DBMS**.**2

**END TERM PROJECT**

**PROJECT NAME:** ONLINE BOOK SHOP

**PRESENTED TO:** AZAMAT SEREK

**PRESENTED BY:**  KANGILOV SULAIMON 210103464

AZHARBAYEV RAIYMBEK 210103300

ASSIMOV ILKHAM 210103427

GARIFULLA AMIR 210103024

**PROJECT DESCRIPTION:**

* This project entails the development of a database management system (DBMS) to support data manipulation and storage in a bookstore context. The objective is to facilitate online sales, provide customers with accurate and timely information on books, effectively manage data on customers and their transactions, as well as comprehensively monitor the inventory of books, sales transactions, and financial transfers.
* The proposed system will enable the bookstore to efficiently store and manage data on the inventory of books, including the quantity in stock, availability, and pricing information. Additionally, the system will allow for the integration of customer data, including personal information, purchase history, and credit card details, to support efficient sales transactions. The system will also capture detailed data on sales, including revenue, cost of goods sold, and profits, and generate reports to provide insight into the performance of the bookstore.
* Furthermore, the system will encompass features for managing online transactions, tracking the delivery of books to buyers, and maintaining records on the status of payments and deliveries. These functionalities will ensure that customers receive timely updates on their orders, and that the bookstore maintains accurate records of all transactions.
* The proposed DBMS is specifically designed to cater to the needs of small and medium-sized businesses, providing them with a comprehensive solution for data storage and manipulation, transaction processing, and financial management. By adopting this system, bookstores can streamline their operations, reduce errors, and enhance the overall customer experience.

**PROJECT STRUCTURE:**

**TABLES:** BOOK, BASKET, SUPPLIER, DELIVERY, TRANSACTION, CUSTOMER, CARD, ORDERS, ORDER\_DETAIL

**BOOK:**

CREATE TABLE "BOOK"

( "BOOK\_ID" NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL ENABLE,

"AMOUNT" NUMBER NOT NULL ENABLE,

"SUPPLIER\_ID" NUMBER NOT NULL ENABLE,

"ISBN" VARCHAR2(20 CHAR) NOT NULL ENABLE,

"AUTHOR" VARCHAR2(50 CHAR) NOT NULL ENABLE,

"NAME" VARCHAR2(50 CHAR) NOT NULL ENABLE,

"DESCRIPTION" VARCHAR2(4000 CHAR) NOT NULL ENABLE,

"PRICE" NUMBER NOT NULL ENABLE,

CONSTRAINT "BOOK\_PK" PRIMARY KEY ("BOOK\_ID")

USING INDEX ENABLE

) ;

ALTER TABLE "BOOK" ADD CONSTRAINT "BOOK\_FK" FOREIGN KEY ("SUPPLIER\_ID")

REFERENCES "SUPPLIER" ("SUPPLIER\_ID") ENABLE;

CREATE OR REPLACE EDITIONABLE TRIGGER "BOOK\_SHOW\_ROWS"

before insert on book

for each row

declare

num number;

begin

select count(\*) into num from book;

dbms\_output.put\_line('Before insertion there are '||num||' rows');

end;

/

ALTER TRIGGER "BOOK\_SHOW\_ROWS" ENABLE;

**BASKET:**

CREATE TABLE "BASKET"

( "CUSTOMER\_ID" NUMBER NOT NULL ENABLE,

"BOOK\_ID" NUMBER NOT NULL ENABLE,

"QUANTITY" NUMBER NOT NULL ENABLE

) ;

ALTER TABLE "BASKET" ADD CONSTRAINT "BASKET\_FK2" FOREIGN KEY ("BOOK\_ID")

REFERENCES "BOOK" ("BOOK\_ID") ENABLE;

ALTER TABLE "BASKET" ADD FOREIGN KEY ("CUSTOMER\_ID")

REFERENCES "CUSTOMER" ("CUSTOMER\_ID") ON DELETE CASCADE ENABLE;

CREATE OR REPLACE EDITIONABLE TRIGGER "BASKET\_SHOW\_ROWS"

before insert on basket

for each row

declare

num number;

begin

select count(\*) into num from basket;

dbms\_output.put\_line('Before insertion there are '||num||' rows');

end;

/

ALTER TRIGGER "BASKET\_SHOW\_ROWS" ENABLE;

**SUPPLIER:**

CREATE TABLE "SUPPLIER"

