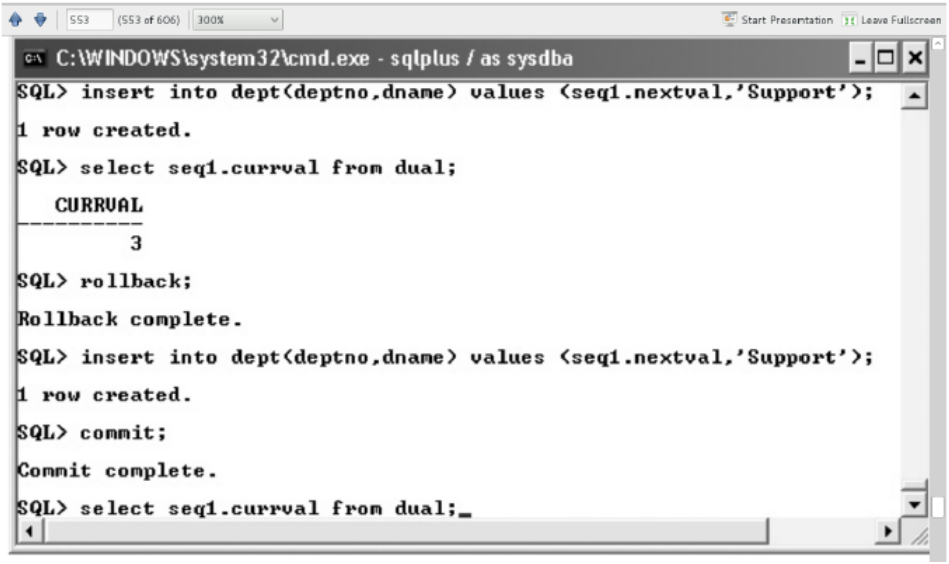

Figure 1: Study the following exhibit.

ated with the option ORDER and INCREMENT BY set to 1, what value will be returned by the final SELECT statement?

A. 2

B. 3

C. 4

The sequence continues to increase, regardless of the rollback since it only applies to transactions.

7. A UNIQUE constraint on a column requires an index. Which of the following scenarios is correct?

A. If a UNIQUE index already exists on the column, it will be used.

B. If a NONUNIQUE index already exists it will be used.

C. If a NONUNIQUE index already exists on the column, a UNIQUE index will be created implicitly.

D. If any index exists on the column, there will be an error as Oracle attempts to create another index implicitly.

In case an index already exists, Oracle uses it and does not create another.

8. This statement will fail:
create unique bitmap index on employees(department_id,hire_date);
Why?

   A. Bitmap indexes cannot be unique.

   B. The two columns are of different data types.

   C. A bitmap index can be on only one column.

   D. There is already a B*Tree index on DEPARTMENT_ID.

   Bitmaps cannot be unique as this type of index is used for few values but repeating values in the column.

9. You have created an index with this statement:
create index ename_i on employees(last_name,first_name);
How can you adjust the index to include the employees' birthdays, which is a date type column called DOB?

   A. A. Use ALTER INDEX ENAME_I ADD COLUMN DOB;

   B. You can't do this because of the data type mismatch.

   C. You must drop the index and re-create it.

   D. This can only be done if the column DOB is NULL in all existing rows.

   You cannot add the employees' birthdays to the created index as they are of a different type from the ones already established

2.2 Activity 2:

Consider the following context issue:
SHOP (shop_id, address, manager).
PRODUCT (product_id, pname, sale_price, purchase_price, provider).
CHANNEL (channel_id, cname).
EMPLOYEE (emp_id, emp_name, emp_lastn, boss_id, address, date_of_birth, gender, beneficiaries)
Figure 1 shows an entity-relationship diagram for a simple system designed to store and analyze sales. The columns for the fact table SALES are as follows:
• SALE_ID System-generated primary key
• CHANNEL_ID Foreign key to CHANNELS.
• PRODUCT_ID Foreign key to PRODUCTS.
• SHOP_ID Foreign key to SHOPS.

