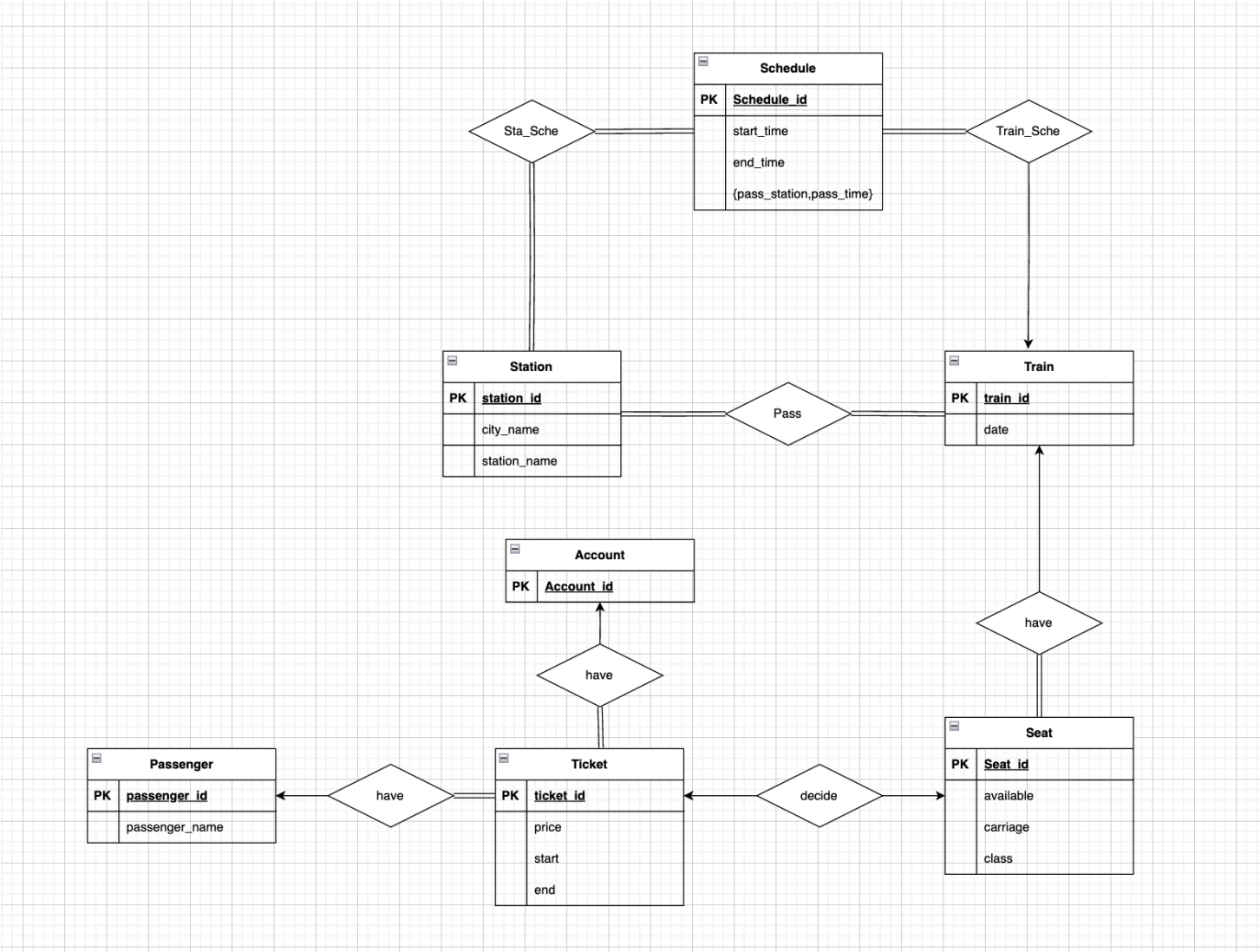
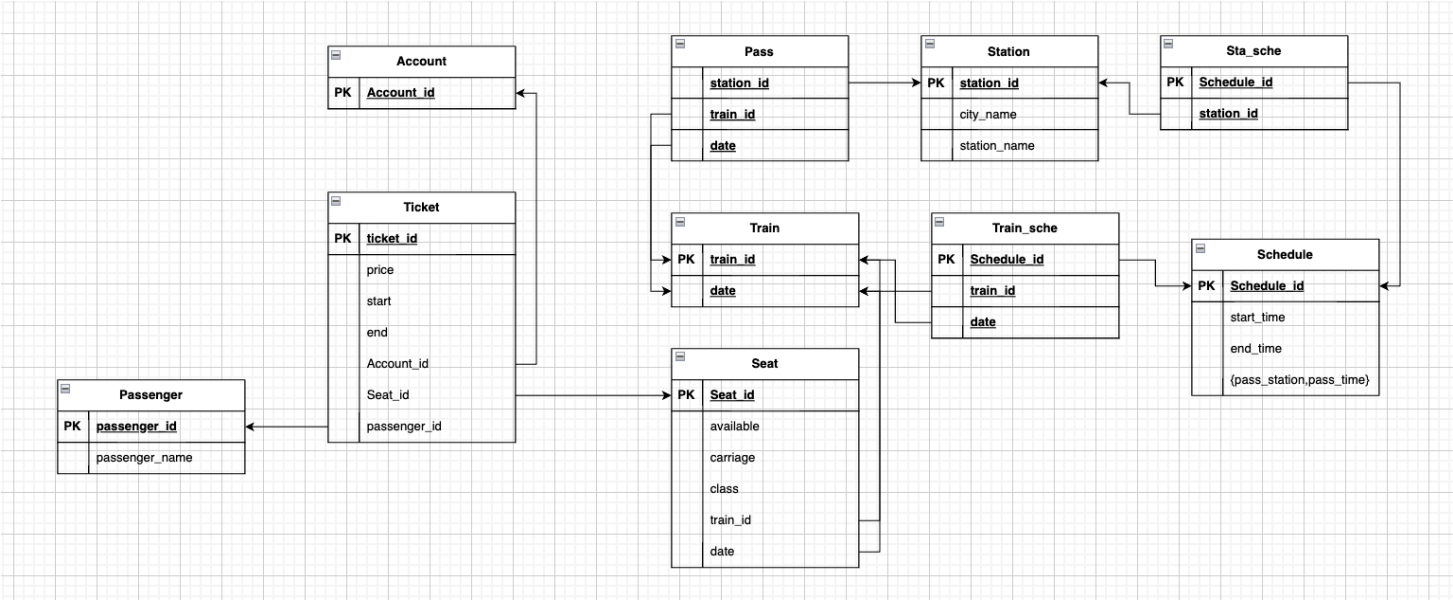
为中国铁路12306数据库设计ER图，并转换成关系模式

