IIITD Furniture Store

Team Xypnix: Shiva (2017190), Sunny(2016270), Ashwani(2016232), Ayush(2017286), Vishal(2016278)

Goal: The goal of this project is to develop the back-end of an online store with extensive use of data entities selection and the relationship between them, data modeling, database access, and data manipulation.

Description: Our team plan to start an Online Furniture store that will provide a platform to seller and renters to register their Furniture products and buyers to browse the products. Admin adds or deletes the selling and renting stores after their request. Admin are us i.e. founders of the store.

The following are the stakeholders:

- Buyers
- Renters
- Sellers
- Shipping Company
- Admin

Workload:

Costumer:

- Search Furniture products by categories and price?
- Who are retailers available in the city?
- What are the Contact Details of the store ?.
- Check Order List.
- Which products are on offer.

Shipping Company:

- What are the retailer’s contact and address?
- What are the buyer’s contact and address?
- How many products are needed to be delivered?
- What is the total cost of the delivery?

- Details of the delivery person.
- Assign delivery person.

Seller:

- How many products left in stock?
- How many people have ordered the products?
- What are the customer's address and contact no. ?
- What is the total price?

Renter:

- How many products left in stock?
- How much rent is due?
- What duration is left for the rented product?
- What are the customer's address and contact no. ?
- How many people have ordered the products?

Admin:

- Accepting(or rejecting) the request of a person as seller, renters, and the shipping company and creating their database.
- Creating a buyer database(id, address, contact information).
- Adding(or deleting) another admin.
- Adding privileges of buyer, seller, renter, shipping company.
- Will have a track of sales.

Database Schema

Seller*(S_ID: int , S_name: String, S_contact: int, S_city: String, S_address: String, S_Zip Code: int, primary key(S_ID),);*

Renter*(R_ID: int, R_name: String, R_contact: int, R_city: String, R_address: String, R_Zip Code: int , primary key(S_ID))*;

Sale Product *(SP_ID: int, RP_name: String, SP_Category_ID: Int, S_ID: int, SP_Price:int , primary key (SP_ID) , foreign key(S_ID) refrences Seller, foreign key(Category_ID) refrences Category);*

Rent Product *(RP_ID: int, RP_name: String, RP_Category_ ID: Int, R_ID:int, RP_Price:int, primary key (RP_ID), foreign key(R_ID) refrences Renter, foreign key(Category_ID) refrences Category));*

Shipping Company *(Ship_ID: int; Del_Person_ID: int, Name: String, Contact: int, primary key(Ship_ID), foreign key(Del_Person_ID) refrences Delivery Person));*

Customer *(Cust_ID: Int, Cust_name: String, Cust_address: String, Cust_state: String, Cust_City: String, Cust_Zip:int, Primary Key(Cust_ID , primary key(Cust_ID));*

Delivery Person *(Order_ID; int, Del_Person_ID: int; DelPerson_name: String, Contact: int, primary key(Del_Person_ID), foreign key(Order_ID) refrences Sales Oder, Rent Orders) ;*

Category *(Category_ID: int, Category_name: String, primary key(Category_ID));*

Sale Orders *(Order_ID: int, SP_ID: int, Customer_ID: int, Shipper_ID: int, Price: int, Order_date: date, Shipping Date: date,primary key(Order_ID), foreign key(SP_ID) refrences Sale Product, foreign key(Shipper_ID) refrences Shipping Company));*

Rent Orders *(Order_ID: int, RP_ID: int, Customer_ID: int, Shipper_ID: int, Rent: int, Order_date: date, Shipping Date: date, ReturnDate: date, primary key(Order_ID) , foreign key(RP_ID) refrences Rent Product, foreign key(Shipper_ID) refrences Shipping Company));*

Admin *(Admin_ID: int, Request_ID: int, Admin_name: string, Admin_mail: String, primary key(Admin_ID) , foreign key(Request_ID) refrences Request);*

Request *(Request_ID: int, Person_name: String, Person_contact: String, Contact_Type: String,primary key(Request_ID));*

Offerpricesaleprod*(SP_ID int,Off int, Primary Key (SP_ID));*

Offerpricerentprod*(RP_ID int, Off int, Primary Key (SP_ID));*

Reach(*username varchar(100),email varchar(100),contact bigint,comments longtext*);

Queries

Indexes

We have already made crucial attributes as primary key and foreign key to save up the cost. From the rest of the attributes, we have made the following indexes.

