به نام خدا

گزارش پروژه پایانی پایگاه داده

عنوان پروژه: فروشگاه آنلاین

اعضای گروه:

تینا صداقت ۹۳۳۱۰۴۴

زهرا اطهاری نیکوروان ۹۳۳۱۰۲۷

فاز اول

سامانه فروشگاه:

مواردی که بین همه کالاها مشترک است، در جدول Utility آمده‌است. این جداول با توجه به مفهوم inheritance پیاده‌سازی شده اند که جدول Utility به عنوان parent در نظر گرفته شده است:

Utility (UtilityName, UtilityCode, UtilityCompany, UtilityCost, UtilitySale, UtilityNExist, UtilityGroup)

توجه داریم که primary keyها با یک خط زیر آن‌ها، و foreign keyها با رنگ قرمز نشان داده شده‌اند.

سه جدول HomeUtility و DigitalUtility و SportUtility که زیرگروه های جدول Utility هستند، در ادامه مشخص شده اند:

HomeUtility (UtilityName, UtilityCode, UtilityCompany, UtilityCost, UtilitySale, UtilityNExist, UtilityGroup, produceDate)
DigitalUtility (UtilityName, UtilityCode, UtilityCompany, UtilityCost, UtilitySale, UtilityNExist, UtilityGroup)
SportUtility (UtilityName, UtilityCode, UtilityCompany, UtilityCost, UtilitySale, UtilityNExist, UtilityGroup, UtilityColor)

در جدول Utility، صفتی وجود ندارد که به کمک آن بتوانیم یک صفت دیگر را به طور یکتا مشخص کنیم.

در واقع، وابستگی‌های تابعی ای که وجود دارد به فرم زیر است:

UtilityCode → UtilityName

UtilityCode → …….

که می‌توان از این وابستگی‌ها صرف نظر کرد؛ زیرا UtilityCode سوپرکلید است.

وابستگی‌های تابعی دیگری نیز وجود ندارد. بنابراین دارای فرم نرمال BCNF است.

مشتریان:

یکی از مفاهیمی که در این پروژه مطرح است، بحث عوامل انسانی است که در نقش‌های مختلفی نظیر مشتری، عوامل پشتیبانی، کارمند و راننده وسایل نقلیه ظاهر می‌شود. بنابراین، می‌توان یک مفهوم کلی به نام person معرفی کرد و نقش‌های مختلف را از آن به ارث برد:

Person (PersonID, PersonName, PersonFamilyName)

مشتریان می‌تواند بدون اشتراک یا دارای اشتراک باشند. مشتریانی که اشتراک دارند، می‌بایست یک username منحصر به فرد داشته باشند؛ علاوه بر این، یک اعتبار هم دارند. توجه داریم که این جدول نیز از جدول person، باید به ارث برده شود:

RegisteredCostumer (CostumerID, PersonName, PersonFamilyName, CostumerUsername, CostumerCredit, RegisteredDate)

UnRegisteredCostumer (CostumerID, PersonName, PersonFamilyName)

در جدول RegisteredCostumer، وابستگی تابعی زیر وجود دارد:

CostumerUsername → CostumerID

این وابستگی باید رفع شود. بنابریان این جدول را به دو جدول زیر تجزیه می‌کنیم:

RegisteredCostumer (CostumerID, PersonName, PersonFamilyName, CostumerCredit, RegisteredDate)

RegisteredCostumerUsernames (CostumerID, CostumerUsername)

هر مشتری، حداقل یک شماره تلفن و آدرس دارد که برای هر کدام از آن‌ها، یک جدول مجزا در نظر می‌گیریم:

Address (City, Street, Pelak)

