VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

VELLORE INSTITUE OF TECHNOLOGY

OCTOBER 2019

HOTEL MANAGEMENT

Submitted by

REDG.NO1:18BIT0137

REDG.NO2:18BIT0147

REDG.NO3:18BIT0126

In partial fulfilment of the requirements for Project J component – ITE 1003

Submitted to

V.Shashikiran sir

INDEX:

INTRODUCTION:

- Here we had taken a hotel management system because now a days the tourism is one of the most popular and economical benefit for a place. As the tourism increases the place needs to be accomplished with the hotels for allocating the people comfortably.
- Here we propose an online webpage where the user can find hotels
- This helps the customer to know the prices of the rooms available in the hotel and the hotel location and facilities.
- So we wanted to create a webpage which would help them keep track of the room availability and we will also daily send newsletters which will be helpful to our customers.
- The webpage needs a good database to be connected with it to work efficiently.
- So here in our project we are going to present a database that helps the webpage to make it available and efficient to the users.
- By encapsulating our database design to the webpage one can be able to build up a web application with front end and back end setup correctly

- Languages and Frameworks used for front-end:
- HTML, CSS, BOOTSTRAP, JAVASCRIPT
- Languages and Frameworks used for back-end:
- PHP
- Database: MYSQL.

Functional Requirements and Conceptual view of the Database Design;

- Our webpage contains a registration page where the user can register with (name, password ,email ,gender ,address).
- Also contain login page where the user is able to login and get to know about rooms available, room rents, payment methods available, Wi-Fi credentials.
- The user can also see the gallery of the hotel where the hotel interior and exterior shown beautifully.
- Here the database is needed to store the hotel room details, to store the users' details and to store the rooms that are booked by the customer.
- Here thereby we need the entities like admin, user details, room registration details, food order detail, and staff in hotel.
- Here we design the E-R Diagram for our better conceptual view and from the entity diagram we are able to know in depth about each entity type.

DOUBLE ELLIPSE-Multivalued attribute

DOUBLE RECTANGLE-Weak attribute

DOTTED ELLIPSE-Derived attribute

ELLIPSE WITH LINE IN IT-Primary key(key attribute)

MULTI ELLIPSE CONNECTED-Composite attribue

DOUBLE KITE-Identifying relationship

Partial participation of the attribute in relationship

Fully participation

Cardinality ratio: This is the one which tells the participation of the entites that are related by a relation ship.

- To make a relational database there should be a like every relation(table) should be related to each other that is a query that can be made upon our database should be able to retriev the values as such.
- So here by insereting the new foreign keys into relation we make the relation to be related with each other.

- This can also be said to be mapping of e-r diagram to a relational schema.The constraints are the which impose or enforce the rules over the database to remove complexity and the possibilites of the redundancy.
- Constraints are of 3 types

1.key constraints

2.Entity integrity constrinats

3.Referential integrity constrints

Possibly we should avoid the null vlaues as less as possible to make our database

In order a desing a database some GUIDELINES are provided

1.INFORMAL GUIDELINES

- Informally each tuple in a relation should represent one entity or relationship instance.
- Design a schema that does not suffer from the insert, delete, and update anomalies.
- Avoid null values.
- There shouldn’t be generation of spurious tuples when 2 tuples are joined.

2.FORMAL GUIDELINES

- Functional dependencies.

MAPPING OF E-R DIAGRAM TO RELATIONAL SCHEMA

Table name:customer

ATTRIBUTE NAME	DATA TYPE	CONSTRAINT
ssn	Varchar2(30)	Primary key
name	Varchar2(30)	Not null
country	Varchar2(30)	Not null
email	Varchar2(30)	Not null
Customer_id	Varchar2(30)	Primary key
Phone_number	Varchar2(20)	Multivalued

Table name:invoice

ATTRIBUTE NAME	DATA TYPE	CONSTRAINT
Invoice_id	Varchar2(30)	Primary key
status	Varchar2(30)	Any from domain
Invoice_description	Varchar2(30)	Not null

Table name:Bill

ATTRIBUTE NAME	DATA TYPE	CONSTRAINT
Bill_id	Varchar2(30)	Primary key
amount	Number(30)	Not null
name	Varchar2(30)	Not null
status	Varchar2(30)	Not null
date	date	Not null

