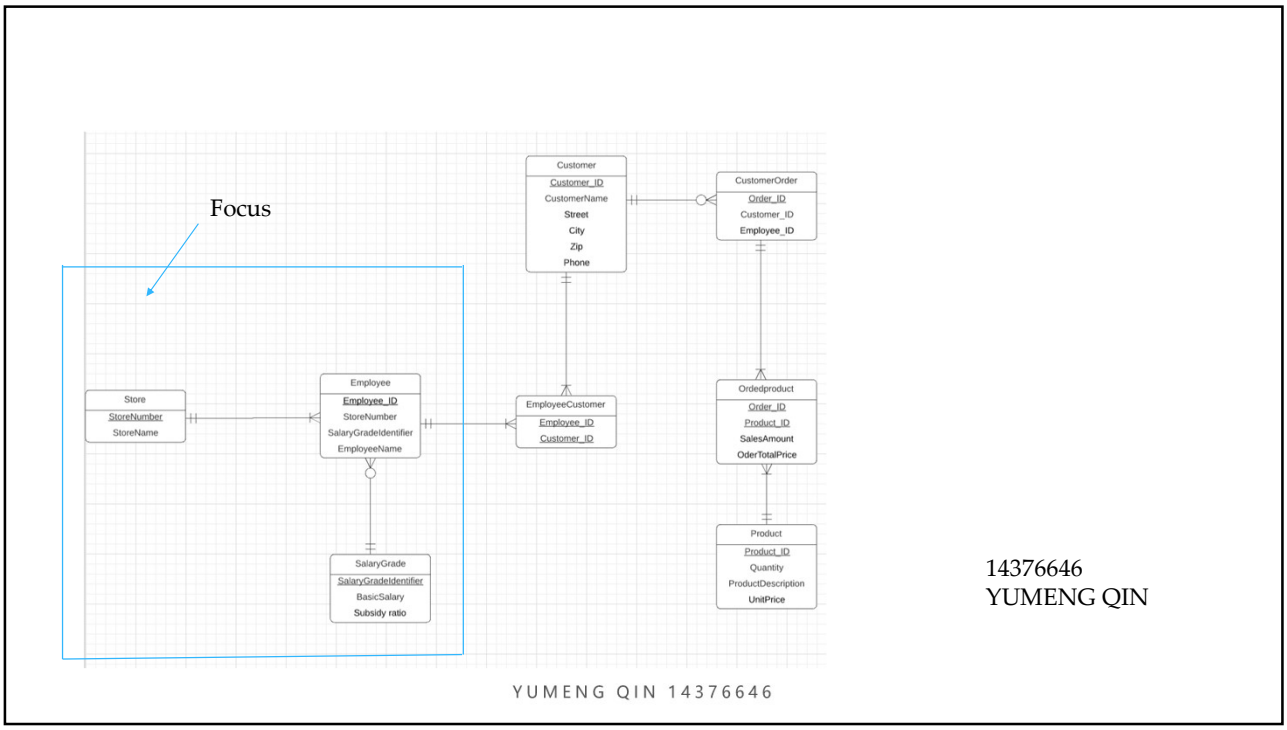

Focus
Customer
Customer_ID
CustomerName
Street
City
Zip
Phone
CustomerOrder
Order_ID
Customer_ID
Employee_ID
Store
StoreNumber
StoreName
Employee
Employee_ID
StoreNumber
SalaryGradeIdentifier
EmployeeName
EmployeeCustomer
Employee_ID
Customer_ID
Ordedproduct
Order_ID
Product_ID
SalesAmount
OderTotalPrice
Product
Product_ID
Quantity
ProductDescription
UnitPrice
SalaryGrade
SalaryGradeIdentifier
BasicSalary
Subsidy ratio
14376646
YUMENG QIN
YUMENG QIN 14376646

1


Agenda
01 / Overview
02 / Entity Relationship
03 / SQL Queries Retrieval
04 / Data Integrity
05 /ON DELETE RESTRICT&ON DELETE CASCADE
06 /The use of a view in my SQL
YUMENG QIN 14376646