( "SUPPLIER\_ID" NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL ENABLE,

"NAME" VARCHAR2(50 CHAR) NOT NULL ENABLE,

"PHONE\_NO" VARCHAR2(20 CHAR) NOT NULL ENABLE,

"EMAIL" VARCHAR2(50 CHAR) NOT NULL ENABLE,

CONSTRAINT "SUPPLIER\_PK" PRIMARY KEY ("SUPPLIER\_ID")

USING INDEX ENABLE

) ;

**DELIVERY:**

CREATE TABLE "DELIVERY"

( "DELIVERY\_ID" NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL ENABLE,

"ORDER\_ID" NUMBER NOT NULL ENABLE,

"STATUS" VARCHAR2(50) NOT NULL ENABLE,

CONSTRAINT "DELIVERY\_PK" PRIMARY KEY ("DELIVERY\_ID")

USING INDEX ENABLE

) ;

ALTER TABLE "DELIVERY" ADD CONSTRAINT "DELIVERY\_FK" FOREIGN KEY ("ORDER\_ID")

REFERENCES "ORDER" ("ORDER\_ID") ENABLE;

CREATE OR REPLACE EDITIONABLE TRIGGER "DELIVERY\_SHOW\_ROWS"

before insert on delivery

for each row

declare

num number;

begin

select count(\*) into num from delivery;

dbms\_output.put\_line('Before insertion there are '||num||' rows');

end;

/

ALTER TRIGGER "DELIVERY\_SHOW\_ROWS" ENABLE;

**TRANSACTION:**

CREATE TABLE "TRANSACTION"

( "TRANSACTION\_ID" NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL ENABLE,

"CUSTOMER\_ID" NUMBER NOT NULL ENABLE,

"TRANSACTION\_DATE" DATE NOT NULL ENABLE,

"STATUS" VARCHAR2(50) NOT NULL ENABLE,

"TOTAL\_SUM" NUMBER NOT NULL ENABLE,

CONSTRAINT "TRANSACTION\_PK" PRIMARY KEY ("TRANSACTION\_ID")

USING INDEX ENABLE

) ;

ALTER TABLE "TRANSACTION" ADD CONSTRAINT "TRANSACTION\_FK2" FOREIGN KEY ("CUSTOMER\_ID")

REFERENCES "CUSTOMER" ("CUSTOMER\_ID") ENABLE;

**CUSTOMER:**

CREATE TABLE "CUSTOMER"

( "CUSTOMER\_ID" NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL ENABLE,

"FIRST\_NAME" VARCHAR2(50 CHAR) NOT NULL ENABLE,

"LAST\_NAME" VARCHAR2(50 CHAR) NOT NULL ENABLE,

"ADDRESS" VARCHAR2(50 CHAR) NOT NULL ENABLE,

"PHONE\_NUMBER" VARCHAR2(50 CHAR) NOT NULL ENABLE,

CONSTRAINT "CUSTOMER\_PK" PRIMARY KEY ("CUSTOMER\_ID")

USING INDEX ENABLE

) ;

CREATE OR REPLACE EDITIONABLE TRIGGER "CUSTOMER\_SHOW\_ROWS"

before insert on customer

for each row

declare

num number;

begin

select count(\*) into num from customer;

dbms\_output.put\_line('Before insertion there are '||num||' rows');

end;

/

ALTER TRIGGER "CUSTOMER\_SHOW\_ROWS" ENABLE;

**CARD:**

CREATE TABLE "CARD"

( "CUSTOMER\_ID" NUMBER NOT NULL ENABLE,

"CARD\_NUM" NUMBER NOT NULL ENABLE,

"CVV" NUMBER NOT NULL ENABLE,

"EXP\_DATE" VARCHAR2(15) NOT NULL ENABLE,

"BALANCE" NUMBER NOT NULL ENABLE,

UNIQUE ("CUSTOMER\_ID")

USING INDEX ENABLE

) ;

ALTER TABLE "CARD" ADD CONSTRAINT "CARD\_FK" FOREIGN KEY ("CUSTOMER\_ID")

REFERENCES "CUSTOMER" ("CUSTOMER\_ID") ENABLE;

CREATE OR REPLACE EDITIONABLE TRIGGER "CARD\_SHOW\_ROWS"

before insert on card

for each row

declare

num number;

begin

select count(\*) into num from card;

dbms\_output.put\_line('Before insertion there are '||num||' rows');

end;