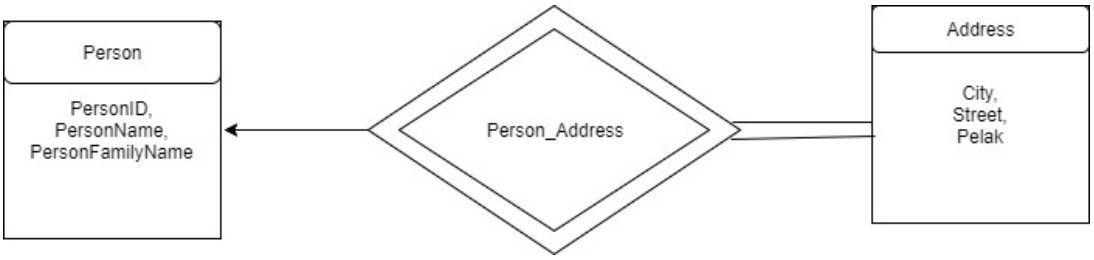
Phone (phoneNumber)

نکته قابل توجه این است که این دو جدول، برای ذخیره آدرس یا شماره تلفن هر فردی استفاده می‌شوند.

این دو از نوع weak entity هستند، زیرا primary key ندارند و discriminator در جدول Address، هر سه صفت و در جدول phone، همان phoneNumber است.

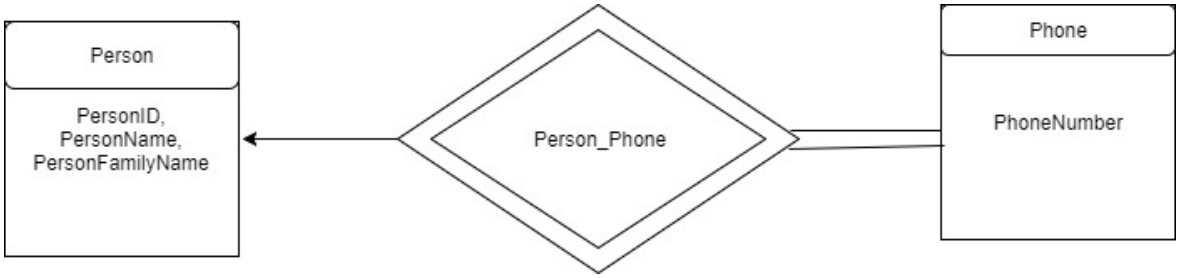
حال باید یک relationship بین جدول Address و جدول Person برقرار کنیم که primary key آن، برابر است با PersonId و City و Street و Pelak.

Person_Address (PersonID, City, Street, Pelak)



همین مسئله برای جدول Phone نیز انجام می‌دهیم. در واقع یک relationship بین جدول Phone و جدول Person برقرار کنیم که primary key آن، برابر است با PersonId و PhoneNumber.

Person_phone (PersonID, PhoneNumber)



جدول سفارش‌های مشتریان به صورت زیر است:

Order (OrderID, CostumerID, UtilityCode)

در این جدول وابستگی تابعی نداریم. بنابراین دارای فرم نرمال BCNF است.

جدول خرید به صورت زیر است. به طور خلاصه می‌توان گفت که این جدول، بیان می‌کند که هر مشتری، چه سبدی را در چه تاریخی خریداری کرده است. همچنین وضعیت پرداخت، وضعیت سبد خریداری شده و آدرس نیز مشخص است. میزان قیمت و مالیات پرداختی نیز مشخص است:

Bought (BoughtID, CostumerID, UtilityTotlalCost, UtilityTotlalTax, BoughtDateTime, PayStatus, CostumerCity, CostumerStreet, CostumerPelak, BoughtStatus)

BoughtStatus بیان می‌کند که سبد خریداری شده در کدام مرحله از فرایند ارسال می‌باشد (قبل از ارسال، در حال ارسال، تحویل داده شده)

در این جدول وابستگی تابعی نداریم. بنابراین دارای فرم نرمال BCNF است.

هر سبد خریداری شده، با یک ID منحصر به فرد شناخته می‌شود. برای این که بدانیم در هر خرید چه اقلامی خریداری شده است، باید از جدول زیر کمک بگیریم:

BoughtDetail (BoughtID, UtilityID, NumberOfUtility)

در این جدول وابستگی تابعی نداریم. بنابراین دارای فرم نرمال BCNF است.

