Uppsala University

Database Design I

1DT072

AltOnline Database Project

Author:
Group IT7
Granroth Arvid
Janse van Rensburg Louis
Tamang Sujata

December 12, 2017

Task 1: ER Model

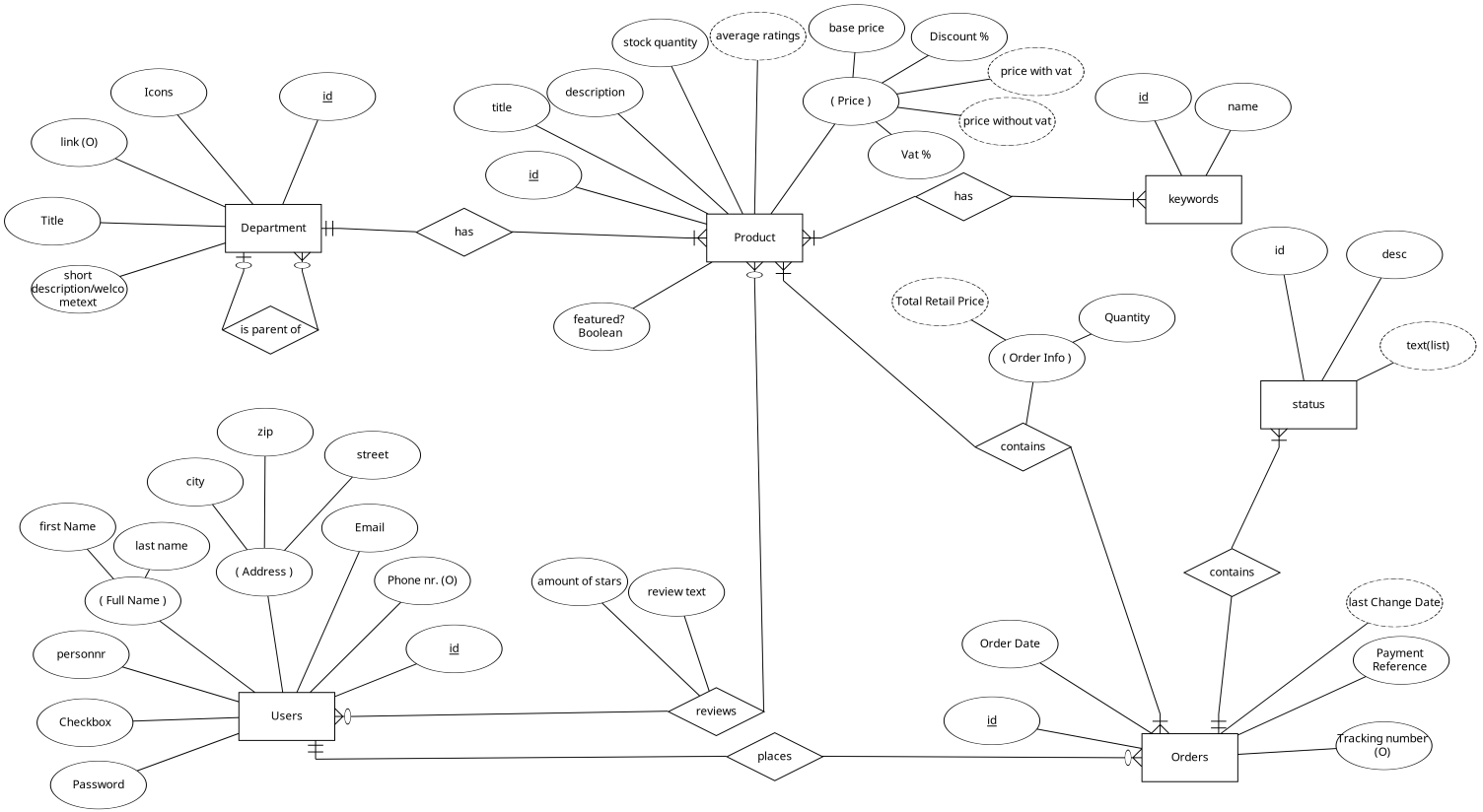

Figure 1: ER diagram representing attributes of, and relationships between, entities Department, Product, User, Order and Keyword.