/

ALTER TRIGGER "CARD\_SHOW\_ROWS" ENABLE;

**ORDER:**

CREATE TABLE "ORDER"

( "ORDER\_ID" NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL ENABLE,

"CUSTOMER\_ID" NUMBER NOT NULL ENABLE,

"TRANSACTION\_ID" NUMBER NOT NULL ENABLE,

CONSTRAINT "ORDER\_PK" PRIMARY KEY ("ORDER\_ID")

USING INDEX ENABLE

) ;

ALTER TABLE "ORDER" ADD CONSTRAINT "ORDER\_FK" FOREIGN KEY ("CUSTOMER\_ID")

REFERENCES "CUSTOMER" ("CUSTOMER\_ID") ENABLE;

ALTER TABLE "ORDER" ADD CONSTRAINT "TRANSACTION\_FK" FOREIGN KEY ("TRANSACTION\_ID")

REFERENCES "TRANSACTION" ("TRANSACTION\_ID") ON DELETE CASCADE ENABLE;

**ORDER\_DETAIL:**

CREATE TABLE "ORDER\_DETAIL"

( "ORDER\_ID" NUMBER NOT NULL ENABLE,

"BOOK\_ID" NUMBER NOT NULL ENABLE,

"QUANTITY" NUMBER NOT NULL ENABLE

) ;

ALTER TABLE "ORDER\_DETAIL" ADD CONSTRAINT "HISTORY\_FK" FOREIGN KEY ("ORDER\_ID")

REFERENCES "ORDER" ("ORDER\_ID") ON DELETE CASCADE ENABLE;

ALTER TABLE "ORDER\_DETAIL" ADD CONSTRAINT "HISTORY\_FK1" FOREIGN KEY ("BOOK\_ID")

REFERENCES "BOOK" ("BOOK\_ID") ON DELETE CASCADE ENABLE;

**ENTITY RELATIONSHIP DIAGRAM (ERD):**

[HERE IS THE LINK TO (ERD)](https://miro.com/app/board/uXjVMTK1_JU=/?share_link_id=593467966364)

**PROCEDURES:**

1)create or replace procedure order\_book

(customer\_id in number)

is

basket\_empty exception;

not\_enough exception;

no\_customer exception;

not\_enough\_book exception;

num\_of\_products number;

row\_basket basket%rowtype;

total\_sum number := 0;

price number;

amount number;

sum\_of\_card number;

order\_id number;

transaction\_id number;

num number;

begin

select count(\*) into num from customer where CUSTOMER\_ID = customer\_id;

if num = 0 then

raise no\_customer;

end if;

select count(\*) into num\_of\_products from basket where CUSTOMER\_ID=customer\_id;

if num\_of\_products = 0 then

raise basket\_empty;

end if;

for row\_basket in (select \* from basket where CUSTOMER\_ID=customer\_id)

loop

select AMOUNT into amount from book where BOOK\_ID=row\_basket.BOOK\_ID;

if amount<row\_basket.QUANTITY then

raise not\_enough\_book;

end if;

end loop;

select BALANCE into sum\_of\_card from card where CUSTOMER\_ID=customer\_id;

for row\_basket in (select \* from basket where CUSTOMER\_ID=customer\_id)

loop

select PRICE into price from book where BOOK\_ID=row\_basket.BOOK\_ID;

total\_sum := total\_sum + (price \* row\_basket.QUANTITY);

end loop;

if total\_sum > sum\_of\_card then

insert into "TRANSACTION" (CUSTOMER\_ID, TRANSACTION\_DATE, STATUS, TOTAL\_SUM) values (customer\_id, sysdate(), 'failed', total\_sum);

raise not\_enough;

end if;

insert into "TRANSACTION" (CUSTOMER\_ID, TRANSACTION\_DATE, STATUS, TOTAL\_SUM) values (customer\_id, sysdate(), 'successful', total\_sum) returning TRANSACTION\_ID into transaction\_id;

insert into "ORDER" (CUSTOMER\_ID, TRANSACTION\_ID) values (customer\_id, transaction\_id) returning ORDER\_ID into order\_id;

for row\_basket in (select \* from basket where CUSTOMER\_ID=customer\_id)

loop

insert into order\_detail(ORDER\_ID, BOOK\_ID, QUANTITY) values(order\_id, row\_basket.BOOK\_ID, row\_basket.QUANTITY);

