Data Warehouse
Lab Module 1
Deadline: 19.10.2012

1. (1pts) Choose the domain of the data warehouse. Describe an example company and the reason why a data warehouse is needed. (max 500 words)

   The domain of my data warehouse should include the data of a company on the beverage market. Let's call the company "drinks.com". It consists of 200 stores spread over 6 countries in Europe. Each store, depending on its location, has different suppliers and customers. So each store has to deal with customer, supplier and product data and has to store them in a database.
   Now management would like to have some information about the overall sales and know if the advertising reaches the possible new customers.
   Example:
   - "How many bottles of Coke have we sold in Italy during the last month?"
   - "Do we sell more beer in Bolzano or Trento?"
   - "Which customers have spent more than 50€ within 4 weeks during the last 3 years after receiving a promotion newsletter?"

   A data warehouse is needed, because the company wants to optimise its business. The data warehouse will help to make decisions. Since the data, which provide a data warehouse, are related to do analytics, we can perform advanced queries that help us deciding to do something or not. Example: to raise the stock of a product or to drop the product.

2. (2pts) What are the business processes that you want to model and what business questions should they help to answer. Consider minimum two business processes. Which granularity level is appropriate for the described processes? (max 500 words)

   Business process 1: Sales

   Business questions:
   a. How many bottles of Coke have we sold in Italy last month?
   b. What are the daily receipts per country?
   a. Who are the 5 customers, which spent at most?
   b. How many customers have used a promotion in the last month?

   Granularity
   Low granularity is needed since we have individual line items on a bill. We are talking about event facts. It is possible to aggregate them over time (SUM, AVG, …)

   Product by Customer by Store by Promotion by Day
   (product, customer, store, promotion, date, number of units, amount)

   Dimensions: product, customer, store, promotion, date
   Measures: number of units, amount

   Business process 2: Order

   Business questions:
   a. What is the total amount of energy drinks ordered from a specific supplier every year?
   b. What is the daily total amount of mineral water ordered last month?

   Granularity
   (product, date, supplier, ordered quantity, receipts, discount)

   Dimensions: product, date, supplier
   Measures: ordered quantity, receipts, discount

Business process 3: Stock of Inventory

Business questions:

a. Which product stocks did run out at least once last week at the same time in every warehouse?
b. What is the average quantity of beer each warehouse orders among their suppliers every month?

Granularity

The granularity doesn't have do be so detailed, because here we have longer time spans (weeks).

(product, date, warehouse, stocked quantity)

Dimensions: product, date, supplier
Measures: stocked quantity

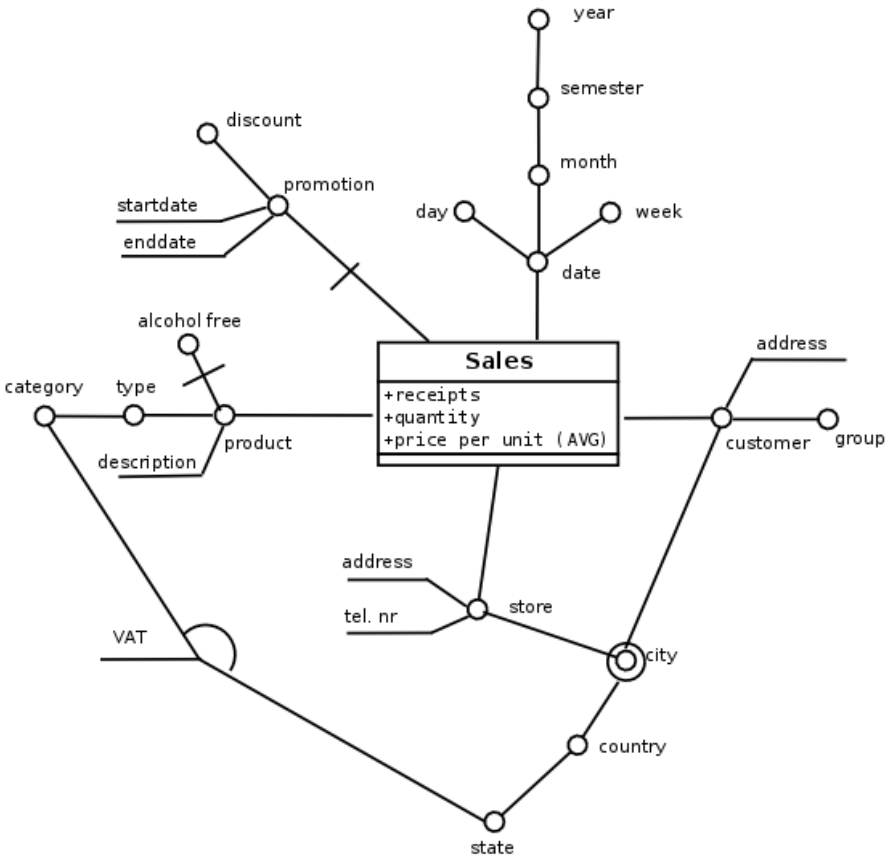
Data Warehouse
Module 1 Task 2
Deadline: 29.10.2012