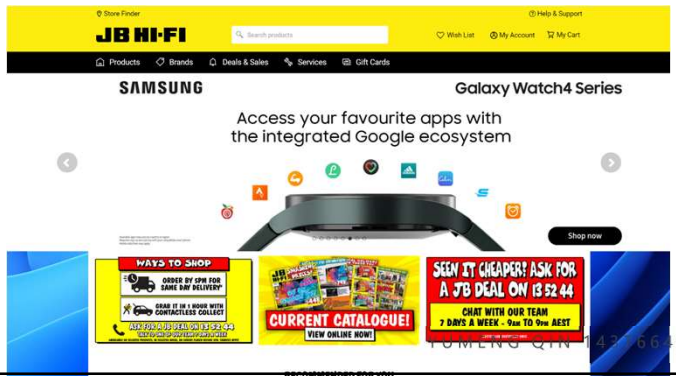
2

01 Overview

part one

3

Goal: JB HI-FI have many store in Australia. Through different entities, we can calculate different employee's salary according to their individual performance. This goal can be achieved by SQL Queries. Also, for each employee, we can see which type of product they are good at to sell. For this reason, we can allocate different employee to sell product which they are good at.



JB HI-FI offer the world's leading brands of Computers, Tablets, TVs, Cameras, Hi-Fi, Speakers, Home Theatre, Portable Audio, the largest range of games, recorded music, DVD music, Blu-Ray and DVD movies and TV shows and stacks more all at cheap prices! Everything you're after is available in one of our stores or online. JB Hi-Fi has it all - best brands, huge range, cheapest prices, convenient locations, but most importantly genuine personal service from our experienced specialist staff.