• QUANTITY The quantity of the product sold.
• EMP_ID The id of the employee that sold a product. Foreign Key to EMPLOYEE.
• SALE_DATE The date of the sale.
It is expected that there will be several million SALES rows per year. The

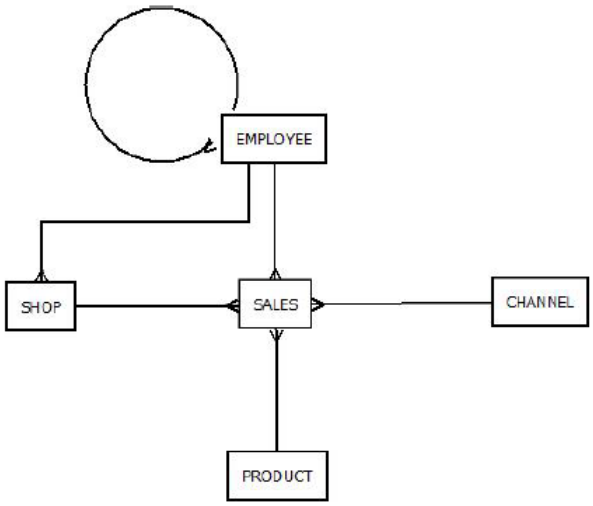

Figure 2: System designed to store and analyze sales.

dimension tables are as follows:
PRODUCT A list of all products, including price (a few hundred).
EMPLOYEE A list of employees in the stores. Hundreds of employees.
CHANNEL Possible sales methods, such as walk-in, Internet, and telephone.
SHOP Details of all the shops (no more than a couple of dozen).
SALES Details of the sales in the different shops. Thousands of sales per day.

- Write code to create the tables (only columns data type specifications);
  - create table Shops(
    shop_id number,
    address varchar2(50) constraint shops_address_nn not null,
    manager number(6));
  - create table Products(
    product_id number,
    pname varchar2(25) constraint products_pname_nn not null,
    sale_price number(6,2) constraint products_sale_price_nn not null,
    purcahse_price number(6,2), constraint products_purchase_pri_nn not null,
    provider varchar(25) constraint products_provider_nn not null);
  - create table Channels(

channel_id number,
cname varchar2(25) constraint channels_cname_nn not null);

- create table Employees(
emp_id number,
emp_name varchar2(25) constraint employee_name_nn not null,
emp_lastn varchar2(25),
boss_id number,
address varchar(50) constraint employee_address_nn not null,
date_of_birth date,
gender varchar2(6) constraint employee_gender_nn not null,
beneficiaries varchar(50));
- create table Sales(
sale_id number,
channel_id number,
product_id number,
shop_id number,
quantity number,
emp_id number,
sale_date date);

- Create indexes (choose appropriate type);
  - create bitmap index shops_addr_idx on Shops(address);
  - create unique index product_pname_idx on Products(pname);
  - create index product_idx on Products(pname, sale_price, purchase_price, provider);
  - create bitmap index channels_cname_idx on Channels(cname);
  - create bitmap index employee_gender_idx on Employees(gender);
  - create index employee_idx on Employees(emp_name, emp_lastn, address, date_of_birth, beneficiaries);
  - create index sales_idx Sales(quantity, sale_date);
- Create constraints (all that it needs).
  - alter table Shops add constraint shops_pk primary key (shop_id);
  - alter table Products add constraint product_pk primary key (product_id);
  - alter table Channels add constraint channel_pk primary key (channel_id);
  - alter table Employees add constraint employee_pk primary key (emp_id);
  - alter table Sales add constraint sales_pk primary key (sale_id);
  - alter table Shops add constraint employee_shop_fk foreign key (manager) references Employees(emp_id);

  - alter table Employees add constraint employee_employee_fk foreign key (boss_id) references Employees(emp_id);
  - alter table Sales add constraint shop_sales_fk foreign key (shop_id) references Shops(shop_id);
  - alter table Sales add constraint product_sales_fk foreign key (product_id) references Products(product_id);
  - alter table Sales add constraint channel_sales_fk foreign key (channel_id) references Channels(channel_id);
  - alter table Sales add constraint employee_sales_fk foreign key (emp_id) references Employees(emp_id);

- Create sequences to be used for primary keys where necessary with the best options.
  - create sequence sq_shops_id NOMAXVALUE NOCYCLE;
  - create sequence sq_products_id NOMAXVALUE NOCYCLE;
  - create sequence sq_channels_id NOMAXVALUE NOCYCLE;
  - create sequence sq_employees_id NOMAXVALUE NOCYCLE;
  - create sequence sq_sales_id NOMAXVALUE NOCYCLE;

- Create short name synonymous for each table.
  - create synonym Sho for Shops;
  - create synonym Prod for Products;
  - create synonym Chann for Channels;
  - create synonym Emp for Employees;
  - create synonym Sal for Sales;

2.3 Activity 3:

Consider these tables: create table vehicle(id_veh number, lic_plates varchar2(10), owner_id number);
create table owner(owner_id number, surname varchar2(10), forename varchar2(10), dateobirth date);

Reports often require information from surname and forename of owners, reports from vehicle license plates, vehicle license plates of specific people.

