8. Databases and JDBC

1. Introduction to Databases

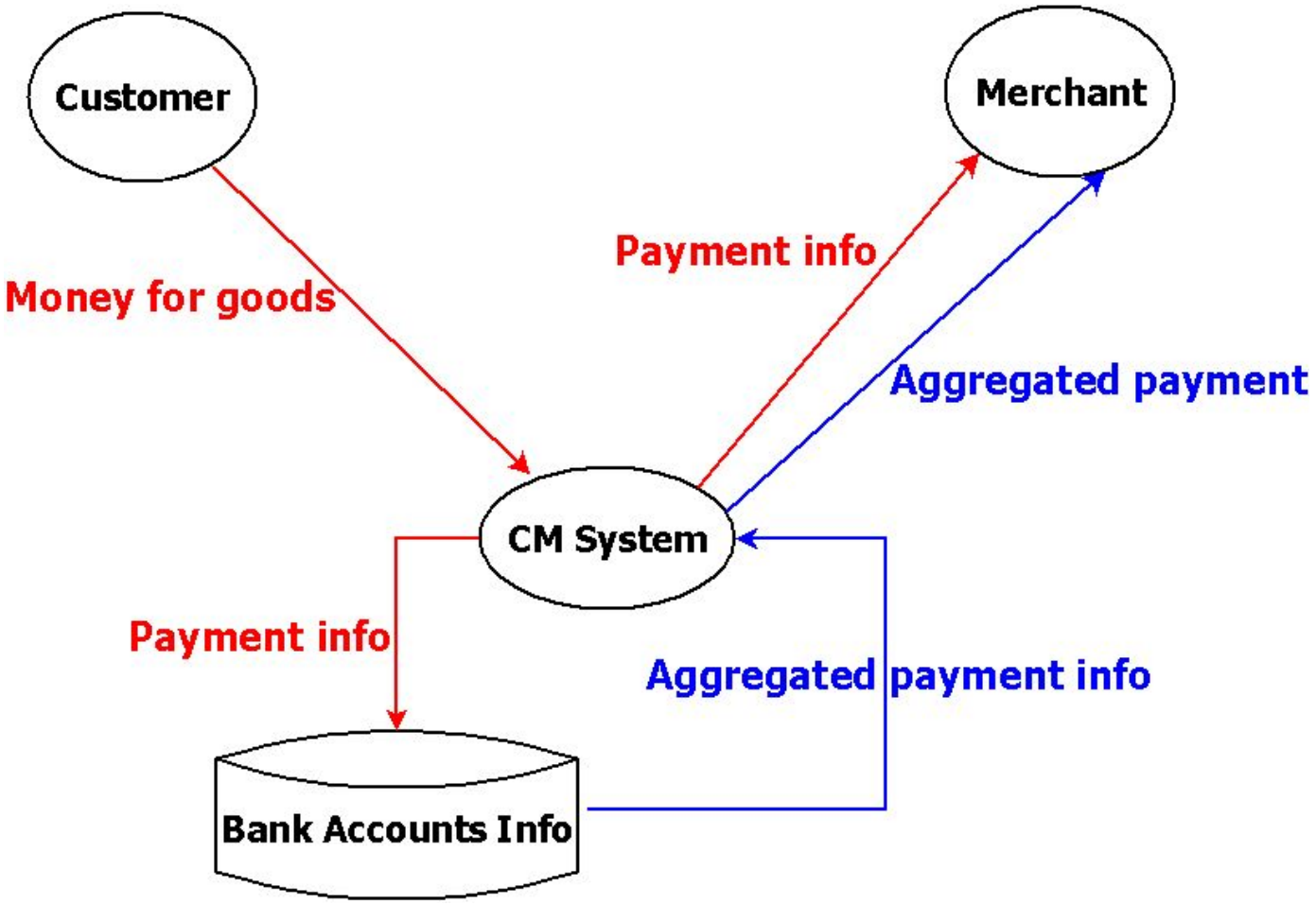
Relational DBMS

- A DBMS in which data is stored in tables and the relationships among the data are also stored in tables
- The data can be accessed or reassembled in many different ways without having to change the table forms.