4

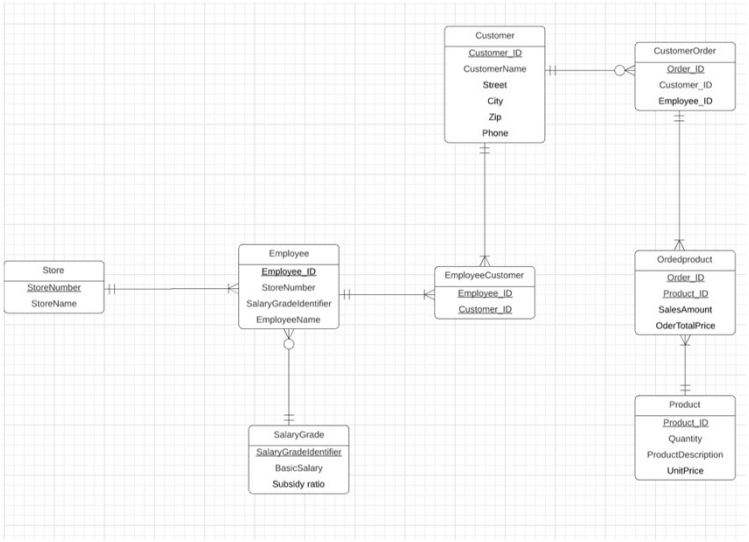
02 Entity Relationship Diagram

part two

YUMENG QIN 14376646

5

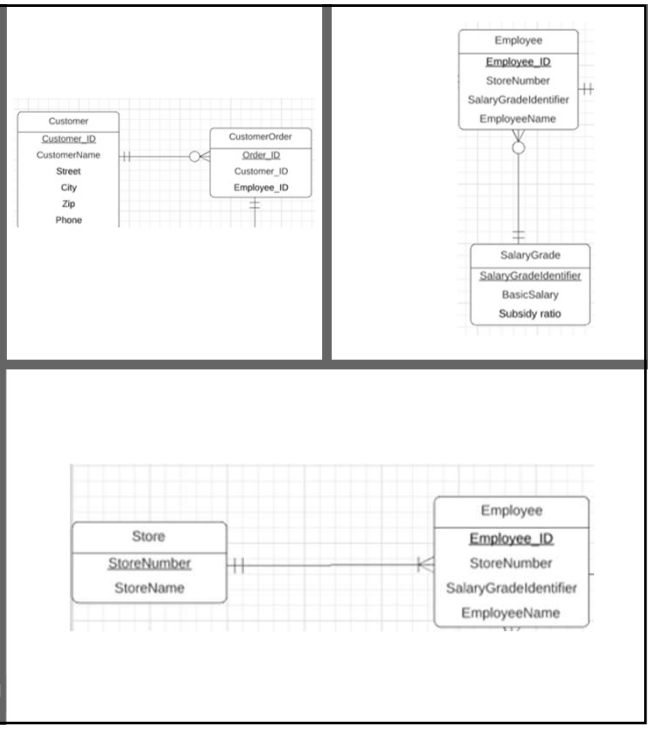
JIBIHIFI



YUMENG QIN 14376646

6

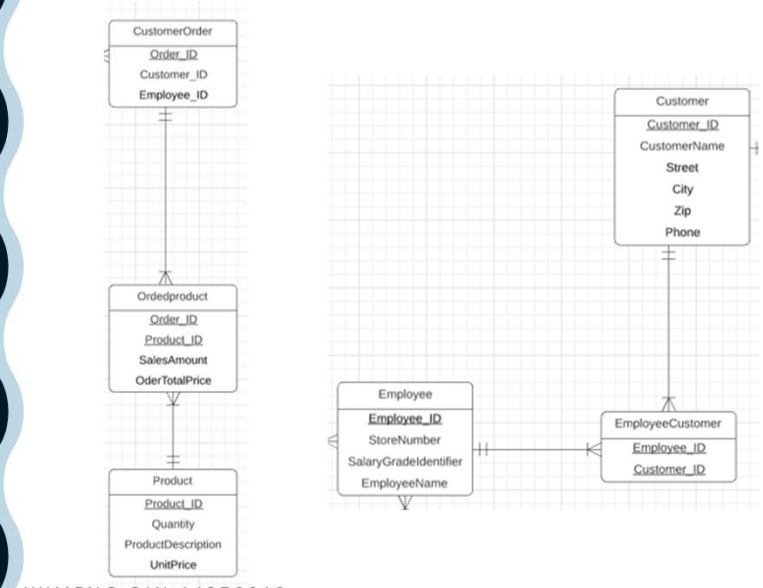
- One to Many



YUMENG QI

7

- Many-Many



YUMENG QIN 14376646

8

9

A simple query of a single table

Goal: find out the amount of employee in each store.

```
postgres=# select Storenumber, count(storenumber) as staffsamount
from employee group by storenumber
order by storenumber ;
```

```
(2 rows)

postgres=# select Storenumber, count(storenumber) as staffsamount from employee group by storenumber order by storenumber ;
 storenumber | staffsamount
-------------+--------------
           1 |            8
           2 |            9
(2 rows)
```


10

A query which uses the words "natural join"

If we want to know each transaction details, we can natural join order entity and ordered product entity. After using this way, we can forward use different queries to get different result through group by query.

select * from CustomerOrder natural join OrderProduct;

```
postgres=# select * from CustomerOrder natural join OrdedProduct;
 customerorder_id | customer_id | employee_id | product_id | salesamount | ordertotalprice
------------------+-------------+-------------+------------+-------------+-----------------
            10001 |         201 |         101 |         11 |           1 |            1499
            10001 |         201 |         101 |         13 |           1 |            1199
            10002 |         202 |         102 |         12 |           1 |             499
            10002 |         202 |         102 |         14 |           1 |             899
            10002 |         202 |         102 |         15 |           1 |            1089
            10003 |         203 |         103 |         13 |           1 |            1199
            10004 |         204 |         105 |         14 |           1 |             899
            10004 |         204 |         105 |         11 |           2 |            2998
            10004 |         204 |         105 |         15 |           3 |            3267
            10005 |         205 |         105 |         15 |           1 |            1089
(10 rows)
```

YUMENG QIN 14376646

11

A query involving a “Group by”, also with a “HAVING”

Goal: find out employee who serve equal or more than 2 customer.
Meaningful: if we define an outstanding employee is who serve equal or more than 2 employee, we can find out outstanding employees.

postgres=# select employee_id from CustomerOrder natural join OrdedProduct group by employee_id having count(customer_id) >=2 ;

```
postgres=# select employee_id from CustomerOrder natural join OrdedProduct group by employee_id having count(customer_id) >=2 ;
 employee_id
-------------
         101
         105
         102
(3 rows)
```

YUMENG QIN 14376646

12

The cross product equivalent to the "natural join" query above

```
select employeename,storename
from employee,store
where store.storenumber = employee.storenumber
order by storename;
```