Task 2: Converting ER Model to Relational Database

The code below creates a relational database with the following tables:

```
+------------------------------+
| Tables_in_fall17_project_it7 |
+------------------------------+
| Department                   |
| Keyword_Products             |
| Keywords                     |
| Order_Products               |
| Orders                       |
| Product                      |
| Status                       |
| User_Product                 |
| Users                        |
+------------------------------+
```

Code for creating the tables (found in the file `Task2_creatingTable.sql`), is given here:

Populating Database

```
CREATE TABLE Department(
  id int NOT NULL PRIMARY KEY,
  icons varchar(255) NOT NULL,
  link varchar(255) NOT NULL,
  title varchar(255) NOT NULL UNIQUE,
  description text NOT NULL,
  parentId int,
  FOREIGN KEY (parentId) REFERENCES Department(id)
  );

CREATE TABLE Product(
  id int NOT NULL PRIMARY KEY,
  title varchar(255) NOT NULL UNIQUE,
  featured bit NOT NULL,
  description text NOT NULL,
  stockQuantity int NOT NULL,
  price_BasePrice decimal(10,2) NOT NULL,
  price_Discount decimal(10,2) NOT NULL,
  price_Vat decimal(10,2) NOT NULL,
    deptID int NOT NULL,
    FOREIGN KEY (deptID) REFERENCES Department (id)
  );

CREATE TABLE Users(
  id int NOT NULL PRIMARY KEY,
  phoneNumber varchar(16),
  email varchar(255) NOT NULL,
  address_city varchar(255) NOT NULL,
  address_zip int(7) NOT NULL,
  address_street varchar(255) NOT NULL,
  fullName_FirstName varchar(255) NOT NULL,
  fullName_LastName varchar(255) NOT NULL,
```

```
  personNr varchar(12),
  checkBox bit NOT NULL,
  password varchar(255) NOT NULL
  );

CREATE TABLE User_Product(
  uid int NOT NULL,
  pid int NOT NULL,
  review_stars int NOT NULL,
  review_text text NOT NULL,
  Constraint uid_FK FOREIGN KEY (uid) REFERENCES Users (id),
  Constraint pid_FK FOREIGN KEY (pid) REFERENCES Product (id),
  Constraint puid_PK PRIMARY KEY (uid, pid)
  );

CREATE TABLE Keywords(
  keyword_id int NOT NULL PRIMARY KEY,
  kname varchar(255) NOT NULL UNIQUE
  );

CREATE TABLE Orders (
  id int NOT NULL PRIMARY KEY,
  orderDate varchar(10) NOT NULL,
  -- lastChangeDate varchar(10) NOT NULL, is a derived attribute
  paymentReference varchar(10) NOT NULL UNIQUE,
  trackingNumber varchar(10) UNIQUE,
  uid int NOT NULL UNIQUE,
  FOREIGN KEY (uid) REFERENCES Users (id)
  );
CREATE TABLE Status (
  statusID int NOT NULL PRIMARY KEY,
  status text NOT NULL,
  orderId int NOT NULL UNIQUE,
  FOREIGN KEY (orderId) REFERENCES Orders (id)
-- textlist i.e. history should be derived att
  );

CREATE TABLE Order_Products(
  orderID int NOT NULL,
  prod_ID int NOT NULL,
  Constraint oid_FK FOREIGN KEY (orderID) REFERENCES Orders (id),
  Constraint proid_FK FOREIGN KEY (prod_ID) REFERENCES Product (id),
  Constraint poid_PK PRIMARY KEY (orderID, prod_ID),
  quantity int NOT NULL

  -- derived attribute is total retail price
);

CREATE TABLE Keyword_Products(
  key_ID int NOT NULL,
  prod_ID int NOT NULL,
  Constraint kid_FK FOREIGN KEY (key_ID) REFERENCES Keywords (keyword_id),
    Constraint prodid_FK FOREIGN KEY (prod_ID) REFERENCES Product (id),
    Constraint kpid_PK PRIMARY KEY (key_ID, prod_ID)
);
```

Task2_creatingTable.sql

Task 3: Populating The Data Base

Insert statements used for populating the Database as found in “Task3_populatingDB.sql” are :