update book set AMOUNT = AMOUNT - row\_basket.QUANTITY where BOOK\_ID = row\_basket.BOOK\_ID;

end loop;

update card set balance = sum\_of\_card - total\_sum where CUSTOMER\_ID = customer\_id;

delete from basket where CUSTOMER\_ID = customer\_id;

insert into delivery (ORDER\_ID, STATUS) values (order\_id, 'on the way');

exception

when basket\_empty then

dbms\_output.put\_line('Basket is empty');

when not\_enough then

dbms\_output.put\_line('Not enough balance in card');

when no\_customer then

dbms\_output.put\_line('There is no such customer');

when not\_enough\_book then

dbms\_output.put\_line('Unfortunately there is not enough amount of books for you');

when others then

dbms\_output.put\_line('Customer doesn''t have card');

end;

2) create or replace procedure confirm\_delivery

(order\_id in number)

is

begin

update delivery set STATUS = 'delivered' where ORDER\_ID = order\_id;

delete from "ORDER" where ORDER\_ID = order\_id;

delete from order\_detail where ORDER\_ID = order\_id;

exception

when others then

dbms\_output.put\_line('There is no such order');

end;

3) create or replace PROCEDURE CUSTOMERS\_BASKET(

CUST\_ID BASKET.CUSTOMER\_ID%TYPE

) IS

F\_NAME CUSTOMER.FIRST\_NAME%TYPE;

L\_NAME CUSTOMER.LAST\_NAME%TYPE;

CURSOR "C\_BASKET" IS

SELECT BOOK."NAME" AS TITLE, BASKET.QUANTITY AS QUANTITY

FROM BASKET JOIN BOOK USING (BOOK\_ID)

WHERE CUSTOMER\_ID = CUST\_ID;

BEGIN

SELECT FIRST\_NAME, LAST\_NAME INTO F\_NAME, L\_NAME FROM CUSTOMER WHERE CUSTOMER\_ID = CUST\_ID;

DBMS\_OUTPUT.PUT\_LINE('Customer full name: ' || F\_NAME || ' ' || L\_NAME);

FOR "BASKET" IN "C\_BASKET" LOOP

DBMS\_OUTPUT.PUT\_LINE('Book name: ' || "BASKET"."TITLE" || ', Quantity: ' || "BASKET"."QUANTITY");

END LOOP;

END;

4) create or replace PROCEDURE SUPPLIER\_BOOKS IS

CURSOR "C\_INFO" IS

SELECT "SUPPLIER\_ID", "SUPPLIER"."NAME" AS "SUPPLIER\_NAME", COUNT(\*) AS "NUMBER\_OF\_BOOKS"

FROM "BOOK" JOIN "SUPPLIER" USING ("SUPPLIER\_ID")

GROUP BY "SUPPLIER\_ID", "SUPPLIER"."NAME"

ORDER BY "SUPPLIER\_ID";

BEGIN

FOR "INFO" IN "C\_INFO" LOOP

DBMS\_OUTPUT.PUT\_LINE('Supplier name: ' || "INFO"."SUPPLIER\_NAME");

DBMS\_OUTPUT.PUT\_LINE('Number of books: ' || "INFO"."NUMBER\_OF\_BOOKS");

DBMS\_OUTPUT.PUT\_LINE('–––––––––––––––');

END LOOP;

END;

**FUNCTION:**

create or replace function count\_rows

(table\_name in varchar)

return number

is

num number;

begin

execute immediate 'select count(\*) from '||table\_name into num;

return num;

exception

when others then

if sqlcode = -942 then

dbms\_output.put\_line('No such table exists');

else

dbms\_output.put\_line('Something else happened');

end if;

end;

**PACKAGE:**

create or replace PACKAGE CARD\_DATA AS

PROCEDURE ADD\_CARD(

CRD\_ID CARD.CUSTOMER\_ID%TYPE,

CRD\_NUM CARD.CARD\_NUM%TYPE,

CRD\_CVV CARD.CVV%TYPE,

CRD\_EXP CARD.EXP\_DATE%TYPE,

CRD\_BALANCE CARD.BALANCE%TYPE

);

