TSE
Toulouse School of Economics
École d'économie de Toulouse

Année Universitaire 2012-2013
SESSION 1

MASTER 1

DATA BASES
(durée 1h30)

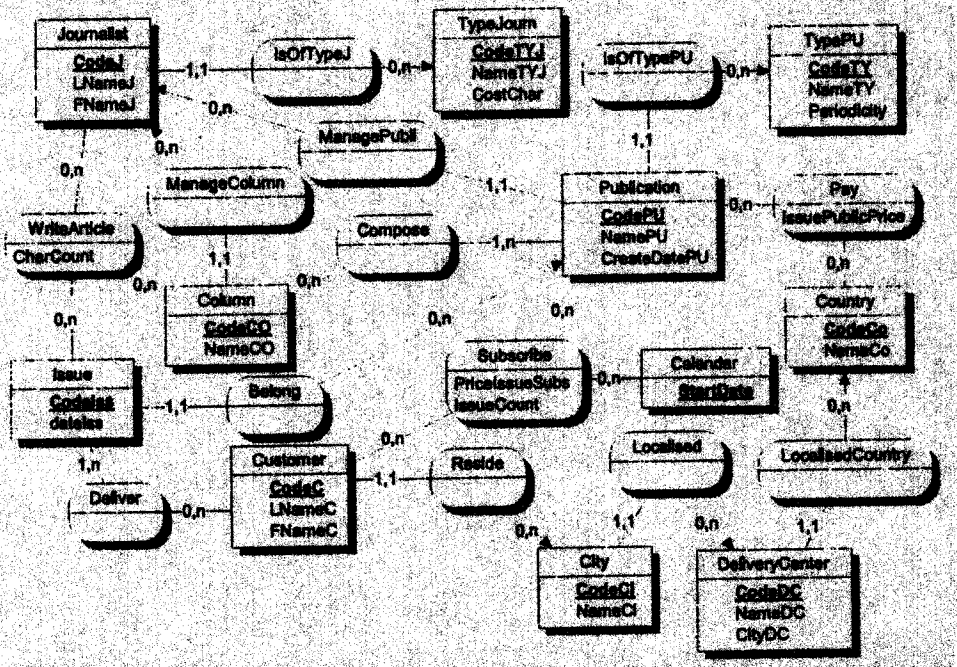
Friday January 11th 2013 – 14h00 –15h30

R. TOURNIER

****

Documents and electronic devices are strictly forbidden.

A journal editor has elaborated a data warehouse to study its publications. This data warehouse analyses the productions (publications) of journalists as well as the customers (subscribers) to different journals. Moreover, this data warehouse stores the subscribers' delivery centres. The conceptual schema of the data warehouse (in entity/relationship formalism) is:



Each journalist belongs to a type (grand reporter, independent, etc.) that indicates the character price these journalists are paid. The wages of a journalist depends on the number of typed characters in his articles. Journalists writes articles in columns (politic, economy, international, etc.), these articles are composed of a certain amount of characters and are published in publication issues. A publication ("Le Monde", "Elle", "Maison&Travaux", etc.) is composed of columns and is created at a certain date. Each column and each publication has a manager who is a journalist. Moreover, each publication is of one type (new, fashion…) with a certain periodicity (weekly, monthly, etc.).
Publications have a public price that depends on the country where the publication is sold. Subscribers are customers that subscribe for a year at a given date (StartDate) and pay a reduced price for each number (PriceIssueSubs) for a subscription that consist in a certain number of issues (IssueCount). This later number depends of the periodicity of the publication. Customers, depending on the city they live in, are associated to a delivery centre (not necessarily in the same town) and this centre is located in a country.
The relational schema of the data warehouse is the following:

- **Journalist**(CodeJ, LNameJ, FNameJ, CodeTYJ#)
- **TypeJourn**(CodeTYJ, NameTYJ, CostChar)
- **Issue**(CodeIss, DateIss, CodePU#)
- **Column**(CodeCO, NameCO, CodeJ#)
- **Publication**(CodePU, NamePU, CreateDatePU, CodeJ#, CodeTY#)
- **TypePU**(CodeTY, NameTY, Periodicity)
- **Customer**(CodeC, LNameC, FNameC, CodeCi#)
- **City**(CodeCi, NameCi, CodeDC#)
- **DeliveryCenter**(CodeDC, NameDC, CityDC, CodeCN#)

- **Country**(<u>CodeCN</u>, NameCN)
- **WriteArticle**(<u>CodeJ#, CodeIss#, CodeCO#</u>, CharCount)
- **Deliver**(<u>CodeIss#, CodeC#</u>)
- **Compose**(<u>CodeCO#, CodePU#</u>)
- **Subscribe**(<u>CodeC#, CodePU#, StartDate</u>, PriceIssueSubs, IssueCount)
- **Pay**(<u>CodePU#, CodeCN#</u>, IssuePublicPrice)

In this schema, primary key attributes are underlined and foreign key attributes are followed by a hash (#).

Work to be done

1. Modelling a multidimensional data mart

The data warehouse is used to create a multidimensional data mart for the sales manager. This manager wishes to know the number of subscriptions as well as the amount that these subscriptions represent. This is done by year, by month, for each publication and for each customer. In order to offer this manager analysis opportunities, all non temporal dimensions should have all the possible aggregation levels that can be extracted from the data warehouse as well as all possible associated information. Note that the amounts correspond to the issue price multiplied by the number of issues.

1.1. Provide the content of the measure dictionary (Columns of this dictionary are: Code, Description, Type, Extraction formula). Note that the description is very important;

1.2. Provide a complete graphical representation of the conceptual schema of this data mart (use the graphical formalism of facts and dimensions studied during the course).

2. SQL querying of the data warehouse

Write the following SQL queries that will be executed on the tables of the data warehouse (note that the tables are described by the **relational schema** of the data warehouse).

2.1. For each journalist and type of journalist, provide the number of written articles;

2.2. Provide the last name and first name of journalists that write articles only in publications that have a "daily" periodicity;

2.3. For each publication that has more than 1,000 subscribers, provide the total number of subscriptions, as well as the type of the publication.

3. Decisional workbook

The department that pays the journalist wages uses an Excel workbook. Answer to the question 3.1 hereafter.

Universal Data Table

Journalist Code	Journalist First Name	Journalist Last Name	Journalist Type	Character cost	Issue	Character Count	Issue Date*	Column Name
J1	Jean	Sairien	Reporter	$0.50	LM4562	1450	2/1/2012	Infos
J1	Jean	Sairien	Reporter	$0.50	EL8755	2340	3/1/2012	Reportages
J1	Marc	Dufaili	Grand Reporter	$0.75	LM4562	2076	2/1/2012	Reportages
J2	Ella	Froid	Trainee	$0.05	EL8755	265	2/1/2012	Politique
J2	Ella	Froid	Trainee	$0.05	MT745	640	4/1/2012	Economie
J2	Ella	Froid	Trainee	$0.05	MT745	185	3/1/2012	Jeux
J3	Charles	De Teilhardie	Retired	$0.45	LM4562	835	2/1/2012	Infos
J3	Charles	De Teilhardie	Retired	$0.45	MT745	460	4/1/2012	Jeux
J4	Al	Quolic	Independent	$0.25	MT745	1650	4/1/2012	Reportages
J5	Ramon	Neur	AFP Journalist	$0.75	EL8755	380	2/1/2012	Infos
J6	Alain	Proviat	Reporter	$0.50	LM4562	1680	3/1/2012	International
J6	Alain	Proviat	Reporter	$0.50	MT745	2380	4/1/2012	Reportages

*Date are in English format

The workbook is composed of a data sheet named **Data** that contains the universal data table (see previous image). This table contains the code, first name, last name and the type of a journalist, as well as his character salary, the issue where he published, the issue date and the column in which the article was published.

