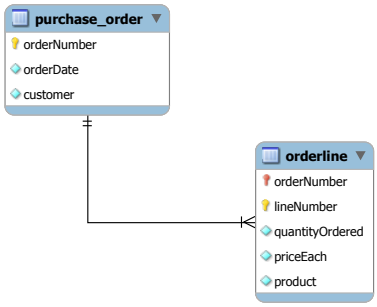
Building a Data Warehouse in MySQL

1 Retailer Data Warehouse

In this lab session you are required to create a data warehouse. Imagine a retailer of scale models of classic cars has contacted you to integrate its different data sources into a data warehouse. In particular the retailer has three different data sources:

- A relational database of purchase orders (this database is stored in the file *orders.sql*) with two tables as follows:



These tables contain the following sample data:

orderNumber	orderDate	customer
10100	2003-01-06	C@363
10101	2003-01-09	C@128
10102	2003-01-10	C@181
10103	2003-01-29	C@121
10104	2003-01-31	C@141
10105	2003-02-11	C@145
10106	2003-02-17	C@278
10107	2003-02-24	C@131

orderNumber	lineNumber	quantityOrdered	priceEach	product
10100	1	49	35.29	P@S24_3969
10100	2	50	55.09	P@S18_2248
10100	3	30	136.00	P@S18_1749
10100	4	22	75.46	P@S18_4409

- The information about customers is stored in a *comma-separated values* (csv) file (named *customerInfo.csv*) as follows:

customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1	addressLine2	city	state	postalCode
103	Atelier graphique	Schmitt	Carine	40.32.2555	54, rue Royale		Nantes		44000
112	Signal Gift Stores	King	Jean	7.03E+09	8489 Strong St.		Las Vegas	NV	83030
114	Australian Collect	Ferguson	Peter	03 9520 45	636 St Kilda Roa	Level 3	Melbourn	Victoria	3004
119	La Rochelle Gifts	Labrune	Janine	40.67.8555	67, rue des Cinquante Otages		Nantes		44000
121	Baane Mini Import	Bergulfsen	Jonas	07-98 9555	Erling Skakkes gate 78		Stavern		4110
124	Mini Gifts Distribu	Nelson	Susan	4.16E+09	5677 Strong St.		San Rafae	CA	97562
125	Havel & Zbyszek C	Piestrzeniewicz	Zbyszek	(26) 642-7	ul. Filtrowa 68		Warszawa		01-012
128	Blauer See Auto, C	Keitel	Roland	+49 69 66 5	Lyonerstr. 34		Frankfurt		60528
129	Mini Wheels Co.	Murphy	Julie	6.51E+09	5557 North Pendale Street		San Franci	CA	94217
131	Land of Toys Inc.	Lee	Kwai	2.13E+09	897 Long Airport Avenue		NYC	NY	10022
141	Euro+ Shopping Ch	Freyre	Diego	(91) 555 94	C/ Moralzarzal, 86		Madrid		28034
144	Volvo Model Repli	Berglund	Christina	0921-12 35	BerguvsvÃ¤gen 8		LuleÃ¥		S-958 22
145	Danish Wholesale	Petersen	Jytte	31 12 3555	VinbÃ¦ltet 34		Kobenhavn		1734
146	Saveley & Henriot,	Saveley	Mary	78.32.5555	2, rue du Commerce		Lyon		69004
148	Dragon Souveniers	Natividad	Eric	+65 221 75	Bronz Sok.	Bronz Apt. 3/6 Te	Singapore		79903
151	Muscle Machine In	Young	Jeff	2.13E+09	4092 Furth Circle	Suite 400	NYC	NY	10022
157	Diecast Classics Inc	Leong	Kelvin	2.16E+09	7586 Pompton St.		Allentowr	PA	70267
161	Technics Stores Inc	Hashimoto	Juri	6.51E+09	9408 Furth Circle		Burlingam	CA	94217
166	Handji Gifts& Co	Victorino	Wendy	+65 224 15	106 Linden Road	2nd Floor	Singapore		69045

Files written in CSV format vary somewhat; there is apparently no formal standard describing the format. However, the general idea is that lines consist of values separated by commas, and values containing internal commas are enclosed within quotes to prevent the commas from being interpreted as value delimiters. It's also common for values containing spaces to be quoted as well.