Indexes for SP_name, RP_name (selling products and rent products name)

Since the cost for searching the selling products name and rent products name was high we created indexes on them to save the cost and memory. Using where clause was searching total 73 rows, but when an index was created on the query, searching become efficient only took the number of rows where our interest of result was. Like searching ‘arm chair’ only took 10 rows to search instead of 73 rows for our database.

EXPLAIN SELECT SP_ID, S_ID, SP_name from saleproducts where SP_name= 'arm chair';
CREATE INDEX SP_name ON saleproducts(SP_name);
show index from saleproducts;
EXPLAIN SELECT SP_ID, S_ID from saleproducts where SP_name= 'arm chair';

EXPLAIN SELECT RP_ID, R_ID, RP_name from rentproduct where RP_name= 'arm chair';
CREATE INDEX RP_name ON rentproduct(RP_name);
show index from rentproduct;
EXPLAIN SELECT RP_ID, R_ID, RP_name from rentproduct where RP_name= 'arm chair';

Indexes for seller or renter’s city.

While searching for the seller or renter’s city for the products, result was generated by searching through all rows, but when the index was created on the city, result only came up by searching only the city in which the customer is interested. For example, searching a seller in New Delhi took 13 rows with Where clause. While searching with the index it took only 3 rows for our database.

Explain SELECT R_ID, R_name from renter where R_city= 'New Delhi';
CREATE INDEX R_city ON renter(R_city);
show index from renter;
Explain SELECT R_ID, R_name from renter where R_city= 'New Delhi';

Explain SELECT S_ID, S_name from seller where S_city= 'New Delhi';
CREATE INDEX S_city ON seller(S_city);
show index from seller;

Explain SELECT S_ID, S_name from seller where S_city= 'New Delhi';

Indexes for prices

Since there are a lot of products in our database, searching their price especially between some range was slow. So, we created indexes for price on both sale products and rent products. As a result, the products whose selling prices were between 500 and 700 took only 11 rows for result which were taking 73 rows earlier.

```
Explain select Sp_ID, S_ID from saleproducts where SP_price between 500 and 700;
CREATE INDEX Price ON saleproducts( SP_Price);
show indexes from saleproducts;
Explain select Sp_ID, S_ID from saleproducts where  SP_price  between 500 and 700;
```

```
Explain select Rp_ID, R_ID from saleproducts where RP_price between 500 and 700;
CREATE INDEX Price ON rentproduct( RP_Price);
show indexes from rentproduct;
Explain select Rp_ID, R_ID from rentproduct where  RP_price  between 500 and 700;
```

Alter

Initially, we mentioned order dates in date time format but later we realized that we only need date. So, we made alteration to our type of order date attribute using alter command.

```
ALTER TABLE salesorders MODIFY order_date date;
ALTER TABLE salesorders MODIFY shipping_date date;

ALTER TABLE rentorders MODIFY order_date date;
ALTER TABLE rentorders MODIFY shipping_date date;
ALTER TABLE rentorders MODIFY return_date date;
```

Relational Queries

Find the product name, category and price by a seller name

select Sp_name,Category_Id, Sp_price from saleproducts natural join seller where seller.s_name="Galaxy Wooden Designer";

Find the product name, category and price by a renter name

```
select Rp_name,Category_Id, Rp_price from rentproduct natural join renter where
renter.r_name="Prince Furniture";
```

Find the seller who sells either two of three types of products and list the names of the products

```
select S1.S_Id, S1.S_name as Seller, P.SP_name as Product, S1.S_contact as Contact,
S1.S_city as city ,S1.S_address as Address ,S1.S_Zipcode as Zipcode from seller S1 natural
join saleproducts P where  P.category_Id= 3 or P.category_ID =4;

select S1.S_Id, S1.S_name as Seller, P.SP_name as Product, S1.S_contact as Contact,
S1.S_city as city ,S1.S_address as Address ,S1.S_Zipcode as Zipcode from seller S1 natural
join saleproducts P where  P.category_Id= 1 or P.category_ID =4;

select S1.S_Id, S1.S_name as Seller, P.SP_name as Product, S1.S_contact as Contact,
S1.S_city as city ,S1.S_address as Address ,S1.S_Zipcode as Zipcode from seller S1 natural
join saleproducts P where  P.category_Id= 1 or P.category_ID =3;
```