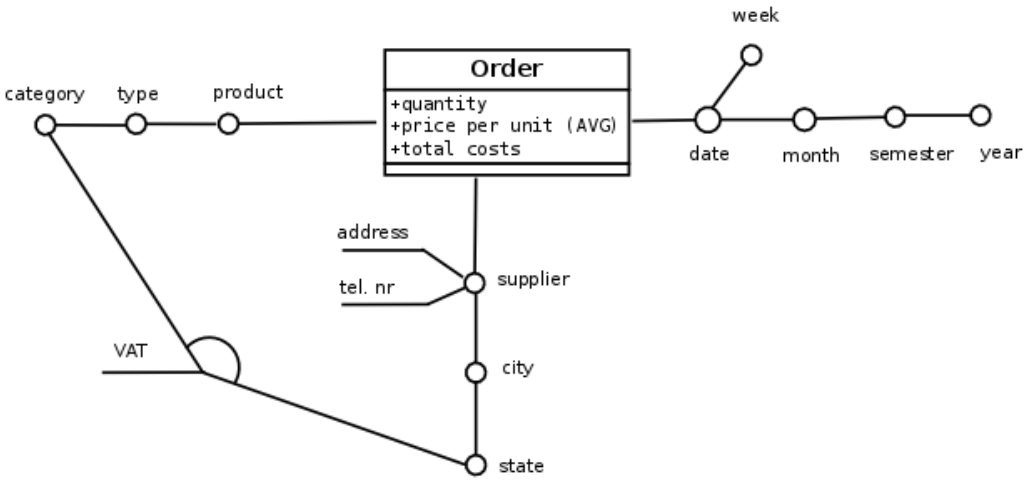
1. (1pt) Choose minimum two facts that you want to model and describe them (max 250 words). Write what do the facts represent, what are their dimensions and measures. To make the facts complicated enough follow the guidelines:
    a. One fact should have minimum four dimensions.
    b. One fact should have minimum two measures with different additivity properties.