برای ذخیره امتیازها و نظرات محصول، از جدول زیر استفاده می‌کنیم:

Opinion (OpinionID, CostumerUsername, UtilityCode, Comment, Rank)

با این فرض که هر کاربر برای یک محصول تنها یک بار اجازه دارد که rank وارد کند، وابستگی تابعی زیر در این جدول وجود دارد:

CostumerUsername, UtilityCode→ Rank

بنابراین باید این جدول را به صورت زیر بازنویسی می‌کنیم:

Opinion (CostumerUsername, UtilityCode, Comment, Rank)

در این حالت، primary key، ترکیب CostumerUsername, UtilityCode می شود. لذا وابستگی تابعی-ای که در بالا ذکر کردیم، مشکل‌ساز نخواهد شد.

عوامل پشتیبانی:

جدول عوامل پشتیبانی هم از جدول Person به ارث می برند و به صورت زیر است:

Suporters (SuporterID, PersonName, PersonFamilyName, SupproterStatus)

در این جدول وابستگی تابعی نداریم. بنابراین دارای فرم نرمال BCNF است.

یک جدول برای رسیدگی به شکایات، دنبال کردن سفارش‌ها و پاسخگویی آنلاین طراحی کردیم که با توجه به TrackingKind معلوم می شود که کدام یک از این عملیات انجام می شود و به صورت زیر است:

Tracking(TrackingID, TrackingKind, CostumerID, SupporterID, BoughtID, TrackingText, TrackingAnswer, TrackingTime)

در این جدول وابستگی تابعی نداریم. بنابراین دارای فرم نرمال BCNF است.

شرکت‌ها:

در رابطه با شرکت ها، چند موضوع مطرح است: اول این که خود شرکت‌ها برای ثبت در سیستم نیاز به یک جدول دارند:

Company (CompanyID, CompanyName, RegisteredDate)

در این جدول وابستگی تابعی نداریم. بنابراین دارای فرم نرمال BCNF است.

دوم این که هر کدام از شرکت‌ها، فرد یا افرادی را به عنوان مسئول ارتباط با فروشگاه معرفی می‌کنند که باید اطلاعات این افراد را نیز در جداول زیر ذخیره کنیم. توجه داریم که مسئول ارتباط نیز از Person به ارث برده می‌شود:

MrLink (PersonID, PersonName, PersonFamilyName)

در این جدول وابستگی تابعی نداریم. بنابراین دارای فرم نرمال BCNF است.

برای فرد مسئول ارتباط، می‌بایست آدرس و شماره تلفن را هم ذخیره کرد که در همان جدول مربوط به تلفن که قبلا ساختیم، ذخیره می‌کنیم:

Phone (phoneNumber)

در ادامه به یک جدول نیاز داریم که مشخص کند مسئول ارتباط هر شرکت کیست. توجه داریم که هر مسئول رابط، تنها برای یک شرکت فعالیت می‌کند.:

Company_Link (CompanyID, LinkID)

در این جدول وابستگی تابعی نداریم. بنابراین دارای فرم نرمال BCNF است.

برای ذخیره شماره تلفن‌ها و آدرس‌های شرکت، از جدول‌های زیر استفاده می‌کنیم:

Company_Address (CompanyID, CompanyCity, CompanyStreet, CompanyPelak)

Company_Phone (CompanyID, CompanyPhone)

در این جدول‌ها وابستگی تابعی نداریم. بنابراین دارای فرم نرمال BCNF هستند.

مسئله دیگری که مطرح است، ذخیره اطلاعات کارمندان یک شرکت است:

Employee (CompanyID, EmployeeID, EmployeeName, EmployeeFamilyName)

فرض بر این است که یک کارمند نمی‌تواند عضو دو شرکت باشد؛ بنابراین کلید اصلی در این جدول EmployeeID می‌شود.

در این جدول وابستگی تابعی نداریم. بنابراین دارای فرم نرمال BCNF است.