- Finally the information about products is maintained in a *tab-separated values* file (named *productInfo.txt*) as follows:

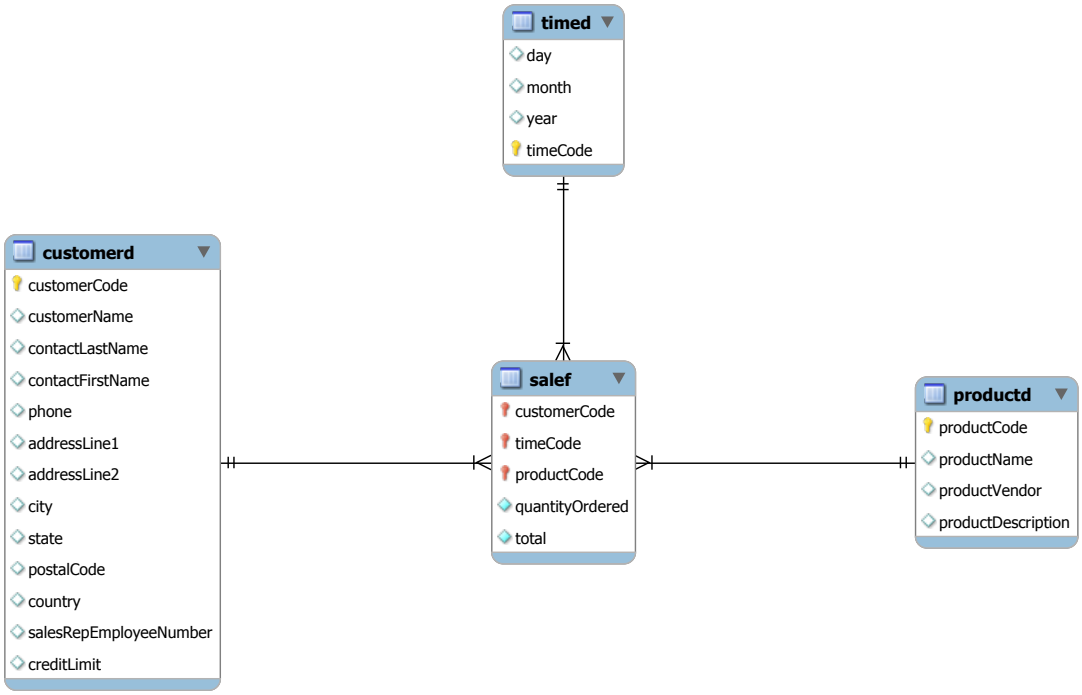
```
"productCode"   "productName"   "productVendor" "productDescription"
"S10_1678"      "1969 Harley Davidson Ultimate Chopper" "Min Lin Diecast"       "This replica features working kickstand, fr
"S10_1949"      "1952 Alpine Renault 1300"      "Classic Metal Creations"       "Turnable front wheels; steering function; d
"S10_2016"      "1996 Moto Guzzi 1100i" "Highway 66 Mini Classics"      "Official Moto Guzzi logos and insignias, saddle bag
"S10_4698"      "2003 Harley-Davidson Eagle Drag Bike"  "Red Start Diecast"     "Model features, official Harley Davidson lo
, precision diecast replica, baked enamel finish, 1:10 scale model, removable fender, seat and tank cover piece for displayi
"S10_4757"      "1972 Alfa Romeo GTA"   "Motor City Art Classics"       "Features include: Turnable front wheels; steering f
"S10_4962"      "1962 LanciaA Delta 16V"        "Second Gear Diecast"   "Features include: Turnable front wheels; steering f
```

This is one of the simplest file structures; lines contain values separated by tab characters. It is also common to quote values.

2 Designing the Data warehouse

The retailer would like to perform different queries related to the sales of individual products, by each customer and day.