populating Database

```
-- populating dept.
```

```
INSERT INTO Department(id, icons, link, title, description, parentid)
VALUES(
   11,
   "/images/logo1.png",
     "www.shop.com",
     "Home of shop.com Clothes & Electronics",
     "We sell damn good clothes & electronics",
     NULL);

INSERT INTO Department(id, icons, link, title, description, parentid)
VALUES(
   1,
     "/images/logo1.png",
     "www.shop.com/clothes",
     "Clothes", "We sell damn good clothes",
     11);

INSERT INTO Department(id, icons, link, title, description, parentid)
VALUES(
   2,
     "/images/logo2.png",
     "www.shop.com/electronics"
     , "Electronics",
     "We sell damn good electronics",
     11);

INSERT INTO Department(id, icons, link, title, description, parentid)
VALUES(
   3,
     "/images/logo1.png",
     "www.shop.com/clothes/mensclothes",
     "Men's Clothes",
     "We sell damn good Men's Clothes",
     1);

INSERT INTO Department(id, icons, link, title, description, parentid)
VALUES(
   4,
     "/images/logo1.png",
     "www.shop.com/clothes/womensclothes",
     "Women's Clothes",
     "We sell damn good Women's Clothes",
     1);

INSERT INTO Department(id, icons, link, title, description, parentid)
VALUES(
   5,
     "/images/logo1.png",
     "www.shop.com/clothes/kidsclothes",
     "Kid's Clothes",
     "We sell damn good Kid's Clothes",
     1);

INSERT INTO Department(id, icons, link, title, description, parentid)
VALUES(
   6,
     "/images/logo1.png",
     "www.shop.com/clothes/fancyclothes",
     "Fancy Clothes",
     "We sell damn good Fancy Clothes",
     1);

INSERT INTO Department(id, icons, link, title, description, parentid)
VALUES(
   7,
     "/images/logo2.png",
     "www.shop.com/electronics/Computers and Tablets",
     "Computers and Tablets",
     "We sell damn good computer and tablets",
     2);

INSERT INTO Department(id, icons, link, title, description, parentid)
VALUES(
   8,
     "/images/logo2.png",
     "www.shop.com/electronics/Desktop ",
     "Desktop",
```

```
    "We sell damn good Desktop",
    2);

INSERT INTO Department(id, icons, link, title, description, parentid)
VALUES(
  10,
    "/images/logo2.png",
    "www.shop.com/electronics/Computers and Tablets",
    "Desktops",
    "We sell damn good computer and tablets",
    2);

INSERT INTO Department(id, icons, link, title, description, parentid)
VALUES(
  9,
    "/images/logo2.png",
    "www.shop.com/electronics/Phones ",
    "Phones",
    "We sell damn good Phones",
    2);


-- populating product
INSERT INTO Product(
  id, title, featured,
    description, stockQuantity,
    price_BasePrice, price_Discount,
    price_VAT, deptID)
VALUES(6, "Men's Jeans", 0,
  "Damn Good Men's Jeans",
    10000, 600, 10, 25, 3);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice, price_Discount,
    price_VAT, deptID)
VALUES(7, "Women's Jeans", 1,
  "Damn Good Women's Jeans",
  50000, 700, 30, 25, 4);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
  price_BasePrice, price_Discount,
    price_VAT, deptID)
VALUES(8, "Kid's Jeans", 0,
    "Damn Good Kid's Jeans",
    100, 400, 0, 25, 5);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
  price_BasePrice, price_Discount,
    price_VAT, deptID)
VALUES(9, "Men's Shirts", 0,
  "Damn Good Men's Shirts",
    1000, 300, 20, 25, 3);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice, price_Discount,
    price_VAT, deptID)
VALUES(10, "Women's Shirts", 0,
  "Damn Good Women's Shirts",
    6000, 200, 40, 25, 3);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice, price_Discount,
    price_vat, deptID)
VALUES(1, "iPhone X", 1,
  "We sell damn good iPhones",
    10, 3000, 15, 25, 2);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice, price_Discount,
    price_vat, deptID)
VALUES(2, "Samsung S8", 1,
```

```
156   "We sell damn good samsung s8",
        50, 2200, 5, 25, 2);
158
INSERT INTO Product(id, title, featured,
160   description, stockQuantity,
        price_BasePrice, price_Discount,
162     price_vat, deptID)
VALUES(3, "Lenovo K6", 0,
164   "We sell damn good lenovo k6",
        100, 1500, 15, 25, 2);
166
INSERT INTO Product(id, title, featured,
168   description, stockQuantity,
        price_BasePrice,price_Discount,
170     price_vat, deptID)
VALUES(4, "Nokia 360", 0,
172   "We sell damn good Nokia 360",
        80, 750, 12, 25, 2);
174
INSERT INTO Product(id, title, featured,
176   description, stockQuantity,
        price_BasePrice, price_Discount,
178     price_vat, deptID)
VALUES(5, "Sony Ericsson", 1,
180   "We sell damn good Ericsson",
        45, 2500, 20, 25, 2);
182
-- populating users
184 INSERT INTO Users(id, phoneNumber, email,
      address_city, address_zip,
186     address_street, fullName_FirstName,
        fullName_LastName, personNr,
188     checkBox, password)
    VALUES(1, "+467345060", "bobmarley@jammin.com",
190   "Kingston Town", 1234, "420 Rasta High Way",
        "Bob", "Marley", "420420BOB",
192     1, "rastamanvibrations");

194 INSERT INTO Users(id, phoneNumber, email,
      address_city, address_zip,
196     address_street, fullName_FirstName,
        fullName_LastName, personNr,
198     checkBox, password)
    VALUES(2, "+467345063", "ironlion@positivevibrations.com",
200   "Kingston Town", 1334, "420 Herb Way",
        "Iron", "Lion", "420420BOB",
202     1, "ironlikealioninzion");

204 -- populating user_products
    INSERT INTO User_Product(uid, pid, review_stars, review_text)
