Chip Production Line Management System

By

Chen Yiwen 120090894
Luo Kaicheng 120090770
Zhu Jiajun 120090498
Zhao Dule 120090080
Long Jiyuan 120090850
Cao Xiao 120090847

* * * * * * * * *

Submitted in fulfillment

of the requirements for

Course Project of CSC3170, 22-23 Fall Semester

THE CHINESE UNIVERSITY OF HONG KONG, SHENZHEN

December, 2022

1 . Introduction

Nowadays, production line around the world is facing a huge challenge due to epidemic situation. We want the production line from customer to plant is as clear as it may seen.

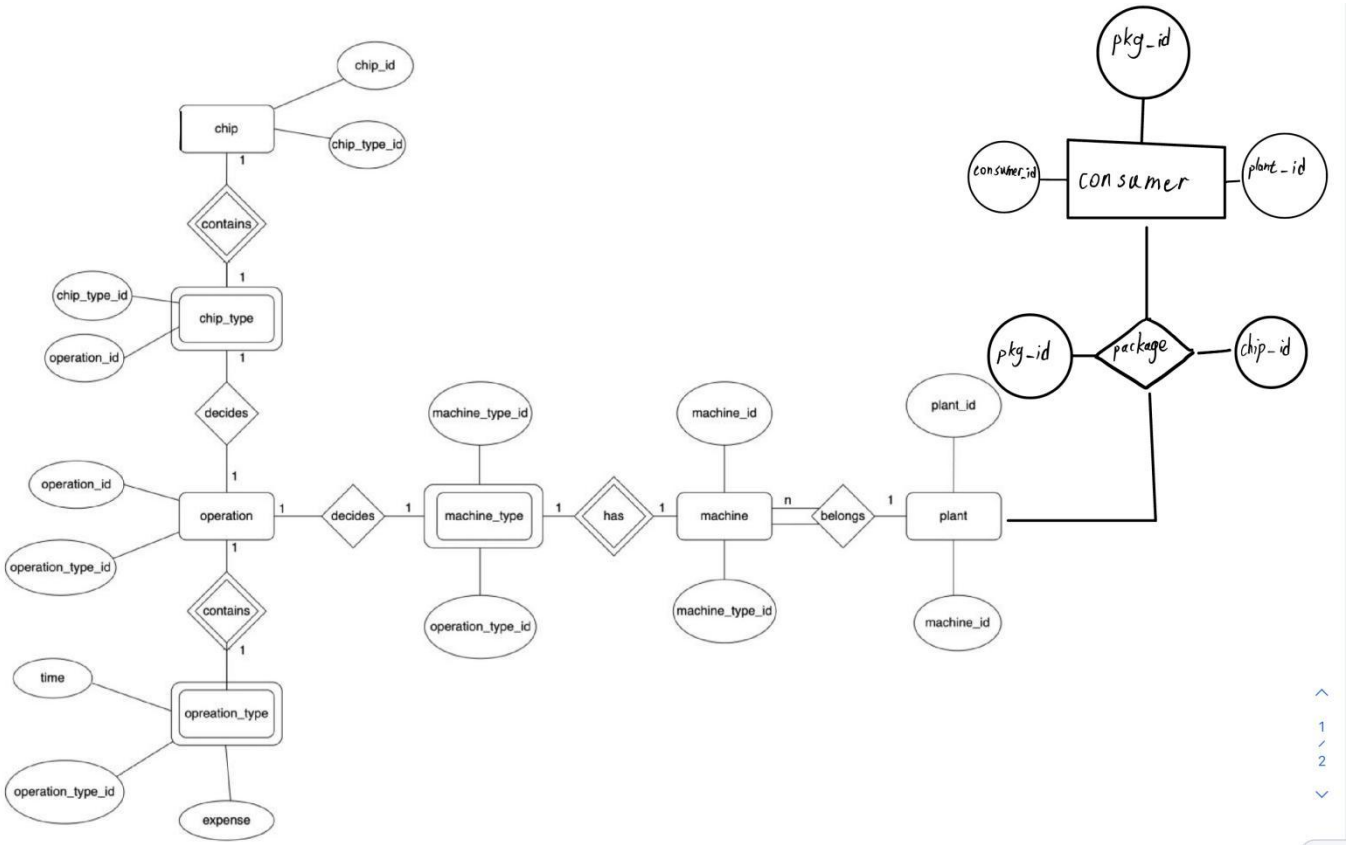
Requirement analysis:

We firstly thick that for customer. They want their package to be produced by some specific plants. So that customer can directly assign their required package to plant.

