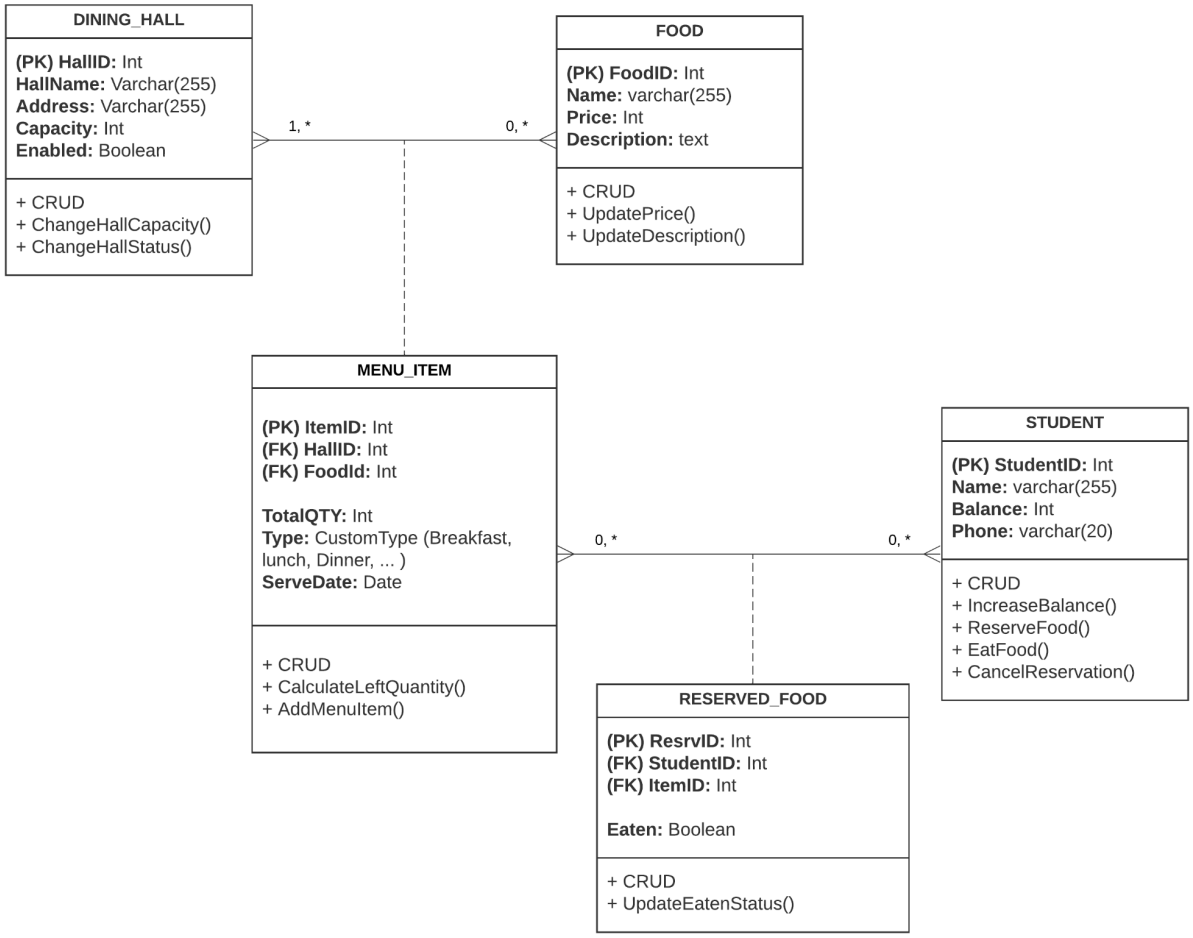
پروژه درس آزمایشگاه پایگاه داده

تحلیل و پیاده‌سازی پایگاه داده سلف دانشگاه

امیرمحمد فلاح - ۹۷۰۱۲۲۶۸۰۰۱۴

در شکل زیر طراحی Class Diagram سامانه سلف دانشگاه آورده شده است. در ادامه به بررسی هر موجودیت، روابط بین آن ها و صفت هایش می پردازیم.



