Lab 1: Data Warehousing and OLAP

Due date: Thursday, April 14, 11:59pm.

Lab Assignment

The assignment consists of two parts:

1. Using software provided to you, you are asked to design a data warehouse from an existing database. The software will allow you to describe the structure of the data warehouse, and will produce as output SQL code to create and populate the data warehouse.

2. Once you construct the data warehouse, you will have to write programs implementing specific OLAP operations for your data warehouse. You can use any programming environment (Java JDBC+ SQL and PL/SQL are two obvious choices, Perl DBI is also possible) to write the code.

Assignment Preparation

This is a group assignment to be performed in groups of three students. We will discuss the formation of the groups at the beginning of the lab period on Tuesday, April 7.

Dataset

The assignment uses the `Extended BAKERY` dataset from Lab 0. The dataset is available from the datasets wiki at the following URL:

http://wiki.csc.calpoly.edu/datasets/wiki/ExtendedBakery

The dataset actually has four sets of data in it: 1000 receipts, 5000 receipts, 20,000 receipts and 75,000 receipts. You may use smaller databases for testing purposes while developing your code. The two larger datasets will be used in testing and evaluation of your submissions.

Note: Each sub-dataset has exactly the same structure and table names, all four datasets won't co-exist on a single Oracle account unless you rename tables (a REALLY big hassle). You will have access to multiple Oracle accounts (at least three for each group), make proper use of them.

The full description of the dataset was distributed to you at the time of Lab 0, and is available from the course web page. The wiki page referenced above contains the link to the README file for the dataset as well.

Each sub-dataset of the `EXTENDED BAKERY` dataset consists of the following files:

Filename	Explanation
`EB-setup.sql`	database setup script (`CREATE TABLE` statements)
`EB-cleanup.sql`	`DROP TABLE` statements
`EB-insert.sql`	runs all other `...-insert.sql` scripts
`EB-build-employee.sql`	inserts data into the `Employees` table
`EB-build-goods.sql`	inserts data into the `Goods` table
`EB-build-items.sql`	inserts data into the `Items` table
`EB-build-location.sql`	inserts data into the `Location` table
`EB-build-receipts.sql`	inserts data into the `Receipts` table
`EB-verify.sql`	reports tuple counts in all database tables

These files can be downloaded one-by-one, or as a single zip file, available from the datasets wiki.

```
> $ORACLE_HOME/bin/sqlplus  <yourAccount>@ora10g
SQL> start EB-setup.sql
...
SQL> start EB-insert.sql
```

Note, the last command, may take some time (it runs for a few mins for the 75K dataset). You can run `start EB-verify.sql` command to verify that all data is there. Tuple counts are available in the `results-of-verify.txt` file, supplied for every sub-dataset.

Oracle Connectivity

All your Java programs shall connect to the Oracle DBMS using JDBC. A short primer on JDBC connectivity can be found in my CSC 365 lecture notes:

http://users.csc.calpoly.edu/∼dekhtyar/365-Spring2008/lectures/lec12.365.pdf

This handout is linked to from the course web page. The course web page also contains links to the `classes12.jar`, Oracle's JDBC jar (in case you cannot include its location in your classpath for some reason), and a simple Java program that shows the main JDBC functionality, `JDBCTest.java`.

Oracle account independence. All programs you submit for this assignment, **must be Oracle account-independent**. To achieve that, your programs will assume that there is a file with the name `connection.info` stored in the current directory. The `connection.info` file shall have the following format:

1. The first line of the file will have the format:

   ```
   login = <loginId>
   ```

   where, `<loginId>` is and Oracle account userId.

2. The second line of the file will have the format:

   ```
   password = <pass>
   ```

   where `<pass>` is the password for the Oracle account.

For example, for a userId `st27` with the password `"e4dkr34_!df"`, the `connection.info` file will be:

```
login = st27
password = e4dkr34_!df
```

Your programs shall read the contents of `connection.info` file and set up the JDBC connection with the `ora10g` server accordingly. When your programs are submitted, I will use `connection.info` files pointing to my grading accounts.