Meanwhile, in order to give customer the time they need to receive their package. We need to calculate the time it need to produce such package. We also need to consider the occupation of those plants. Because one plants can only conduct one operation at a time. For that we need to have a table to restore those information.

2 . Design and Analysis

2 . 1 ER diagram and relational schema



1. Consumer

```
DROP TABLE IF EXISTS consumer;

CREATE TABLE consumer(
consumer_id INT(20) UNSIGNED  NOT NULL,
package_id INT(20) UNSIGNED,
plant_id INT(20) UNSIGNED
/* FOREIGN key(plant_id) REFERENCES plant(plant_id),
FOREIGN KEY(package_id) REFERENCES package(package_id) */
);
```

For consumer, consumer_id represents each consumer is unique. Package_id represents consumer's requirement. Plant_id represents their required plant.

2. Plant

```sql
DROP TABLE IF EXISTS plant;

CREATE TABLE plant(
plant_id INT(20) UNSIGNED  NOT NULL,
machine_id INT(20) UNSIGNED
/* FOREIGN KEY(machine_id) REFERENCES machine(machine_id) */
);
```

For plant, plant_id represents their uniqueness. Machine_id represents the machine in the plant. Because every machine is unique in machine table, so we use machine_id to represents the machine type and machine number in a plant.

3. Package

```sql
DROP TABLE IF EXISTS package;

CREATE TABLE package(
package_id INT(20) UNSIGNED  NOT NULL,
chip_id INT(20) UNSIGNED NOT NULL
/* FOREIGN KEY(chip_id) REFERENCES chip(chip_id) */

);
```

For package, package_id represents their uniqueness. Chip_id is very much like the same in plant(machine_id). Because every chip is unique in chip table, so we use chip_id to represents the chip type and chip number in a package.

4. Chip

```sql
DROP TABLE IF EXISTS chip;

CREATE TABLE chip(
chip_id INT(20) UNSIGNED  NOT NULL,
chip_type_id INT(20) UNSIGNED NOT NULL
/* FOREIGN KEY(chip_type_id) REFERENCES chip_type(chip_type_id) */
);
```

For chip, chip_id represents their uniqueness. Chip_type_id represents their chip type. It connects to table chip_type.

5. Chip_type

```
DROP TABLE IF EXISTS chip_type;

CREATE TABLE chip_type(

chip_type_id INT(20) UNSIGNED  NOT NULL,
operation_id INT(20) UNSIGNED NOT NULL,
operation_order INT(20) UNSIGNED NOT NULL DEFAULT 0
/* FOREIGN KEY(operation_id) REFERENCES operation(operation_id) */
);
```

For chip_type, chip_type_id represents their uniqueness. Because each chip_type need to be produced using specific operation and order, so operation_id and operation_order represents the operation type and where they are in a production activity.

6. Operation

```
DROP TABLE IF EXISTS operation;

CREATE TABLE operation(
operation_id INT(20) UNSIGNED  NOT NULL,
operation_type_id INT(20) UNSIGNED NOT NULL
/* FOREIGN KEY(operation_type_id) REFERENCES operation_type(operation_type_id) */
);
```

For operation, operation_id represents their uniqueness. Because each operation has a operation type, so it also needs an identifier like operation_type_id.

7. Operation type

```
DROP TABLE IF EXISTS operation_type;

CREATE TABLE operation_type(
operation_type_id INT(20) UNSIGNED  NOT NULL,
times INT(20),
expense INT(20)
);
```

For operation_type, operation_type_id represents their uniqueness. Each operation type also contains its time and expense.

8. Machine

```
DROP TABLE IF EXISTS machine;

CREATE TABLE machine(
machine_id INT(20) UNSIGNED  NOT NULL,
machine_type_id INT(20) UNSIGNED NOT NULL,
statuis int (20)
/* FOREIGN KEY(machine_type_id) REFERENCES machine_type(machine_type_id) */
);
```

For machine, machine_id represents their uniqueness. Each machine has some machine_type, and their occupation situation can be presented through status.

