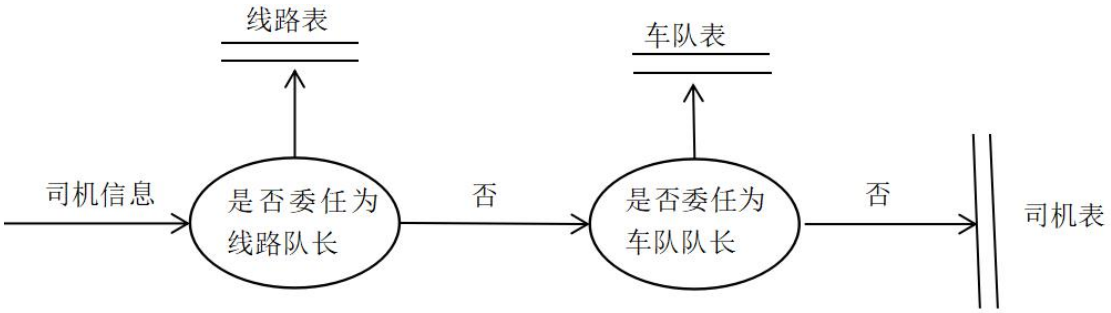
一、 需求分析

1. 处理要求

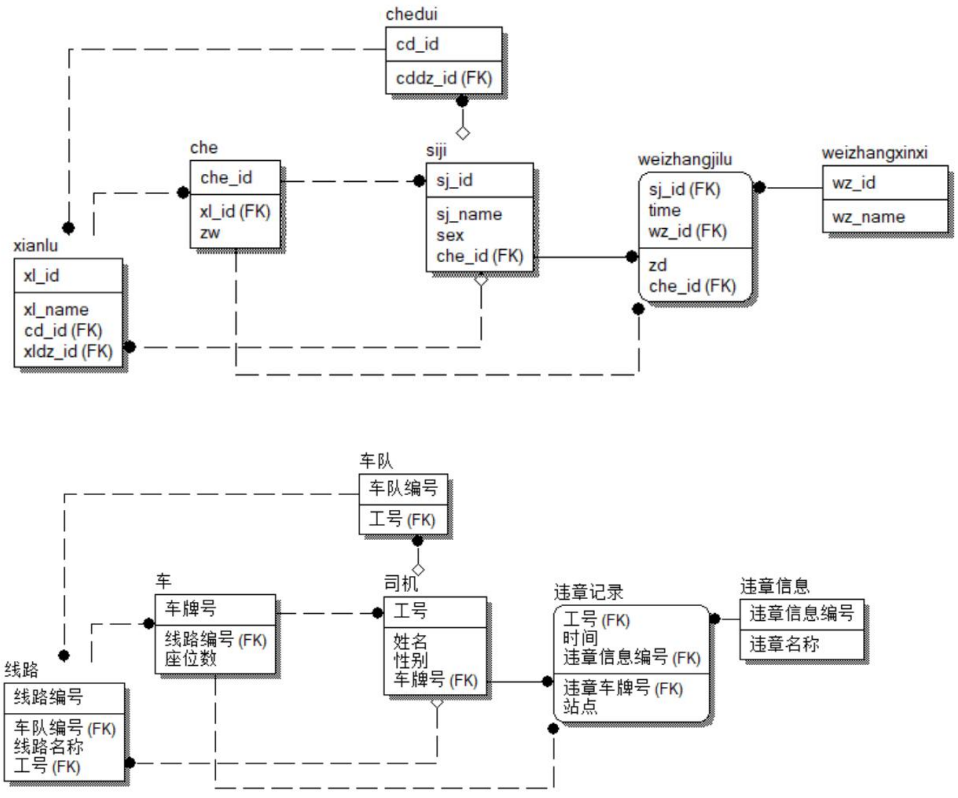
1. 要求能够录入司机的基本信息，如工号，姓名，性别等；
2. 要求能够录入汽车基本信息，如车牌号，座位数等；
3. 要求能够录入司机的违章信息；
4. 要求能够查询车队下全体司机信息
5. 要求能够查询某名司机在某个时间段内的违章记录
6. 要求能够查询某个车队在某段时间内的违章统计信息，输出应该为一句话。



2. 完整性要求

1. 每个车队下只有一名路队长，他只管理，不开车
2. 每条线路下有一名路队长，除了开车外还进行管理工作
3. 每名司机只在一条线路上开车
4. 性别只能为男女
5. 车牌号唯一