Goal: find out every employee work in which store

```
employeename |      storename
-------------+---------------------
Crystal      | JB Hi-Fi City
John         | JB Hi-Fi City
Jennifer     | JB Hi-Fi City
Jo           | JB Hi-Fi City
Martina      | JB Hi-Fi City
Nan          | JB Hi-Fi City
Pasty        | JB Hi-Fi City
Nina         | JB Hi-Fi City
Daisy        | JB Hi-Fi City
Deb          | JB Hi-Fi Leichhardt
Jenny        | JB Hi-Fi Leichhardt
Anna         | JB Hi-Fi Leichhardt
Bob          | JB Hi-Fi Leichhardt
Chris        | JB Hi-Fi Leichhardt
Julia        | JB Hi-Fi Leichhardt
Celia        | JB Hi-Fi Leichhardt
Cheryl       | JB Hi-Fi Leichhardt
```

YUMENG QIN 14376646

13

The cross product equivalent to the "natural join" query above

- postgres=# select employeesales.employee_id,personalperformance,basicsalary,subsidyratio,personalperformance*subsidyratio+basicsalary*40 as salary from employeesales natural join employee,salarygrade where salarygrade.salarygradeidentifier = employee.salarygradeidentifier;

Goal: we can calculate different employee's salary according to their individual performance.

```
postgres=# select employeesales.employee_id,personalperformance,basicsalary,subsidyratio,personalperformance*subsidyratio+basicsalary*40 as salary from employeesales natural join
employee,salarygrade
where salarygrade.salarygradeidentifier = employee.salarygradeidentifier;
 employee_id | personalperformance | basicsalary | subsidyratio | salary
-------------+---------------------+-------------+--------------+---------
         101 |               14637 |          45 |          0.2 |  4727.4
         102 |               14637 |          28 |          0.3 |  5511.1
         103 |               14637 |          40 |          0.2 |  4527.4
         104 |               43911 |          35 |          0.3 | 14573.3
         105 |               58548 |          28 |          0.3 | 18684.4
(5 rows)
```

YUMENG QIN 14376646

14

A query which uses a sub query

Goal: find out which product has the minimize stock.
Meaning: manager can ask employee to recommend other product instead of minimized stock product to customer.

postgres=# select product_id, quantity from product where quantity <= (select min(quantity) from product) ;

```
postgres=# select product_id, quantity from product where quantity <= ( select min(quantity) from product) ;
 product_id | quantity
------------+----------
         11 |      501
(1 row)
```

YUMENG QIN 14376646

15

A cross product which cannot be implemented using the words "natural join" (e.g. self join)

Goal: According to self join, we can see different sales's manager with their store number

postgres=# select A.EmployeeName as sales, B.EmployeeName as salesmanager, A.storenumber from Employee A,Employee B where A.Bossnumber = B.Employee_id;

```
postgres=# select A.EmployeeName as sales, B.EmployeeName as salesmanager, A.storenumber
from Employee A,Employee B
where A.Bossnumber = B.Employee_id;
   sales   | salesmanager | storenumber
-----------+--------------+-------------
 Daisy     | John         |           2
 Crystal   | John         |           2
 Jenny     | Julia        |           1
 Anna      | Julia        |           1
 Bob       | Jenny        |           1
 Chris     | Jenny        |           1
 Deb       | Jenny        |           1
 Celia     | Anna         |           1
 Cheryl    | Anna         |           1
 Jennifer  | John         |           2
 Jo        | Crystal      |           2
 Martina   | Crystal      |           2
 Nan       | Jo           |           2
 Pasty     | Jo           |           2
 Nina      | Jo           |           2
(15 rows)
```

YUMENG QIN 14376646

16

17

Check statement

```
postgres(#   SalaryGradeIdentifier   integer     NOT NULL,
postgres(#   BasicSalary      integer,
postgres(#   SubsidyRatio    Decimal,
postgres(#   CONSTRAINT SalaryGradePK PRIMARY KEY (SalaryGradeIdentifier),
postgres(#   CONSTRAINT di_table_SalaryGrade_SubsidyRatio CHECK
postgres(#              ((SubsidyRatio >= 0) AND (SubsidyRatio <=1))
postgres(# );
CREATE TABLE
postgres=#
```

