
Amirkabir
University of Technology

به نام خدا

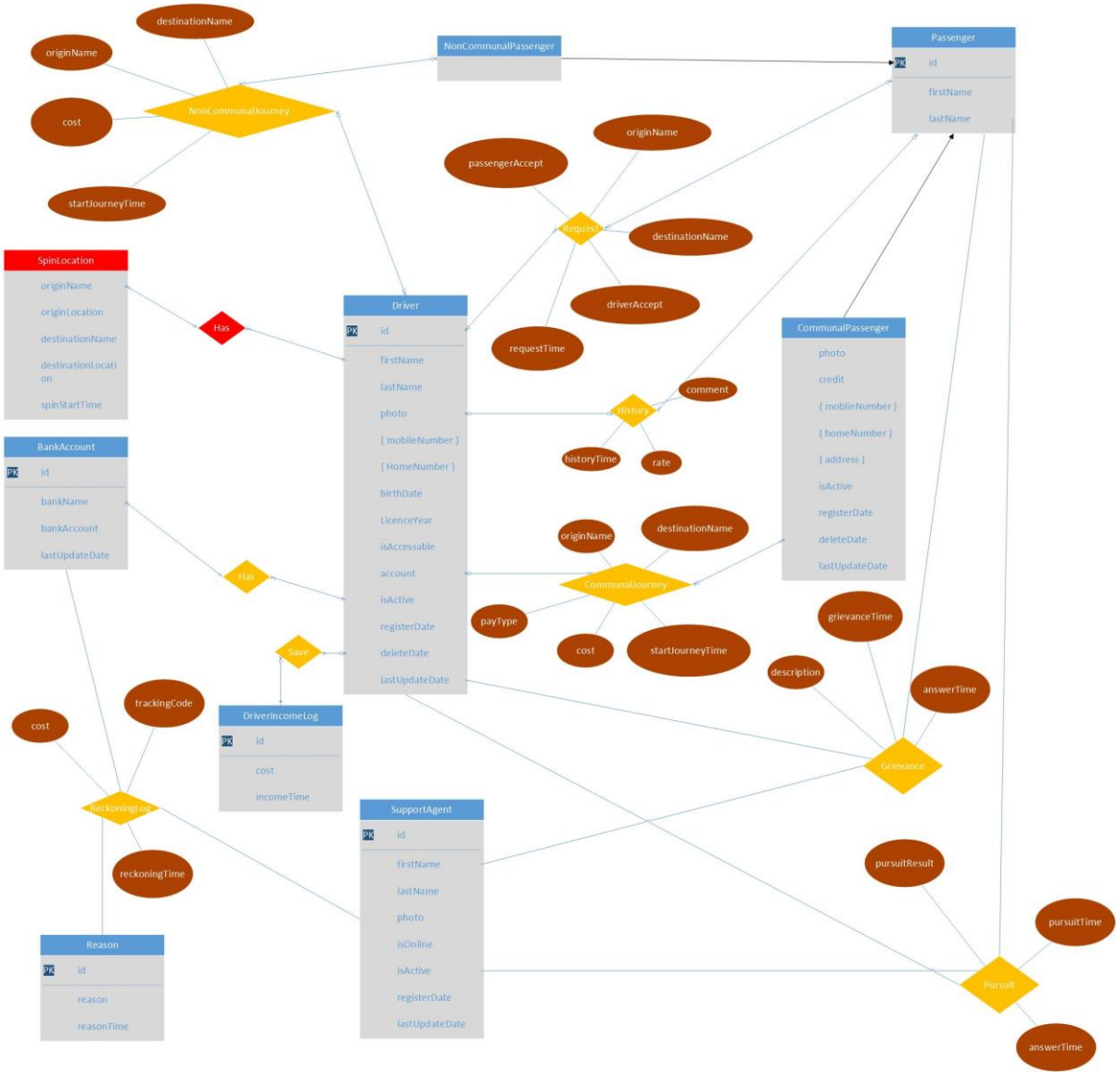
# طراحی پایگاه داده شرکت تاکسیرانی

سید نوید کرمی نژاد

تیر ۱۳۹۶

این گزارش مربوط به طراحی پایگاه داده یک شرکت تاکسیرانی است. کاربران این شرکت عبارت اند از : مدیر شرکت، رانندگان، مسافران و عوامل پشتیبانی سیستم

باتوجه به پروژه تعریف شده ERD مربوط به این پروژه به صورت زیر است:




که در آن Weak Entity ها با رنگ قرمز مشخص شده‌اند.

همچنین تمام روابط بین انتیتی ها با رنگ زرد نشان داده شده‌اند.