Journalist Wages

Journalist			
Code	J1	Last Name	Sairien
		First Name	Jean
Pays			
Article Count	3		
Amount to pay	$4 051.25		

The previously defined universal table is used as a support for the **Stat** data sheet (see previous image). In this sheet, only cell B4 is an input cell. The sheet displays the journalist whose code is typed in B4 as well as the amount of money that has to be paid to him.

3.1. Provide the formula in E4 and knowing that a journalist is paid according to the number of characters he used; provide also the formula in cell B9.

Note: a possible solution for B9 is using additional calculated columns in the Data sheet. If such a solution is considered, use, in the Data sheet, the J column, then the K column, etc. Moreover, for each new column created in this way, provide the formula typed in line 4 (that is the formulas typed in J4, then in K4, etc.).

Appendix 1: Excel formulas that might be required for this exam

Search Functions:

Syntax	**VLOOKUP(Searched_Value ; Range ; Col_Number ; Approximate_Search)** ♦ **Searched_Value**: is the value looked up in the first column of the table provided (Range). This can be a value, a reference or a character string. ♦ **Range**: is the table where the search is performed (Searched_Value is lookedup only in the first column of Range). ♦ **Col_Number**: is the column where the value found is returned. **Col_Number = 1** means the returned value is in the first column of Range, **Col_Number = 2** means the returned value is in the second column of Range, etc. ♦ **Approximate_Search**: ♦ **TRUE or NOTHING**: is used if the search is to be approximate (in other words, the search is to find an equal or smaller value than the value given in Searched_Value). This option requires that the values in the first column of Range are ordered in an ascending order. ♦ **FALSE**: if the function has to search an exact match of the value provided in Searched_Value. This option does not require a ordering on the values in the first line of Range.
Result	This function searches a value in the first column of a table (Range) and then returns the value in the same line but Col_Number columns right of it.

Syntax	**OFFSET(Reference_Cell ; NbLine ; NbCol)** ♦ **Reference_Cell** the reference cell (an address of a cell) ♦ **NbLine** the number of lines to move (positive or negative integer) ♦ **NbCol** the number of columns to move (positive or negative integer)
Result	This function returns the value in the cell that correspond to the reference cell moved **NbLine** lines downwards (if **NbLine** is positive) or upwards (if **NbLine** is negative) and moved **NbCol** rightwards (if **NbCol** is positive) of leftwards (if **NbCol** is negative).

MATCH(Searched_Value ; Range ; Search_Type) returns the relative position of Searched_Value (value or reference to a cell) in the table **Range** (a serie of adjacent cells). **Search_Type** indicates which search is to be used (Search_Type can be 1, 0 or -1):

- **1**: the EQUIV function returns the highest value which is smaller or equal to **Searched_Value**. Values in Range should be ordered in ascending order.
- **0**: The EQUIV function returns the first value which is equal to **Searched_Value**. No order is necessary in Range.
- **-1** : the EQUIV function returns the smallest value which is greater or equal to **Searched_Value**. Values in Range should be ordered in descending order.

Conditional Functions

Syntax	Result
SUMIF(RangeC; criteria; RangeS)	Adds the values in the cells of the range "RangeS" if the criterion ("criteria") applied to the cells of "RangeC" is TRUE.

COUNTIF(Range; criteria)	Counts the number of cells inside a range ("Range") such that the criterion ("criteria") applied on these cells is TRUE.

Note: A criterion can be an operator associated to a constant, an operator associated to a reference, or an operator and an calculus expression. The criteria must be a character string (expressed between quotes), example: ">10" or "="&A1 (references to cells must be associated with the "&" operator).

Appendix 2: general syntax of a query

The general syntax of a selection query is the following:

```
SELECT R1.GroupingAttribute1, Function(R2.Aggreg_Attribute) AS [Column header],...
FROM Relation1 As R1, Relation2 As R2, ...
WHERE Join_conditions (...) AND Selection_Conditions (...)
GROUP BY R1.GroupingAttribute1,...
HAVING Grouping_Conditions AND ...
ORDER BY R1.Attribute order_type... ;
```