206 VALUES(1, 10, 5, "great shirt");

208 INSERT INTO User_Product(uid, pid, review_stars, review_text)
    VALUES(2, 6, 2, "the jeans are too small");
210
    -- note inserting into Orders, must be followed up by inserting which product is
         selected into Order Products.
212 INSERT INTO Orders(id, orderDate,
      paymentReference, trackingNumber, uid)
214 VALUES(1, "22/11/2017",
      "ref-1", "TN12345678", 1);
216
    INSERT INTO Orders(id, orderDate,
218   paymentReference, trackingNumber, uid)
    VALUES(2, "24/12/2017",
220   "ref-2", "TN54356781",2);

222 -- populating Order_Products
    INSERT INTO Order_Products(orderID, prod_ID, quantity)
224 VALUES(1, 4, 102);
    INSERT INTO Order_Products(orderID, prod_ID, quantity)
226 VALUES(1, 3, 203);
    INSERT INTO Order_Products(orderID, prod_ID, quantity)
228 VALUES(1, 6, 1000);
    INSERT INTO Order_Products(orderID, prod_ID, quantity)
230 VALUES(1, 1, 234);
    INSERT INTO Order_Products(orderID, prod_ID, quantity)
```

```
232 VALUES(1, 2, 2);
    INSERT INTO Order_Products(orderID, prod_ID, quantity)
234 VALUES(2, 9, 12);
    INSERT INTO Order_Products(orderID, prod_ID, quantity)
236 VALUES(2, 7, 23);
    INSERT INTO Order_Products(orderID, prod_ID, quantity)
238 VALUES(2, 5, 100);
    INSERT INTO Order_Products(orderID, prod_ID, quantity)
240 VALUES(2, 8, 34);
    INSERT INTO Order_Products(orderID, prod_ID, quantity)
242 VALUES(2, 10, 24);

244 -- populating keywords
    INSERT INTO Keywords(keyword_id, kname)
246 VALUES(1, "pants");
    INSERT INTO Keywords(keyword_id, kname)
248 VALUES(2, "jeans");
    INSERT INTO Keywords(keyword_id, kname)
250 VALUES(3, "kid's clothes");
    INSERT INTO Keywords(keyword_id, kname)
252 VALUES(4, "shirts");
    INSERT INTO Keywords(keyword_id, kname)
254 VALUES(5, "clothes");
    INSERT INTO Keywords(keyword_id, kname)
256 VALUES(6, "computers");
    INSERT INTO Keywords(keyword_id, kname)
258 VALUES(7, "phones");
    INSERT INTO Keywords(keyword_id, kname)
260 VALUES(8, "tablets");
    INSERT INTO Keywords(keyword_id, kname)
262 VALUES(9, "brands");
    INSERT INTO Keywords(keyword_id, kname)
264 VALUES(10, "smart phone");

266

    -- populating the many to many
268 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(1, 6);
270 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(1, 7);
272 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(1, 8);
274 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(2, 6);
276 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(2, 7);
278 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(2, 8);
280 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(3, 8);
282 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(4, 9);
284 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(4, 10);
286 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(5, 6);
288 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(5, 7);
290 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(5, 8);
292 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(5, 9);
294 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(5, 10);
296 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(6, 3);
298 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(6, 5);
300 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(7, 1);
302 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(7, 2);
304 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(7, 4);
306 INSERT INTO Keyword_Products(key_ID, prod_ID)
    VALUES(7, 5);
308 INSERT INTO Keyword_Products(key_ID, prod_ID)
```

```
VALUES(8, 1);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(8, 5);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(8, 3);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(9, 1);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(9, 2);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(9, 3);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(9, 4);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(9, 5);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(10, 1);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(10, 2);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(10, 3);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(10, 4);
INSERT INTO Keyword_Products(key_ID, prod_ID)
VALUES(10, 5);

-- more insert statements just to overpopulate (required for task5 - optimization
    section)
INSERT INTO Product(id, title, featured,
  description, stockQuantity,
  price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(12, "Dell XPS 15", 1,
  "We sell damn good Dell XPS laptops",
    100, 23000, 25, 25, 7);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(13, "HP Pavilion g6", 0,
  "We sell damn good HP Pavilion g6s",
    50, 5000, 30, 25, 7);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(14, "Black Shuttle", 0,
  "We sell damn good Black Shuttles",
    300, 5000, 12, 25, 8);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(15, "Ant Miner", 1,
  "We sell damn good Ant Miner",
    28, 30000, 7, 25, 8);
-- ********************************************************************************

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
  price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(16, "Lenovo", 0,
  "We sell damn good Lenovo",
    100, 1500, 28, 25, 7);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(17, "Surface Pro", 0,
  "We sell damn good Surface Pro",
    41, 1900, 14, 25, 7);