1. Given the data available, design a data warehouse that meets these requirements.
2. Compare your solution to the model in the next page



3 Creating the Tables in the MySQL

1. Use DDL statements to create the different tables. Note that some fields on the customer file have not been included in the dimensional model (this has been intentionally done to show how to discard some columns when loading the data). To facilitate your work the DDL statements needed are included below:

```
CREATE TABLE customerd (
  customerCode int(11) NOT NULL,
  customerName text,
  contactLastName text,
  contactFirstName text,
  phone text,
  addressLine1 text,
  addressLine2 text,
  city text,
  state text,
  postalCode int(11) DEFAULT NULL,
  country text,
  PRIMARY KEY (customerCode)
);

CREATE TABLE productd (
  productCode varchar(10) NOT NULL,
  productName text,
  productVendor text,
  productDescription text,
  PRIMARY KEY (productCode)
);
```

```
CREATE TABLE timed (
  timeCode int(11) NOT NULL AUTO_INCREMENT,
  day int(2) DEFAULT NULL,
  month int(2) DEFAULT NULL,
  year int(4) DEFAULT NULL,
  PRIMARY KEY (timeCode)
);


CREATE TABLE salef (
  customerCode int(11) NOT NULL,
  timeCode int(11) NOT NULL,
  productCode varchar(11) NOT NULL,
  quantityOrdered int(11) DEFAULT NULL,
  total decimal(20,2),
  PRIMARY KEY (customerCode,timeCode,productCode),
  CONSTRAINT fkCustomer FOREIGN KEY (customerCode) REFERENCES customerd (customerCode),
  CONSTRAINT fkProduct FOREIGN KEY (productCode) REFERENCES productd (productCode),
  CONSTRAINT fkTime FOREIGN KEY (timeCode) REFERENCES timed (timeCode)
);
```

4 Importing Data into MySQL

Once the tables for the Data Warehouse have been created we need to import the different datasets available into the database.

1. We will start by importing the *orders.sql* file. Since the two tables in this file do not correspond to any table in the dimensional model we will import them into two new tables.

   (a) Open the *orders.sql* file with a text editor and analyse its contents.

   (b) As you can see it is a SQL script created by MySQL, thus it can be used directly to create the `orderline` and `purchase_order` tables. To execute this SQL script file you can use the MySQL `SOURCE` command as follows:

   ```
   mysql > SOURCE orders.sql
   ```

   Note the that the command above requires that you are located in the directory where the file *orders.sql* is located. If this is not the case, you will have to specify the pathname of the file.

2. Next, we will import the product data into the product dimension table (`productd`).

   (a) Open the *productInfo.txt* file with a text editor and analyse its contents.

   (b) As you can see it is tab-separated values file in which fields are enclosed by double quotes ("). Moreover, the first line in the file contains the names of the different fields and should be ignored when importing the data.

   (c) Usually a newline, also known as end of line (EOL), is represented with one or two control characters. To know the the end of line used in the *productInfo.txt* file you can use the command `file` in Unix as follows:

   ```
   file productInfo.txt
   productInfo.txt: UTF-8 Unicode English text, with very long lines, with CRLF line
                    terminators
   ```

   Thus, this file uses CR followed by LF to start a new line. Note that CRLF corresponds to the characters '\r\n'.

   (d) To load the data use the `LOAD DATA` command as follows:

```
mysql> LOAD DATA LOCAL INFILE 'productInfo.txt' INTO TABLE productd
    -> FIELDS TERMINATED BY '\t' ENCLOSED BY '"'
    -> LINES TERMINATED BY '\r\n'
    -> IGNORE 1 LINES;
```

`LOAD DATA` statement that acts as a bulk data loader. `LOAD DATA` provides options to address import issues such as the line-ending sequence for recognizing how to break input into records, the column value delimiter that permits records to be broken into separate values, the quoting character that may enclose column values, quoting and escaping conventions within values, and NULL value representation. In the example above we have:

- Indicated that the first line should be ignored, data values are separated by tab characters (note the tab corresponds to character '\t'), that data values may have quotes around them that should be stripped, and that new lines are represented by CRLF.