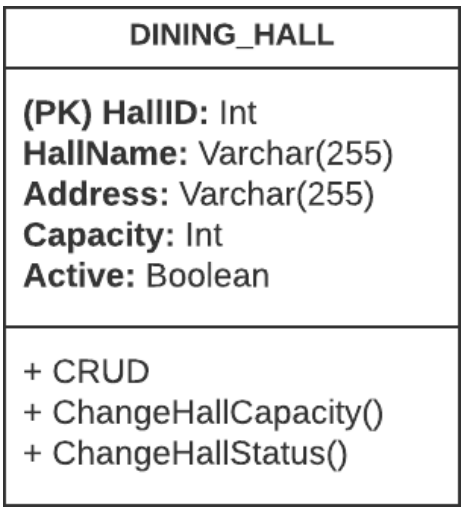
در مرحله اول یک دیتابیس جداگانه برای سرویس غذا خوری دانشگاه ایجاد میکنیم. قرار است تمام داده های مربوط در جدول‌هایی که در ادامه معرفی می‌شوند در این دیتابیس ذخیره شوند.

```
CREATE DATABASE guilan_uni_food_service;
show databases;

+------------------------+
| Database               |
+------------------------+
| guilan_uni_food_service |
| information_schema     |
| mysql                  |
| performance_schema     |
| sys                    |
+------------------------+
```

موجودیت‌ها

- **سالن غذا خوری (DINING_HALL):** در یک دانشگاه ممکن است سالن های غذا خوری متعددی وجود داشته باشد. به طور مثال سلف مرکزی دانشگاه گیلان، سلف علوم پایه و غیره. یک سالن غذاخوری می‌تواند دارای صفت های زیر باشد.
  - شناسه سالن HallID
  - نام Name
  - آدرس Address
  - ظرفیت سالن Capacity
  - وضعیت فعال بودن سالن Active



برای این کلاس، به طور مثال، ۲ قابلیت تغییر ظرفیت سالن و تغییر وضعیت سالن در نظر گرفته شده است.
برای ساخت جدول مورد نظر برای ذخیره سازی رکوردهای سالن غذاخوری می‌توانیم از کد SQL زیر استفاده کنیم.

```
CREATE TABLE DINING_HALL (
HallID int not null auto_increment,
HallName varchar(255),
Address text,
Capacity int,
Active boolean,
primary key (HallID)
);
```

سپس برای شبیه سازی سلف های دانشگاه گیلان، تعدادی سالن غذا خوری اضافه اضافه می‌کنیم.

```
INSERT INTO DINING_HALL (HallName, Address, Capacity, Active)
      VALUES("100 ,"دانشگاه گیلان، جاده تهران","سلف مرکزی", true);

INSERT INTO DINING_HALL (HallName, Address, Capacity, Active)
      VALUES("50 ," دانشگاه گیلان، منظریه","سلف علوم پایه", true);

INSERT INTO DINING_HALL (HallName, Address, Capacity, Active)
      VALUES("30 ," دانشگاه گیلان، منظریه"," سلف تربیت بدنی", false);

SELECT * FROM DINING_HALL;
```

پس از اجرا کردن دستورات بالا با پاسخ زیر مواجه می‌شویم.

Active	Capacity	Address	HallName	HallID
TRUE	100	دانشگاه گیلان، جاده تهران	سلف مرکزی	3
FALSE	30	دانشگاه گیلان، جاده تهران	سلف تربیت بدنی	2
TRUE	50	دانشگاه گیلان، منظریه	سلف علوم پایه	1

- **غذا (FOOD):** غذا عضوی از یک منو است و یک منو می‌تواند چند غذا داشته باشد، نحوی ذخیره سازی مقدار موجودی غذا در ادامه توضیح داده می‌شود. یک غذا می‌تواند دارای صفت های زیر باشد.
  - نام
  - قیمت
  - توضیحات

```
CREATE TABLE FOOD (
    FoodID int not null auto_increment,
    Name varchar(255),
    Price int,
    Description text,
    primary key (FoodID)
    );
```

افزودن چند رکورد در این جدول برای مثال:

```
INSERT INTO FOOD (Name, Price, Description)
      VALUES ("به همراه ماست" ,15000 ,"قیمه");

INSERT INTO FOOD (Name, Price, Description)
      VALUES ("به همراه ترشی" ,65000 ,"کتلت");

INSERT INTO FOOD (Name, Price, Description)
      VALUES ("ماهی سفید" ,25000 ,"سبزی پلو با ماهی");

select * from FOOD;
```

پس از اجرا کردن دستورات بالا با پاسخ زیر مواجه می‌شویم.