```

```
INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(18, "Razer", 1,
  "We sell damn good Razer",
    28, 30000, 16, 25, 7);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(19, "Alien ware", 1,
  "We sell damn good Alien ware",
  99, 30000, 10, 25, 7);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
  price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(20, "Chrome Book", 1,
  "We sell damn good Chrome Book",
    50, 1700, 25, 25, 7);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(21, "Asus", 1,
  "We sell damn good Asus",
    52, 1900, 23, 25, 7);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
    price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(22, "Acer Aspire", 1,
  "We sell damn good Acer Aspire",
    76, 2100, 21, 25, 7);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
  price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(23, "MacBook Air", 1,
  "We sell damn good MacBook Air",
    29, 3100, 5, 25, 7);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
  price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(24, "MSI", 1,
  "We sell damn good MSI",
    79, 1850, 21, 25, 7);

INSERT INTO Product(id, title, featured,
  description, stockQuantity,
  price_BasePrice,price_Discount,
    price_vat, deptID)
VALUES(25, "Dell Latitude", 1,
  "We sell damn good Dell Latitude",
    94, 2580, 12, 25, 7);
```

Task3_populatingDB.sql

Task 4: SQL Queries

Output for queries are as follows:

1) Welcome text for the homepage

```
+----------------------------------------+
| WelcomeMessage                         |
+----------------------------------------+
| We sell damn good clothes & electronics |
+----------------------------------------+
```

2) List of the top level departments with fields needed for the homepage: –

```
+----+-------------------+-------------------------+-------------+-------------------------------+----------+
| id | icons             | link                    | title       | description                   | parentId |
+----+-------------------+-------------------------+-------------+-------------------------------+----------+
|  1 | /images/logo1.png | www.shop.com/clothes    | Clothes     | We sell damn good clothes     |       11 |
|  2 | /images/logo2.png | www.shop.com/electronics | Electronics | We sell damn good electronics |       11 |
+----+-------------------+-------------------------+-------------+-------------------------------+----------+
```

3) List of the featured products with fields needed for the homepage

```
+----+--------------+----------+-----------------------------+---------------+-----------------+----------------+-----------+--------+
| id | title        | featured | description                 | stockQuantity | price_BasePrice | price_Discount | price_Vat | deptID |
+----+--------------+----------+-----------------------------+---------------+-----------------+----------------+-----------+--------+
|  1 | iPhone X     |          | We sell damn good iPhones   |            10 |         3000.00 |          15.00 |     25.00 |      2 |
|  2 | Samsung S8   |          | We sell damn good samsung s8 |            50 |         2200.00 |           5.00 |     25.00 |      2 |
|  5 | Sony Ericsson |          | We sell damn good Ericsson  |            45 |         2500.00 |          20.00 |     25.00 |      2 |
|  7 | Women's Jeans |          | Damn Good Women's Jeans     |         50000 |          700.00 |          30.00 |     25.00 |      4 |
+----+--------------+----------+-----------------------------+---------------+-----------------+----------------+-----------+--------+
```

4) Given a product, list all keyword-related products (The product selected was "Sony Ericsson", which has Product.id 5):

```
+------------+
| title      |
+------------+
| iPhone X   |
| Lenovo K6  |
| Nokia 360  |
| Samsung S8 |
+------------+
```

5) Given a department, list of all its products (title, short description, current retail price) with their average rating (selected department was "electronics", dept id = 2):

```
+----+---------------+-------------------------------+-----------------------------+-----------------+
| id | title         | avg(User_Product.review_stars) | description                 | price_BasePrice |
+----+---------------+-------------------------------+-----------------------------+-----------------+
|  1 | iPhone X      |                          NULL | We sell damn good iPhones   |         3000.00 |
|  2 | Samsung S8    |                          NULL | We sell damn good samsung s8 |         2200.00 |
|  3 | Lenovo K6     |                          NULL | We sell damn good lenovo k6  |         1500.00 |
|  4 | Nokia 360     |                          NULL | We sell damn good Nokia 360  |          750.00 |
|  5 | Sony Ericsson |                          NULL | We sell damn good Ericsson   |         2500.00 |
|  6 | Men's Jeans   |                        2.0000 | Damn Good Men's Jeans        |          600.00 |
|  7 | Women's Jeans |                          NULL | Damn Good Women's Jeans      |          700.00 |
|  8 | Kid's Jeans   |                          NULL | Damn Good Kid's Jeans        |          400.00 |
|  9 | Men's Shirts  |                          NULL | Damn Good Men's Shirts       |          300.00 |
| 10 | Women's Shirts |                        5.0000 | Damn Good Women's Shirts     |          200.00 |
+----+---------------+-------------------------------+-----------------------------+-----------------+
```

NOTE: only two products have been rated.