Find the renter who either two of three types of products on rent and list the names of the products

```
Select R1.R_Id, R1.R_name as Renter, P.RP_name as Product, R1.R_contact as Contact,
R1.R_city as city ,R1.R_address as Address ,R1.R_Zipcode as Zipcode  from renter R1 natural
join rentproduct P where  P.category_Id= 3 or P.category_ID =4;

Select R1.R_Id, R1.R_name as Renter, P.RP_name as Product, R1.R_contact as Contact,
R1.R_city as city ,R1.R_address as Address ,R1.R_Zipcode as Zipcode  from renter R1 natural
join rentproduct P where  P.category_Id= 1 or P.category_ID =4;

Select R1.R_Id, R1.R_name as Renter, P.RP_name as Product, R1.R_contact as Contact,
R1.R_city as city ,R1.R_address as Address ,R1.R_Zipcode as Zipcode  from renter R1 natural
join rentproduct P where  P.category_Id= 1 or P.category_ID =3;
```

Find the avg sale of a given month

Example- Month 2

```
select avg(quantity*saleprice(S.SP_ID)) as av from salesorders S where month(S.order_date)
between 2 and 3;
```

Find the rent of a given month

Example- Month 2

*select avg(quantity*rentprice(S.RP_ID)) as av from rentorders S where month(S.order_date) between 2 and 3;*

Find the avg sale of a day
Example - on 2019-04-03

*select avg(quantity*saleprice(S.SP_ID)) as averagesale from salesorders S where S.order_date ='2019-04-03';*

Find the avg rent of a day

*select avg(quantity*rentprice(S.RP_ID)) as averagerent from rentorders S where S.order_date='2019-04-03';*

Find the returning date of products along with rent to the renter having id =1

select order_id, rentproductname(R1.RP_ID) as Name, return_date from rentorders R1 where exists (select R_ID from rentproduct Rp where Rp.R_ID=1 AND R1.RP_ID = Rp.RP_ID);

Name of the Shipping company of order given by customer 1 of the order on 2019-03-03

select Sc.Comp_name , Sc.Contact from shippingcomp Sc where sc.ship_id = (select ship_id from salesorders s1 where cust_id=1 and order_date= '2019-05-03');

Delivery person who has received an order made by costumer 1 of on 2019-03-03

select Del_Id, DelPerson_name from deliveryperson p where p.ship_id =(select Sc.ship_id from shippingcomp Sc where sc.ship_id = (select ship_id from salesorders s1 where cust_id=1 and order_date= '2019-05-03'));

Procedures and functions call

History of customer's orders with given ID
call Saleorderinvoice(1); for customer with id =1
call rentorderinvoice (9); for customer with id=9

History of products sold by seller with given ID

call GetSellerHistory(2); id=2

History of products on rent by renter with given ID

call Getrenterhistory(15); id=15

Total price of a product by product id sold according to date

*select SP_ID, Order_Date, sum(quantity*saleprice(SP_ID)) over(partition by SP_ID order by order_date) as totalsale from salesorders where SP_ID=6;*

Total rent of a product by product id sold according to date

*select RP_ID, Order_Date, sum(quantity*rentprice(RP_ID)) over(partition by RP_ID order by order_date) as totalrent from rentorders where RP_ID= 15;*

Filter selling products and rent products by name

call filtersaleproducts("double bed");
call filterentproducts("double bed");

Bonus

Register a new seller or renter

We have considered a scenario where seller or renters send the requests to admin of the site to become a part of this business. Now, admin will look into request table. After consulting with other, if he wants to add the seller or renter he will just call the register procedure with input as request ID. The procedure will look about the details of a seller or renter and add to them to their respective database.

call register(1); - registers a seller
call register(2); - renter

Price when an offer is going on

Sales and offers are quite common on Ecommerce websites.We have that feature too. Whenever a sale month will come our store will automatically show the reduced price according to the offer. For example- product with ID=1 has 500 price on normal days, but if the offer is running on 4th Month, the price would be 60% less i.e. 200.

call offerpriceselling(15);

select offerpricesell(S.SP_ID) as price from salesorders S where S.order_date ='2019-02-03';
## this date has 2 products with ID 1 and 15
select saleprice(S.SP_ID) as price from salesorders S where S.order_date ='2019-02-03'; ##

Given below are the PL/SQL Functions and Procedures which we’ve used.

Procedures

CREATE DEFINER=`root`@`localhost` PROCEDURE `**filterentproducts**`(IN product varchar(40))
BEGIN
select RP_Id, R_ID, RP_name,Category_ID,RP_Price from saleproducts where RP_name= product;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `**filtersaleproducts**`(IN product varchar(40))
BEGIN
select SP_Id, S_ID, SP_name,Category_ID,SP_Price from saleproducts where SP_name= product;