Data Warehouse Design

Design Tool

For this lab you are provided with a data warehouse design tool, created by our alum Steven Weigand as his senior project and modified by our TA, Jacob Verburg. A separate handout provides some information on interaction with the tool and includes a walk-through for a simple example.

Before you use the tool, you must at least create[1] the database/ database tables from which you want to extract the data warehouse. The tool allows you to connect to your Oracle account. It collects information on the

[1]The tool analyzes the structure of the database stored at the account, but not the content, so it is important to have the tables in place, when it runs, even if the tables are empty. The tool ouputs SQL code to run on the database to create a user-defined data warehouse. Obviously before you actually run the code, you want to have the database appropriately populated.

database stored in in your tablespace, and allows you to define a data warehouse structure by specifying the dimensions of the data warehouse and the aggregate information to be collected. The tool then outputs SQL DDL for creating the warehouse-style tables. You create the warehouse by running the SQL commands provided by the tool.

The installation and operation of the design tool is described in a separate handout.

IMPORTANT NOTE: The tool is in the prototype stage and cannot create correct code for every possible data warehouse organization. It **is**, however, set up to **ensure that the correct data warehouse creation DDL is generated** for the **correct** data warehouse schema for this assignemnt[2].

The tool is available from the course web page:

http://www.csc.calpoly.edu/~dekhtyar/466-Spring2008/DWDesignTool/DataWareHouse.jar

Data Warehouse

The bakery chain has hired your group to provide some analytical support for their operations. In particular, the management of the bakery chain wants to obtain answers to the following questions:

- What are good locations for expansion?
- Conversely, what are the locations where the bakery locations are underperforming?
- Which items on the menu are universally popular? Which items have only limited geographical appeal?
- Does it make financial sense to keep part-time employees? Do they generate as many sales as full-time staff?
- What is the best allocation of staff throughout the week? What are the busy/light times?

Your first task it to design the architecture of a data warehouse that would contain information sufficient for answering these questions:

1. Identify the dimensions of the data warehouse.
2. For each dimension, identify its internal hierarchy.
3. Identify the measures of the data warehouse.

[2]Use it to your advantage. If the tool outputs something you do not believe to be correct, this probably means, you have a conceptual error in your warehouse design.

4. Design the schema for the multidimensional data table for the warehouse.
5. Design the schemas for the dimension tables (if any are needed) for the warehouse.
6. Using the data warehouse design tool, produce the SQL DDL and DML for creating the data warehouse.
7. Create and populate the data warehouse by running the produced SQL commands.

Deliverables. Your deliverables for this stage are:

1. A text file (word-processed) containing the schema of the data warehouse.
2. A diagram of the star (snowflake, if needed) schema of the data warehouse (use Powerpoint or other drawing program to create it).
3. The SQL DDL and DML output of the data warehouse design tool.

OLAP Operations

For this part of the assignment your task is to implement a number of special-purpose OLAP operations over the data warehouse you have constructed.

Roll-up and sort by location

Output sales data by individual location, city, zip, and state. Rank sales data by (i) number of sales made, (ii) total number of sold items, (iii) total amount of sales, (iv) sales per store.

```
java LocationRollUp <rollUp-Level>  [<sort>]  [<sortModifier>] [<direction>]
```

`<rollUp-Level>`: `location, zip, city` or `state`.

`<sortModifier>`: 0 = use totals; 1 = use averages per location. Optional (no parameter means use totals).

`<sort>`: 0= unsorted, 1 = sort by number of sales (receipts), 2 = sort by number of sold items, 3 = sort by total amount of sales. Optional. No parameter means unsorted.

`<direction>`: `d` = descending; `a` = ascending. Optional. No parameter means descending.

For example, to output sales data by city, sorting the results in descending order by the total amount of sales, we will issue the command:

```
java LocationRollUp city 3 0
```

To output sales data in ascending order by the average number of items sold per store by store zip code, we will issue the command:

```
java LocationRollUp zip 2 1 a
```

Roll-up by location with Slice by food type

Output sales data by individual location, city and state for a specified menu item, or category of menu items. Order sales data as above.

```
java LocationRollUpSlice <rollUp-Level>  <sliceLevel>  <sliceValue>
                         [<sort>]  [<sortModifier>] [<direction>]
```

`<rollUp-Level>`, `<sort>`, `<sortModifier>`, `<direction>` are all as for `LocationRollUp`.