YUMENG QIN 14376646

18

```
postgres=# CREATE TABLE Product
postgres-#  (
postgres(#   Product_ID             integer    NOT NULL,
postgres(#   Quantity               integer,
postgres(#   ProductDescription     text,
postgres(#   UnitPrice              integer,
postgres(#   CONSTRAINT ProductPK PRIMARY KEY (Product_ID),
postgres(# CONSTRAINT di_table_Product_UnitPrice CHECK
postgres(#               (UnitPrice >= 0) ,
postgres(# CONSTRAINT di_table_Product_Quantity CHECK
postgres(#               (Quantity >= 0)
postgres(# );
CREATE TABLE
postgres=#
```

Check statement

YUMENG QIN 14376646

19

- Check statement

```
postgres=# CREATE TABLE Employee
postgres-# (
postgres(#   Employee_ID           integer,
postgres(#   StoreNumber      integer,
postgres(#   SalaryGradeIdentifier integer,
postgres(#   EmployeeName          text    NOT NULL,
postgres(#
postgres(#   CONSTRAINT EmployeePK PRIMARY KEY (Employee_ID),
postgres(#   FOREIGN KEY (StoreNumber)
postgres(#   REFERENCES Store,
postgres(#   FOREIGN KEY (SalaryGradeIdentifier)
postgres(#   REFERENCES SalaryGrade,
postgres(#   CONSTRAINT di_table_Employee_StoreNumber CHECK
postgres(#               (StoreNumber <= 87)
postgres(#
postgres(# );
CREATE TABLE
postgres=#
```

YUMENG QIN 14376646

20

21

One delete restrict

```
postgres=# delete from Customer ;
ERROR:  update or delete on table "customer" violates foreign key constraint "employeecustomer_customer_id_fkey" on table "employeecustomer"
DETAIL:  Key (customer_id)=(201) is still referenced from table "employeecustomer".
postgres=# delete from Employee ;
ERROR:  update or delete on table "employee" violates foreign key constraint "employeecustomer_employee_id_fkey" on table "employeecustomer"
DETAIL:  Key (employee_id)=(101) is still referenced from table "employeecustomer".
postgres=# delete from Employee where Employee_id = 101 ;
ERROR:  update or delete on table "employee" violates foreign key constraint "employeecustomer_employee_id_fkey" on table "employeecustomer"
DETAIL:  Key (employee_id)=(101) is still referenced from table "employeecustomer".
```

YUMENG QIN 14376646

22

After I add **ON DELETE CASCADE**

- On delete Cascade

```
CREATE TABLE EmployeeCustomer
(
  Employee_ID            integer   NOT NULL,
  Customer_ID            integer   NOT NULL,
  CONSTRAINT EmployeeCustomerPK PRIMARY KEY (Employee_ID,Customer_ID),
  FOREIGN KEY (Employee_ID)
  REFERENCES Employee ON DELETE CASCADE,
  FOREIGN KEY (Customer_ID)
  REFERENCES Customer ON DELETE CASCADE
);
```

```
postgres=# select * from employee ;
 employee_id | storenumber | salarygradeidentifier | employeename
-------------+-------------+-----------------------+--------------
         101 |           1 |                     4 | Julia
         102 |           2 |                     1 | John
         103 |           2 |                     3 | Daisy
         104 |           2 |                     2 | Crystal
         105 |           1 |                     1 | Jenny
(5 rows)
```

```
postgres=# delete from store where storenumber = 1 ;
DELETE 1
postgres=# select * from employee ;
 employee_id | storenumber | salarygradeidentifier | employeename
-------------+-------------+-----------------------+--------------
         102 |           2 |                     1 | John
         103 |           2 |                     3 | Daisy
         104 |           2 |                     2 | Crystal
(3 rows)
```

YUMENG QIN 14376646

23

06 the use of a view in my SQL

YUMENG QIN 14376646

24

The use of a view in SQL

Goal: build a new 'employeesales' entity to show each employee's sales performance

```
postgres=# create view employeesales as
postgres-# select employee_id, SUM(ordertotalprice) as PersonalPerformance
postgres-# from customerorder, ordedproduct
postgres-# group by employee_id ;
CREATE VIEW
postgres=#
postgres=# select * from employeesales;
 employee_id | personalperformance
-------------+--------------------
         101 |               14637
         103 |               14637
         104 |               43911
         105 |               58548
         102 |               14637
(5 rows)
```

YUMENG QIN 14376646