برای تایید این که یکی از مشتریان فروشگاه، از کارمندان شرکت هست یا نه، می‌بایست فیلدهای جدول بالا با فیلدهایی که مشتری وارد کرده است مقایسه شود. در صورت یکسان بودن، فرد می‌تواند از تخفیفات مورد نظر استفاده کند.

حمل و نقل:

اطلاعات راننده‌ها در جدول زیر ذخیره می‌شود:

TransportMan (TransportManID, TransportManName, TransportManFamilyName, TransportManPelak, TransportManCar, TransportManShenasname, TransportManStatus)

در این جدول، چند وابستگی تابعی وجود دارد:

TransportManPelak→ TransportManCar

فرض را بر این می‌گذاریم که هر راننده تنها یک ماشین دارد. بنابراین TransportManPelak نیز یکتا می‌شود و می‌تواند به عنوان primary key استفاده شود. در نتیجه جدول بالا را به صورت زیر بازنویسی می‌کنیم:

TransportMan (TransportManName, TransportManFamilyName, TransportManPelak, TransportManCar, TransportManShenasname, TransportManStatus)

در این جدول نیز همچنان وابستگی تابعی زیر وجود دارد:

TransportManShenasname→ TransportManName, TransportManFamilyName

بنابراین باید جدول را تجزیه کنیم:

TransportMan (TransportManPelak, TransportManCar, TransportManShenasname, TransportManStatus)

TransportManIndividual (TransportManName, TransportManFamilyName, TransportManShenasname)

با این که وابستگی تابعی بالا در جدول دوم همچنان وجود دارد، اما چون TransportManShenasname دراین جدول سوپرکلید است، مشکلی به وجود نخواهد آمد.

برای ذخیره آدرس و شماره تلفن راننده‌ها، از جداول اصلی مربوط به آدرس و شماره تلفن، استفاده می‌کنیم:

Address (City, Street, Pelak)

Phone (phoneNumber)

جدول زیر تعیین می‌کند که هر راننده، چه سبدی را برده است و کی تحویل داده‌است:

TransportDetail (TransportManPelak, BoughtID, RecievedDate)

در این جدول وابستگی تابعی نداریم. بنابراین دارای فرم نرمال BCNF است.

فاز دوم

برای نشان دادن log، نیاز به پیاده سازی trigger داریم. کد زیر یک نمونه از trigger را نشان می‌دهد که برای update شدن جدول employee استفاده شده است:

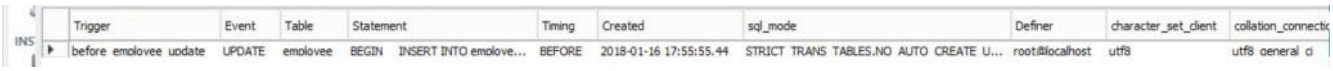
```
CREATE TABLE employees_audit (
EmployeeID INT primary key,
  CompanyID VARCHAR(45) references Company(CompanyID),
   action VARCHAR(50) DEFAULT NULL
);
DELIMITER $$
CREATE TRIGGER before_employee_update
   BEFORE UPDATE ON employee
   FOR EACH ROW
BEGIN
   INSERT INTO employees_audit
   SET action = 'update',
  employeeID = OLD.employeeID,
  CompanyID = OLD.CompanyID;
      — changedat = NOW();
END$$
DELIMITER ;

UPDATE employee
SET
   companyID = 'Phan'
WHERE
```

```
  employeeID = 2001;
```

در این کد، هنگامیکه employeeID = 2001 است، مقدار companiID را به صورت companyID = 'Phan'، update می‌کنیم که در این صورت اگر جدول employees_audit را مشاهده کنیم، می‌بینیم که مقدار employeeID = 2001 ثبت شده است:

```
SHOW TRIGGERS;
SELECT
  *
FROM
  employees_audit;
```



	Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client	collation_connectic
INS	before_employee_update	UPDATE	employee	BEGIN INSERT INTO employe...	BEFORE	2018-01-16 17:55:55.44	STRICT_TRANS_TABLES,NO_AUTO_CREATE_U...	root@localhost	utf8	utf8_general_ci