END
CREATE DEFINER=`root`@`localhost` PROCEDURE **`Getrenterhistory`**(In ID integer)
BEGIN

select Order_ID,rentorders.RP_ID as RP_ID ,rentproduct.Rp_name, rentorders.quantity as RP_quantity, rp_price*quantity as total_price, Cust_ID, Ship_id, Order_date ,Shipping_date, return_date from rentproduct,rentorders where rentproduct.R_ID= ID and rentproduct.RP_ID= rentorders.RP_ID;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE **`GetSellerhistory`**(In ID integer)
BEGIN

```
 select Order_ID,salesorders.SP_ID as SP_ID ,saleproducts.Sp_name, salesorders.quantity as
SP_quantity, sp_price*quantity as total_price, Cust_ID, Ship_id, Order_date ,Shipping_date
from saleproducts,salesorders where saleproducts.S_ID= ID and saleproducts.SP_ID=
salesorders.SP_ID;


END

CREATE DEFINER=`root`@`localhost` PROCEDURE `register`(IN ID integer)
BEGIN
declare nam varchar(50);
declare num bigint default 0;
declare typ varchar(50);
declare cit varchar (50);
declare ad varchar (200);
declare zip bigint;
declare s int default 0;
declare r int default 0;
select Request_type, Store_name , Person_contact,city,address,zipcode  into
typ,nam,num,cit,ad,zip from request where request.Request_Id=ID;
set s= get_id_seller(); -- get last id
set r = get_id_renter();-- get last id
if typ="New Seller"
        then insert into seller (S_ID,S_name,S_contact,S_city,S_address, S_zipcode) values (
s+1,nam, num, cit,ad, zip);
end if;
 if typ="New Renter"
        then insert into renter (R_ID, R_name,R_contact,R_city,R_address, R_zipcode) values
(r+1, nam, num,  cit,ad, zip);
end if;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `rentorderinvoice`(IN ID integer)
BEGIN
select Order_ID, Ship_ID , Cust_ID ,
sum(rentprice(RP_ID)*S1.quantity) as Total,
Order_date, Shipping_date, return_Date
from rentorders s1
where Cust_ID=ID;
END
CREATE DEFINER=`root`@`localhost` PROCEDURE `Saleorderinvoice`(IN ID integer)
BEGIN
```

```
select Order_ID, Ship_ID , Cust_ID ,
sum(saleprice(SP_ID)*S1.quantity) as Total,
Order_date, Shipping_date
from salesorders s1
where Cust_ID=ID
group by Order_ID;

END
```

Functions

```
CREATE DEFINER=`root`@`localhost` FUNCTION `get_id_renter`() RETURNS int(11)
   READS SQL DATA
   DETERMINISTIC
BEGIN
declare q int default 0;
SELECT R_ID into q FROM renter order BY R_id DESC LIMIT 1;
RETURN q;

END
CREATE DEFINER=`root`@`localhost` FUNCTION `get_id_seller`() RETURNS int(11)
   READS SQL DATA
   DETERMINISTIC
BEGIN
declare q int default 0;
SELECT S_ID into q FROM seller order BY S_id DESC LIMIT 1;
RETURN q;

END


END
CREATE DEFINER=`root`@`localhost` FUNCTION `saleprice`(ID Int) RETURNS int(11)
   READS SQL DATA
   DETERMINISTIC
BEGIN
DECLARE q int default 0;
select SP_Price into q
from SaleProducts
where SP_ID =ID;
RETURN q;
END
```