Table name:Hotel

ATTRIBUTE NAME	DATA TYPE	CONSTRAINT
Name	Varchar2(30)	Not null
Location	Varcha2(30)	Not null
Hotel_id	Varchar2(30)	Primary key
Gst_id	Varchar2(30)	Primary key

Table name:Todays price

ATTRIBUTE NAME	DATA TYPE	CONSTRAINT
Hotel_id	Varchar2(30)	Primary key(foreign)
Price	Number(30)	Not null
Availability	Varchar2(30)	Any value from domain
Date	date	Not null

Table name:Room_category

ATTRIBUTE NAME	DATA TYPE	CONSTRAINT
Customer_id	Varchar2(30)	Primary key(foreign)
Name	Varchar2(30)	Not null
Hotel_id	Varchar2(30)	Primary key(foreign)
Room_num	Varchar2(30)	Primary key(foreign)
Room_type	Varchar2(30)	Not null

Table name:Rooms

ATTRIBUTE NAME	DATA TYPE	CONSTRAINT
Room_nos	Varchar2(30)	Not null87
Room_type	Varchar2(30)	Not null

Table name: Reservation

ATTRIBUTE NAME	DATA TYPE	CONSTRAINT
Reservation_id	Varchar2(30)	Primary key
period	Varchar2(30)	Not null
Start_date	date	Not null
End_date	date	Not null
Customer_id	Varchar2(30)	Primary key(foreign)

Table name:Payment

ATTRIBUTE NAME	DATA TYPE	CONSTRAINT
Payment_method	Varchar2(30)	Not null
Date	date	Not null

Table name:Phone_number

ATTRIBUTE NAME	DATA TYPE	CONSTRAINT
Customer_id	Varchar2(30)	Forigen key
Ph1	Varchar2(30)	From the domain
Ph2	Varchar2(30)	From the domain
Ph3	Varchar2(30)	From the domain
Ph4	Varchar2(30)	From the domain
Ph5	Varchar2(30)	From the domain

CUSTOMER

ssn	name	Country	email	Customer_id

Constraints

1.Constraint cus_pk PRIMARY KEY(ssn);

2.Constraint name_chek CHECK(name!=NULL)

Constraint name_chek1 CHECK(Country!=NULL)

4.Constraint name_chek2 CHECK(email!=NULL)

5.constraint name_chek3 FOREIGN KEY(CUSTOMER_ID) REFERENCES CSN(CUSTOMER_ID);

FUNCTIONAL DEPENDENCIES

1.ssn->{name,country,email,customer_id};

2.customer_id->{ssn};

Customer:::

customer_id	name	Country	Email	SSN
123456789	AJAY	INDIA	ajay@gmail.com	1234567
234567891	SUVAS	AUSTRILIA	Suvas1234@hotmail.com	2345671
345678912	JAYANTH	U.S.A	Jayanth223@yahoo.com	3456712
456789123	PRAKASH	BRAZIL	parakash@rediff.com	4567123
567891234	SUMANTH	NEWZELAND	sumanth@gmail.com	5671234

INVOICE

Invoice_id	status	Invoice_description	Customer_id

Constraints

1.Constraint in_pk PRIMARY KEY(Invoice_id);

2.Constraint in_fk3 FOREIGN KEY(Customer_id) references SCN(Customer_id);

3.Constraint invo_chek CHECK(Invoice_description!=NULL)

FUNCTIONAL DEPENDENCIES

1.invoice_id->{status,invoice_description,customer_id};

2.Customer_id->Invoice_id;

Invoice::

Invoice_id	status	Invoice_description	Customer_id
12345678	generated	21,000 RS	123456789
23456781	generated	10,000 RS	234567891
34567812	generated	30,000 RS	345678912
45678123	generated	40,000 RS	456789123
56781234	generated	50,000 RS	567891234
67812345	generated	22,000 RS	123456789

BILL

Bill_id	amount	status	date	Customer_id	Invoice_id

Constraints

1.Constraint bil_pk PRIMARY KEY(Bill_id);

2.Constraint bil_fk3 FOREIGN KEY(Customer_id) references SCN(Customer_id);

3.Constraint bill_chek4 CHECK(name!=NULL);

4.Constraint bill_chek3 CHECK(date!=NULL);