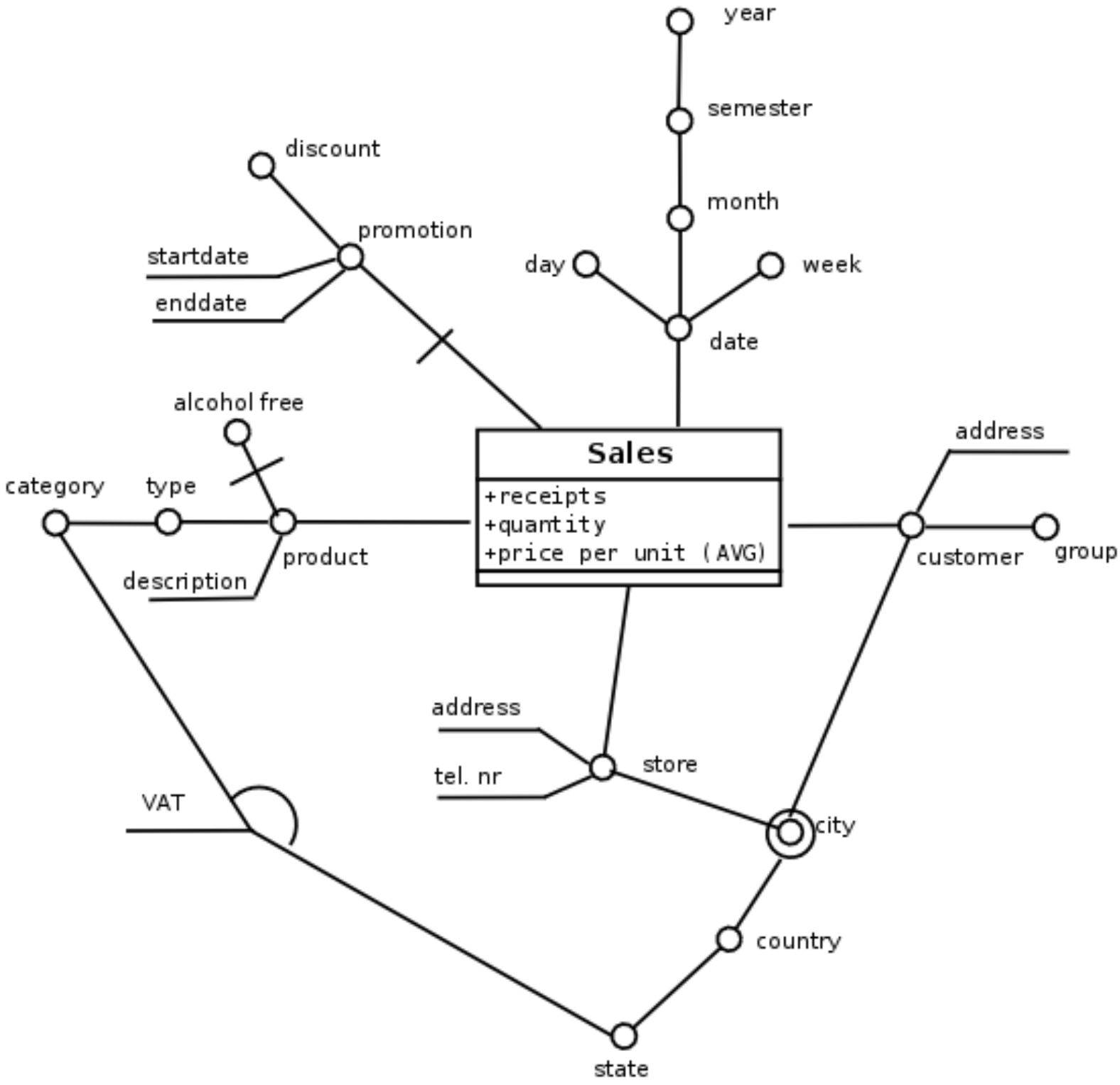
The first fact I choose is the "**Sales Fact**", which helps to decide about different selling strategies. Date, Product, Store, Customer and Promotion are defined as dimensions. The dimension *promotion* is tagged as optional like the attribute *alcohol free* of the dimension product. The attributes *address, tel. nr, description, startdate* and *enddate* cannot be used for aggregation and are tagged as non-dimension descriptive attributes. The fact attribute *receipts* has to be calculated (SUM(quantity*price)). The fact attribute *quantity* is added (Count(Sales)). *Price per unit* is the average price of all given products. The "*VAT*" is a cross-dimensional attribute since its value is defined by the combination of the products category and the state. Finally, there is the *city* attribute, which is defined as a shared hierarchy.