二、 概念结构设计



ER 图设计

1. 关于公交公司我的认知有问题，所以认为司机和车是多对一的关系，同时我认为这样便于追责到人。
2. 我设计的时候将他们看作多个对象，分别是车队，线路，车辆，司机，违章。其中车队和线路是一对多，线路和车辆是一对多，车辆和司机是一对多。其中车队队长和线路队长都看作司机的特例，他们三者首先都是员工，其次才有等级划分。这样可以保证每个车队只有一个队长，每条线路只有一个路队长，同时人员调动时只需要修改一张表。
3. 这样做遇到的问题是，车队队长比较特殊，他不开车。数据中的表达是他的 che_id 字段是 null。相应的需要在一些存储过程中加入条件插入或者在前端判断是否是车队队长能做的事。
4. 对于违章记录，我新建了一张违章信息表，用来存储各种违章的名字和对应的编号，这样可以减少存储占用的空间。
5. 在违章记录表中，通过(站点,违章时间,工号)组合作为主码，没有非主属性对码的传递或部分函数依赖，且主属性间没有依赖，是完全的一一对应关系，达到四范式。对于这张表也考虑到可能有司机在违章之后出现车队变动的情况，所以设置了违章车牌的字段。其余表也显然达到了四范式。

三、 逻辑结构设计

根据 ER 图转换为关系模型，其中关系模式包括

1. 车队(车队编号，工号)
2. 线路(线路编号，线路名称，线路队长编号)
3. 车(车牌号，座位数)
4. 司机(司机工号，姓名，性别，车牌号)
5. 违章记录(司机工号，违章时间，违章信息编号，违章车牌号，站点编号)
6. 违章信息(违章信息编号，违章名称)

四、 软件开发环境及应用环境

1. 数据库: MYSQL
2. 后端：JAVA
3. 前端：JAVA
4. 所有的存储过程在 create_procedure.sql 中
5. 所有的创建视图都在 create_view.sql 中
6. 所有插入数据的操作放在 insert_data 中，包括 6 个车队，200 条线路，1200 名司机。

五、 应用程序设计中遇到的问题及解决方法

1. 由于 MYSQL 中 CHECK 约束没有用，所以另加了一张性别表，通过外键约束来限制司机表中的性别，同时前端也设置成了二选一。
2. 查询全体司机信息时，想输出每个员工的职务。最初是通过建立一个视图，里面只包含必要的信息，然后对每个员工单独判断是否在车队队长表中，或者线路队长表中实现。但是后来加入 1200 名司机后发现这样做会使数据库断线抛出异常，猜测是由于判断职务时过于频繁的连接数据库导致。后来改用新的视图，其中包括如下字段

sj_id	sex	sj_name	che_id	xl_id	cd_id	xldz_id	cddz_id

然后通过判断司机编号与车队队长编号或线路队长编号是否相等来判断职务。
3. 建立上面这个视图时遇到了新的问题是，由于车队队长不开车，所以一般司机那种通过车表自然连接线路表得到视图的方法就行不通。我是通过司机表直接

连接车队表，通过司机编号与车队队长编号连接，同时给视图中线路编号和线路队长编号赋 null，得到第二个视图。然后将两个视图通过 union 联合得到想要的结果表。分别对应视图 che_xl_cd,cddz_cd,sj_xl_cd。

```
#sj_xl_cd的基础：将车表，线路表，车队表组合
CREATE OR REPLACE VIEW che_xl_cd
AS
        SELECT xl.xl_id,cd.cd_id,xl.xldz_id,cd.cddz_id
        FROM che,xianlu xl,chedui cd
        WHERE che.`xl_id` = xl.xl_id
        AND xl.cd_id = cd.cd_id
$
#用于联合查询
CREATE OR REPLACE VIEW cddz_cd
AS
        SELECT sj.sj_id,sj.sex,sj.sj_name,NULL xl_id,NULL xldz_id,cd.cddz_id,cd.cd_id
        FROM siji sj
        LEFT OUTER JOIN chedui cd
        ON cd.cddz_id = sj_id
        WHERE sj.che_id IS NULL
$
#所有司机的详细信息并包括车队队长编号，线路队长编号
CREATE OR REPLACE VIEW sj_xl_cd
AS
        SELECT sj.sj_id,sj.sex,sj.sj_name,tl.xl_id,cd_id,tl.xldz_id,cddz_id
        FROM siji sj,che_xl_cd tl
        WHERE sj.che_id = tl.che_id
        UNION
        SELECT sj_id,sex,sj_name,xl_id,cd_id,xldz_id,cddz_id
        FROM cddz_cd
        ORDER BY sj_id
$
```