5.Constraint bill_chek2 CHECK(amount!=NULL);

6.Constraint bill_chek1 CHECK(status!=NULL);

FUNCTIONAL DEPENDENCIES

1.BILL_ID->{amount,name,type,date,customer_id,invoice_id};

BILL::

Bill_id	amount	status	date	Customer_id	Invoice_id
098765	21,000	paid	23-Aug-2019	123456789	12345678
087659	10,000	paid	22-Jun-2019	234567891	23456781
076598	20,000	paid	19-Jul-2019	345678912	34567812
065987	40,000	paid	12-May-2019	456789123	45678123
059876	30,000	waiting	19-Sep-2019	123456789	56781234

HOTEL

Name	location	Hotel_id	Gst_id

Constraints

1.Constraint hote_pk PRIMARY KEY(Hotel_id,Gst_id);

2. Constraint hote_chek CHECK(name!=NULL);

FUNCTIONAL DEPENDENCIES

1.Hotel_id->{name,location,gst_id};

HOTEL:::

Name	location	Hotel_id	Gst_id
Hotel Ashoka	BANGLORE	123111000	Gst110-210-310
The Grand Charriot	GOA	234222000	Gst220-320-420
Hotel Grand Krishna	PONDICHEERY	345333000	Gst330-430-530
Taj Mahal Tower	MUMBAI	456444000	Gst440-540-640
The St. Rgis	CHENNAI	567555000	Gst550-650-750

TODAYS_PRICE

Hotel_id	Price	Availability	Date	Room_type	Room_nos

Constriants

1. Constraint tod_fk2 FOREIGN KEY(Hotel_id) references hotel(hotel_id);
2. Constraint tp_chek CHECK(date!=NULL);
3. Constraint tp_chek2 CHECK(price!=NULL);

FUNCTIONAL DEPENDENCIES

1.{hotel_id,room_nos}->{price,availability,date,room_type};

2.{hotel_id,room_type}->{price,room_nos,date,availability};

TODAYS_PRICE::

Hotel_id	Price	Availability	Date	Room_type	Room_nos
123111000	5,000	10	19-Sep-2019	A.C	201-220
123111000	2,000	20	19-Sep-2019	N.A.C	101-120
234222000	4,000	10	19-Sep-2019	A.C	210-240
345333000	3,000	5	19-Sep-2019	A.C	100-140
345333000	1,000	2	19-Sep-2019	N.A.C	160-180
456444000	2,500	13	19-Sep-2019	A.C	100-140
567555000	1,500	8	19-Sep-2019	N.A.C	120-180

ROOMS

Room_nos	Room_type	Hotel_id

Constraints

2.Constraint rom_fk FOREIGN KEY(Hotel_id) references Hotel(Hotel_id);

3.Constraint rooms_chek CHECK(room_type!=NULL);

FUNCTIONAL DEPENDENCIES

1.{hotel_id,room_type}->{room_nos};

ROOMS::

Room_nos	Room_type	Hotel_id
201-220	A.C	123111000
101-120	N.A.C	123111000
210-240	A.C	234222000
100-140	A.C	345333000
160-180	N.A.C	345333000
100-140	A.C	456444000
120-180	N.A.C	567555000

ROOMS_CATEGORY

Room_num	Customer_id	Room_type	Hotel_id	Check_in	Check_out

Constraints

1.Constraint roc_fk2 FOREIGN KEY(Hotel_id) references Hotel(Hotel_id);

2.Constraint tod_fk3 FOREIGN KEY(Customer_id) references SCN(Customer_id);

3.Constraint rc_chek CHECK(room_type!=NULL);

4.Constraint rc_chek2 CHECK(check_out!=NULL);

5.Constraint rc_chek3 CHECK(check_in!=NULL);

6.Constraint rc_chek4 CHECK(name!=NULL);

FUNCTIONAL DEPENDENCIES

1.{room_num,hotel_id}->{name,room_type,check_in,check_out};

ROOMS_CATEGORY::

Room_num	Customer_id	Room_type	Hotel_id	Check_in	Check_out
202	123456789	A.C	123111000	21-08-19	23-08-19
111	234567891	N.A.C	123111000	20-06-19	22-06-19
212	345678912	A.C	234222000	17-07-19	19-07-19
102	456789123	A.C	345333000	10-05-19	12-05-19
164	567891234	N.A.C	345333000	17-09-19	19-09-19
114	123456789	A.C	456333000	12-08-19	14-08-19
126	234567891	N.A.C	567555000	13-09-19	15-09-19