Note you can can load files located either on the server host, or on the client host (your machine). Telling MySQL where to find your datafile is a matter of knowing the rules that determine where it looks for the file (particularly important for files not in your current directory). By default, the MySQL server assumes that the datafile is located on the server host. You can load local files that are located on the client host using `LOAD DATA LOCAL` rather than `LOAD DATA`. If the `LOAD DATA` statement includes the `LOCAL` keyword, your client program reads the file on the client host and sends its contents to the server. Again the command above requires that you are located in the directory where the file *productInfo.txt* is located.

3. Finally, we will import the customer data into the customer dimension table (`customerd`).

   (a) Open the *customerInfo.csv* file with a text editor and analyse its contents.

   (b) As you can see it is comma-separated values file in which non-numeric fields are enclosed by double quotes ("). The first line in the file also contains the names of the different fields and should be ignored when importing the data. NULL values are represented as empty strings ("") and not all of the fields in the file need to be imported into the `customerd` table (some fields are not needed in the dimensional model).

   (c) To know the the end of line used in the *customerInfo.csv* file you can use the command `file` in Unix as follows:

```
file customerInfo.csv
customerInfo.csv: UTF-8 Unicode English text
```

   In this case we can assume the file uses the default end of line witch corresponds to the character '\n'.

   (d) To load the data use the `LOAD DATA` command as follows:

```
mysql> LOAD DATA LOCAL INFILE 'customerInfo.csv' INTO TABLE customerd
->FIELDS TERMINATED  BY ',' ENCLOSED BY '"'
->LINES TERMINATED BY '\n'
->IGNORE 1 LINES
->(@customerNumber,@customerName,@contactLastName,@contactFirstName,@phone,
->@addressLine1,@addressLine2,@city,@state,@postalCode,@country,@dummy,@dummy)
->SET customerCode = @customerNumber,
->customerName=@customerName,
->contactLastName = @contactLastName,
->contactFirstName = @contactFirstName,
->phone = @phone,
->addressLine1 = IF(@addressLine1='',NULL,@addressLine1),
->addressLine2 = IF(@addressLine2='',NULL,@addressLine2),
->city = @city,
->state = IF(@state='',NULL,@state),
->postalCode = @postalCode,
->country = @country;
```

   By default, `LOAD DATA` expects the datafile to have the same number of columns as the table into which you load it, with the columns present in the same order as in the table. If the file column

number or order differ from the table, you can specify which columns are present and their order. In particular, the above statement stores the values in the different fields into user variables and then uses them to set the values in the different columns on the `customerd` table —note user variables are written as `@var_name`.

4. At this point the `productd` and `customerd` tables should be populated with the data in the *productInfo.txt* and *customerInfo.csv* files, respectively. Select the rows on both tables to check that they contain the corresponding information.

5 Adding the data into the Time Dimension Table

The information that is needed for the time dimension can be obtained by selecting distinct rows from the `purchase_order`.

1. Create a query that selects the distinct day, month and year from purchase dates. Recall you can use the date functions `DAY, MONTH` and `YEAR`.

2. Use your previous statement in a `INSERT` statement to populate the `timed` table. Recall that the `timeCode` is an `AUTO_INCREMENT` column and MySQL sets its value automatically.

6 Adding the data into the Sales Fact Table

Finally to add the data into the sales fact table we need to use the information on the `purchase_order` and `orderline`. However, there are a few things that need a little bit of care:

1. The information on the tables `purchase_order` and `orderline` needs to be joined. Create a select statement that joins these to tables using the orderNumber.

2. Modify the previous query to retrieve the information need for the sales fact table (i.e., `productCode, customerCode, quantityOrdered, total`).

3. Note that both customer and product codes contain the prefixes 'P@' and 'C@', respectively. Modify the previous select to remove these prefixes. Recall that you can use the function `SUBSTRING`.

4. The `timeCode` needs to be retrieved. Modify the previous select to join the time dimension table and include the `timeCode` corresponding to each purchase order.

5. Use your previous statement in a `INSERT` statement to populate the `salef` table.

7 Personal Study

There are many other useful things that can be done with the `LOAD DATA` statement. Read the information about it here:

`http://dev.mysql.com/doc/refman/5.7/en/load-data.html`