1. Define required indexes (choose adequate type).
   - create unique index vehicle_lic_plates_idx on vehicle(lic_plates);
   - create index owner_sur_for_name on owner(surname,forename);
2. Add required constraints.

- alter table vehicle add constraint vahicle_pk primary key (id_veh);
- alter table vehicle add constraint owner_vehicle_fk foreign key (owner_id) references owner(owner_id);
- alter table owner add constraint owner_pk primary key (owner_id);

2.4 Activity 4:

Propose a response to the following issues:

- You are involved in designing a database to be used for online order entry and offline financial reporting. What should you consider with regard to data consulting, synonyms, and indexes?
  - The indexes that are created help us to make the query process much faster, in addition to the fact that with the public synonyms indicated we can access the tables from any scheme that we require.
- Should sequences always be used for primary keys?
  - If the primary key is a number, it is highly recommended to use the sequences to define the value of the primary key, but since the primary key is not always numbers, it is not always possible to use the sequences for the primary keys.

2.5 Activity 5:

In this exercise, you will create some sequences and use them. You will need two concurrent sessions.

1. Log on to your database twice, as YOURSCHEME in separate sessions. Consider one to be your A session and the other to be your B session.
2. In your A session, create a sequence as follows:
   create sequence seq1 start with 10 nocache maxvalue 15 cycle;
3. Execute the following commands in the appropriate session in the correct order to observe the use of NEXTVAL and CURRVAL and the cycling of the sequence:

	In Your A Session	In Your B Session
1st	select seq1.nextval from dual;	
2nd		select seq1.nextval from dual;
3rd	select seq1.nextval from dual;	
4th		select seq1.nextval from dual;
5th	select seq1.currval from dual;	
6th		select seq1.nextval from dual;
7th	select seq1.nextval from dual;	
8th		select seq1.currval from dual;
9th	select seq1.nextval from dual;	
10th		select seq1.nextval from dual;

Figure 3: Commands.

Results: As it exceeds the limits, it starts again for the cycle.

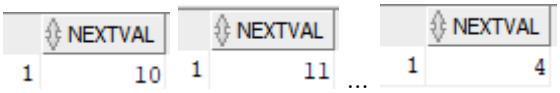

Figure 4: Results.

4. Create a table with a primary key:
create table seqtest(c1 number,c2 varchar2(10)); alter table seqtest add constraint seqtest_pk primary key (c1);

```
Table SEQTEST creado.


Table SEQTEST alterado.
```

Figure 5: seqtest.

5. Create a sequence to generate primary key values: create sequence seqtest_pk_s;

```
Sequence SEQTEST_PK_S creado.
```

Figure 6: sequence.

6. In your A session, insert a row into the new table and commit:
insert into seqtest values(seqtest_pk_s.nextval, 'first');
commit;

```
1 fila insertadas.

Confirmación terminada.
```

Figure 7: Insert and commit.

7. In your B session, insert a row into the new table and do not commit it:
   insert into seqtest values(seqtest_pk_s.nextval, 'second');

```
1 fila insertadas.
```

Figure 8: Insert.

8. In your A session, insert a third row and commit:
   insert into seqtest values(seqtest_pk_s.nextval, 'third');
   commit;

```
1 fila insertadas.

Confirmación terminada.
```

Figure 9: Insert and commit.

9. In your B session, roll back the second insertion:
   rollback;

10. In your B session, see the contents of the table:
    select * from seqtest;

	C1	C2
1	1	first
2	2	second
3	3	third

Figure 10: Select.

11. Tidy up:
    drop table seqtest;
    drop sequence seqtest_pk_s;
    drop sequence seq1;

2.6 Activity 6:

In this exercise, create indexes on a copy of the EMPLOYEES table in the HR schema.

1. Connect to your database as your user.
2. Create a table that is a copy of HR.EMPLOYEES:
   create table emps as select * from hr.employees;
   describe emps;

```
Nombre          ¿Nulo?   Tipo
--------------- -------- ------------
EMPLOYEE_ID              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)
```

Figure 11: Describe emps.