RESERVATION

Reservation_id	Period	Customer_id	Room_num	Hotel_id

Constraints

1.Constraint res_fk FOREIGN KEY(Customer_id) references SCN(Customer_id);

2.Constraint res_pk PRIMARY KEY(Reservation_id);

3.Constraint res_chek1 CHECK(end_date!=NULL);

4.Constraint res_chek3 CHECK(start_date!=NULL);

5.Constraint res_chek2 CHECK(period!=NULL);

6. Constraint res_fk FOREIGN KEY(Hotel_id) references Hotel(Hotel_id);

FUNCTIONAL DEPENDENCIES

1.RESERVATION_ID->{period,start_date,end_date,customer_id,room_num};

3.customer_id->reservation_id;

RESERVATION::

Reservation_id	Period	Customer_id	Room_num	Hotel_id
0001234567	2	123456789	202	123111000
0002345671	2	234567891	111	123111000
0003456712	2	345678912	212	234222000
0004567123	2	456789123	102	345333000
0005671234	2	567891234	164	345333000
0006712345	2	123456789	114	456333000
0071234567	2	234567891	126	567555000

PAYMENT

Payment_method	Date	Customer_id	Bill_id

Constraints

1. Constraint pay_fk3 FOREIGN KEY(Customer_id) references SCN(Customer_id);
2. Constraint pay_fk4 FOREIGN KEY(Bill_id) references Bill(Bill_id);
3. Constraint pay_chek3 CHECK(payment_method!=NULL);
4. Constraint pay_chek2 CHECK(date!=NULL);

FUNCTIONAL DEPENDENCIES

1. Customer_id->Bill_id;
2. Bill_id->{payment_method,date};

PAYMENT::

Payment_method	Date	Customer_id	BILL_ID
Card payment	23-08-19	123456789	098765
Card payment	22-06-19	234567891	087659
Card payment	19-07-19	345678912	076598
Net banking	12-05-19	456789123	065987
Net banking	19-09-19	567891234	059876
Card payment	14-08-19	456789123	055557

Phone_number

<u>Customer_id</u>	Ph1	Ph2	Ph3	Ph4	Ph5

Constraints

1. Constraint pho_fk FOREIGN KEY(Customer_id) references SCN(Customer_id);

FUNCTIONAL DEPENDENCIES

1.customer_id->{ph1,ph2,ph3,ph4,ph5};

Phone_number:::

<u>Customer_id</u>	Ph1	Ph2	Ph3	Ph4	Ph5
123456789	9959892263	9459892364	NULL	NULL	NULL
234567891	9912620066	9991120123	NULL	NULL	NULL
345678912	9010366473	9100366374	6305762343	NULL	NULL
456789123	6303186334	9633018633	9874352367	9105442323	NULL
567891234	6305164354	9630516454	9458723490	9845672131	NULL

SCN

Customer_id	SSN

Constraints

1. Constraint SC_fk FOREIGN KEY(SSN) references CUSTOMER(SSN);
2. CONSTRAINT SC_PK PRIMARY KEY(CUSTOMER_ID);

FUNCTIONAL DEPENDENCIES

1.customer_id->{SSN};

Customer_id	SSN
123456789	1234567
234567891	2345671
345678912	3456712
456789123	4567123
567891234	5671234

HOW THE TABLES ARE IN 1NF

WE can say that one given relation is in 1NF by eliminating the

1.) multi-valued and creating new table for it

2) composite attribute and creating new table for it

3.) nested relations

- Above all the relations has no multivalued attribute
  except the customer table has the phone_number
- So we have created a separate table for the phone_number with customer_id as key.
- And in the remaining tables there are no multivalued attributed or composite attributes and no nested relations.
- There by here we can conclude that the above formed relations are in '1NF'.
- AS we are now done with 1NF we are now going to decompose the realtion to 2NF
- The decomposition must be functional dependency preserving and loss less join

HOW THE TABLES IN 2NF

X->A BELONGS TO “F” {THE FUNCTIONAL DEPENDENCIES SET}

X MUST BE A PRIME ATTTRIBUTE.