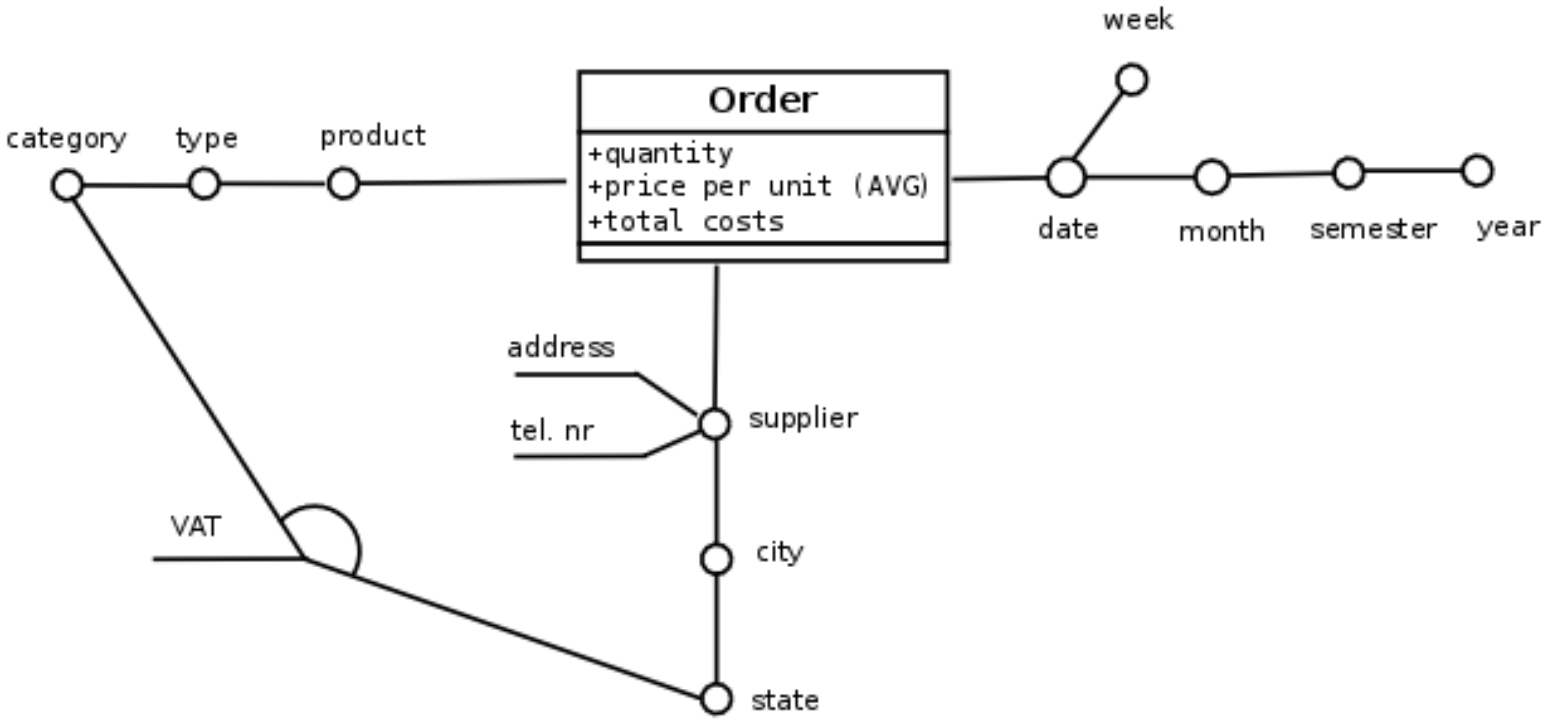
The second fact is the "**Order Fact**". With the order fact it is possible to analyse the orders of different suppliers and to see whether it is better to negotiate with a supplier or to change to another one. The "Order Fact" has 3 dimensions: Product, Date and Supplier. Descriptive attributes are the *address* and the *telephone number* of the supplier. The fact attribute *total costs* has to be calculated (SUM(quantity*price)). The fact attribute *quantity* is added (Count(Orders)). *Price per unit* is the average price of all given products. The "*VAT*" is a cross-dimensional attribute since its value is defined by the combination of the products category and the state.

2. (4pts) Draw fact schema in the DFM for all the chosen facts. Mark properly on the schema the following concepts: fact, dimensions with hierarchies, measures, descriptive attributes, non-additivity of the measures, cross-dimensional attributes, convergence, shared hierarchies, multiple arcs, optional arcs, recursive hierarchies.





