DBMS Project

R.S. Drycleaners

Submitted To:

Dr. Parteek Bhatia

Submitted By:

Naman Ahuja(101203059)

Nishant Gupta(10203066)

Nitesh Singh(101203067)

Sarah Afrin(101213048)

Department Of Computer Science and Engineering

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of DBMS Project undertaken during the 5th semester. We owe special debt of gratitude to our mentor Dr.Parteek Bhatia, Assistant Professor, Department of Computer Science & Engineering, Thapar University, Patiala for his constant support and work throughout the course of our work .His sincerity, thoroughness and perseverance have been a constant source of inspiration for us.We would like to acknowledge and thank our group members and seniors of our college for their sincere advice, constant guidance, supervision and continuous encouragement throughout the preparation of the project.At last, but not the least we would like to thank our college management for providing us the good environment to do this report.

CONTENTS

REQUIREMENT ANALYSIS

1. Introduction

In this project we Aim to Automate the Laundry Service in Thapar University. Currently it is a manually operated shop located in COS Complex whose customers are both students and faculty members.

2. Current system

A) Currently the laundry service is manual and there is no way of verifying the ownership of the clothes.

B) The Process of tagging clothes of a customer is not fool proof .It is prone to error and clashes.

C) The billing System is manual sometimes leading to mistakes in calculation.

3. Proposed system

3.1 Overview

We plan to automate this laundry system by :

A) Making a system for tagging of clothes

B) Method for Verifying Identity

C) Generating Bills

D) Online status of the job

3.2 Functional requirements

What the laundry requires:

1. Unique tagging of clothes from a single customer.
2. Easy entering of data.
3. To enter no of clothes along with the job to be done.
4. Automatic billing of all clothes.
5. Online updating whether the job is done or not.
6. Fool proof technique to return clothes to the right customer.
7. To include all the chores that can be done by the laundry shop.

3.3 Nonfunctional requirements

A) Accessibility: this software will be available for all vendors.
B) Secure :Admin and user logins
C) Easy To use (made for people with little knowledge of Computers)
D) Training the staff to use the software
E) Software should ask only the most important details enabling least time spent per customer
F) Error handling to prevent incorrect information entering the system

The complete working of the system is explained for easy understanding.

Home Page

As soon as the admin opens the application an admin login page appears. It verifies the administrator using the system and his permissions. After logging in the home page appears. The title of this page is the name of the shop ‘RS Drycleaners’.

This page has four tabs which link to other pages. A brief description of these pages are(complete description explained later) :

a) Take in:
This is clicked when a customer gives an order. When this tab is clicked the user is asked for the roll no (for students), faculty id (for faculty) or any other unique id corresponding to the customer. This id is searched in the database. If a match is found then it’s a registered customer otherwise the customer is new. Then the admin is redirected to two different pages depending whether it is a new user or an already registered one. If the user is new his/her details are to be entered in the new page which further opens the next page where order is placed. If the customer is a student the details to be added include his name, roll no (both mandatory fields), room no and mobile number. Similar details are required for an entry of faculty member. A unique tag id is generated automatically as soon as a new customer is registered. This id is used for all transactions and also identification of clothes.
If the customer is already registered, the page where order is to be placed is opened directly. The number of clothes, items etc are added along with the chore associated to it. The expected date of return or delivery is eneterd. Then admin proceeds to the billing page. The total bill is calculated and a softcopy of bill is generated.

b) Delivery:
This tab is clicked when the shopkeeper returns the clothes to a customer and asks for unique id of the customer (eg. Roll no, Faculty id etc). It retrieves from the database the details of customer order and displays it in the form of bill. This bill includes the the tag code assigned to that customer. The shopkeeper searches the items corresponding to the code. If customer is a student direct cash payment at the shop is done and if it is a faculty member then bill is added to faculty account and items are delivered at residence.

c) Status Updates:
When the job is completed this status has to be updated in the database. When the admin clicks on this tab the list of all items grouped by customer appears whose delivery date is the current day. There are two options regarding this list: job done, delay. The entries whose job (like dry-cleaning) is done are selected and job doe button is clicked. If the job is not done and the date of delivery is to be postponed then delay is clicked. By default one day is increased. The option of search for entries is also available on this page.

d) Reports:
The admin can view daily and monthly reports of his shop. He can check his monthly or daily earnings.
Also he can check how many items are pending to be delivered on any day. By this the shopkeeper can check which items have been pending for a long time (delayed) and deliver them first. The date is selected in a drop down menu.

Status Update

When the job is completed this status has to be updated in the database. When the admin clicks on this tab the list of all items grouped by customer appears whose delivery date is the current day ,the admin can select other date as per his needs. There are two options regarding this list: job done, delay. The entries show are grouped by the unique id .The fields shown here are:

Roll No.	Key	No. of Clothes	Bill	Expected Date of Delivery
101203067	NSJ	12	328	19/11/2014
101203066	NGJ	4	28	20/11/2014
101203059	NAJ	1	38	18/11/2014
101213048	SAP	1	38	17/11/2014

The entries whose job (like dry-cleaning) is done are selected and job done button is clicked. If the job is not done and the date of delivery is to be postponed then delay is clicked. By default one day is increased. The option of search for entries is also available on this page. In search field ,the shopkeeper has a option to type Key or the unique id row corresponding to that entry is shown up and the above possible option are applied .

The Rows shown will be color coded

Red: Order Received

Yellow: Order Complete but not delivered

Green: Order Delivered

When the user will select REPORT tab from the home page , he will be taken to the report page which will have a simple and nice user interface . This page can be used by the shop owner to see what all clothes have to be delivered and what all clothes are pending for job.

This can be sorted according to the dates which will be a drop – down list .