6) List of all products on sale sorted by the discount percentage (starting with the biggest discount):

id	title	price_Discount
10	Women's Shirts	40.00
7	Women's Jeans	30.00
9	Men's Shirts	20.00
5	Sony Ericsson	20.00
1	iPhone X	15.00
3	Lenovo K6	15.00
4	Nokia 360	12.00
6	Men's Jeans	10.00
2	Samsung S8	5.00
8	Kid's Jeans	0.00

7) List of all new orders sorted by the order date (id, order date, customer's name and the city, and the total price):

orderID	orderDate	fullName_LastName	fullName_FirstName	address_city	TotalPrice
2	24/12/2017	Lion	Iron	Kingston Town	302655.000000
1	22/11/2017	Marley	Bob	Kingston Town	1888875.000000

8) 10 best-selling products (in last 30 days):

bestsellers	prod_ID	title
1000	6	Men's Jeans
234	1	iPhone X
203	3	Lenovo K6
102	4	Nokia 360
100	5	Sony Ericsson
34	8	Kid's Jeans
24	10	Women's Shirts
23	7	Women's Jeans
12	9	Men's Shirts
2	2	Samsung S8

The code found in `Task4_queries.sql` is as follows:

caption

```
-- QUERIES

-- Welcome text for the home page
Select description AS WelcomeMessage
from Department
where Department.id = 11;

-- List of the top level departments with fields needed for the homepage
Select *
from Department
where Department.id = 1 OR Department.id = 2;

-- List of the featured products with fields needed for the homepage
Select *
from Product
where Product.featured = 1;

-- Given a product, list all keyword related products
-- NOTE: the chosen product here is 5: "Sony Ericsson" It returns 4 electronic products,
-- because they are all related by all keywords related to "Sony Ericsson"
CREATE VIEW view6 AS
SELECT key_ID
```

```
FROM Keyword_Products
WHERE prod_ID = 5;

CREATE VIEW view7 AS
SELECT prod_ID
FROM Keyword_Products
WHERE prod_ID IN (SELECT prod_ID from Keyword_Products where key_ID IN (SELECT key_ID
    from view6));

SELECT Product.title
FROM Product
WHERE id in (SELECT * from view7) AND NOT id = 5;

-- Given a department,
-- list of all its products (title, short description,
-- current retail price) with their average rating
SELECT Product.id, Product.title, avg(User_Product.review_stars), Product.description,
    Product.price_BasePrice
FROM Product
LEFT JOIN User_Product ON User_Product.pid = Product.id
GROUP BY Product.id, Product.title;

-- List of all products on sale sorted by the discount percentage (starting with the
    biggest discount)
SELECT id, title, price_Discount
FROM Product
ORDER BY price_Discount DESC;

-- List of all new orders sorted by the order date
CREATE VIEW view1 AS
Select Order_Products.orderID, Order_Products.prod_ID, Order_Products.quantity, Orders.
    orderDate, Orders.uid
FROM Order_Products
LEFT JOIN Orders ON Orders.id = Order_Products.orderID
GROUP BY Orders.id, Order_Products.prod_ID;

CREATE VIEW view2 AS
Select view1.orderID, view1.prod_ID, view1.quantity, view1.orderDate, view1.uid, Users.
    fullName_LastName, Users.fullName_FirstName, Users.address_city
FROM Users
LEFT JOIN view1 ON view1.uid = Users.id
GROUP BY view1.uid, view1.prod_ID, Users.id;

CREATE VIEW view3 AS
Select view2.orderID, view2.prod_ID, view2.quantity, view2.orderDate, view2.uid, view2.
    fullName_LastName, view2.fullName_FirstName, view2.address_city, Product.
    price_BasePrice, Product.price_Discount, Product.price_Vat
FROM Product
LEFT JOIN view2 ON view2.prod_ID = Product.id
GROUP BY view2.uid, view2.prod_ID, Product.id;

CREATE VIEW view4 AS
Select view3.orderID, sum(quantity*(price_BasePrice+price_BasePrice*(price_Vat*0.01)-
    price_BasePrice*(price_Discount*.01))) AS TotalPrice from view3 Group by orderID;

CREATE VIEW view5 AS
SELECT view3.orderID, view3.orderDate, view3.fullName_LastName, view3.fullName_FirstName
    , view3.address_city, view4.TotalPrice
FROM view4
LEFT JOIN view3 ON view3.orderID = view4.orderID
Group by orderID
Order By convert(view3.orderDate, date);

Select * from view5;

-- 10 best-selling products (in last 30 days)
Select sum(Order_Products.quantity) AS bestsellers, Order_Products.prod_ID, Product.
    title
from Product
LEFT JOIN Order_Products on Order_Products.prod_ID = Product.id
Group BY Order_Products.prod_ID ORDER BY sum(Order_Products.quantity) DESC limit 10;



drop view view1;
drop view view2;
```

```
drop view view3;
drop view view4;
drop view view5;
drop view view6;
drop view view7;
```