3. Create an index to be used for the primary key constraint:
   create unique index emps_empid_i on emps(employee_id);
4. Demonstrate that a unique index cannot accept duplicates, even before a constraint is defined: insert into emps(employee_id, last_name, email, hire_date, job_id) values(198, 'Watson', 'jw@bplc.co.za', sysdate, 'IT_PROG');

```
Error que empieza en la línea: 7 del comando :
insert into emps(employee_id,last_name,email,hire_date,job_id)
values(198,'Watson','jw@bplc.co.za',sysdate,'IT_PROG')
Informe de error -
ORA-00001: unique constraint (HR.EMPS_EMPID_I) violated
```

Figure 12: Inserting error.

5. Create additional indexes on columns that are likely to be used in WHERE clauses, using B*Tree for columns of high cardinality and bitmap for

columns of low cardinality:
create index emps_name_i on emps(last_name, first_name);
create index emps_tel_i on emps(phone_number);
create bitmap index emps_mgr_i on emps(manage_id);
create bitmap index emps_dept_i on emps(department_id);

6. Define some constraints:
   alter table emps add constraint emps_pk primary key (employee_id);
   alter table emps add constraint emps_email_uk unique(email);
   alter table emps add constraint emps_tel_uk unique(phone_number);

7. Display the index names and their type:
   select index_name,index_type,uniqueness from user_indexes where table_name='EMPS';
   Bitmap indexes do not appear due to the use of the Oracle version.

	INDEX_NAME	INDEX_TYPE	UNIQUENESS
1	EMPS_EMPID_I	NORMAL	UNIQUE
2	EMPS_NAME_I	NORMAL	NONUNIQUE
3	EMPS_TEL_I	NORMAL	NONUNIQUE
4	EMPS_EMAIL_UK	NORMAL	UNIQUE

Figure 13: Display the index.

8. Tidy up by dropping the EMPS table, and confirm that all the indexes have also gone:
   drop table emps;
   select index_name from user_indexes where table_name='EMPS';

INDEX_N...

Figure 14: Drop emps table.

2.7 Activity 7:

Consider the following context issues employed in past practices. For each one:

2.7.1 User case 1. Bookstore. The establishment of activity 1/practice 1.

- Write code to create the tables (only the create table statements with basic constrains: default, check, not null, . . . , and column data type specifications).

- create table Customers(
  customer_id number,
  fname varchar2(25) constraint customer_fname_nn not null,
  lname varchar2(25),
  shopping_cart_id number);
- create table Shopping_Cart(
  shopping_cart_id number);
- create table Establishments(
  establish_id number,
  name varchar2(25) constraint establesh_name_nn not null,
  address_id number);
- create table Address(
  address_id number,
  country varchar2(25) constraint address_country_nn not null,
  postal_code char(12) constraint address_postal_c_nn not null,
  city varchar2(25) constraint address_city_nn not null,
  street varchar2(25) constraint address_street_nn not null,
  anumber number constraint address_anumber_nn not null);
- create table Warehouse(
  warehouse_id number,
  address_id number);
- create table Products(
  product_id number,
  price number(6,2) constraint product_price_nn not null);
- create table Product_warehouse(
  warehouse_id number,
  product_id number);
- create table Buy(
  shopping_cart_id number,
  product_id number,
  date_buy date);
- create table Book(
  ISBN char(12),
  product_id number,
  title varchar2(25) constraint book_title_nn not null);
- create table Cassettes(
  cassette_id number,
  product_id number,
  capacity number(5,2));
- create table Electronic_cards(
  card_id number,
  product_id number,
  capacity number(5,2));

- create table Blue_Ray(
  blue_ray_id number,
  product_id number,
  capacity number(5,2));
- create table CDs(
  cd_id number,
  product_id number,
  capacity number(5,2));
- create table Publisher(
  publisher_id number,
  fname varchar2(25) constraint publisher_fname_nn not null,
  lname varchar2(25));
- create table Author(
  author_id number,
  fname varchar2(25) constraint author_fname_nn not null,
  lname varchar2(25));