Suppose user selects 'Pending For Job' and he selects today from date drop-down then all the jobs which are pending for today will be displayed along with yesterday's job .

Roll No	Key	No. of clothes	Bill	Expected date of delivery
101203066	NGJ	13	324	19/11/2014

Now if he selects 'Pending For Delivery' and he selects today from date drop-down then all the jobs which are pending for delivery will be displayed along with tomorrow's job .

Roll No	Key	No. of clothes	Bill	Expected date of delivery
101203063	NJK	11	456	21/03/2014

Color Coding will be followed in the whole application . Color Coding is as follows :

Green - Order Delivered

Yellow - Order Completed but not Delivered

Red   - Order Received

On clicking the TAKE IN Button On The Home Page , a drop down text box opens in which the user enters his/her roll number or faculty id.

If the user is not registered then he will be taken to the New User Page where there will be various fields like

- Roll Number
- Name
- Hostel
- Phone Number
- Gender
- Room Number

Which he will have to fill. After submitting those details he will be taken to a new page same as below.

If the user is already registered then he will be taken to Already Registered Page. That Page Will be divided in three sections .   On the rightmost section , there will be tabs of various services available to the user like

- Dry Cleaning
- Alteration
- Washing
- Ironing

Suppose the user clicks Washing , then a scrollable drop – down meu will appear in which there will be options of selecting

- Shirt
- Pants
- Jeans
- Shoes
- Others
- Jacket

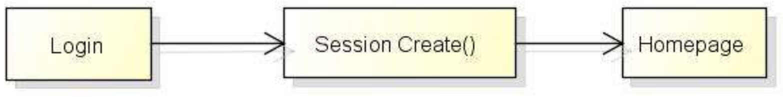
Now when the user select an item (say shirt) then on left section of the page entry corresponding to Washing machine -> Shirt -> Qty :1 will be inserted . However if the user selects shirt again in washing machine tab then the quantity will be increased by one i.e. Washing machine -> Shirt -> Qty :1 . will be updated to Washing machine -> Shirt -> Qty :2 . Similar is for the other items.

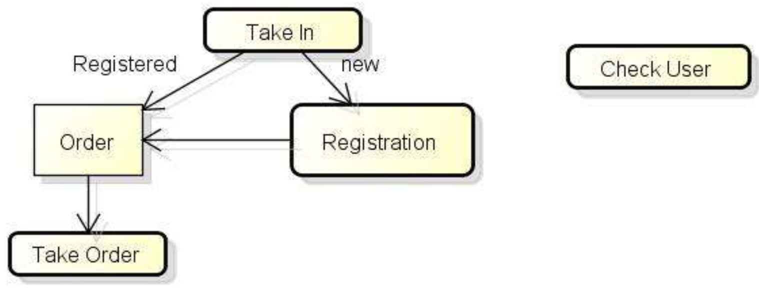
There is also available a remarks section in which any remarks regarding the condition of clothes or any other remark can be entered.

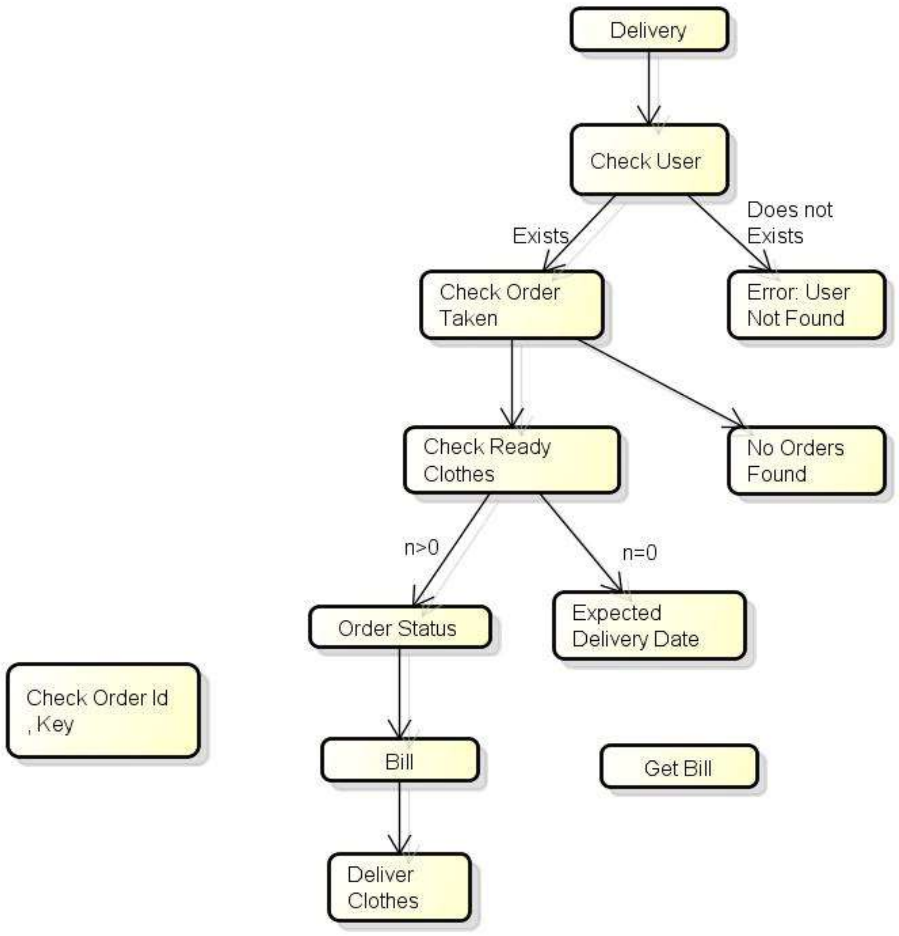
After that when the user clicks on the submit button a computerised bill will be generated .