Description	Price	Name	FoodID
به همراه ماست	15000	قیمه	1
به همراه ترشی	65000	کتلت	2
ماهی سفید	25000	سبزی پلو با ماهی	3

- **دانشجو (STUDENT):** دانشجو ها می‌توانند با مراجعه به سامانه، غذا مورد نظر خود را در سالن غذاخوری مد نظر، رزرو کنند. یک دانشجو می‌تواند دارای صفت های زیر باشد.
  - نام
  - موجودی حساب
  - شماره دانشجویی
  - شماره تماس

```
CREATE TABLE STUDENT (
    StudentId int not null auto_increment,
    Name varchar(255),
    Balance int default 0,
    Phone varchar(20),
    primary key (StudentID)
);
```

افزودن چند رکورد در این جدول برای مثال:

```
INSERT INTO STUDENT (Name, Phone)
      Values("09123456789" ,"امیرمحمد فلاح");

INSERT INTO STUDENT (Name, Phone)
      Values("John Doe", "09123456790");

SELECT * FROM STUDENT;
```

پس از اجرا کردن دستورات بالا با پاسخ زیر مواجه می‌شویم.

Phone	Balance	Name	StudentID
09123456789	10000	امیرمحمد فلاح	1
09123456790	0	John Doe	2

- **رابطه تعداد غذا و سالن غذا خوری (MENU_ITEM):** برای اینکه آمار غذا های موجود در یک روز مشخص و در یک وعده مشخص که در یک سالن غذاخوری مشخص سرو می شود را در دیتابیس نگه داری کنیم از رابطه ی زیر می توانیم استفاده کنیم. این رابطه می‌تواند دارای صفت های زیر باشد. این رابطه دارای صفت های زیر باشد.
  - شناسه آیتم منو ItemID
  - سالن غذاخوری HallID
  - شناسه غذا FoodID
  - تعداد کل TotalQTY
  - روز سرو ServeDate
  - وعده سرو MealType

برای ذخیرسازی نوع وعده، از قابلیت دامین استفاده شد تا بتوانیم نوع داده جدید موردنیازمان را بسازیم.

```
CREATE DOMAIN meal_type varchar(20)
CHECK (
    VALUE IN ('breakfast', 'lunch', 'dinner')
);

create table menu_item (
      ItemID int not null auto_incerement,
      HallID int,
      FoodID int,
      totalQTY int not null,
      MealType meal_type,
      ServeDate date,
      PRIMARY KEY(ItemId),
      FOREIGN KEY(HallID) REFERENCES dining_hall(HallID),
      FOREIGN KEY(FoodID) REFERENCES food(FoodID)
      );
```

افزودن چند رکورد در این جدول برای مثال:

```
insert into menu_item (HallID, FoodID, totalQTY, MealType, ServeDate)
values (1,1,100,'lunch','2023-01-20');
insert into menu_item (HallID, FoodID, totalQTY, MealType, ServeDate)
values (2,2,100,'lunch','2023-01-21');
insert into menu_item (HallID, FoodID, totalQTY, MealType, ServeDate)
values (3,3,50,'lunch','2023-01-21');
SELECT * FROM menu_item;
```

پس از اجرا کردن دستورات بالا با پاسخ زیر مواجه می‌شویم.

ServeDate	MealType	TotalQTY	FoodID	HallID	ItemID
2023-01-20	lunch	100	1	1	1
2023-01-21	dinner	100	2	1	3
2023-01-21	lunch	100	2	2	4
2023-01-21	lunch	50	3	3	5

- **رابطه دانشجوییان و آیتم منو رزرو شده:** برای نشان داده آیتم منو های رزرو شده توسط دانشجوییان از رابطه زیر استفاده می‌کنیم. این رابطه می‌تواند دارای صفت های زیر باشد. این رابطه دارای صفت های زیر باشد.
  - شناسه رزرواسیون ReservID
  - شناسه دانشجو StudentID
  - شناسه آیتم ItemID

```
create table reserved_food (
      ReservID serial not null,
      StudentID int not null,
      ItemID int not null,
      PRIMARY KEY(ReservID),
      FOREIGN KEY(StudentID) REFERENCES student(StudentID),
      FOREIGN KEY(ItemID) REFERENCES menu_item(ItemID));
```

افزودن چند رکورد در این جدول برای مثال:

```
insert into reserved_food (StudentID, ItemID)
values (1,1);
```

پس از اجرا کردن دستورات بالا با پاسخ زیر مواجه می‌شویم.