برای نمایش دادن لاگ به کمک تریگرها از دستور SHOW TRIGGERS; استفاده می کنیم. این دستور همه ی عملیات انجام شده (update,insert,delete) را به همراه زمان ایجاد تغییر نمایش می دهد. برای نمایش دادن زمان آخرین ویرایش در تریگر، در جدول هایی که به عنوان history نگه داشته ایم صفتی به نام changedat اضافه می کنیم که با استفاده از تابع now(); در بخش تریگر، زمان آخرین ویرایش را نشان می دهد.

برای هر جدولی که بخواهیم اطلاعاتش بعد از delete کردن، ذخیره شود، از جدولی به عنوان history استفاده می کنیم که در زمان تریگر روی delete، در این جدول اطلاعات لازم را insert می کنیم. به عنوان مثال:

```
CREATE TABLE RegisteredCostumerUsernames(
 CostumerID INT primary key,
 CostumerUsername VARCHAR(45) );
 create table history_of_deleted_RegisteredCostumerUsernames(
 CostumerID INT primary key,
 CostumerUsername VARCHAR(45),
 action VARCHAR(50) DEFAULT NULL,
```

```
  changedat DATETIME DEFAULT NULL  );


  DELIMITER $$
CREATE TRIGGER before_deleting_RegisteredCostumerUsernames
    BEFORE DELETE ON RegisteredCostumerUsernames
    FOR EACH ROW
BEGIN
    INSERT INTO history_of_deleted_RegisteredCostumerUsernames
    SET action = 'delete',
          CostumerID = OLD.CostumerID,
          CostumerUsername = OLD.CostumerUsername,
          changedat = NOW();
END$$
DELIMITER ;
```

برای انجام فاز دوم، در برخی از قسمت‌ها نیاز به استفاده از view داشتیم که در ادامه به توضیح چگونگی استفاده از view می‌پردازیم.

در صورت پروژه آمده است که سیستم به ازای هر ۱۰۰ هزار تومانی که مشتریان ثبت‌شده، اعتبار خود را افزایش می‌دهند، به آنها ۱۰هزار تومان هدیه می‌دهد. برای پیاده‌سازی این قسمت از view استفاده شده‌است. به این صورت که در یک جدول عادی، میزانی که هر کاربر اعتبار خود را افزایش می‌دهد ثبت می‌شود. یک view می‌سازیم که بررسی می کند آیا میزان شارژ کردن حساب به ۱۰۰هزار تومان رسیده است یا خیر. بنابراین CostumerCredit را بر ۱۰۰ تقسیم می‌کنیم. سپس مقدار به دست آمده را در ۱۰ ضرب می‌کنیم و با CostumerCredit قبلی جمع می‌کنیم.

```
CREATE TABLE RegisteredCostumer(
CostumerID INT primary key references Person(PersonID),
CostumerCredit INT NULL,
RegisteredDate DATETIME NULL);


create view creditWithGift(CostumerID,CostumerCredit) as
select CostumerID,10 * (RegisteredCostumer.CostumerCredit / 100) +
RegisteredCostumer.CostumerCredit
from RegisteredCostumer;
```

یکی دیگر از قسمت‌هایی که از view استفاده شده‌است، قسمت ۵,۱ است:

۵,۱. تعداد و قیمت تمام کالاهایی که در هر دسته‌بندی تا به حال فروخته شده‌اند به همراه نام دسته و شناسه دسته.) نیازی به ذکر هر مورد نیست تنها در هر دسته چه تعداد کالا و قیمت کل فروش هر دسته را به همراه نام دسته و شناسه آن مورد نیاز می‌باشد).