Relational DBMS

- Commercial
  - Oracle
  - MS SQL Server
  - DB2
- Free
  - Derby (Java DB)
  - MySQL

Cash Management System



Merchant Info

- Name
- Bank
- Bank account
- Charge percent
- Aggregation period
- Minimal sum

Customer Info

- Name
- Address
- Email
- Credit card No
- Credit card type
- Credit card maturity date

Payment info

- Date
- Customer
- Merchant
- Goods description
- Sum

Java DB

- Java DB is Oracle's supported distribution of the Apache Derby open source database
- It supports standard ANSI/ISO SQL through the JDBC and Java EE APIs
- Java DB is included in the JDK
- http://www.oracle.com/technetwork/java/javadb/overview/index.html

Eclipse & Java DB

- Creating a Driver Definition for Apache Derby
- Creating an Apache Derby Connection Profile
- Connecting to Apache Derby
- Creating and Executing a SQL Query

Driver Definition (1 of 2)

- Start Eclipse
- Menu Window -> Preferences
- Expand Data Management -> Connectivity -> Driver Definitions
- Click Add button
- Select “Derby Embedded JDBC Driver” in Name/Type tab

Driver Definition (2 of 2)

- Select derby.jar in Jar list tab and click Add JAR/Zip button
- Select full path to derby.jar (usually C:\Program Files\Java\jdk1.7.0_05\db\lib)
- Click Open button
- Click Ok button

Eclipse & Java DB

- Creating a Driver Definition for Apache Derby
- Creating an Apache Derby Connection Profile
- Connecting to Apache Derby
- Creating and Executing a SQL Query

Connection Profile

- Switch to the Database Development perspective
- In Data Source Explorer, right-click Database Connections and select New
- Select Derby, change Name of profile (optionally) and click Next
- Select Database location and click Finish

Eclipse & Java DB

- Creating a Driver Definition for Apache Derby
- Creating an Apache Derby Connection Profile
- Connecting to Apache Derby
- Creating and Executing a SQL Query

Connecting to the Database

- In the Database Development perspective, expand Database Connections in the Data Source Explorer
- Right-click the connection profile that you created and select Connect

Eclipse & Java DB

- Creating a Driver Definition for Apache Derby
- Creating an Apache Derby Connection Profile
- Connecting to Apache Derby
- Creating and Executing a SQL Query

SQL Query

- In the Database Development perspective, expand Database Connections in the Data Source Explorer
- Right-click the connection profile that you created and select “Open SQL Scrapbook”
- Select database
- Create SQL query in the editor field
- Right-click in the editor and select Execute All.

Merchant Info

- Name
- Bank
- Bank account
- Charge percent
- Aggregation period
- Minimal sum

Create Merchant Table

```
CREATE TABLE merchant
(
   id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
   name VARCHAR(60) NOT NULL,
   bankName VARCHAR (100) NOT NULL,
   swift VARCHAR (40) NOT NULL,
   account VARCHAR (20) NOT NULL,
   charge DECIMAL(5,2) NOT NULL,
   period SMALLINT NOT NULL,
   minSum DECIMAL (19,2) NOT NULL,
   total DECIMAL(19,2),
   PRIMARY KEY (id)
);
```

Fill Merchant Table

```
INSERT INTO merchant
(name, charge, period, minSum,
 bankName, swift, account)
VALUES('Jim Smith Ltd.', 5.1, 1, 100.0,
        'Chase Manhatten', 'AA245BXW',
        '247991002');
```

Display Merchant Data

- select * from merchant;

ID	NAME	CHARGE	PERIOD	MINSUM	BANKNAME	SWIFT	ACCOUNT	TOTAL
1	Jim Smith Ltd.	5.10	1	100.00	Chase Manhatten	AA245BXW	247991002	NULL
2	Domby and sun Co.	2.80	2	20.00	Paribas	XTW2NNM	1188532009	NULL
3	Victoria Shop	3.40	3	500.00	Swedbank	SWEE34YY	557880234	NULL
4	Software & Digital goods	4.90	1	160.00	Credi Leone	FRTOPM	367920489	NULL

Create Customer Table

- Customer Info
  - Name
  - Address
  - Email
  - Credit card No
  - Credit card type
  - Credit card maturity date

Create Customer Table

```
CREATE TABLE customer
(
   id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
   name VARCHAR(60) NOT NULL,
   address VARCHAR(300) NOT NULL,
   email VARCHAR(90) NOT NULL,
   ccNo VARCHAR(20) NOT NULL,
   ccType VARCHAR(60) NOT NULL,
   maturity DATE,
   PRIMARY KEY (id)
);
```

Fill Customer Table

```
INSERT INTO customer
(name, address, email, ccNo, ccType, maturity)
    values('Dan Nelis',
    'Vosselaar st. 19, Trnaut, Belgium',
    'Dan@adw.com',
    '113456946712l4',
    'MasterCard',
    '2014-07-31');
```