4. 创建委任车队队长的存储过程时，直接委任可能导致有人兼职了两个车队队长。采用条件插入，如果他原来是车队队长，那么将车队表中原车队队长编号置为 null，否则不变。之后再更新司机表中该司机车为 null，并将相应的车队队长设为他。对应存储过程 appoint_cddz

```
CREATE  PROCEDURE       appoint_cddz(IN id INTEGER,IN cd INTEGER)
BEGIN
        UPDATE chedui SET cddz_id =
        (
        CASE
        WHEN (
                SELECT sj_id
                FROM sj_xl_cd sj
                WHERE sj_id = id
                AND cddz_id = sj_id
             )
        IS NOT NULL
        THEN NULL
        ELSE cddz_id
        END
        )WHERE cd_id =  (
                SELECT cd_id
                FROM sj_xl_cd sj
                WHERE sj_id = id
        );

        UPDATE chedui SET cddz_id = id
        WHERE cd_id = cd;
        UPDATE siji SET che_id = NULL
        WHERE sj_id = id;
END $
```

5. 任命线路队长，由于一辆车只能属于一个线路，为了简化操作，规定只能被任命为所在线路的线路队长，所以 appoint_xldz 这个存储过程中也使用了条件插入。

6. 对于录入司机违章信息的存储过程，存在的问题是第一要保证司机和车辆在同一个线路，第二要保证当前是车队队长的不能被录入，因为他不开车，第三是要保证违章时间不能超过当前时间。对于第一个问题我是在存储过程中加入判断被录入司机的线路是否与车辆所在线路相同，第二个问题通过查询 2 中提到的 sj_xl_cd 视图完成，第三个问题直接和 NOW()比较即可。

```
CREATE PROCEDURE insert_weizhangjilu(IN zd INTEGER ,IN t DATETIME,IN sj_id INTEGER,IN wz_id INTEGER,IN che VARCHAR(20))
BEGIN
	INSERT INTO weizhangjilu(`zd`,`TIME`,`sj_id`,`wz_id`,`che_id`)
	SELECT zd,t,sj_id,wz_id,che
	FROM DUAL
	WHERE EXISTS (
	SELECT *
	FROM che_xl_cd c
	WHERE c.che_id  = che
	AND c.xl_id = (
		SELECT xl_id
		FROM sj_xl_cd sj
		WHERE sj.sj_id = sj_id
		)
	)
	AND NOT EXISTS(
	SELECT *
	FROM sj_xl_cd sj
	WHERE sj.sj_id=sj_id
	AND sj.sj_id = sj.cddz_id
	)
	AND  t<=NOW();
END $
```

7. 查询某个车队违章统计信息的时候，分组的依据起初是用违章名来分组，后来听老师课上讲的例题了解到用违章编号和违章名共同分组更好，因为违章名在所查询的视图中不是唯一的。已改正。

8. 考虑到队长可能被降职，所以会出现没有车辆又不是队长的司机。所以增加了一个分配车辆的存储过程，其中要注意的问题是不能利用这个功能给车队队长分配车辆，同样是通过条件插入完成。

```
CREATE PROCEDURE query_che_xl(IN xl_id INTEGER)
BEGIN
	SELECT * FROM che WHERE che.xl_id = xl_id;
END $
#更新司机的车辆
DROP PROCEDURE update_che $
CREATE PROCEDURE update_che(IN sj_id INTEGER,IN che_id VARCHAR(20))
BEGIN
	UPDATE siji sj SET sj.che_id =
	(
	CASE
	WHEN (
		SELECT sj.sj_id
		FROM sj_xl_cd sj
		WHERE sj.cddz_id != sj.sj_id
		AND sj.sj_id = sj_id
	)IS NULL
	THEN  NULL
	ELSE che_id
	END
	)WHERE sj.sj_id = sj_id;
END $
```

9. 存储过程相对于直接调用 sql 语句更快也更清晰，但是我发现对于我通过条件插入排除的那些数据返回值都是“受影响行数为 0”，所以无法直接通过返回

值在前端给出相应的提示信息。没想到好的办法，最后是在前端通过一些简单的 sql 来判断。

10. 前后端其实也有不少问题，但是都是 JAVA 问题不多赘述。

六、 总结

1. 需求分析非常重要，前期没有和老师沟通车和司机的关系，又一直在自己做导致出了问题，下次一定注意。

2. 通过本次实验，不仅练习了大量的 SQL，还让我明白是视图的重要性。第一版数据库设计有冗余，通过建立一些视图才减小了我前后端的修改量。

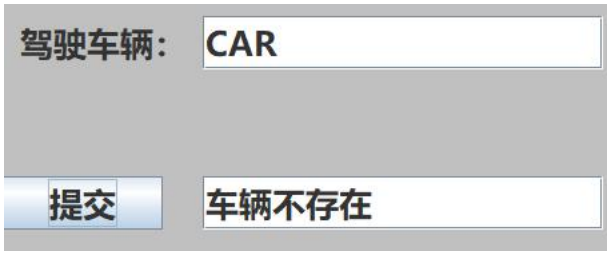
3. 练习使用了存储过程，这一部分是自学的，只使用了一些基础的功能，同时对于有限制的插入好像应该用触发器实现，以后会多多学习。