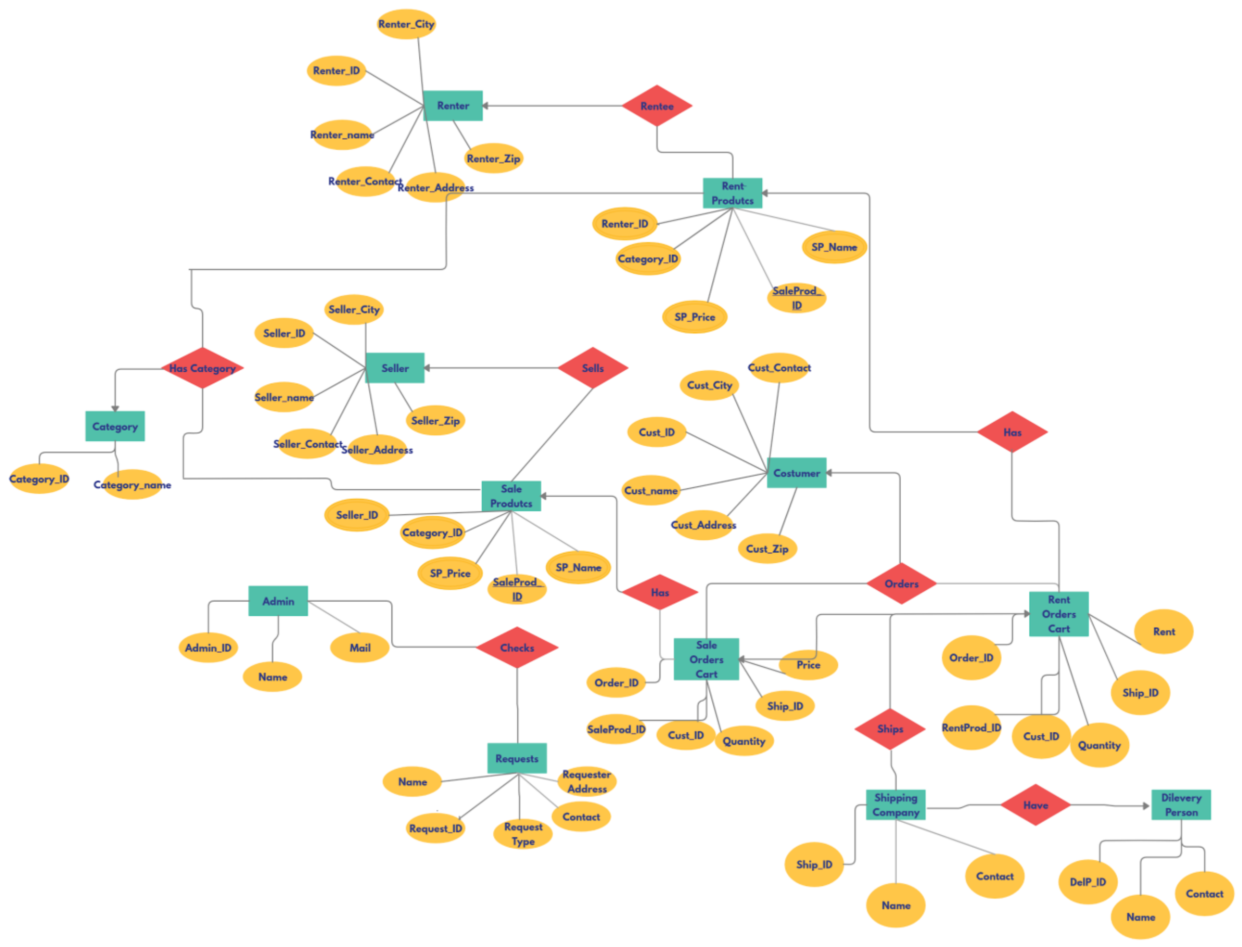
```
CREATE DEFINER=`root`@`localhost` FUNCTION `rentprice`(ID Int) RETURNS int(11)
    READS SQL DATA
    DETERMINISTIC
BEGIN
DECLARE q int default 0;
select RP_Price into q
from rentproduct
where RP_ID =ID;
RETURN q;
END

CREATE DEFINER=`root`@`localhost` FUNCTION `offerpricesell`(ID Int) RETURNS int(11)
    READS SQL DATA
    DETERMINISTIC
BEGIN
DECLARE q int default 0;
##DECLARE f int default 0;
if month(curdate())= 04
then select (S.Sp_price - S.Sp_price*O.off*0.01) into q from saleproducts S, offerpricesaleprod
O where S.SP_ID=O.SP_ID AND S.SP_ID= ID ;
end if;
if  month(curdate())<> 04
then select (S.Sp_price) into q from saleproducts S where S.SP_ID= ID ;
end if;

RETURN q;

END
```

E-R Diagram Of The Store


Renter_City
Renter_ID
Renter
Rentee
Renter_name
Renter_Zip
Renter_Contact
Renter_Address
Rent Produtcs
Renter_ID
Category_ID
SP_Name
SaleProd_ID
SP_Price
Seller_City
Seller_ID
Seller
Sells
Has Category
Seller_name
Seller_Zip
Seller_Contact
Seller_Address
Category
Category_ID
Category_name
Cust_Contact
Cust_City
Cust_ID
Has
Costumer
Cust_name
Cust_Address
Cust_Zip
Sale Produtcs
Seller_ID
Category_ID
SP_Name
SP_Price
SaleProd_ID
Admin
Admin_ID
Name
Mail
Has
Orders
Checks
Rent Orders Cart
Rent
Order_ID
Ship_ID
Sale Orders Cart
Price
Order_ID
Ship_ID
SaleProd_ID
Cust_ID
Quantity
Ships
RentProd_ID
Cust_ID
Quantity
Requests
Name
Requester Address
Request_ID
Request Type
Contact
Shipping Company
Have
Dilevery Person
Ship_ID
Contact
Name
DelP_ID
Contact
Name

Thank You..