Task1:E-R图与关系模式





Task2:回帖点评

刘韬 2分钟前

0

致远同志你好～你的设计成功支持了购票查询等核心功能，E-R图绘制清晰，令我深受启发。对于time_table和schedule的设计我有些疑惑，如果将二者合一也能够满足核心功能的需求，这也许是一个可能的优化方向。

0 评论

Task3:改进

1. 座位偏好选择：当前座位偏好只有位置，比如ABC EF，可以增加座位类型，比如是否静音，是否有电源等等，这些可以在数据库中Seat表中增加字段。
2. 实时动态通知：在实际的列车运行中由于各种原因和意外情况，往往会发生列车晚点的情况，在每次更新列车情况后及时推送给用户，让用户可以及时调整行程。

Task4:sql实现

```sql
create table train(
    train_id int primary key,
    date time primary key,
);

create table passanger(
    passanger_id int primary key,
    passanger_name varchar(20),
);

create table account(
    account_id int primary key
);

create table station(
    station_id int primary key,
    station_name varchar(20),
    city_name varchar(20),
);

create table schedule(
    schedule_id int primary key,
    start_time time,
    end_time time,
    stop_time time,
    start_station_id int,
    end_station_id int,
    foriegn key(start_station_id) references station(station_id),
    foriegn key(end_station_id) references station(station_id),
);

create table seat(
    seat_id int primary key,
    seat_availability boolean,
    seat_carriage int,
    seat_class varchar(20),
    foriegn key(train_id) references train(train_id),
    foriegn key(date) references train(date),
);
```

```
create table train_sche(
    train_id int,
    schedule_id int,
    date time,
    foriegn key(train_id) references train(train_id),
    foriegn key(schedule_id) references schedule(schedule_id),
    foriegn key(date) references train(date),
    primary key(train_id, schedule_id, date),
);

create table sta_sche(
    station_id int,
    schedule_id int,
    foriegn key(station_id) references station(station_id),
    foriegn key(schedule_id) references schedule(schedule_id),
    primary key(station_id, schedule_id),
);

create table pass(
    station_id int,
    train_id int,
    date time,
    foriegn key(station_id) references station(station_id),
    foriegn key(train_id) references train(train_id),
    foriegn key(date) references train(date),
    primary key(station_id, train_id, date),
);

create table ticket(
    ticket_id int primary key,
    ticket_price int,
    start_station_id int,
    end_station_id int,
    account_id int,
    seat_id int,
    passanger_id int,
    foriegn key(start_station_id) references station(station_id),
    foriegn key(end_station_id) references station(station_id),
    foriegn key(account_id) references account(account_id),
    foriegn key(seat_id) references seat(seat_id),
    foriegn key(passanger_id) references passanger(passanger_id),
    primary key(ticket_id),
);
```