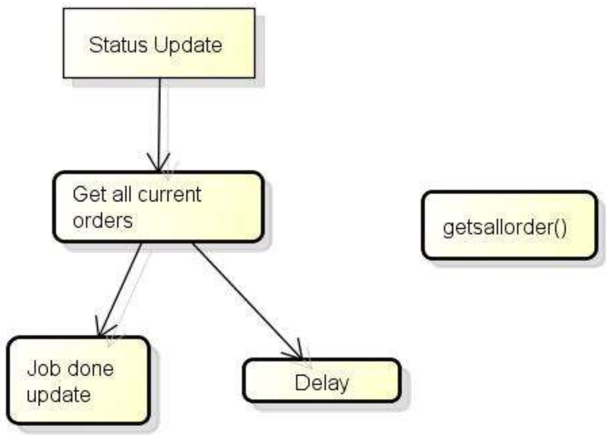
That Particular bill can now be printed .

Flow Diagram

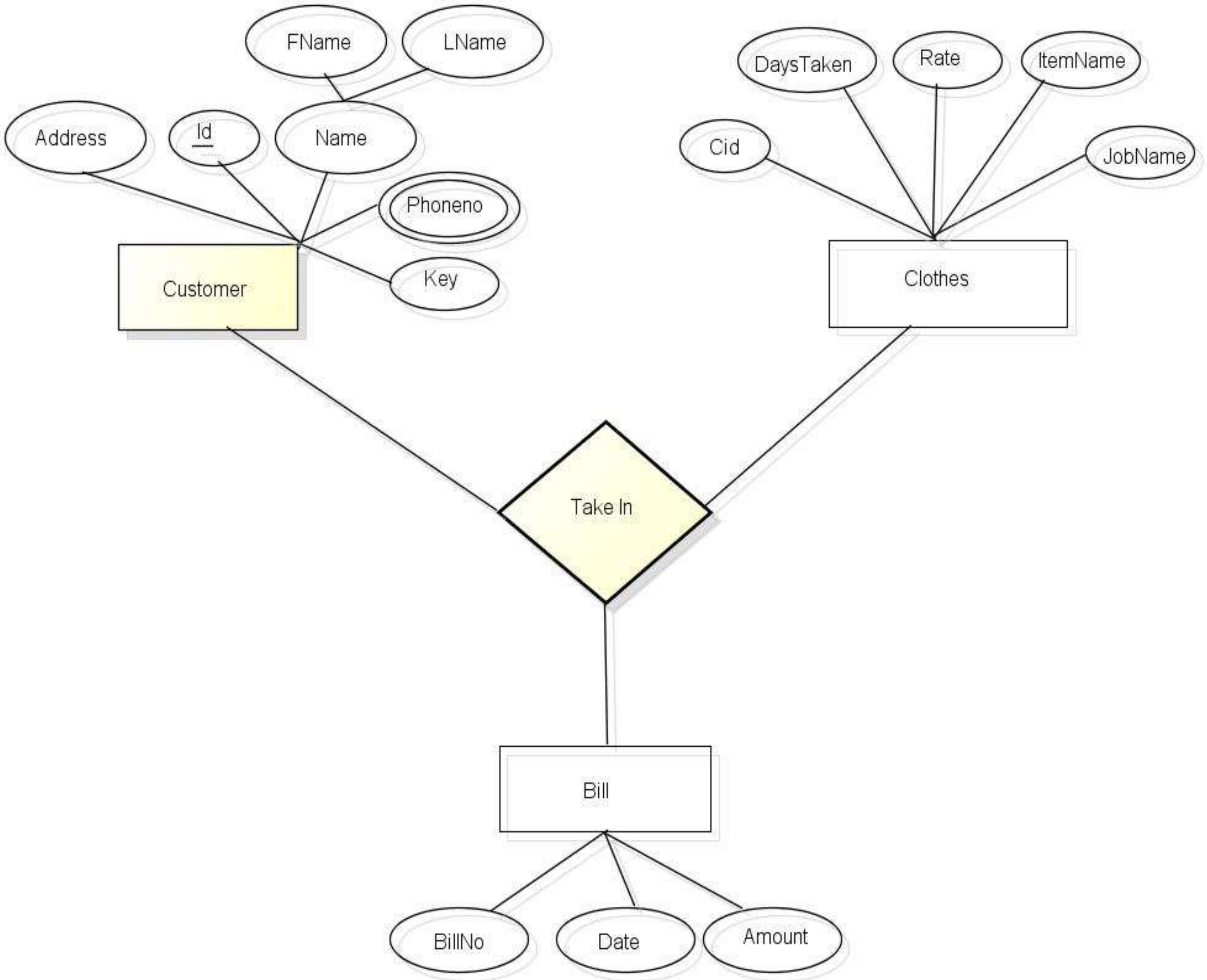








E R Diagram


FName
LName
Address
Id
Name
Phoneno
Key
Customer
DaysTaken
Rate
ItemName
Cid
JobName
Clothes
Take In
Bill
BillNo
Date
Amount

ER To Tables

Customer

Id	Varchar2
Firstname	Varchar2
Lastname	Varchar2
Key	Varchar2

Primary Key : Id

Phoneno

Id	Varchar2
Phoneno	Number(12)

Primary Key : Id

Clothes

Ak	Number(5)
Itemname	Varchar2
Jobname	Varchar2
Rate	Number(5)
Daystaken	Number(4)

Primary Key: ak

Takein

Key	Varchar2
Ak	Number(5)
Quantity	Number(5)
Datein	Date
Status	Varchar2
Remark	Varchar2