باتوجه به شکل بالا "گشت‌زنی" (Spin Location) یک Weak Entity محسوب می‌شود که Driver، Strong Entity آن می‌باشد.

براساس Modality و Cardinality روابط جداول نهایی به صورت زیر درمی‌آید :

```
create table Driver (
  id serial primary key,
  firstName varchar(20) not null,
  lastName varchar(20) not null,
  photo varchar(100) not null,
  birthDate date not null,
  LicenceYear date not null,
  isAccessable int not null check (isAccessable between 0 and 2), --  0 : offline , 1 : online , 2 : servising
  account real default 0,
  isActive boolean not null, -- false : acount deleted
  registerDate timestamp not null,
  deleteDate timestamp,
  lastUpdateDate timestamp
);
```

```
create table DriverMobileNumber(
  id serial primary key,
  driverId int not null,
  phoneNumber varchar(11) not null,
  lastUpdateDate timestamp,
  foreign key (driverId) references Driver(id) on update cascade
);
```

```
create table DriverHomeNumber(
    id serial primary key,
    driverId int not null,
    phoneNumber varchar(11) not null,
    lastUpdateDate timestamp,
    foreign key (driverId) references Driver(id) on update cascade
);
```

```
create table DriverBankAccount(
  id serial primary key,
  driverId int not null,
  bankName varchar(10) not null,
  bankAccount varchar(16) not null,
  lastUpdateDate timestamp,
  foreign key (driverId) references Driver(id) on update cascade
);
```

```
create table SpinLocation (
  id int primary key,
  originName varchar(10) not null,
  originLocation varchar(100) not null,
  destinationName varchar(10) not null,
  destinationLocation varchar(100) not null,
  spinStartTime timestamp,
  foreign key (id) references Driver(id) on update cascade
);
```

```sql
create table Passenger(
  id serial primary key,
  firstName varchar (10) not null,
  lastName varchar (10) not null
);
```

```sql
create table CommunalPassenger(
  id int primary key,
  photo varchar(100) not null,
  credit real not null,
  isActive boolean not null, -- false : deleted account
  registerDate timestamp not null,
  deleteDate timestamp,
  lastUpdateDate timestamp not null,
  foreign key (id) references Passenger(id) on update cascade
);
```

```sql
create table NonCommunalPassenger(
  id int primary key,
  foreign key (id) references Passenger(id) on update cascade
);
```

```sql
create table CommunalPassengerMoblieNumber(
  id serial primary key,
  passengerId int not null,
  phoneNumber varchar(11) not null,
  lastUpdateDate timestamp,
  foreign key (passengerId) references CommunalPassenger(id) on update cascade
);
```

```sql
create table CommunalPassengerAddress(
  id serial primary key,
  passengerId int not null,
  city varchar(10) not null,
  street varchar(10) not null,
  alley varchar(10) not null,
  plate int not null,
  lastUpdateDate timestamp,
  foreign key (passengerId) references CommunalPassenger(id) on update cascade
);
```

```sql
create table CommunalPassengerHomeNumber(
  id serial primary key,
  passengerId int not null,
  phoneNumber varchar(11) not null,
  lastUpdateDate timestamp,
  foreign key (passengerId) references CommunalPassenger(id) on update cascade
);
```

```sql
create table History(
  id serial primary key,
  driverId int not null,
  passengerId int not null,
  comment varchar(100),
  rate int check (rate between 0 and 20),
  historyTime timestamp,
  foreign key (driverId) references Driver(id) on update cascade,
  foreign key (passengerId) references Passenger(id) on update cascade
);
```

```sql
create table Request(
  id serial primary key,
  driverId int not null,
  passengerId int not null,
  originName varchar(10),
  destinationName varchar(10),
  driverAccept boolean not null,
  passengerAccept boolean not null,
  requestTime timestamp not null,
  foreign key (driverId) references Driver(id) on update cascade,
  foreign key (passengerId) references Passenger(id) on update cascade
);
```

```sql
  create table CommunalJourney(
  id serial primary key,
  driverId int not null,
  passengerId int not null,
  originName varchar(10) not null,
  destinationName varchar(10) not null,
  cost real not null,
  payType varchar(20) not null check (payType in ('online','credit')),
  startJourneyTime timestamp not null,
  foreign key (driverId) references Driver(id) on update cascade,
  foreign key (passengerId) references CommunalPassenger(id) on update cascade
);
```

```sql
create table NonCommunalJourney(
  id serial primary key,
  driverId int not null,
  passengerId int not null,
  originName varchar(10) not null,
  destinationName varchar(10) not null,
  cost real not null,
  startJourneyTime timestamp not null,
  foreign key (driverId) references Driver(id) on update cascade,
  foreign key (passengerId) references NonCommunalPassenger(id) on update cascade
);
```

```sql
create table Grievance(
  id serial primary key,
  driverId int,
  passengerId int,
  supportAgentId int,
  description varchar(200) not null,
  grievanceTime timestamp not null,
  answerTime timestamp null,
  foreign key (driverId) references Driver(id) on update cascade,
  foreign key (passengerId) references Passenger(id) on update cascade,
  foreign key (supportAgentId) references SupportAgent(id) on update cascade
);
```

```sql
create table Pursuit(
  id serial primary key,
  driverId int,
  passengerId int,
  supportAgentId int,
  pursuitResult varchar(100),
    pursuitTime timestamp not null,
  answerTime timestamp not null,
  foreign key (driverId) references Driver(id) on update cascade,
  foreign key (passengerId) references Passenger(id) on update cascade,
  foreign key (supportAgentId) references SupportAgent(id) on update cascade
);
```

```sql
create table DriverIncomeLog(
  id serial primary key,
  driverId int,
  cost real not null,
  incomeTime timestamp not null,
  foreign key (driverId) references Driver(id) on update cascade
);
```

```sql
create table Reason(
  id serial primary key,
  reason varchar(100),
  reasonTime timestamp
);
```

```sql
create table ReckoningLog(
  id serial primary key,
  bankAccountId int,
  supportAgentId int,
  reasonId int,
  cost real not null,
  trackingCode varchar(12) not null,
  reckoningTime timestamp not null,
  foreign key (bankAccountId) references DriverBankAccount(id) on delete cascade on update cascade,
  foreign key (supportAgentId) references SupportAgent(id) on update cascade,
  foreign key (reasonId) references Reason(id) on delete cascade on update cascade
);
```