- Create indexes (choose appropriate type).
  - create bitmap index esta_name_idx on Establishments(name);
  - create index esta_addr_idx on Establishments(address_id);
  - create index address_idx on Address(country,postal_code,city);
  - create index ware_addr_idx on Warehouse(address_id);
  - create index pro_ware_idx on Product_warehouse(warehouse_id,product_id);
  - create index book_title_idx on Book(title);
  - create index cassette_capa_idx on Cassettes(capacity);
  - create index elect_capa_idx on Electronic_cards(capacity);
  - create index blue_ray_capa_idx on Blue_Ray(capacity);
  - create index cd_capa_idx on CDs(capacity);
  - create index publisher_name_idx on Publisher(fname,lname);
  - create index author_name_idx on Author(fname,lname);

- Create constraints (unique, primary key, foreign keys, . . .).
  - alter table Customers add constraint customer_pk primary key (customer_id);
  - alter table Customers add constraint shoppin_customer_fk foreign key (shopping_cart_id) references Shopping_Cart(shopping_cart_id);
  - alter table Establishments add constraint establish_pk primary key (establish_id);
  - alter table Establishments add constraint addr_establish_fk foreign key (address_id) references Address(address_id);

- alter table Address add constraint address_pk primary key (address_id);
- alter table Warehouse add constraint warehouse_pk primary key (warehouse_id);
- alter table Warehouse add constraint addr_warehouse_fk foreign key (address_id) references Address(address_id);
- alter table Product add constraint product_pk primary key (product_id);
- alter table Product_warehouse add constraint pro_ware_pk primary key (warehouse_id,product_id);
- alter table Product_warehouse add constraint pro_pro_ware_fk foreign key (product_id) references Products(product_id);
- alter table Product_warehouse add constraint ware_pro_ware_fk foreign key (warehouse_id) references Warehouse(warehouse_id);
- alter table Buy add constraint buy_pk primary key (shopping_cart_id, product_id, date_buy);
- alter table Buy add constraint shoppin_buy_fk foreign key (shopping_cart_id) references Shopping_Cart(shopping_cart_id);
- alter table Buy add constraint product_buy_fk foreign key (product_id) references Products(product_id);
- alter table Book add constraint book_pk primary key (ISBN,product_id);
- alter table Book add constraint product_book_fk foreign key (product_id) references Products(product_id);
- alter table Cassettes add constraint cassette_pk primary key (cassette_id,product_id);
- alter table Cassettes add constraint product_cassette_fk foreign key (product_id) references Products(product_id);
- alter table Electronic_cards add constraint elec_card_pk primary key (card_id,product_id);
- alter table Electronic_cards add constraint product_card_fk foreign key (product_id) references Products(product_id);
- alter table Blue_Ray add constraint blue_ray_pk primary key (blue_ray_id,product_id);
- alter table Blue_Ray add constraint product_blue_ray_fk foreign key (product_id) references Products(product_id);
- alter table CDs add constraint cd_pk primary key (cd_id,product_id);
- alter table CDs add constraint product_cd_fk foreign key (product_id) references Products(product_id);
- alter table Publisher add constraint publisher_pk primary key (publisher_id);
- alter table Author add constraint author_pk primay key (author_id);

- Create sequences to be used for primary keys where necessary with the best options.
  - create sequence sq_cus_id NOMAXVALUE NOCYCLE;
  - create sequence sq_shop_id NOMAXVALUE NOCYCLE;
  - create sequence seq_est_id NOMAXVALUE NOCYCLE;
  - create sequence seq_addr_id NOMAXVALUE NOCYCLE;
  - create sequence seq_ware_id NOMAXVALUE NOCYCLE;
  - create sequence seq_pro_id NOMAXVALUE NOCYCLE;
  - create sequence seq_prowa_id NOMAXVALUE NOCYCLE;
  - create sequence seq_buy_id NOMAXVALUE NOCYCLE;
  - create sequence seq_book_id NOMAXVALUE NOCYCLE;
  - create sequence seq_cass_id NOMAXVALUE NOCYCLE;
  - create sequence seq_car_id NOMAXVALUE NOCYCLE;
  - create sequence seq_blue_id NOMAXVALUE NOCYCLE;
  - create sequence seq_cd_id NOMAXVALUE NOCYCLE;
  - create sequence seq_pub_id NOMAXVALUE NOCYCLE;
  - create sequence seq_aut_id NOMAXVALUE NOCYCLE;
- Create short name synonymous for each table.
  - create synonym Cus for Customers;
  - create synonym Sho_Ca for Shopping_Cart;
  - create synonym Estab for Establishments;
  - create synonym Addr for Address;
  - create synonym Ware for Warehouse;
  - create synonym Prod for Products;
  - create synonym Prod_Ware for Product_warehouse;
  - create synonym Buy for Buy;
  - create synonym Book for Book;
  - create synonym Cass for Cassettes;
  - create synonym Elec_Card for Electronic_cards;
  - create synonym BRay for Blue_Ray;
  - create synonym CD for CDs;
  - create synonym Publi for Publisher;
  - create synonym Auth for Author;