`<sliceLevel>`: `Item`, `Food`, `Flavor`, `Type`.

`<sliceValue>` is as follows:

`<sliceLevel>`	`<sliceValue>`
`Type`	`P[astry]` `D[rink]`
`Food`	any value from the `Goods.Food` field (e.g., `Cake`, `Coffee`, `Cookie`, `Tart`)
`Flavor`	any value from the `Goods.Flavor` field (e.g., `Chocolate`, `Almond`, `Grapefruit`, `Vanilla`)
`item`	any combination of `Goods.Flavor` and `Goods.Food` (e.g., `Chocolate Cookie`, `Grapefruit Juice`, `Opera Cake`)

For example, to output sales data for chocolate cakes by individual location, sorted in descending order by total number of cookies sold, issue the following command:

```
java LocationRollUpSlice location item Chocolate Cookie 2 0 d
```

To output sales of drinks by state sorted in descending order by the average number of receipts per store, issue the command:

```
java LocationRollUpSlice state type Drink 1 1
```

Note: When `<sliceLevel>`= `Item`, the value of `<SliceValue>` will actually occupy two arguments, not one.

Roll Up by Time and Location

Output sales data by time and, optionally, by location.

```
java TimeRollUp <TimeRollUp-Level>  [<LocationRollUp-Level>]
                [<sort>]  [<sortModifier>] [<direction>]
```

<LocationRollUp-Level> is the same as <rollup-Level> for LocationRollUp; if not specified, report total sales data.

<TimeRollUp-Level>: day, week, month, year, dow (day of week - weekday sales vs. weekend sales).

<sort> and <direction> are all same as for LocationRollUp, except, for <sort> the value of 0 means sort in chronological order.

<sortModifier>: 0 = use totals; 1 = use averages per day.

Slice by Time and Employee

Output sales data for a specific category of employees for a designated time slice. Time is aggregated at the level of one month.

```
java EmployeeSlice [<EmployeeCategory>] [<TimeSliceStart>] [<TimeSliceEnd>]
```

<EmployeeCategory>: Part, Full, Barista, Cashier, Manager. (note: Manager includes both store managers and shift managers). If absent, aggregate for each value of Employee.Position.

<TimeSliceStart>, <TimeSliceEnd>: January, Februrary, March, April, May, June, July, August, September, October, November, December. When only one is present, output information for given month only. When none are present, output information for all months.

Output in the chronological order of months.

To find sales data for the entire year for full-time employees:

```
java EmployeeSlice Full
```

To find sales data for summer months for managers:

```
java EmployeeSlice Manager June August
```

To find sales data for all employee positions for December:

```
java EmployeeSlice December
```

Rank Stores

Find the rank of individual stores within given sales situation.

```
java StoreRank <LocationId> <criterion> [<GeoContext>] [<Month>] [<Flavor>] [<Food>]
```

<LocationId>: unique id of the specific location.

<criterion>: 1 = total number of sales; 2 = total number of sold items; 3 = total amount of sales.

<GeContext>: city, state: find rank among the stores within the same geographical context (city, or state). If omitted, find rank among all stores.

<Month>: January, Februrary, March, April, May, June, July, August, September, October, November, December. Find rank for specific month of sales. If omitted, find rank for overall sales.

<Flavor>, <Food> take values from Goods.Flavor, Goods.Food respectively. Find rank for sales of specified categories of items. Additionally, <Flavor> can take values Drink or Pastry (in such a case, no <Food> value will be specified). If both are omitted, use sales data for all types of sold items.