برای نگهداری لاگ برخی از جداول نیاز به ایجاد چند جدول جدید احساس می‌شد که در زیر مشاهده می‌کنید:

```
create table DriverLog(
  id serial primary key,
  driverId int,
  columnName text,
  oldValue text,
  newValue text,
  changer text,
  currentUpdateDate timestamp
);
```

```
create table PassangerLog(
  id serial primary key,
  passengerId int,
  columnName text,
  oldValue text,
  newValue text,
  changer text,
  currentUpdateDate timestamp
);
```

```
create table SupportAgentLog(
  id serial primary key,
  supportAgentId int,
  columnName text,
  oldValue text,
  newValue text,
  changer text,
  currentUpdateDate timestamp
);
```

```
create table SpinLocationLog(
  id serial primary key,
  driverId int,
  originName varchar(10),
  originLocation varchar(100),
  destinationName varchar(10),
  destinationLocation varchar(100),
  spinStartTime timestamp,
  spinEndTime timestamp
);
```

```sql
create table CommunalJourneyLog(
  id serial primary key,
  driverId int,
  passengerId int,
  originName varchar(10),
  destinationName varchar(10),
  cost real not null,
  payType varchar(20),
  startJourneyTime timestamp,
  endJourneyTime timestamp
);
```

```sql
create table NonCommunalJourneyLog(
  id serial primary key,
  driverId int,
  passengerId int,
  originName varchar(10),
  destinationName varchar(10),
  cost real,
  startJourneyTime timestamp,
  endJourneyTime timestamp
);
```

برای وارد کردن داده‌ها در جداول مربوط به لاگ تریگر های زیر تعریف شده‌است:

- driverUpdate
- driverMobileUpdate
- driverHomeUpdate
- DriverBankAccountUpdate

پس از تغییر در جداول مربوط به رانندگان اطلاعات قبلی و جدید در جدول مربوطه ذخیره می‌شود.

- PassengerUpdate
- CommunalPassengerUpdate
- CommunalPassengerHomeNumberUpdate
- CommunalPassengerMoblieNumberUpdate
- CommunalPassengerAddressUpdate

به‌طور مشابه برای مسافران بااشتراک نیز چنین چیزی تعریف شده‌است.

- supportAgentUpdate

به‌طور مشابه برای عوامل پشتیبانی نیز چنین چیزی تعریف شده‌است.

- spinLocationDelete
- communalJourneyDelete
- nonCommunalJourneyDelete

در برخی از جداول مانند گشت‌زنی و سفرها (اشتراکی و غیراشتراکی) با حذف یک داده نباید اطلاعات آن از بین برود و باید قابل بازیابی باشد. لذا پس از حذف از این جداول اطلاعات را در جداول لاگ مربوط به آن‌ها ذخیره می‌کنیم.

- ReckoningLogInsert

افزودن داده جدید به این جدول تغییراتی را باعث می‌شود که باید اعمال شوند. برای مثال باید مقدار ذخیره شده در ستون هزینه (Cost) باید از حساب کاربری راننده کم شود.