ReservID	StudentID	ItemID
1	1	1

با دستور زیر می توانیم بعد از رزرو کردن غذا موجودی حساب دانشجو را کم کرده و یا بعد از شارژ کردن موجودی آن را افزایش دهیم.

```
UPDATE STUDENT
SET Balance=10000
WHERE StudentId=1;
```

چند نمونه Query

نشان دادن غذا های رزرو شده توسط دانشجویی با کد دانشجویی ۱ در روز 2023-01-20 در وعده نهار.

```
select mealtype, student.name, servedate, food.name, dining_hall.hallname from
reserved_food
inner join menu_item on menu_item.itemid = reserved_food.itemid
inner join food on menu_item.foodid = food.foodid
inner join student on reserved_food.studentid = reserved_food.studentid
inner join dining_hall on menu_item.hallid = dining_hall.hallid
where student.studentid=1 and mealtype='lunch' and servedate='2023-01-20';
```

lunch	امیرمحمد فلاح	2023-01-20	قیمه	سلف علوم پایه

نشان دادن تعداد غذا های رزرو شده در یک سالن غدا خوری در یک روز و وعده.

```
select count(*) from menu_item
where hallid=1 and mealtype='lunch' and servedate='2023-01-20';
```

برای استفاده کردن از توابع رشته ای می توانیم به صورت زیر عمل کنیم. عبارت "%Rasht%" نشان دهنده این است که در بخشی از رشته عبارت Rasht آمده باشد.

```
INSERT INTO DINING_HALL (HallName, Address, Capacity, Active)
    VALUES ('Central Hall','No. 5, Rasht', 100, true);

select * from dining_hall where address like'%Rasht%';
```

که جواب Query به صورت زیر خواهد بود.

hallid	hallname	address	capacity	active
4	Central Hall	No. 5, Rasht	100	TRUE

برای حذف رکورد ها می توان از عبارت delete استفاده کرد.

```
delete from dining_hall where hallid=4;
```

این دستور رکوردی که hallID برابر با ۴ دارد را پاک می‌کند.

Tablespace در PostgreSQL به معنای فضایی است که برای ذخیره جداول و دیگر اشیاء پایگاه داده ایجاد می شود. یک Tablespace می تواند شامل یک یا چند فایل در دیسک باشد و برای جداسازی داده ها از دیگر اجزای سیستم فایل استفاده می شود.
در PostgreSQL، هر جدول می تواند در یک Tablespace خاصی ذخیره شود. این کار می تواند برای مدیریت فضای دیسک و بهبود عملکرد پایگاه داده مفید باشد. با استفاده از Tablespace، می توانید جداول را بر اساس نیازهای خود، مانند اندازه و قابلیت اطمینان، به گروه بندی کنید و آنها را در فضای دیسک مناسبی ذخیره کنید.
برای ایجاد یک Tablespace در PostgreSQL، ابتدا باید دستور CREATE TABLESPACE را با نام Tablespace و مسیر دلخواه برای ذخیره فایل های Tablespace اجرا کنید. سپس، با استفاده از دستور ALTER TABLE، می توانید جدول را به Tablespace جدید منتقل کنید.
با استفاده از Tablespace در PostgreSQL، می توانید به راحتی مدیریت فضای دیسک پایگاه داده خود را بهبود دهید و از عملکرد بهتری برخوردار شوید.

برای نمونه، می‌توانید با دستورات زیر یک Tablespace در PostgreSQL ایجاد کنید و یک جدول را به آن منتقل کنید:

1. ابتدا باید با استفاده از دستور زیر یک Tablespace جدید ایجاد کنید:
   این دستور، یک Tablespace با نام "my_tablespace" ایجاد می کند که در مسیر "path/to/my_tablespace/" قرار دارد.

```
CREATE TABLESPACE my_tablespace LOCATION '/path/to/my_tablespace';
```

2. حالا باید یک جدول ایجاد کنید و آن را به Tablespace جدید منتقل کنید. برای مثال:

```
CREATE TABLE my_table (id SERIAL, name TEXT) TABLESPACE my_tablespace;
```

این دستور، یک جدول با نام "my_table" و دو ستون "id" و "name" ایجاد می کند و آن را در Tablespace ""my_tablespace" ذخیره می کند.