(key,ak): Primary Key

Bill

Billno	Number(7)
Key	Varchar2
Quantity	Number(5)
Edd	Date
Balance	Number(4)
Total	Number(4)
Status	Varchar2

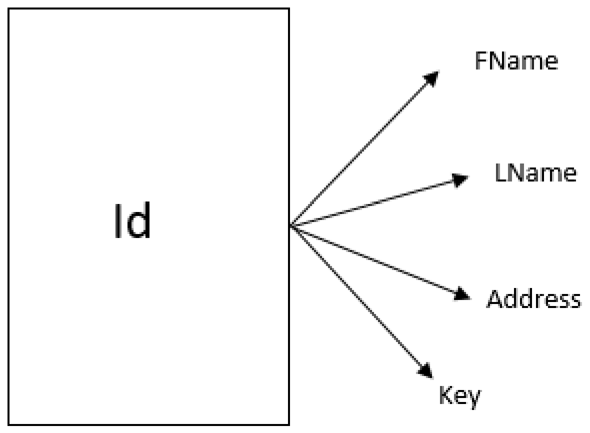
Billno: Primary Key

Report

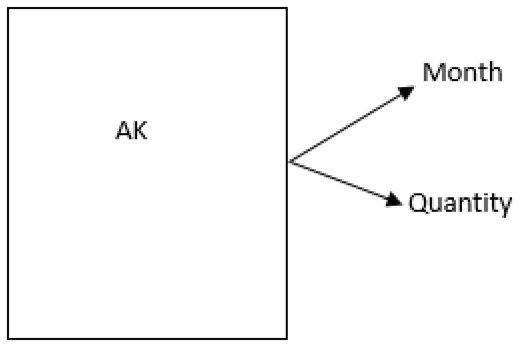
Ak	Number(5)
Quantity	Number(5)
Months	Number(4)

Ak: Primary Key

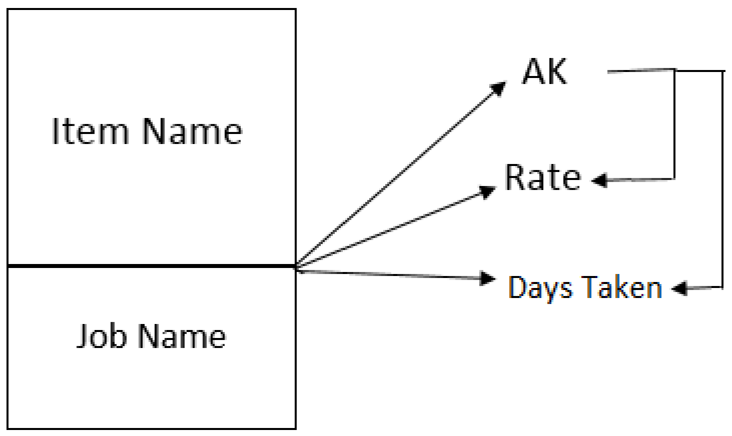
Functional Dependence Diagrams



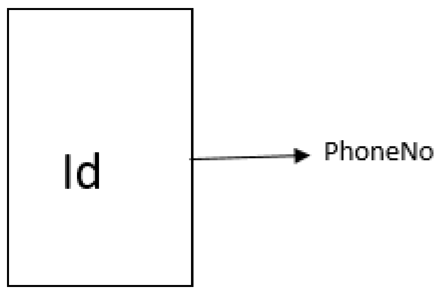
Customer



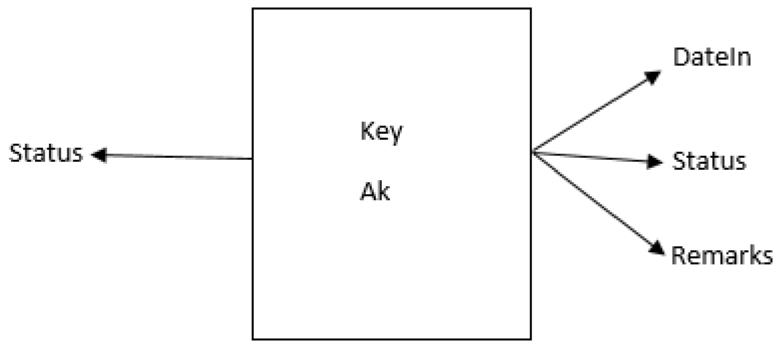
Report



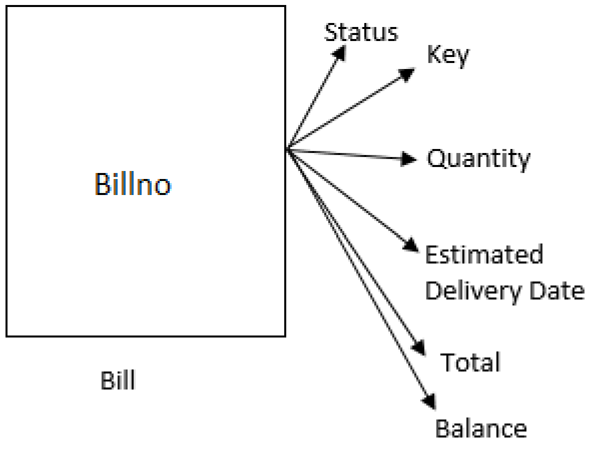
Clothes



Phone Number



Take In



Bill

Normalization

First Normal Form

A relation is said to be in First Normal Form (1NF) if and only if every entry of the relation (the intersection of a tuple and a column) has at most a single value. In other words "a relation is in First Normal Form if and only if all underlying domains contain atomic values or single value only."

Second Normal Form

A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully functional dependent on the primary key.

Third Normal Form

A relation R is in Third Normal Form (3NF) if and only if the following conditions are satisfied simultaneously:

(1) R is already in 2NF

(2) No nonprime attribute is transitively dependent on the key.