Task4_queries.sql

Task 5: Analysis and Optimization of SQL Queries using Indicies

[*Note* :] Where ever views are used to generate the Query, we analyze each SELECT statement of the views to create index on them.
The data we have is not sufficient to notice the performance difference as a result of indexing. However, we have created index where it seems useful for fast querying and assume that with large database, they might result in observable difference of performance measure.

Query 1: Welcome Text

The query for this includes WHERE clause with id. Since, Department.id is a primary key, it already has an index of type BTREE.

Query 2: List of the top level department

This Query is same as Query 1 which uses id in the WHERE clause.

Query 3 : List of the featured products

Here, since we make are trying to get the list of products from Product table WHERE featured is 1. We can create index on featured column to make the query faster.

```
CREATE INDEX ind_featured
  ON Product (featured);
```

Listing 1: Index on featured column

Query 4 : list of keyword related product

We have used two views to run this query and so we
For both views used, we have WHERE clause for $Keyword\_Products.prod\_ID$ that has to be the id of chosen product.
Therefore, we create index on $Keyword\_Products.prod\_ID$

```
CREATE INDEX ind_kp_pid
   ON Keyword_Products (prod_Id);
```

Query 5 :Given a department, list of all its products (title, short description, current retail price) with their average rating

As this query uses GROUP BY $Product.title$, we use index on Product.title.

```
CREATE INDEX prod_id_title
  ON Product (title);
```

Query 6: List of all products on sale sorted by the discount percentage (starting with the biggest discount)

There are two things to notice in this query. one is the use of WHERE and the other is ORDER BY. Both needs to search the table based on $Product.price_Discount$. Therefore, we create index on it as shown below :

```
CREATE INDEX prod_id_title
  ON Product (title);
```

Query 7 :List of all new orders sorted by the order date (id, order date, customer's name and the city, and the total price)

This query is a mix of views again. The views here uses $Order\_Products.prod\_ID$ to group by the result. The end result is supposed to be grouped by Orders.Date.
We create two index for this Query as below :

```
create index prodid
  ON Order_Products (prod_ID);
create index orderdate
  ON Orders (orderDate);
```

We can see time taken to return the rows drops to 0.038 seconds from 0.041 seconds even for only 3 rows. Although, it's insignificant, this can be a lot for larger database as mentioned earlier as well.

Query 8 : 10 best-selling products (in last 30 days)

Here, we create index on $Order\_Products.prod\_ID$ and $Order_products.quantity$ together so that finding quantity associated with that particular product id becomes easier. The index is created as below :

```
create index id_quan
  on Order_Products (prod_ID, quantity);
```

Additional Query: product with review larger than 3

The Query is as following :

```
SELECT title ,description ,review_stars
FROM User_Product
LEFT JOIN Product on User_Product.pid = Product.id
WHERE review_stars > 3 ;
```

The index is as following :

```
create index stars
  on User_Product (review_stars);
```

Additional Query:price from low to high, less than 2000

Query to sort products based on low to high price and is larger than 2000 is as following :

```
select *
  from Product
    where price_BasePrice < 2000
    group by price_BasePrice
    order by price_BasePrice ASC;
```

The index to optimize this query is as following :

```
create index price
on Product (price_BasePrice);
```

Task 6: FFDs, candidate keys, and NFs

The tables `Product, Users` and `Keyword_Products` are represented as follows:

`Product`:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
title	varchar(255)	NO	UNI	NULL	
featured	bit(1)	NO		NULL	
description	text	NO		NULL	
stockQuantity	int(11)	NO		NULL	
price_BasePrice	decimal(10,2)	NO		NULL	
price_Discount	decimal(10,2)	NO		NULL	
price_Vat	decimal(10,2)	NO		NULL	
deptID	int(11)	NO	MUL	NULL	

`Users`:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
phoneNumber	varchar(16)	YES		NULL	
email	varchar(255)	NO		NULL	
address_city	varchar(255)	NO		NULL	
address_zip	int(7)	NO		NULL	
address_street	varchar(255)	NO		NULL	
fullName_FirstName	varchar(255)	NO		NULL	
fullName_LastName	varchar(255)	NO		NULL	
personNr	varchar(12)	YES		NULL	
checkBox	bit(1)	NO		NULL	
password	varchar(255)	NO		NULL	