4. 关于如何使用存储过程，同时又向前端给出插入或者更新失败的具体原因，还没有找到方法，目前是在存储过程中有限制，同时前端也会检查，感觉是挺多余的操作。

5. 总的来说，本次实验相对完整的做了一个管理系统，其中遇到的问题比较多样，通过本次实验复习，学习，练习，令我受益匪浅。

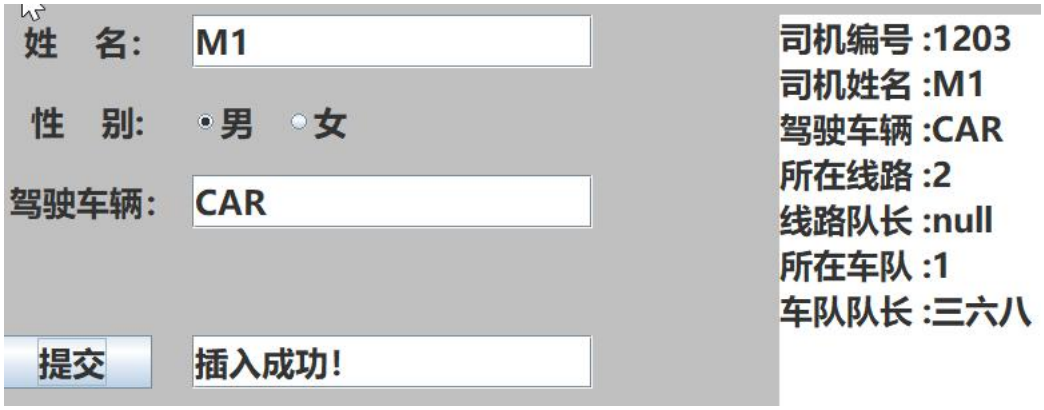
七、 附录 程序截图，以及一些错误提示。

1. 新建司机会验证车辆存在性



2. 录入相同车牌号会提示，同时检查线路存在

3. 成功录入司机信息会回显

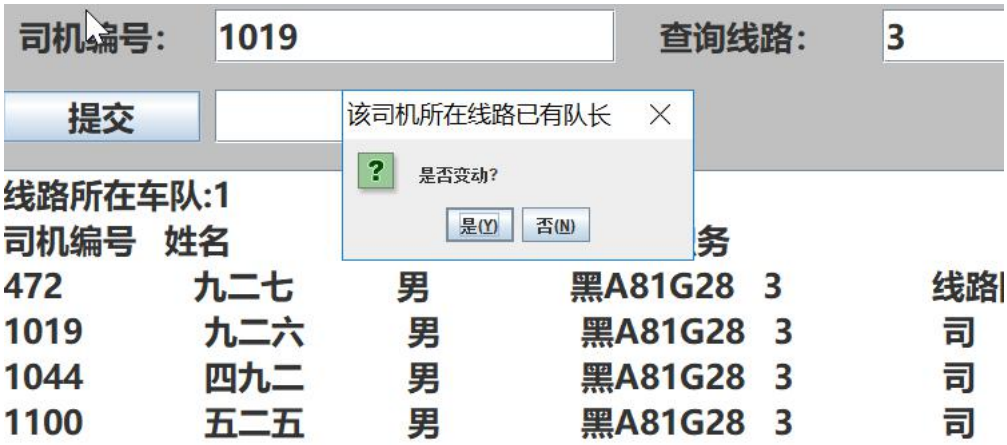


4. 新建行路，线路名称设为了 unique，也会输出相应信息。同时检查车队存在

5. 录入违章信息时，会检查司机是否存在且不是车队队长，违章编号是否存在，时间是否超过当前时间

6. 委任车队队长会检查车队和司机的存在，同时如果目标车队有队长，则会再次询问。

7. 委任线路队长时，会检查该司机是否是车队队长，车队队长不开车所以不能兼职，同时也会检查对应线路是否有路队长。



8. 分配车辆的时候，检查该司机是否是车队队长。

9. 总体图