BCNF

A relation R is in Boyce/Codd N/F (BCNF) if and only if every determinant is a candidate key. Here determinant is a simple attribute or composite attribute on which some other attribute is fully functionally dependent.

Fourth Normal Form

A relation R is in Fourth Normal Form (4NF) if and only if the following conditions are satisfied simultaneously:

(1) R is already in 3NF or BCNF.

(2) If it contains no multi-valued dependencies

Fifth Normal Form

A relation R is in Fifth Normal Form (5NF) if and only if the following conditions are satisfied simultaneously:

(1) R is already in 4NF.

(2) It cannot be further non-loss decomposed.

1. **Customer Table**

- The table is in 1st Normal Form.
- The Table is in 2nd Normal Form.
- The Table is in 3rd Normal Form.
- The Table is in Boyce/Codd Normal Form.
- The Table is in 4th Normal Form.
- The Table is in 5th Normal Form.

2. **Phoneno Table**

- The table is in 1st Normal Form.
- The Table is in 2nd Normal Form.
- The Table is in 3rd Normal Form.
- The Table is in Boyce/Codd Normal Form.
- The Table is in 4th Normal Form.
- The Table is in 5th Normal Form.

3. **Clothes Table**

- The table is in 1st Normal Form.
- The Table is in 2nd Normal Form.
- The table is **not** 3rd Normal Form. So we break the tables in three tables as shown on next page.
- The Table is in 4th Normal Form.
- The Table is in 5th Normal Form.

4. **Takein**

- The table is in 1st Normal Form.
- The Table is in 2nd Normal Form.
- The table is in 3rd Normal Form.
- The Table is in 4th Normal Form.
- The Table is in 5th Normal Form.

5. **Bill**

- The table is in 1st Normal Form.
- The Table is in 2nd Normal Form.
- The table is in 3rd Normal Form.
- The Table is in 4th Normal Form.
- The Table is in 5th Normal Form.

6. **Report**

- The table is in 1st Normal Form.
- The Table is in 2nd Normal Form.
- The table is in 3rd Normal Form.

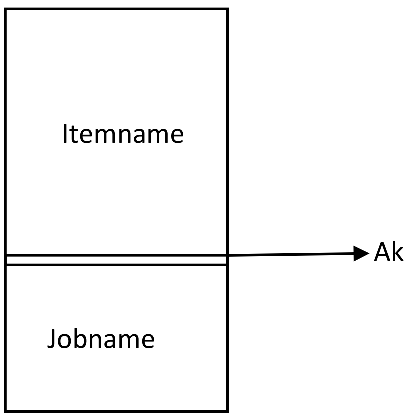
- The Table is in 4th Normal Form.
- The Table is in 5th Normal Form.

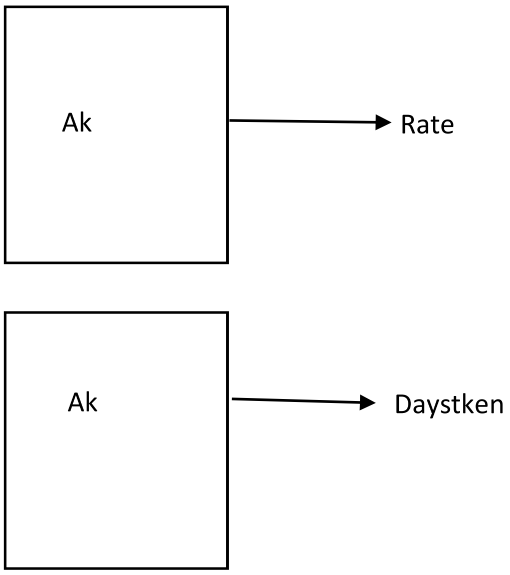
Updated Table and FD After Normalization

Ak	Number(5)
Item name	Varchar2
Job name	Varchar2

Ak	Number(5)
Rate	Number(5)

Ak	Number(5)
Daystaken	Number(4)





Creation of Tables

```
Create table customer_ns(

id varchar(10) primary key,

firstname varchar(20),
lastname varchar(20),
address varchar(40),

key varchar(10) not null

);


Create table phoneno(

id varchar2(10) references customer_ns(id),

phoneno number(12),primary key(id,phoneno));


Create table clothes_ns(

itemname varchar(10) not null ,

jobname varchar(15),

rate number(3),

ak number(6) primary key,

daystaken number(2)

);


Create table takein(

key varchar(10),

ak number(6),

quantity number(2),

datein date,

remark varchar(40),

status varchar(10) check (status in (‘taken’,jobdone’,’delivered’)),

primary key(key,ak)

);
```

```
Create table bill(
key number(6),
edd date,
billno number(7) primary key,
total number(4),
quantity number(2),
status varchar(10) check (status in (‘taken’, ‘pending’,’delivered’)
);

Create table report(
months varchar(3),
quantity number(2),
ak number(6) references clothes_ns(ak)
);

create sequence c_seq
increment by 1
start with 1
nocycle;

create sequence b_seq
increment by 1
start with 200
nocycle;



INSERT
Insert into table clothes_ns
values(‘Shirt/Tshirt’,’Wash’,10,c_seq.nextval,2);
Insert into table clothes_ns values(‘Trouser’,’Wash’,10,
c_seq.nextval,2);
Insert into table clothes_ns values(‘Jeans’,’Wash’,15, c_seq.nextval,2);
Insert into table clothes_ns values(‘Towel’,’Wash’,10, c_seq.nextval,2);
Insert into table clothes_ns values(‘Shoes’,’Wash’,80, c_seq.nextval,2);
Insert into table clothes_ns values(‘Jacket/Sweater’,’Wash’,30,
c_seq.nextval,2);

Insert into table clothes_ns values(‘Bedsheet’,’wash’,30,
c_seq.nextval,2);

Insert into table customer_ns(id,firstname,lastname,address)
values(‘101213048’,‘sarah’,’afrin’,’pg hostel’);

Insert into table customer_ns(id,firstname,lastname,address)
values(‘101203067’,‘nitesh’,’singh’,’j hostel’);

Insert into table customer_ns(id,firstname,lastname,address)
values(‘101203060’,‘naman’,’ahuja’,’j hostel’);

Insert into table customer_ns(id,firstname,lastname,address)
values(‘101203066’,‘nishant’,’gupta’,’j hostel’);

Insert into table customer_ns(id,firstname,lastname,address)
values(‘101’,‘varun’,’sharma’,’teacher quarters’);

Insert into phoneno values(‘101213048’,9876702523);

Insert into phoneno values(‘101213048’,9875348222);

Insert into phoneno values(‘101203067’,8288838038);

Insert into phoneno values(‘101203066’,9876754432);

Insert into takein
values(‘nsj’,2,3,to_date(‘dd/mm/year’,’4/12/2014’),null,’taken’);

Insert into takein
values(‘naj’,4,1,to_date(‘dd/mm/year’,’4/12/2014’),null,’jobdone’);

Insert into takein values(‘vsp’,5,1,
to_date(‘dd/mm/year,’4/12/2014’),null,’delivered’);
```