Keyword_Products:

```
+---------+---------+------+-----+---------+-------+
| Field   | Type    | Null | Key | Default | Extra |
+---------+---------+------+-----+---------+-------+
| key_ID  | int(11) | NO   | PRI | NULL    |       |
| prod_ID | int(11) | NO   | PRI | NULL    |       |
+---------+---------+------+-----+---------+-------+
```

Fully Functional Dependencies and Non-Fully Functional Dependencies

- For `Product`, the FFDs are such that all attributes are dependent on `Product.id`. That is,
  {`Product.id`} $\longrightarrow$ {`Product.*`}
- For `Users`, the FFDs are such that all attributes are dependent on `Users.id`. That is,
  {`Users.id`} $\longrightarrow$ {`Users.*`}
- For `Keyword_Products`, the non-fully functional dependencies are such that both `Keyword_Products.key_id` and `Keyword_Products.prod_id` depend on each other. That is,
  {`Keyword_Products.key_id`} $\longrightarrow$ {`Keyword_Products.prod_id`}
  AND
  {`Keyword_Products.prod_id`} $\longrightarrow$ {`Keyword_Products.key_id`}.
  Both attributes mutually map to multiple values, and so there is no fully functional dependency.

Candidate Keys

Given the FFDs and NFFDs above:

- `Product.id` is a candidate key for the table `Product` as it determines the value of all other attributes in `Product`.
- `Users.id` is a candidate key for the table `Users` as it determines the value of all other attributes in`Users`.

- `Keyword_Products.key_ID` and `Keyword_Products.prod_ID` together form a super-key for the table `Keyword_Products`.

Determining NFs of `Product`, `Users` and `Keyword_Products`

1NF

- `Product`, `Users` and `Keyword_Products` are already in 1NF, that is, they satisfy the conditions that attribute values are atomic and single-valued, and that there are no composite or multivalued attributes and there no nested relationships.

2NF

- `Product` and `Users` are in 2NF as each non-prime attribute is fully functionally dependent on their respective table's candidate key.
- `Keyword_Products` is also in 2NF as all attributes constitute the candidate key which is in fact a super-key.

3NF

- `Product`is not in 3NF as the the dependency relations are not trivial (i.e. `Product.deptID` has multiple occurrences), the candidate key `Product.id` is not a super-key, and `Product.deptID` is a non-prime attribute.
- `Users` is in 3NF as every attribute apart from the candidate key is prime.
- `Keyword_Products` is in 3NF as the all attributes comprise a super-key.

BCNF

- `Product` is not in BCNF as it is not in 3NF.
- `User` is in BCNF as the dependency relations are trivial.
- `Keyword_Products` is in BCNF as the all attributes comprise a super-key.

Task7: Python program which connects to the database

Program 1: list child departments

```
#!/usr/bin/python
# Database Design Project
# Group : IT7
#Task 7 - part 1

import MySQLdb

conn = MySQLdb.connect("back.db1.course.it.uu.se", "fall17_it7",
    "aaaCbaQp", "fall17_project_it7")
conn.autocommit(True)
cursor = conn.cursor()

while True:
    department_id = int(raw_input("Enter Department id (0 to
    exit): "))
    if department_id == 0:
        break

    cursor.execute("""
        SELECT id, title
        FROM Department
        WHERE parentId = %s
    """, (department_id,))

    for row in cursor:
        print "%4d | %32s" % row

cursor.close()
conn.close()
```

Listing 2: Lists all child departments

Program 2: Display and change product discount

```
#!/usr/bin/python
# Database Design Project
# Group : IT7
#Task 7 - part 2

import MySQLdb

conn = MySQLdb.connect("back.db1.course.it.uu.se", "fall17_it7",
    "aaaCbaQp", "fall17_project_it7")
conn.autocommit(True)
cursor = conn.cursor()

while True:
    product_id = int(raw_input("Enter Product id (0 to exit): ")
    )
    if product_id == 0:
        break

    cursor.execute("""
        SELECT id,title, price_Discount
        FROM Product
        WHERE id = %s
    """, (product_id,))

    for row in cursor:
        print "%4d | %32s | %4d"
        print "Change the discount value to : "
        changed_discount = int(raw_input("Enter the discount
    value(0 to exit)"))
        cursor.execute("""
                UPDATE Product
                SET price_Discount = %s
                WHERE id = %s
        """,(changed_discount,product_id))

cursor.close()
conn.close()
```

Listing 3: display and change discount