9. Machine type

```
DROP TABLE IF EXISTS machine_type;

CREATE TABLE machine_type(
machine_type_id INT(20) UNSIGNED  NOT NULL,
operation_type_id INT(20) UNSIGNED NOT NULL
/* FOREIGN key(operation_type_id) REFERENCES operation_type(operation_type_id) */
);
```

For machine_type, machine_type_id represents their uniqueness. Each machine type can produce some operation type.

10. Record

```
drop table if exists record;

create table record(
    start_time INT(20),
    end_time int(20),
    expense int(20)
) ;
```

For record, start_time,end_time and expense of some package is recorded. So that we can make conclusions about the start time and end time of a new package, including the expense it takes.

3 . Implementation

```sql
-- 插入数据

insert into operation_type values
(0,0,0),
(1,6,10),
(2,66,20),
(3,60,666),
(4,20,50),
(5,20,10),
(6,20,20),
(7,30,40),
(8,40,30);

insert into operation values
(0,0),
(1,1),
(2,2),
(3,3),
(4,4),
(5,5),
(6,6),
(7,7),
(8,8);


insert into machine_type values
(0,0),
(1,1),
(2,2),
(3,3),
(4,4),(4,5),
(5,5),(5,7),
(6,6),(6,8),
(7,7),(7,4),(7,6),
(8,8),(8,4),(8,5),(8,7);

insert into machine values
(1,1,0),(2,1,0),(3,1,0),
(4,2,0),(5,2,0);


insert into plant VALUES
(0,0),
(1,1),(1,2),(1,3),
(2,4),(2,5);

insert into chip_type values -- id op_id op_order
(0,0,0),
(1,1,0),(1,2,0), -- operation type id = 1
(2,2,0),
(3,3,0),(3,4,1),
(4,5,0),(4,4,1),
(5,8,0),(5,7,1);
```

```
insert into chip values
(0,0),
(1,1),
(2,1),
(3,1),
(4,2),
(5,3),
(6,3);

insert into record values(0,0,0);

insert into package values
(0,0),
(1,1),(1,2),(1,3),(1,4), -- type1
(2,4),(2,5);

insert into consumer values
-- (1,0,0),
-- (2,1,1);
(1,1,1),
(2,2,1);
```

4 . Function

```
DROP FUNCTION IF EXISTS get_chip_type_time;

DELIMITER //
CREATE FUNCTION get_chip_type_time(cid INT)
RETURNS INT(20)
DETERMINISTIC
BEGIN
DECLARE result int;
select sum(times) into result from operation_type
where operation_type_id != 0 and operation_type_id
in (select operation.operation_type_id from operation where operation_id in (select operation_id from chip_type where chip_type_id = cid));
RETURN result;
END//

DELIMITER ;
```

Function get_chip_type_time is used to get the time a chip type need to produce. We select those chip that in chip_type and find their time in operation_type by chip_type_id. Then sum them up to have the result.

```
DROP FUNCTION IF EXISTS get_chip_type_expense;

DELIMITER //
CREATE FUNCTION get_chip_type_expense(cid INT)
RETURNS INT(20)
DETERMINISTIC
BEGIN
DECLARE result int;
select sum(expense) into result from operation_type
where operation_type_id != 0 and operation_type_id
in (select operation.operation_type_id from operation where operation_id in (select operation_id from chip_type where chip_type_id = cid));
RETURN result;
END//

DELIMITER ;
```

Function get_chip_type_expense is used to get the expense a chip type need to produce. The method is very much like get_chip_type_time.

```
/* select count(*) from (select chip_type_id from chip where chip_id in (select package.chip_id from package where package_id = 1) ) as test; */

DROP procedure IF EXISTS get_package_time;
DELIMITER //
create procedure get_package_time(pid int)
BEGIN
    -- 声明变量
    DECLARE cid INT;
    DECLARE done INT default 0;
    DECLARE tmp INT ;
-- 创建游标，并设置游标所指的数据（这里设置ID不为1是因为ID为1的是总的大类）
    DECLARE totaltimes int default 0;
    DECLARE cur CURSOR for
        select chip_type_id from chip where chip_id in (select package.chip_id from package where package_id = pid);
-- 游标执行aaaaaaa完，即遍历结束。设置done的值为1
    DECLARE CONTINUE HANDLER for not FOUND set done = 1;
-- 开启游标a
    open cur;
    update record set start_time = end_time;
-- 执行循环
    posLoop:
    LOOP
        -- 如果done的值为1，即遍历结束，结束循环
        FETCH cur INTO tmp;
        IF done = 1 THEN
            /* select totaltimes from record; */
            /* update end_time =  totaltimes from record; */
            update record set end_time = start_time + totaltimes;
            /* update start_time  = start_time + end_time from record; */
            /* update record set start_time = start_time + end_time; */
            LEAVE posLoop;
-- 注意，if语句需要添加END IF结束IF
        END IF;
        SET totaltimes = totaltimes + get_chip_type_time(tmp);

-- 从游标中取出cid
-- 关闭循环
    END LOOP posLoop;
-- 关闭游标
    CLOSE cur;
-- 关闭分隔标记
END //
DELIMITER ;
```