Procedure

Procedure to display information of user by id

```
Create or replace procedure find_userinfo_id(id1 in varchar)
is
user1 customer_ns%rowtype;
Begin
select * into user1 from customer_ns where id=id1;
dbms_output.put_line(user1.id ||" "||user.firstname||" " ||user1.lastname||" "||user.address||" " ||user1.key);
check_order_id(id1);

Exception
When TOO_MANY_ROWS then
dbms_output.put_line("Invalid");
When NO_DATA_FOUND then
dbms_output.put_line(id1||" is a Invalid Id");
When others then
dbms_output.put_line("Try again later");
end;
```

Procedure to display information of user by key

```
Create or replace procedure find_userinfo_key(key1 in varchar)
is
user1 customer_ns%rowtype;
Begin
select * into user1 from customer_ns where key=key1;
dbms_output.put_line(user1.id||" " ||user1.firstname||" "||user1.lastname||" " ||user1.address||" " ||user1.key);
check_order_key(key1);
Exception
When TOO_MANY_ROWS then
dbms_output.put_line("Invalid");
```

```
When NO_DATA_FOUND then
dbms_output.put_line(key1||" is a Invalid key");
When others then
dbms_output.put_line("Try again later");
end;
```

Procedure to display all the orders

```
create or replace procedure getallorders() is
cursor c1 is select * from takein where status<>'delivered' order by key,datein desc;
BEGIN
for r in c1 loop
 EXIT when c1%nodatafound;
 dbms_output.put_line(r.key||" "||r.ak||" "||r.quantity||" "||r.datein||" "||r.status||" "||r.remark);
end loop;
EXCEPTION
WHEN NO_DATA_FOUND then
dbms_output.put_line('There are no orders');
END;

create or replace procedure getreport_mnth(m1 in number(3)) is
cursor c1 is select * from report where  months=m1 ;
j clothes_ns%rowtype;
BEGIN
for r in c1 loop
 EXIT when c1%NODATAFOUND;
 select * into j from clothes_ns where ak=r.ak;
 dbms_output.put_line(r.ak||" "||r.quantity||" "||m1||" "||j.rate*quantity||" "j.itemname||" "||j.jobname);
end loop;
EXCEPTION
WHEN NO_DATA_FOUND then
dbms_output.put_line('There are no reports!!');
END;
```

Functions

Q. Function to check whether user is registered

```
Create or replace  function check_user(id1 in varchar) return number
is
a customer_ns%rowtype;
Begin
Select * into a from customer_ns where id=id1;
dbms_output.put_line('User already exists!!');
Return(1);
Exception
When NO_DATA_FOUND then
dbms_output.put_line('User does not exists!!');
Return(0);
When others then
dbms_output.put_line('Try again later');
Return(0);
End;
```

Q. Function to get user registered

```
Create or replace function create_user(id1 in varchar , firstname in varchar ,lastname in varchar , phoneno in number ,address in varchar  ) return number is
--Localkey  varchar(5);
Begin
---- Trigger for key
Insert into customer_ns(id,firstname,lastname,address) values(id1,firstname,lastname,address);
Insert into phoneno(id,phoneno) values(id1,phoneno);
dbms_output.put_line('Succesfully inserted!!');
Return(1);
Exception
When others then
dbms_output.put_line('Try again later');
```

```
Return(0);
End;
```