برای این قسمت یک view به صورت زیر درست کرده‌ایم که مجموع تعداد فروخته شده از هر کالا را به دست آورد:

```
create view cost(utilityID, totalnumber) as
    select utilityId, sum(numberofutility)
```

```
    from boughtdetail
    group by utilityId;
```

حال برای پاسخگویی به پرسش بالا، تنها لازم است مجموع تعداد هر کالا را در قیمت واحد آن ضرب کرده و در نهایت میزان کل کالاهای فروخته شده در هر دسته‌بندی را با group by کردن به دست آوریم:

```
select cost.totalnumber,sum(cost.totalnumber*utility.utilitycost*(1-utilitySale)) as
totalcost ,utility.utilitygroup
    from cost,utility
    where cost.utilityID=utility.utilityCode

   group by utility.utilitygroup
```

قسمت دیگری که از view استفاده شده‌است، قسمت ۵,۱۲ است که باید درآمد فروشگاه را در ۲ ماه اخیر به دست آوریم. یک view به صورت زیر می‌سازیم که تاریخ فروش هر کالا را داشته باشد:

```
create view cost2(utilityID, numberofutility,BoughtDateTime) as
    select utilityId, numberofutility , Bought.BoughtDateTime
    from boughtdetail natural join bought;
```

حال برای به دست آوردن درآمد فروشگاه در ۲ ماه اخیر، از query زیر استفاده می‌کنیم:

```
select sum(cost2.numberofutility*utility.utilitycost) as totalcost
    from cost2,utility
    where cost2.utilityID=utility.utilityCode and 08 <
month(cost2.BoughtDateTime) < 10 and year(cost2.BoughtDateTime)=2017
```

برای قسمت ۵,۱۷ نیز نیاز به استفاده از view داریم:
۵,۱۷. مجموع خرید کارمندان شرکت ها در ماه جاری.

```
create view cost3(employeeID ,utilityID, numberofutility,BoughtDateTime,
companyName) as
```

```
    select Employee.EmployeeID ,utilityId, numberofutility ,
Bought.BoughtDateTime, company.Companyname
    from boughtdetail, bought, Employee, company
    where bought.boughtId = boughtDetail.boughtID and Employee.EmployeeID =
bought.costumerID and employee.CompanyID=company.CompanyID ;
```

```
    create view q17(employeeID ,totalcost , companyname) as
    select cost3.employeeID, sum(cost3.numberofutility*utility.utilitycost) as
totalcostpercustomer , cost3.companyname
        from cost3, utility
        where cost3.utilityID=utility.utilityCode and
month(cost3.BoughtDateTime) = 10 and year(cost3.BoughtDateTime)=2017
        group by cost3.employeeID;
```

برای جداسازی و درک بهتر از جداول، از چند view استفاده شده‌است. در cost3 اطلاعات خرید کارمندان شرکت ذخیره می‌شود. سپس در q17 مجموع خرید هر یک از کارمندان شرکت‌ها محاسبه و ذخیره می‌گردد. یکی از نکاتی که در تعریف پروژه آمده بود، این است که در هر صورت حسابی که کارمندان شرکت پرداخت می‌کنند ده درصد تخفیف برایشان تا سقف ۵۰۰ هزار تومان لحاظ می‌گردد. منظور از این که تخفیف ۱۰ درصدی تا سقف ۵۰۰هزار تومان باشد، این است که تا وقتی مجموع به ۵۰۰۰۰۰۰ نرسیده، ۱۰ درصد از مبلغ کل کاسته شود. اما هنگامی که مجموع خرید از ۵۰۰۰۰۰۰ بیشتر شد، تنها ۵۰۰۰۰۰ تومان از مبلغ کسر شود.
برای هندل کردن این نکته، یک view با عنوان employeewithdiscount ساخته شده است که از دستور select if استفاده می‌کند.

```
    create view employeewithdiscount(employeeID, totalcost, companyname) as
    select q17.employeeID, if(totalcost<5000000,0.9*totalcost,totalcost-500000),
q17.companyname
        from q17;
```

حال به راحتی می‌توان با استفاده از query زیر قسمت ۵,۱۷ را جواب داد:

```
select companyname,sum(totalcost) as totalcost
from employeewithdiscount
group by companyname
```