Function get_package_time is used to get the total time of those chips that in the package.

We used a cursor to repeatedly go through the chips in package. And get their time through get_chip_type_time. Then sum them into ‘totaltimes’ and update those information to table record.

```
DROP procedure IF EXISTS get_package_expense;
DELIMITER //
create procedure get_package_expense(pid int)
BEGIN
    -- 声明变量
    DECLARE cid INT;
    DECLARE done INT default 0;
    DECLARE tmp INT ;
-- 创建游标，并设置游标所指的数据（这里设置ID不为1是因为ID为1的是总的大类）
    DECLARE totaltimes int default 0;
    DECLARE cur CURSOR for
        select chip_type_id from chip where chip_id in (select package.chip_id from package where package_id = pid);
-- 游标执行aaaaaaa完，即遍历结束。设置done的值为1
    DECLARE CONTINUE HANDLER for not FOUND set done = 1;
-- 开启游标a
    open cur;

-- 执行循环
    posLoop:
    LOOP
        -- 如果done的值为1，即遍历结束，结束循环
        FETCH cur INTO tmp;
        IF done = 1 THEN
            /* select totaltimes from record; */
            /* update end_time =  totaltimes from record; */
            update record set expense = totaltimes;
            /* update start_time  = start_time + end_time from record; */
            /* update record set start_time = start_time + end_time; */
            LEAVE posLoop;
-- 注意，if语句需要添加END IF结束IF
        END IF;
        SET totaltimes = totaltimes + get_chip_type_expense(tmp);

-- 从游标中取出cid
-- 关闭循环
    END LOOP posLoop;
-- 关闭游标
    CLOSE cur;
-- 关闭分隔标记
END //
DELIMITER ;
```

Function get_package_expense is very much like get_package_time, except for expense.

```
DROP procedure IF EXISTS reset;

DELIMITER //
CREATE procedure reset( )
DETERMINISTIC
BEGIN
update record set start_time = 0;
update record set end_time = 0;
update record set expense = 0;
END//

DELIMITER ;
```

Function reset is used to reset table record. Since we repeatedly add things into record, it needs to be reset.

```
DROP procedure IF EXISTS get_time_expense;

DELIMITER //
CREATE procedure get_time_expense(ptid INT)

DETERMINISTIC
BEGIN


set @i := 1;
while @i <= ptid DO
    call get_package_time(@i);
    call get_package_expense(@i);
    set @i := @i + 1;
end while;
select * from record;
call reset();

END//

DELIMITER ;
```

Function get_time_expense is used to get the start_time, end_time, expense of a particular package. Like:

```
mysql> call get_time_expense(1);
+------------+----------+---------+
| start_time | end_time | expense |
+------------+----------+---------+
|          0 |      282 |     110 |
+------------+----------+---------+
1 row in set (0.01 sec)

Query OK, 1 row affected (0.01 sec)
```

```
mysql> call get_time_expense(2);
+------------+----------+---------+
| start_time | end_time | expense |
+------------+----------+---------+
|        282 |      428 |     736 |
+------------+----------+---------+
1 row in set (0.01 sec)

Query OK, 1 row affected (0.01 sec)

mysql>
```

Division of labor and contribution of each team member: Shown below.

Team member	Task	Total Contribution
Chen Yiwen	Code,Reference,discussion,e-r-diagram	30%
Luo Kaicheng	Code,Reference,discussion,e-r-diagram	30%
Zhu Jiajun	Discussion,support	8%
Zhao Dule	Discussion,support	8%
Long Jiyuan	Discussion,support	16%
Cao Xiao	Discussion,support	8%

Presentation link：
https://www.bilibili.com/video/BV1SA411D7fE/?vd_source=6edfc1fd386df970e76e5f924c52a035