AND A IS A NON-PRIME ATTRIBUTE.

- IN THE ABOVE TABLES THE IMPLIED FUNCTIONAL DEPENDECIES ON EACH RELATION OF TYPE ‘X->A’ THE ‘X’IS A PRIME ATTRIBUTE IN EACH FUNCTIONAL DEPENDENCIES WHERE AS THE ‘A’ IS THE NON-PRIME ATTRIBUTE.

CUSTOMER:::

ssn->{name,country,email,customer_id};

- THE ABOVE ARE THE FUNCTIONAL DEPENDENCIES FROM THE TABLE ‘CUSTOMER’.HERE ABOVE WE OBSERVE THAT FOR EACH FUNCTIONAL DEPENDENCY THE X->A X IS A PRIME ATTRIBUTE.
- AND FINALLY THE TABLES ARE IN 2NF

STORED PROCEDURES

- TO FIND HOTELS IN A PARTICULAR LOCATION

```
CREATE OR REPLACE PROCEDURE HOTELSIN(LOC VARCHAR2) AS

CURSOR HOTELSF_CUR

IS  SELECT HOTEL_ID,ROOM_NOS,ROOM_TYPE FROM ROOMS

WHERE HOTEL_ID IN(SELECT HOTEL_ID FROM HOTEL WHERE
LOCATION=HOTELSIN.LOC);

H_REC HOTELSF_CUR%ROWTYPE;

CURSOR HNAME(HO_ID VARCHAR2) IS SELECT NAME FROM HOTEL
WHERE HOTEL_ID=HO_ID;

HN_REC HNAME%ROWTYPE;

BEGIN

OPEN HOTELSF_CUR;

LOOP

FETCH HOTELSF_CUR INTO H_REC;

EXIT WHEN HOTELSF_CUR%notfound;

OPEN HNAME(H_REC.HOTEL_ID);

FETCH HNAME INTO HN_REC;

DBMS_OUTPUT.PUT_LINE(HN_REC.NAME||'->'||H_REC.ROOM_NOS||'-
>'||H_REC.ROOM_TYPE);

CLOSE HNAME;

END LOOP;

CLOSE HOTELSF_CUR;

END ;
```

/

- TO FETCH THE ROOM DETAILS WITH CUSTOMER_ID AND HOTEL_ID;

```
CREATE OR REPLACE PROCEDURE CDFETCH(C_ID VARCHAR2,H_ID VARCHAR2) AS

CURSOR C_DET IS SELECT * FROM CUSTOMER WHERE CUSTOMER_ID=CDFETCH.C_ID;

CURSOR H_DET IS SELECT HOTEL_ID,NAME FROM HOTEL WHERE HOTEL_ID=CDFETCH.H_ID;

CURSOR R_DET IS SELECT ROOM_NUM FROM ROOM_CATEGORY WHERE HOTEL_ID=CDFETCH.H_ID AND CUSTOMER_ID=CDFETCH.C_ID;

C_REC C_DET%ROWTYPE;

H_REC H_DET%ROWTYPE;

R_REC R_DET%ROWTYPE;

BEGIN

OPEN C_DET;

OPEN H_DET;

OPEN R_DET;

FETCH C_DET INTO C_REC;

FETCH H_DET INTO H_REC;

LOOP

FETCH R_DET INTO R_REC;

EXIT WHEN R_DET%notfound;
```

```
DBMS_OUTPUT.PUT_LINE(R_REC.ROOM_NUM||'->'||C_REC.NAME||'->'||C_REC.EMAIL||'->'||H_REC.NAME);

END LOOP;

CLOSE C_DET;

CLOSE R_DET;

CLOSE H_DET;

END;

/
```