2.7.2 User case 3. Sales. The orders scenario of activity 3/practice 1.

- Write code to create the tables (only the create table statements with basic constrains: default, check, not null, ..., and column data type specifications).
  - create table Customers(
    customer_id number,
    organisation_or_person varchar2(25),
    organisation_name varchar2(25) constraint cus_org_name_nn not null,
    gender varchar2(10),
    first_name varchar2(25) constraint cus_fir_name_nn not null,
    middle_name varchar2(25),
    last_name varchar2(25) constraint cus_las_name_nn not null,
    email_address varchar2(50),
    login_name varchar2(25) constraint cus_log_name_nn not null,
    login_password varchar2(25) constraint cus_log_pass_nn not null,
    phone_number char(15),
    address_line_1 varchar2(25),
    address_line_2 varchar2(25),
    address_line_3 varchar2(25),
    address_line_4 varchar2(25),
    town_city varchar2(25),
    county varchar2(25),
    country varchar2(25));
  - create table Ref_Payment_Methods(
    payment_method_code char(2),
    payment_method_description varchar2(25));
  - create table Customer_Payment_Methods(
    customer_payment_id number,
    customer_id number,
    payment_method_code char(2),
    credit_card_number char(15),
    payment_method_details varchar2(25));
  - create table Ref_Product_Types(
    product_type_code varchar2(15),
    parent_product_type_code varchar2(15),
    product_type_description varchar2(25));
  - create table Products(
    product_id number,
    product_type_code varchar2(15),
    return_merchandise_authori varchar2(25),
    product_name varchar2(25) constraint product_name_nn not null,

product_price number(6,2) constraint product_price_nn not null,
product_color char(7),
product_size varchar2(10),
product_description varchar2(30),
other_product_details varchar2(30));

- create table Ref__Status_Codes(
  order_status_code varchar2(9),
  order_status_description varchar2(30));
- create table Orders(
  order_id number,
  customer_id number,
  order_status_code varchar2(9),
  date_order_placed date,
  order_details varchar2(30));
- create table Re_Order_Item_Status_Codes(
  order_item_status_code varchar2(9),
  order_item_stauts_description varchar2(30));
- create table Order_Items(
  order_item_id number,
  product_id number,
  order_id number,
  order_item_status_code varchar2(9),
  order_item_quantity number,
  order_item_price number(6,2),
  RMA_number number,
  RMA_issued_by varchar2(25),
  RMA_issued_date date,
  other_order_item_details varchar2(30));
- create table Ref_Invoice_status_Codes(
  invoice_stauts_code varchar2(6),
  invoice_status_description varchar2(30));
- create table Invocies(
  invoice_number number,
  order_id number,
  invoice_stauts_code varchar2(6),
  invoice_date date,
  invoice_details varchar2(30));
- create table Payments(
  payment_id number,
  invoice_number number,
  payment_date date,
  payment_amount number(7,2));
- create table Shipments(
  shipment_id number,

order_id number,
invoice_number number,
shipment_tracking_number number,
shipment_date date,
other_shipment_details varchar2(30));

  - create table Shipment_Items(
shipment_id number,
order_item_id number);