To find the rank of store number 1 in chocolate cake sales ($$) in the state for the entire year, use

```
java StoreRank 1 3 state Chocolate Cake
```

To find the rank of store number 1 in total number of drinks sold for the month of July, use

```
java StoreRank 1 2 July Drink
```

Program Output

We have not provided exact instructions on how you should format the output of your programs. Instead, this section contains examples of output provided by the T.A.'s implementation. You are encouraged to output results in a similar manner. Formatting results into columns has the benefit of producing readable output that can later be collected and imported into a spreadsheet or used for further processing.

Note, that this document does not include **all possible** output from the programs, only samples.

The results of running a number of tests on the T.A.'s implementations will be made available to you on the course web page.

LocationRollUp

```
java LocationRollUp city 3 0

city , Number of Sales , Total number of sold items , Total $$ made
Los Angeles     10838                   32563                   132653.04
Flagstaff       3794                    11444                   47212.52
Seattle  3747                   11385                   46757.83
Sacramento      3716                    11225                   44969.38
```

Bakersfield	3647	10980	44144.01
San Luis Obispo	3546	10532	44118.12
Santa Barbara	3517	10620	43895.68
Olympia	3471	10540	43566.12
Reno	3539	10607	43100.83
Monterey	3562	10740	42924.11
Stockton	3481	10444	42884.59
Phoenix	3414	10395	42837.68
San Jose	3526	10615	42647.26
San Francisco	3564	10671	42579.21
Portland	3612	10757	42416.7
Las Vegas	3473	10449	42063.7
Tucson	3465	10373	41180.61
San Diego	3218	9626	38733.51

LocationRollUpSlice

```
java LocationRollUpSlice location item Chocolate Cake 2 0 d
```

Store	Number of Sales	Total number of sold items	Total $$ made
19	427	1458	13049.1
2	388	1269	11357.55
8	331	1015	9084.25
18	306	974	8717.3
11	281	947	8475.65
14	337	941	8421.95
4	328	929	8314.55
5	284	877	7849.15
13	265	781	6989.95
17	221	638	5710.1
3	174	510	4564.5
7	154	467	4179.65
16	154	463	4143.85
12	140	460	4117
10	142	437	3911.15
15	136	413	3696.35
9	128	394	3526.3
20	123	382	3418.9
1	111	341	3051.95
6	93	271	2425.45

TimeRollUp

```
java TimeRollUp month location 0 0 d
```

Month	location	Number of Sales	Total number of sold items	Total $$ made
12	1	347	1024	3874.22
11	1	455	1374	5341.69
10	1	403	1289	5767.98
09	1	470	1412	5993.24
08	1	311	928	4240
07	1	362	1077	4627.11
06	1	405	1243	4987.83

```
05      1       326                 966                 3859.05
04      1       399                 1161                4606.98
03      1       287                 893                 3066.49
02      1       300                 867                 3204.77
01      1       305                 901                 3292.91
01      2       13058               38597               170841.59
01      3       5480                16530               69477.39
01      4       13769               41344               171159.89
01      5       12733               38526               164649.93
01      6       5172                15401               62088.4
...
```

EmployeeSlice

```
java EmployeeSlice Full January May

Employee Type, Month, Number of Sales, Total # of sold items, Total $$ made
Barista         01              5940            18015           75775.75
Cashier         01              12970           38983           158050.76
Manager         01              144009          433372          1828000.03
Shift Manager   01              9190            27697           112812.66
Manager         02              300             867             3204.77
Manager         03              287             893             3066.49
Manager         04              399             1161            4606.98
Manager         05              326             966             3859.05
```

StoreRank

```
java StoreRank 1 3 state Chocolate Cake

Rank of location 1: 11 out of 20
```

Submission Instructions

You will use the handin tool to submit your files. Each groups submits exactly one copy of all materials. You will submit the following files:

- `README`. Shall contain the name of the group, and the names and email addresses of all students in the group.
- design. The files containing the design of your data warehouse. This includes textual description of the warehouse schema, the diagram of the data warehouse and the DDL/DML SQL commands.
- `LocationRollUp.java`, `LocationRollUpSlice.java`, `TimeRollUp.java`, `EmployeeSlice.java` and `StoreRank.java`.

To submit use the following command:

```
$ handin dekhtyar-grader lab01 lab01.zip
```