- TO FETCH THE CUSTOMER DETAILS FROM ROOM_NUM AND HOTEL_ID

```
CREATE OR REPLACE PROCEDURE CFETCH(C_ID VARCHAR2,H_ID VARCHAR2) AS

CURSOR H_DET IS SELECT HOTEL_ID,NAME FROM HOTEL WHERE HOTEL_ID=CFETCH.H_ID;

CURSOR R_DET IS SELECT CUSTOMER_ID FROM ROOM_CATEGORY WHERE HOTEL_ID=CFETCH.H_ID AND ROOM_NUM=CFETCH.C_ID;

CURSOR C_DET(CC_ID VARCHAR2) IS SELECT * FROM CUSTOMER WHERE CUSTOMER_ID=CC_ID;

H_REC H_DET%ROWTYPE;

C_REC C_DET%ROWTYPE;

R_REC R_DET%ROWTYPE;

BEGIN

OPEN H_DET;

OPEN R_DET;
```

```
FETCH H_DET INTO H_REC;

LOOP

FETCH R_DET INTO R_REC;

OPEN C_DET(R_REC.CUSTOMER_ID);

FETCH C_DET INTO C_REC;

EXIT WHEN R_DET%notfound;

DBMS_OUTPUT.PUT_LINE(C_REC.CUSTOMER_ID||'-
>'||C_REC.NAME||'->'||C_REC.EMAIL||'->'||H_REC.NAME);

CLOSE C_DET;

END LOOP;

CLOSE R_DET;

CLOSE H_DET;

END;

/
```

- TO GENERATE INVOICE

```
CREATE OR REPLACE PROCEDURE INVOICEGEN(C_ID
VARCHAR2,INV VARCHAR2,INV_D VARCHAR2) AS

CURSOR CC_D IS SELECT * FROM CUSTOMER WHERE
CUSTOMER_ID=INVOICEGEN.C_ID;

C_REC CC_D%ROWTYPE;

K NUMBER;

BEGIN

OPEN CC_D;
```

```
K:=0;

LOOP

FETCH CC_D INTO C_REC;

EXIT WHEN CC_D%NOTFOUND;

K:=K+1;

END LOOP;

IF(K>0) THEN

INSERT INTO INVOICE
VALUES(INVOICEGEN.INV,'GENERATED',INVOICEGEN.INV_D,INVOI
CEGEN.C_ID);

ELSE

DBMS_OUTPUT.PUT_LINE('THERE WAS NO CUSTOMER WITH THAT
ID KINDLY FIRST CREATE CUSTOMER PROFILE AND PROCEED');

COMMIT;

END;/
```

- TO CHANGE THE ROOMNOS

```
CREATE OR REPLACE PROCEDURE UPDATEROOMNOS(HO_ID
VARCHAR2,ACR VARCHAR2,NACR VARCHAR2) AS

ACROOMS ROOMS.ROOM_NOS%TYPE;

NACROOMS ROOMS.ROOM_NOS%TYPE;

BEGIN

ACROOMS:=UPDATEROOMNOS.ACR;

NACROOMS:=UPDATEROOMNOS.NACR;

UPDATE ROOMS SET ROOM_NOS=ACROOMS WHERE
ROOM_TYPE='A.C' AND HOTEL_ID=UPDATEROOMNOS.HO_ID;
```

```
UPDATE ROOMS SET ROOM_NOS=NACROOMS WHERE
ROOM_TYPE='N.A.C' AND HOTEL_ID=UPDATEROOMNOS.HO_ID;

END;

/
```

TRIGGER

1. TRIGGER ON HOTEL TABLE

```
CREATE OR REPLACE TRIGGER

 UPDATE_ROOMS_TABLE

 AFTER INSERT ON HOTEL FOR EACH ROW

 DECLARE

 HOT_ID ROOMS.HOTEL_ID%TYPE;

ACN ROOMS.ROOM_NOS%TYPE;

NACN ROOMS.ROOM_NOS%TYPE;

AAC ROOMS.AVAILABLE%TYPE;

ANAC ROOMS.AVAILABLE%TYPE;

BEGIN

ACN:='&ACN';

NACN:='&NACN';

AAC:='&AAC';

ANAC:='&ANAC';

HOT_ID:=:NEW.HOTEL_ID;

INSERT INTO ROOMS
(ROOM_NOS,ROOM_TYPE,HOTEL_ID,AVAILABLE)
VALUES(ACN,'A.C',HOT_ID,AAC);
```

```
INSERT INTO ROOMS
(ROOM_NOS,ROOM_TYPE,HOTEL_ID,AVAILABLE)
VALUES(NACN,'N.A.C',HOT_ID,ANAC);

DBMS_OUTPUT.PUT_LINE('THE CORRESPONDING TABLES HAVE
BEEN UPDATED');

END;

 /
```

**********END**********