- Create indexes (choose appropriate type).
  - create bitmap index cus_gender_idx on Customers(gender);
  - create index cus_organisation on Customers(organisation_or_person, organisation_name);
  - create index cus_name on Customers(first_name, middle_name, last_name);
  - create unique index cus_log_name on Customers(login_name);
  - create index cus_pay_me_cus on Customer_Payment_Methods(customer_id);
  - create index cus_pay_me_pay_me on Customer_Payment_Methods (payment_method_code);
  - create index ref_pro_typ on Ref_Product_Types(parent_product_type_code);
  - create bitmap index prod_size on Products(product_size);
  - create index product_name_idx on Products(product_name, product_price);
  - create index prod_type_idx on Products(product_type_code);
  - create bitmap index order_status on Orders(order_status_code);
  - create index cus_order on Orders(customer_id);
  - create index order_item_prod on Order_Items(product_id);
  - create index order_item_order on Order_Items(order_id);
  - create index order_item_status on Order_Items(order_item_status_code);
  - create index invoice_order on Invocies(order_id);
  - create index invoice_status on Invocies(invoice_stauts_code);
  - create index invo_paym on Payments(invoice_number);
  - create index payment_date on Payments(payment_date);
  - create index ship_order on Shipments(order_id);
  - create index ship_invoice on Shipments(invoice_number);
  - create index ship_date on Shipments(shipment_date);
- Create constraints (unique, primary key, foreign keys, . . .).
  - alter table Customers add constraint customer_pk primary key (customer_id);

- alter table Ref_Payment_Methods add constraint ref_paym_met_pk primary key (payment_method_code);
- alter table Customer_Payment_Methods add constraint custo_pay_met_pk primary key (customer_payment_id);
- alter table Customer_Payment_Methods add constraint cus_cus_pay_met_fk foreign key (customer_id) references Customers(customer_id);
- alter table Customer_Payment_Methods add constraint ref_cus_pay_met_fk foreign key (payment_method_code) references Ref_Payment_Methods (payment_method_code);
- alter table Ref_Product_Types add constraint ref_pro_ty_pk primary key (product_type_code);
- alter table Ref_Product_Types add constraint ref_ty_ref_pro_ty_fk foreign key (parent_product_type_code) references Ref_Product_Types (product_type_code);
- alter table Products add constraint product_pk primary key (product_id);
- alter table Products add constraint ref_ty_prod_fk foreign key (product_type_code) references Ref_Product_Types (product_type_code);
- alter table Ref_Order_Status_Codes add constraint ref_or_status_pk primary key (order_status_code);
- alter table Orders add constraint order_pk primary key (order_id);
- alter table Orders add constraint cus_order_fk foreign key (customer_id) references Customers(customer_id);
- alter table Orders add constraint status_order_fk foreign key (order_status_code) references Ref_Order_Status_Codes(order_status_code);
- alter table Ref_Order_Item_Status_Codes add constraint item_status_pk primary key (order_item_status_code);
- alter table Order_Items add constraint order_item_pk primary key (order_item_id);
- alter table Order_Items add constraint pro_order_item_fk foreign key (product_id) references Products(product_id);
- alter table Order_Items add constraint ord_order_item_fk foreign key (order_id) references Orders(order_id);
- alter table Order_Items add constraint status_order_item_fk foreign key (order_item_status_code) references Ref_Order_Item_Status_Codes (order_item_status_code);
- alter table Ref_Invoice_status_Codes add constraint ref_inv_stat_pk primary key (invoice_stauts_code);
- alter table Invocies add constraint invoice_pk primary key (invoice_number);

  - alter table Invocies add constraint order_invoice_fk foreign key (order_id) references Orders(order_id);
  - alter table Invocies add constraint stauts_invoice_fk foreign key (invoice_stauts_code) references Ref_Invoice_status_Codes(invoice_stauts_code);
  - alter table Payments add constraint payment_pk primary key (payment_id);
  - alter table Payments add constraint invoice_payment_fk foreign key (invoice_number) references Invocies(invoice_number);
  - alter table Shipments add constraint shipment_pk primary key (shipment_id);
  - alter table Shipments add constraint order_shipment_fk foreign key (order_id) references Orders(order_id);
  - alter table Shipments add constraint invoice_shipment_fk foreign key (invoice_number) references Invocies(invoice_number);
  - alter table Shipment_Items add constraint shipment_item_pk primary key (shipment_id,order_item_id);
  - alter table Shipment_Items add constraint ship_ship_item_fk foreign key (shipment_id) references Shipments(shipment_id);
  - alter table Shipment_Items add constraint order_ship_item_fk foreign key (order_item_id) references Order_Items(order_item_id);

- Create sequences to be used for primary keys where necessary with the best options.
  - create sequence cus_id NOMAXVALUE NOCYCLE;
  - create sequence pay_met_id NOMAXVALUE NOCYCLE;
  - create sequence prod_id NOMAXVALUE NOCYCLE;
  - create sequence order_id NOMAXVALUE NOCYCLE;
  - create sequence order_item_id NOMAXVALUE NOCYCLE;
  - create sequence inv_id NOMAXVALUE NOCYCLE;
  - create sequence order_id NOMAXVALUE NOCYCLE;
  - create sequence inv_sta_id NOMAXVALUE NOCYCLE;
  - create sequence pay_id NOMAXVALUE NOCYCLE;
  - create sequence ship_id NOMAXVALUE NOCYCLE;