Q. Function to take order from customer_ns

```
CREATE OR REPLACE TYPE Items AS VARRAY(200) OF VARCHAR(50);
CREATE OR REPLACE TYPE quantities AS VARRAY(200) OF Number(5);
CREATE OR REPLACE TYPE jobs AS VARRAY(200) OF VARCHAR(50);
CREATE OR REPLACE TYPE rates AS VARRAY(200) OF Number(5);
CREATE OR REPLACE TYPE remarks AS VARRAY(200) OF VARCHAR(50);

Create or replace  function take_order(id in varchar2,item in items,quantity in quantities,job in jobs,remark in  remarks) return number is
localkey varchar(5);
ak number(5);
total number(4):=0;
edd date;
date1 date;
quantity1 number(5):=0;
rate number(5);
begin
Select daystaken into edd from clothes_ns where job=job(1);

For i in 1..id.count loop
Select key into localkey from customer_ns where id=id(i);
Select ak into ak from clothes_ns where jobname=job(i) and itemname=item(i);
Insert into takein(key,ak,quantity,remark,status)  values(localkey,ak,quantity(i),remark(i),'taken');
quantity1:=quantity1+quantity(i);
Select rate into rate from clothes_ns where ak=ak;
total:=total+rate*quantity(i);
Select daystaken into date1 from clothes_ns where jobname=job(i);
if date1>=edd then
edd:=date1;
end if;
end loop;

Insert into bill(key,quantity,edd,total,status) values(localkey, quantity1,edd,total,'taken');
dbms_output.put_line('Order Succesfully taken');
Return(1);
Exception
When NO_DATA_FOUND then
dbms_output.put_line('No data found ');
return(0);
When TOO_MANY_ROWS then
dbms_output.put_line('Try again later');
return(0);
When OTHERS then
dbms_output.put_line('try again later');
return(0);
End;
```

Q. Function to update status of clothes_ns of customer_nss

```
CREATE OR REPLACE TYPE aka AS VARRAY(200) OF number(5);
CREATE OR REPLACE TYPE key AS VARRAY(200) OF varchar(50);
CREATE OR REPLACE TYPE checka AS VARRAY(200) OF number(50);

Create or replace user function updatestatus(keyq in key , check1 in  checka, ak in aka,quantity in quantities)
return number is
--Localkey varchar(5);

Begin
for i in 1..id.count loop
if check1(i)=1 then
```

```
update takein set  status='jobdone' where key=keyq(i) and ak=ak(i);
else
 null;
end if;
end loop;
dbms_output.put_line('Status updated!!');
return(1);

Exception
When NO_DATA_FOUND then
dbms_output.put_line('No data found ');
When TOO_MANY_ROWS then
dbms_output.put_line('Try again later');
When others then
dbms_output.put_line('Try again late');
Return(0);
End;
```

Q. Function to check whether a user has given clothes_ns by id

```
Create or replace function check_order_id(id1 in varchar) return number
is
keyy varchar(10);
Begin
keyy:=find_key(id1);
Select * from takenin where key=keyy and status<>'delivered';
dbms_output.put_line(id1||''Has Given clothes_ns'');
Return(1);
Exception
When TOO_MANY_ROWS then
dbms_output.put_line(id1||''Has Given clothes_ns'');
```

```
Return(1);
When NO_DATA_FOUND then
dbms_output.put_line(id1||"Has not Given clothes_ns");
Return(0);c
When others then
dbms_output.put_line('Try again later');
end;
```

Q. Function to check whether a user has given clothes_ns by key

```
Create or replace function check_order_key(key1 in varchar) return number
is
idd varchar(10);
Begin
idd:=find_id(key1);
Select * from takenin where key=key1 and status<>'delivered';
dbms_output.put_line(key1||"Has Given clothes_ns"||" The customer_ns Id is "||idd);
Return(1);
Exception
When TOO_MANY_ROWS then
dbms_output.put_line(key1||"Has Given clothes_ns");
Return(1);
When NO_DATA_FOUND then
dbms_output.put_line(idd||"Has not Given clothes_ns");
Return(0);
When others then
dbms_output.put_line("Try again later");
end;
```

Q.Function to find id from key

```
Create or replace function find_id(key1 in varchar) return varchar
is
id1 customer_ns.id%type;
Begin
select id into id1 from customer_ns where key=key1;
dbms_output.put_line(key1||" Has id"||id1);
Return id1;
Exception
When TOO_MANY_ROWS then
dbms_output.put_line("Invalid");
Return null;
When NO_DATA_FOUND then
dbms_output.put_line(key1||" is a Invalid Key");
Return null;
When others then
dbms_output.put_line("Try again later");
return null;
end;
```

Q.Function to find key from id

```
Create or replace function find_key(id in varchar) return varchar
is
key1 customer_ns.key%type;
Begin
select key into key1 from customer_ns where id=id;
dbms_output.put_line(id||" Has key"||key1);
Return key1;
Exception
When TOO_MANY_ROWS then
dbms_output.put_line("Invalid");
Return null;
```

```
When NO_DATA_FOUND then
dbms_output.put_line(id||" is a Invalid id");
Return null;
When others then
dbms_output.put_line("Try again later");
return null;
end;
```

Q. Function to get all the bills by id

```
create or replace  function getbills(idd in varchar)   return number is
bill_t bill%rowtype;
Begin
Select * into bill_t from bill where id=idd ;
if(check_order_id(idd)=1) then
{
checkstatus_id(id);
dbms_output.put_line(bill_t.key||" "||bill_t.quantity||" "||bill_t.edd||" " ||bill_t.total||" "||bill_t.balance||" "||bill_t.status);
return (1);
}
else
dbms_output.put_line("");
endif;
Exception
When NO_DATA_FOUND then
Return (0);
When others then
dbms_output.put_line("Try again later");
Return (0);
End;
```

Cursors

Q. Cursor to check status of clothes_ns pending by passing id

```
create or replace  function checkstatus_id(id in varchar)   return number is
Cursor checkstatusid(keyy varchar) IS Select * from takein where key=keyy ;
jobname takein.jobname%type;
itemname takein.jobname%type;
Begin
if(check_order_id(id)=1)
{

foreach order1 in checkstatusid( find_key(id)) loop
        --if (order.status='jobdone') then
                select itemname,jobname into itemname,jobname from clothes_ns where ak=order1.ak;
                dbms_output.put_line(itemname||" "||jobname||" "||order1.status||" "||order1.datein);
        --endif;
end loop;
Return (1);
}
else
dbms_output.put_line("No clothes_ns pending");
end if;
Exception
When NO_DATA_FOUND then
Return (0);
When others then
dbms_output.put_line("Try again later");
Return (0);
End;
```