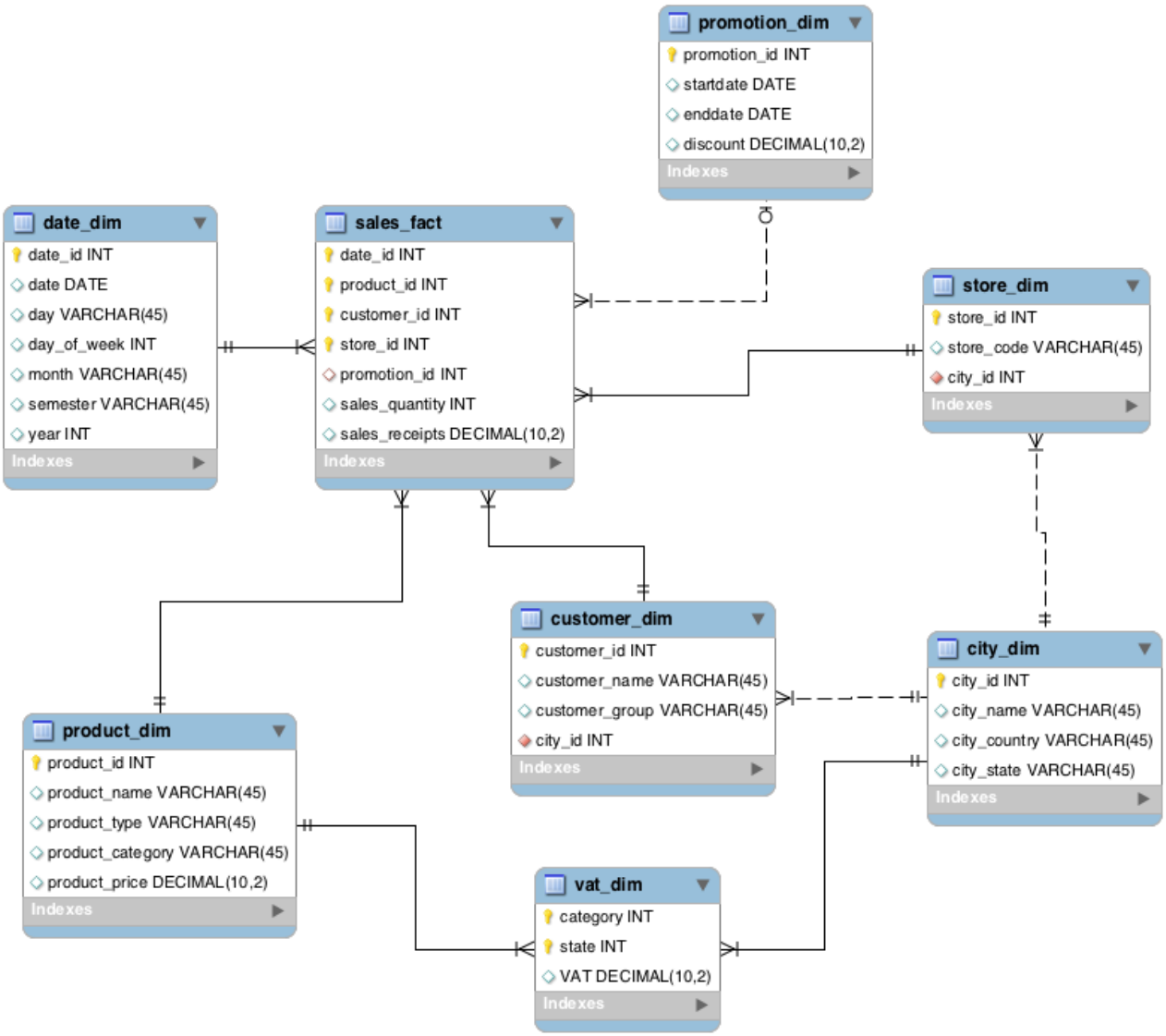

year
semester
month
day
week
date
discount
promotion
startdate
enddate
alcohol free
category
type
product
description
Sales
+receipts
+quantity
+price per unit (AVG)
address
customer
group
address
tel. nr
store
VAT
city
country
state


week
Order
+quantity
+price per unit (AVG)
+total costs
category
type
product
date
month
semester
year
address
tel. nr
supplier
VAT
city
state

Data Warehouse
Module 1 Task 3
Deadline: 09.11.2012

1. (2pts) Draw a star schema (or snowflake schema if it suits better) for your facts (choose main two) including primary/foreign key relations, attributes and cardinalities. For clarity, do not consider too many descriptive attributes from the DFM schema, e.g. name is enough. Describe all the non-trivial choices that you made, e.g. how did you model a multiple arc or a recursive hierarchy.