Display Customer Data

- select * from customer

ID	NAME	ADDRESS	EMAIL	CCNO	CCTYPE	MATURITY
1	Dan Nelis	Vosselaar st. 19, Trnaut, Belgium	Dan@adw.com	11345694671214	MasterCard	2014-07-31
2	Mark Wolf	Olaf st. 11, Stockholm, Sweden	mwolf@yahoo.com	44402356988712	Visa	2012-09-30
3	Stein Brown	Oxford st. 223, Stockholm, Sweden	steinB@yahoo.com	41233576012434	Visa	2015-11-30

Create Payment Table

- Payment info
  - Date
  - Customer
  - Merchant
  - Goods description
  - Sum

Create Payment Table

```
CREATE TABLE payment
(
   id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
   dt TIMESTAMP NOT NULL,
   merchantId INT CONSTRAINT mer_fk references merchant,
   customerId INT CONSTRAINT cust_fk references customer,
   goods VARCHAR(500),
   total DECIMAL(15,2),
   charge DECIMAL(15,2),
   PRIMARY KEY (id)
);
```

Fill Payment Table

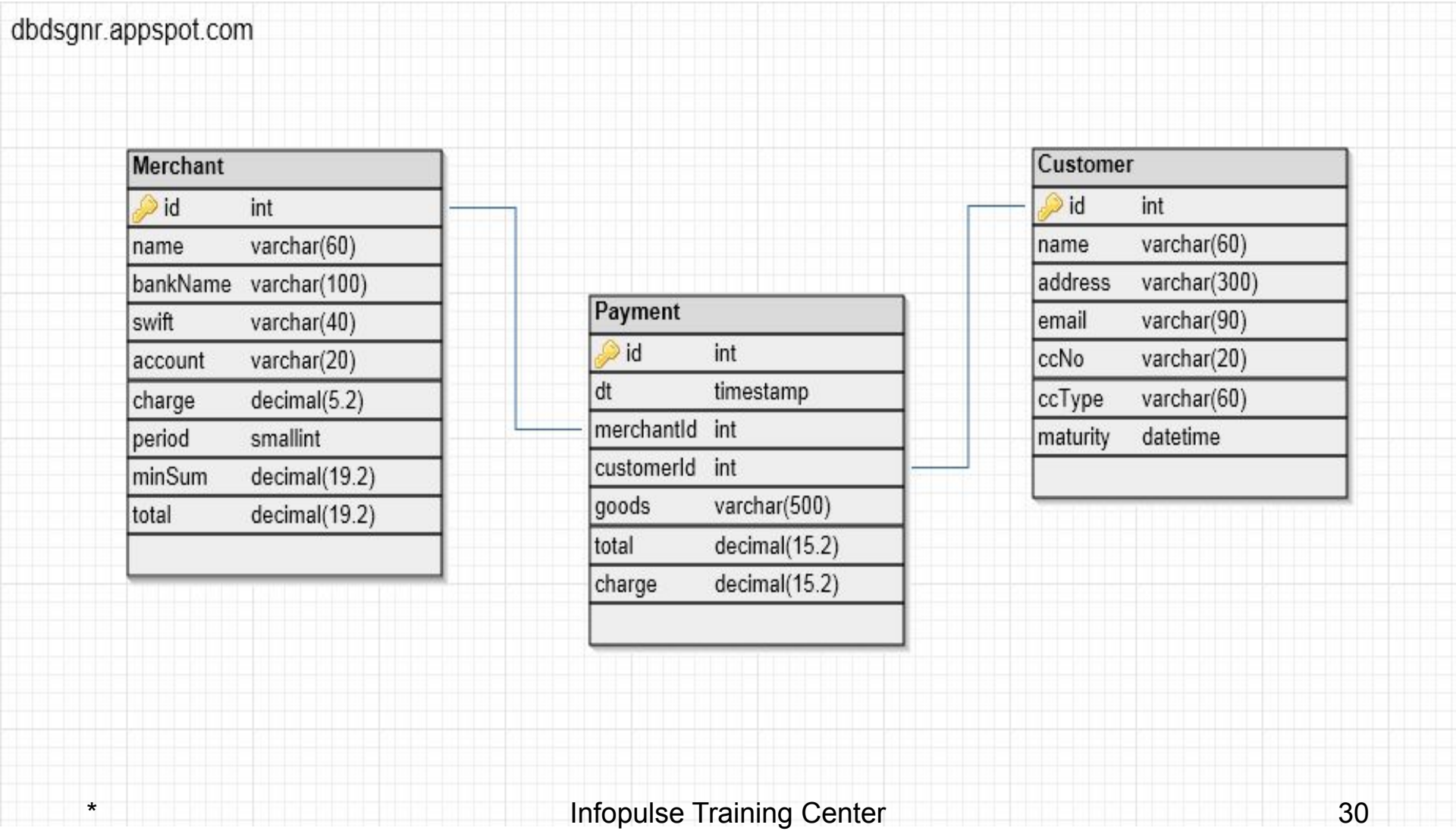
```
insert into payment
(dt, merchantId, customerId, goods, total)
values('2012-07-12 10:00:14', 3, 1,
        'CD Europe Maps', 12.08);
```