Q. Cursor to check status of clothes_ns pending by passing key

```
create or replace  function checkstatus_key(key1 in varchar)   return number is
Cursor checkstatuskey(keyy varchar) IS Select * from takein where key=keyy ;
jobname takenin.jobname%type;
itemname takenin.jobname%type;
```

```
Begin
if(check_order_key(key1)=1)
{
foreach order1 in checkstatuskey loop
--        if (order.status='jobdone') then
                    select itemname,jobname into itemname,jobname from clothes_ns where ak=order.ak;
                    dbms_output.put_line(itemname||" "||jobname||" "||order1.status||" "||order1.datein);
--endif;
end loop;
Return (1);
}
else
dbms_output.put_line("No clothes_ns pending");
endif;
Exception
When NO_DATA_FOUND then
Return (0);
When others then
dbms_output.put_line("Try again later");
Return (0);
End;
```

Q. Cursor to deliver clothes_ns and update status of clothes_ns which are delivered

```
create or replace function deliver(id in varchar)  return number is
Cursor delivering(key1 varchar) is Select * from takein where key=key1;
jobname takenin.jobname%type;
itemname takenin.jobname%type;
total number(10);
keyy varchar(10);
count1 number(10);
Begin
if(check_order_id(id)=1) then
count1:=0;
getbills(id);
keyy:=find_key(id);
```

```
foreach dv in delivering(keyy) loop
        if (dv.status='jobdone') then
                update takein set status ='delivered' where ak=dv.ak and key=keyy;
                Select rate into rate from clothes_ns where ak=dv.ak;
                total:=total+rate*dv.quantity;
        end if;
        count1:=count1+1;
end loop;
dbms_output.put_line(''The Total For clothes_ns ready is ''||total);
if(count1=(select count(ak) from takein group by key)) then
update bill set status='delivered' where key=keyy;
else if(count1>0)  then
update bill set status='pending' where key=keyy;
end if;
else
dbms_output.put_line(''No clothes_ns pending'');
end if;
Return (1);
Exception
When NO_DATA_FOUND then
Return (0);
When others then
dbms_output.put_line(''Try again later'');
Return (0);
End;
```

Q. Cursor to get all the orders.

```
create or replace procedure getallorders() is
cursor c1 is select * from takein where status<>'delivered' order by key,datein desc;
BEGIN
for r in c1 loop
 EXIT when c1%nodatafound;
 dbms_output.put_line(r.key||'' ''||r.ak||'' ''||r.quantity||'' ''||r.datein||'' ''||r.status||'' ''||r.remark);
end loop;
```

```
EXCEPTION

WHEN NO_DATA_FOUND then

dbms_output.put_line('There are no orders');

END;
```

Q. Cursor to get report month wise

```
create or replace procedure getreport_mnth(m1 in number(3)) is

cursor c1 is select * from report where  months=m1 ;

j clothes_ns%rowtype;

BEGIN

for r in c1 loop

 EXIT when c1%NODATAFOUND;

 select * into j from clothes_ns where ak=r.ak;

 dbms_output.put_line(r.ak||'' ''||r.quantity||'' ''||m1||'' ''||j.rate*quantity||'' ''j.itemname||'' ''||j.jobname);

end loop;

EXCEPTION

WHEN NO_DATA_FOUND then

dbms_output.put_line('There are no reports!!');

END;
```

Triggers

```
--Foreign key constrain of ID in phoneno
--We use trigger as id is not a Primary in the customer_ns
Create or replace trigger foreign_id_phone
Before insert or update on phoneno for each row
Declare
id customer_ns.id%type;
Begin
Select id into id from customer_ns where id=:new.id;
Exception
When no_data_found then
Raise_application_error(-20004, 'FOREIGN KEY VIOLATED
BECAUSE VALUE IS NOT FOUND IN THE PARENT TABLE');
end;
<><><><><><><><><><><><><>
--Foreign key constrain of key in takein
--We use trigger as id is not a Primary in the parent table
Create or replace trigger foreign_key_key
Before insert or update on takein for each row
```

```
Declare
key customer_ns.id%type;
Begin
Select key into key from customer_ns  where key=:new.key;
Null;
Exception;
When NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20004, 'FOREIGN KEY VIOLATED
BECAUSE VALUE IS NOT FOUND IN THE PARENT TABLE');
End;
```

<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

```
Create or replace trigger foreign_key_ak
Before insert or update on takein for each row
Declare
ak1 clothes_ns.ak%type;
Begin
Select ak into ak1 from clothes_ns where ak=:new.ak;

Exception
When NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20004, 'FOREIGN KEY VIOLATED
BECAUSE VALUE IS NOT FOUND IN THE PARENT TABLE');
end;
```

<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

```
--Foreign key constrain of key in takein
--We use trigger as id is not a Primary in the parent table


Create or replace trigger foreign_key_bill_key

Before insert or update on bill for each row

Declare

key customer_ns.id%type;

Begin

Select key into key from customer_ns  where key=:new.key;

Null;

Exception

When NO_DATA_FOUND THEN

RAISE_APPLICATION_ERROR(-20004, 'FOREIGN KEY VIOLATED

BECAUSE VALUE IS NOT FOUND IN THE PARENT TABLE');

End;
```

<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

Trigger to update date:

```
Create or replace trigger date_takein
After insert on customer_ns for each row
Declare
date1  takein.datein%type;
Begin
select sysdate into date1 from dual;
Update takein set datein=date1 where key=:new.key;
Exception
When NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20004, 'FOREIGN KEY VIOLATED
BECAUSE VALUE IS NOT FOUND IN THE PARENT TABLE');
End;
```