END CARD\_DATA;

create or replace PACKAGE BODY CARD\_DATA AS

PROCEDURE ADD\_CARD(

CRD\_ID CARD.CUSTOMER\_ID%TYPE,

CRD\_NUM CARD.CARD\_NUM%TYPE,

CRD\_CVV CARD.CVV%TYPE,

CRD\_EXP CARD.EXP\_DATE%TYPE,

CRD\_BALANCE CARD.BALANCE%TYPE

) IS

lcount NUMBER;

BEGIN

SELECT COUNT(\*) INTO lcount FROM CUSTOMER WHERE CUSTOMER\_ID IN (SELECT CUSTOMER\_ID FROM CARD WHERE CUSTOMER\_ID = CRD\_ID);

IF lcount = 0 THEN

INSERT INTO CARD

(CUSTOMER\_ID, CARD\_NUM, CVV, EXP\_DATE, BALANCE)

VALUES

(CRD\_ID, CRD\_NUM, CRD\_CVV, CRD\_EXP, CRD\_BALANCE);

DBMS\_OUTPUT.PUT\_LINE('CARD IS SUCCESSFULLY CREATED!');

ELSE

DBMS\_OUTPUT.PUT\_LINE('THERE IS ALREADY EXIST CARD WITH THIS ID, PLEASE WRITE ANOTHER');

END IF;

END ADD\_CARD;

END CARD\_DATA;

/

create or replace PACKAGE CUSTOMER\_DATA AS

PROCEDURE ADD\_CUSTOMER(

CUST\_FNAME CUSTOMER.FIRST\_NAME%TYPE,

CUST\_LNAME CUSTOMER.LAST\_NAME%TYPE,

CUST\_ADDRESS CUSTOMER.ADDRESS%TYPE,

CUST\_PHONE CUSTOMER.PHONE\_NUMBER%TYPE

);

PROCEDURE DELETE\_CUSTOMER(

CUST\_ID CUSTOMER.CUSTOMER\_ID%TYPE

);

END CUSTOMER\_DATA;

/

create or replace PACKAGE BODY CUSTOMER\_DATA AS

PROCEDURE ADD\_CUSTOMER(

CUST\_FNAME CUSTOMER.FIRST\_NAME%TYPE,

CUST\_LNAME CUSTOMER.LAST\_NAME%TYPE,

CUST\_ADDRESS CUSTOMER.ADDRESS%TYPE,

CUST\_PHONE CUSTOMER.PHONE\_NUMBER%TYPE

) IS

c\_id NUMBER;

c\_id1 NUMBER;

BEGIN

SELECT COUNT(\*) INTO c\_id FROM CUSTOMER;

c\_id1:= c\_id+1;

INSERT INTO CUSTOMER

(CUSTOMER\_ID, FIRST\_NAME, LAST\_NAME, ADDRESS, PHONE\_NUMBER)

VALUES

(c\_id1, CUST\_FNAME, CUST\_LNAME, CUST\_ADDRESS, CUST\_PHONE);

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER IS SUCCESSFULLY CREATED!');

END ADD\_CUSTOMER;

PROCEDURE DELETE\_CUSTOMER(

CUST\_ID CUSTOMER.CUSTOMER\_ID%TYPE

) IS

BEGIN

DELETE FROM CARD WHERE CUSTOMER\_ID = CUST\_ID;

DELETE FROM CUSTOMER WHERE CUSTOMER\_ID = CUST\_ID;

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER AND CARD WAS SUCCESSFULLY DELETED!');

END DELETE\_CUSTOMER;

END CUSTOMER\_DATA;

**FUNCTIONAL DEPENDENCY (FD):**

**Keys:** CUSTOMER\_ID, CARD\_ID, BOOK\_ID, SUPPLIER\_ID, DELIVERY\_ID, ORDER\_ID, TRANSACTION\_ID

**The minimal cover of FDs:**  
CUSTOMER\_ID => CARD\_ID  
CUSTOMER\_ID => ORDER\_ID  
CUSTOMER\_ID => BOOK\_ID  
CUSTOMER\_ID => TRANSACTION\_ID  
BOOK\_ID => SUPPLIER\_ID  
ORDER\_ID => DELIVERY\_ID

**The super key:** CUSTOMER\_ID  
I. CUSTOMER\_ID => TRANSACTION\_ID, BOOK\_ID, ORDER\_ID, CARD\_ID  
II. BOOK\_ID => SUPPLIER\_ID  
III. ORDER\_ID => DELIVERY\_ID