Display Payment Data

- Select * from payment

ID	DT	MERCHANTID	CUSTOMERID	GOODS	TOTAL
1	2012-07-12 10:00:14.0	3	1	CD Europe Maps	12.08
2	2012-06-22 18:21:10.0	4	3	NOD32 Antivirus	33.80
3	2012-07-02 00:00:17.0	1	1	Railway return ticket Brussel-Paris	255.58
4	2012-07-06 11:22:40.0	1	2	Railway ticket Stockholm - Oslo	1325.00
5	2012-07-10 11:10:45.0	3	2	CD African music	7.65
6	2012-06-30 12:00:00.0	2	1	Acer computer	654.00
7	2012-07-02 22:28:50.0	4	2	NOD32 Antivirus	33.80
8	2012-07-09 02:12:53.0	4	3	MS Office	400.23
9	2012-07-15 22:28:50.0	2	2	Dell computer	768.00

CM Database Schema



Select Statement

- Don’t use * in select!

Select Statement

- SELECT dt, merchantId, customerId, goods, total FROM payment WHERE merchantId = 3;

Select Statement

- SELECT dt, merchantId, customerId, goods, total FROM payment WHERE merchantId = 3;

DT	MERCHANTID	CUSTOMERID	GOODS	TOTAL
2012-07-12 10:00:14.0	3	1	CD Europe Maps	12.08
2012-07-10 11:10:45.0	3	2	CD African music	7.65

Select Statement

- SELECT dt, merchantId, customerId, goods, total FROM payment ORDER BY merchantId;

Select Statement

- SELECT dt, merchantId, customerId, goods, total FROM payment ORDER BY merchantId;

DT	MERCHANTID	CUSTOMERID	GOODS	TOTAL
2012-07-06 11:22:40.0	1	2	Railway ticket Stockholm - Oslo	1325.00
2012-07-02 00:00:17.0	1	1	Railway return ticket Brussels-Paris	255.58
2012-07-15 22:28:50.0	2	2	Dell computer	768.00
2012-06-30 12:00:00.0	2	1	Acer computer	654.00
2012-07-10 11:10:45.0	3	2	CD African music	7.65
2012-07-12 10:00:14.0	3	1	CD Europe Maps	12.08
2012-07-09 02:12:53.0	4	3	MS Office	400.23
2012-07-02 22:28:50.0	4	2	NOD32 Antivirus	33.80
2012-06-22 18:21:10.0	4	3	NOD32 Antivirus	33.80

Select Statement

```
SELECT sum(total) FROM payment WHERE customerId = 2;
```

Select Statement

SELECT sum(total) FROM payment WHERE customerId = 2;

Output is 2134.45

Select Statement

- SELECT merchantId, count(*) as n, sum(total) as total FROM payment GROUP BY merchantId;

Select Statement

- SELECT merchantId, count(*) as n, sum(total) as total FROM payment GROUP BY merchantId;

MERCHANTID	N	TOTAL
1	2	1580.58
2	2	1422.00
3	2	19.73
4	3	467.83

Select Statement

- SELECT customerId, sum(total) FROM payment GROUP BY customerId HAVING count(*)>2;

Select Statement

- SELECT customerId, sum(total) FROM payment GROUP BY customerId HAVING count(*)>2;

CUSTOMERID	2
1	921.66
2	2134.45

Join Operations

```
SELECT p.dt, m.name as merchant, c.name as customer,
        p.goods, p.total
FROM payment p
    LEFT OUTER JOIN merchant m on m.id = p.merchantId
    LEFT OUTER JOIN customer c on c.id = p.customerId;

SELECT p.dt, m.name as merchant, c.name as customer,
    p.goods, p.total
FROM payment p, merchant m, customer c
WHERE m.id = p.merchantId and c.id = p.customerId;
```