功能1:

```sql
SELECT t.train_id, s.start_time, s.end_time
FROM train_sche ts
JOIN train t ON ts.train_id = t.train_id
JOIN schedule s ON ts.schedule_id = s.schedule_id
JOIN station start ON s.passing_station_id = start.station_id
JOIN station end ON s.station_id = end.station_id
WHERE start.city_name = 'A' AND end.city_name = 'B'
ORDER BY s.start_time;
```

功能2:

```sql
SELECT t.train_id, s.start_time, s.end_time, seat.seat_id, seat.seat_availability
FROM train_sche ts
JOIN train t ON ts.train_id = t.train_id
JOIN schedule s ON ts.schedule_id = s.schedule_id
JOIN station start ON s.passing_station_id = start.station_id
JOIN station end ON s.station_id = end.station_id
JOIN seat ON seat.train_id = t.train_id AND seat.date = ts.date
WHERE start.city_name = 'A' AND end.city_name = 'B'
ORDER BY s.start_time;
```

功能3:

```sql
SELECT p.train_id
FROM pass p
WHERE p.station_id = 'x' AND p.date = 'y';
```

功能4:

```sql
SELECT p.station_id, s.start_time, s.end_time, s.start_station_id, s.end_station_id,s.s
FROM pass p
JOIN train_sche ts ON p.train_id = ts.train_id AND p.date = ts.date
JOIN schedule s ON ts.schedule_id = s.schedule_id AND ts.date = s.date
WHERE p.station_id = 'x' AND p.date = 'y';
```

功能5:

```sql
-- 1. 找到该列车的车厢号
SELECT DISTINCT seat_carriage
FROM seat
WHERE train_id = {train_id};

-- 2. 找到该列车在同一时间段的所有乘客
SELECT DISTINCT t.passanger_id
FROM ticket t
JOIN seat s ON t.seat_id = s.seat_id
JOIN train_sche ts ON s.train_id = ts.train_id AND t.date = ts.date
WHERE ts.train_id = {train_id} -- 替换为要查询的列车ID
AND ts.date = {date}; -- 替换为要查询的日期

-- 3. 筛选出与该乘客同乘一个车厢的所有乘客
SELECT DISTINCT p.passanger_id, p.passanger_name
FROM ticket t1
JOIN seat s1 ON t1.seat_id = s1.seat_id
JOIN seat s2 ON s1.seat_carriage = s2.seat_carriage
JOIN ticket t2 ON s2.seat_id = t2.seat_id
JOIN passanger p ON t2.passanger_id = p.passanger_id
WHERE t1.passanger_id = {passenger_id}; -- 替换为要查询的乘客ID
```

功能6:

```
BEGIN TRANSACTION;

-- 替换下面的参数为具体的值
DECLARE @passenger_id INT = <passenger_id>; -- 乘客ID
DECLARE @train_id INT = <train_id>; -- 列车ID
DECLARE @start_station_id INT = <start_station_id>; -- 起始站ID
DECLARE @end_station_id INT = <end_station_id>; -- 终点站ID
DECLARE @seat_class VARCHAR(20) = '<seat_class>'; -- 座位等级，例如 'economy' 或 'business
DECLARE @seat_carriage INT = <seat_carriage>; -- 车厢号
DECLARE @ticket_price INT = <ticket_price>; -- 票价
DECLARE @account_id INT = <account_id>; -- 用户账号ID

-- 检查所选座位是否可用
IF NOT EXISTS (
    SELECT 1
    FROM seat
    WHERE train_id = @train_id
    AND seat_carriage = @seat_carriage
    AND seat_availability = true
) BEGIN
    -- 座位不可用，回滚事务并返回错误消息
    ROLLBACK TRANSACTION;
    PRINT 'Seat is not available.';
    RETURN;
END

-- 插入票信息到 ticket 表中
INSERT INTO ticket (ticket_price, start_station_id, end_station_id, account_id, seat_id
VALUES (@ticket_price, @start_station_id, @end_station_id, @account_id,
        (SELECT seat_id FROM seat WHERE train_id = @train_id AND seat_carriage = @seat_
        @passenger_id);

-- 更新座位的可用性
UPDATE seat
SET seat_availability = false
WHERE train_id = @train_id
AND seat_carriage = @seat_carriage
AND seat_class = @seat_class;

-- 提交事务并返回成功消息
```

```
COMMIT TRANSACTION;
PRINT 'Ticket booked successfully.';
```