- Create short name synonymous for each table.
  - create synonym Cus for Customers;
  - create synonym Ref_Pay_Met for Ref_Payment_Methods;
  - create synonym Cus_Pay_Met for Customer_Payment_Methods;
  - create synonym Ref_Pro_Typ for Ref_Product_Types;

- create synonym Prod for Products;
- create synonym Ref_Ord_Status for Ref_Order_Status_Codes;
- create synonym Ord for Orders;
- create synonym Ref_Ord_It_Sta_Co for Ref_Order_Item_Status_Codes;
- create synonym Ord_Ite for Order_Items;
- create synonym Ref_Inv_Stat_Co for Ref_Invoice_status_Codes;
- create synonym Invo for Invocies;
- create synonym Paym for Payments;
- create synonym Ship for Shipments;
- create synonym Ship_Ite for Shipment_Items;

2.7.3 User case 4. Documents.

- Write code to create the tables (only the create table statements with basic constrains: default, check, not null, ..., and column data type specifications).
  - create table Templates(
    template_id integer,
    version_number integer,
    template_type_code char(15),
    date_effective_from datetime,
    date_effective_to datetime,
    template_details varchar(255));
  - create table Ref_Template_Types(
    template_type_code char(15),
    template_type_description varchar(255));
  - create table Documents(
    document_id integer,
    template_id integer,
    document_name varchar(255),
    document_description varchar(255),
    other_details varchar(255));
  - create table Paragraphs(
    paragraph_id integer,
    document_id integer,
    paragraph_text varchar(255),
    other_details varchar(255));
- Create indexes (choose appropriate type).
  - create index temp_version on Templates(version_number);
  - create index temp_time on Templates(date_effective_from, date_effective_to);

  - create index doc_temp on Documents(template_id);
  - create index doc_name on Documents(document_name);
  - create index para_doc on Paragraphs(document_id);
  - create index para_txt on Paragraphs(paragraph_text);

- Create constraints (unique, primary key, foreign keys, ...).
  - alter table Templates add constraint template_pk primary key (template_id);
  - alter table Templates add constraint tem_type_template_fk foreign key (template_type_code) references Ref_Template_Types(template_type_code);
  - alter table Ref_Template_Types add constraint template_type_pk primary key (template_type_code);
  - alter table Documents add constraint document_pk primary key (document_id);
  - alter table Documents add constraint temp_doc_fk foreign key (template_id) references Templates(template_id);
  - alter table Paragraphs add constraint paragraph_pk primary key (paragraph_id);
  - alter table Paragraphs add constraint doc_paragraph_fk foreign key (document_id) references Documents(document_id);

- Create sequences to be used for primary keys where necessary with the best options.
  - create sequence temp_id NOMAXVALUE NOCYCLE;
  - create sequence doc_id NOMAXVALUE NOCYCLE;
  - create sequence para_id NOMAXVALUE NOCYCLE;

- Create short name synonymous for each table.
  - create synonym Temp for Templates;
  - create synonym Temp_Typ for Ref_Template_Types;
  - create synonym Doc for Documents;
  - create synonym Para for Paragraphs;

3 Pre-Assessment:

- Practices pre-Assessment for Database Systems Laboratory II

Practice	Pre-Assessment
COMPLIES WITH THE REQUESTED FUNCTIONALITY	
HAS THE CORRECT INDENTATION	X
HAS AN EASY WAY TO ACCESS THE PROVIDED FILES	X
HAS A REPORT WITH IDC FORMAT	X
REPORT INFORMATION IS FREE OF SPELLING ERRORS	X
DELIVERED IN TIME AND FORM	X
IS FULLY COMPLETED (SPECIFY THE PERCENTAGE COMPLETED)	95

Table 1: Pre-Assessment.

4 Conclusion:

Without the good handling of the sentences, a database with many errors is obtained as a result and this will cause a very slow database, on the other hand, if the correct DDL sentences are applied and all the tools that Oracle has to use are used. perform the required scheme, it will be much more optimized, resulting in the insertions, deletions, and queries being much faster.