Join Operations

DT	MERCHANT	CUSTOMER	GOODS	TOTAL
2012-07-12	Victoria Shop	Dan Nelis	CD Europe Maps	12.08
2012-06-22	Software & Digital goods	Stein Brown	NOD32 Antivirus	33.80
2012-07-02	Jim Smith Ltd.	Dan Nelis	Railway return ticket Brussel-Paris	255.58
2012-07-06	Jim Smith Ltd.	Mark Wolf	Railway ticket Stockholm - Oslo	1325.00
2012-07-10	Victoria Shop	Mark Wolf	CD African music	7.65
. . . . .	. . . . .	. . . . .	. . . . .	. . . .
2012-07-15	Domby and sun Co.	Mark Wolf	Dell computer	768.00

Update Payments

DATE	MER_ID	GOODS	TOTAL	CHARGE
2012-07-12	3	CD Europe Maps	12.08	NULL
2012-06-22	4	NOD32 Antivirus	33.80	NULL
2012-07-02	1	Railway return ticket Brussel-Paris	255.58	NULL
2012-07-06	1	Railway ticket Stockholm - Oslo	1325.00	NULL
2012-07-10	3	CD African music	7.65	NULL
2012-06-30	2	Acer computer	654.00	NULL
2012-07-02	4	NOD32 Antivirus	33.80	NULL
2012-07-09	4	MS Office	400.23	NULL
2012-07-15	2	Dell computer	768.00	NULL

Update Statement

```
UPDATE payment SET charge = total * 0.034 WHERE id = 1;
```

Update Statement

UPDATE payment SET charge = total * 0.034 WHERE id = 1;

DATE	MER_ID	GOODS	TOTAL	CHARGE
2012-07-12	3	CD Europe Maps	12.08	**0.41**
2012-06-22	4	NOD32 Antivirus	33.80	NULL
2012-07-02	1	Railway return ticket Brussel-Paris	255.58	NULL
2012-07-06	1	Railway ticket Stockholm - Oslo	1325.00	NULL
2012-07-10	3	CD African music	7.65	NULL
2012-06-30	2	Acer computer	654.00	NULL
2012-07-02	4	NOD32 Antivirus	33.80	NULL
2012-07-09	4	MS Office	400.23	NULL
2012-07-15	2	Dell computer	768.00	NULL

Update Statement

- UPDATE payment
  SET charge = (SELECT p.total * m.charge / 100.0
  FROM payment p, merchant m
  WHERE m.id = p.merchantId and p.id = 2)
  WHERE id = 2;

Update Statement

DATE	MER_ID	GOODS	TOTAL	CHARGE
2012-07-12	3	CD Europe Maps	12.08	**0.41**
2012-06-22	4	NOD32 Antivirus	33.80	**1.65**
2012-07-02	1	Railway return ticket Brussel-Paris	255.58	NULL
2012-07-06	1	Railway ticket Stockholm - Oslo	1325.00	NULL
2012-07-10	3	CD African music	7.65	NULL
2012-06-30	2	Acer computer	654.00	NULL
2012-07-02	4	NOD32 Antivirus	33.80	NULL
2012-07-09	4	MS Office	400.23	NULL
2012-07-15	2	Dell computer	768.00	NULL

Update Statement

- UPDATE payment p SET charge = total * (SELECT charge FROM merchant m WHERE m.id = p.merchantId) / 100.0

Update Statement

DATE	MER_ID	GOODS	TOTAL	CHARGE
2012-07-12	3	CD Europe Maps	12.08	**0.41**
2012-06-22	4	NOD32 Antivirus	33.80	**1.65**
2012-07-02	1	Railway return ticket Brussel-Paris	255.58	**13.03**
2012-07-06	1	Railway ticket Stockholm - Oslo	1325.00	**67.57**
2012-07-10	3	CD African music	7.65	**0.26**
2012-06-30	2	Acer computer	654.00	**18.13**
2012-07-02	4	NOD32 Antivirus	33.80	**1.65**
2012-07-09	4	MS Office	400.23	**19.61**
2012-07-15	2	Dell computer	768.00	**21.50**

Update Merchants

ID	NAME	MINSUM	TOTAL
1	Jim Smith Ltd.	100.00	NULL
2	Domby and sun Co.	20.00	NULL
3	Victoria Shop	500.00	NULL
4	Software & Digital goods	160.00	NULL

Update Merchants

- UPDATE merchant m SET total =
  (SELECT sum(total - charge)
  FROM payment p WHERE p.merchantId=m.id)

Update Merchants

ID	NAME	MINSUM	TOTAL
1	Jim Smith Ltd.	100.00	1499.98
2	Domby and sun Co.	20.00	1382.19
3	Victoria Shop	500.00	19.06
4	Software & Digital goods	160.00	442.92

Manuals

- http://docs.oracle.com/javadb/10.8.2.2/ref/refderby.pdf
- http://docs.oracle.com/javadb/10.8.2.2/devguide/derbydev.pdf