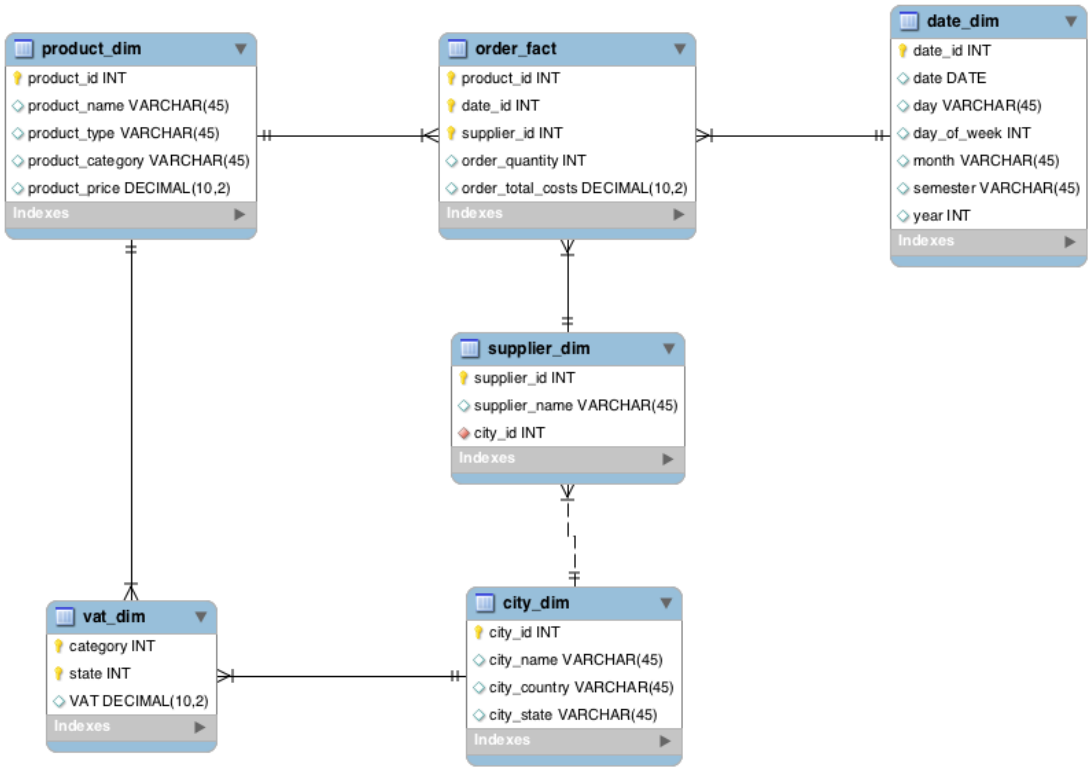
Snowflake Schema for the SALES Fact



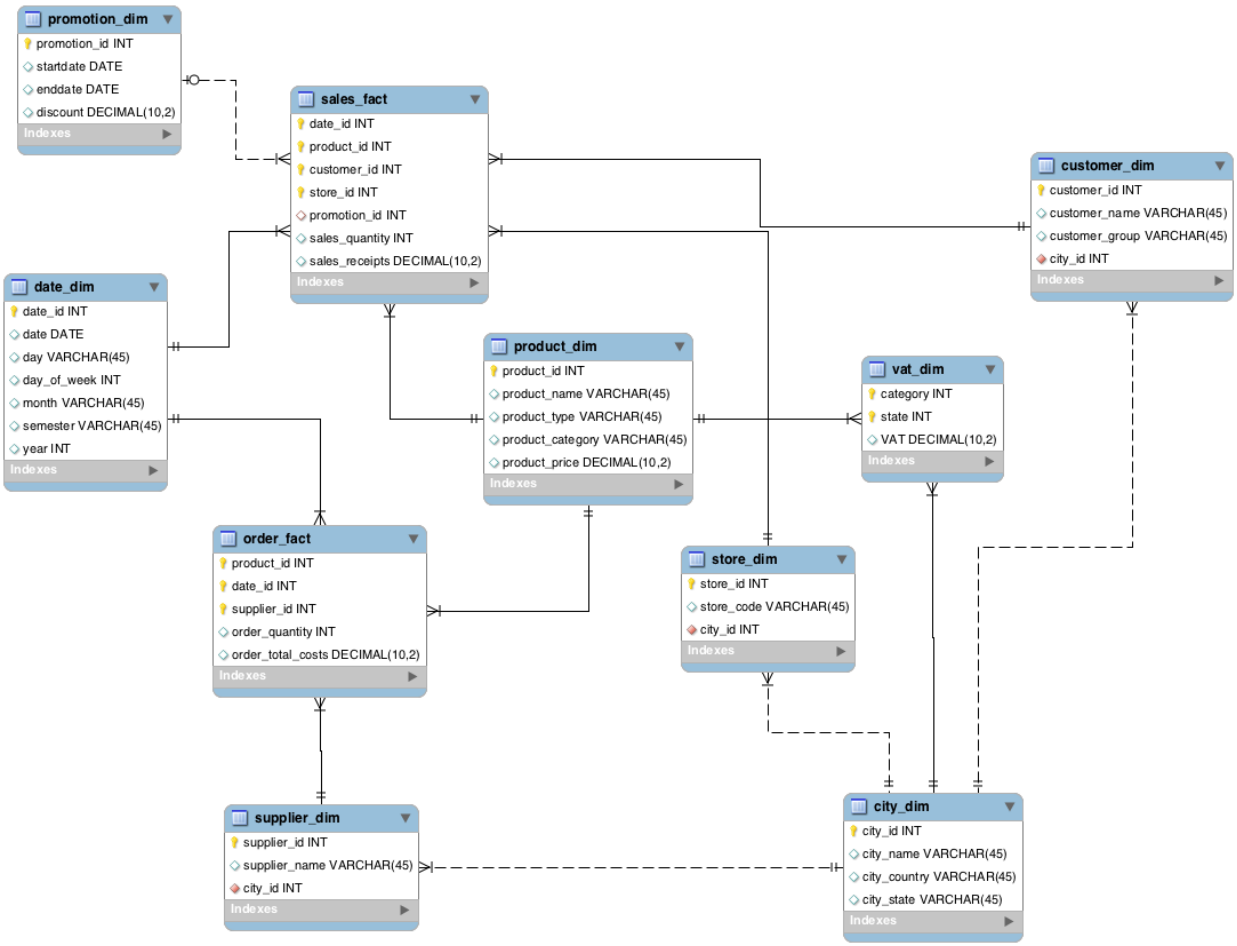
The cross-dimensional attribute *VAT* defines a many to many association between the city dimension and the product dimension. So I have to insert a new table, which primary key is composed by the attribute category (foreign key to product_category) and state (foreign key to city_state).

In addition I have to model a shared hierarchy with partial sharing for the city dimension. To avoid redundancy I insert a new Table. A disadvantage of that solution could be that additional costs for queries will appear, since more joins have to be done.

Snowflake Schema for the ORDER Fact



The combination of the two schemas



2. (3pts) Choose two of the business questions that you find important (one for each fact) and write SQL queries that solve them. Draw sample instances of the tables that are involved in the queries (several rows for the dimension tables and up to 15 rows for the fact tables). Show the results of the queries on the instances.

   a) What are the daily receipts per country?

   Involved tables: date_dim, sales_fact, store_dim, city_dim

   Table: date_dim

※date_id	date	day	day_of_week	month	semester	year
1	2010-01-21	Thursday	4	January	1	2010
2	2010-02-12	Friday	5	February	1	2010
3	2011-05-23	Monday	1	May	1	2011
4	2011-08-11	Thursday	4	August	2	2011
5	2011-01-02	Tuesday	2	January	1	2011
6	2012-11-10	Saturday	6	November	2	2012
7	2012-04-04	Wednesday	3	April	1	2012

   Table: sales_fact

※date_id	※product_id	※customer_id	※store_id	promotion_id	sales_quantity	sales_receipts
1	5	1	3	5	2	15.00
1	9	10	2	3	5	8.25
2	3	2	4	5	2	3.50
2	4	4	1	5	4	10.00
2	7	2	3	3	1	35.20
2	7	6	4	1	10	352.00
3	1	3	4	2	4	13.00
3	1	8	2	1	5	16.25
3	5	6	1	1	7	52.50
3	6	9	1	4	9	13.95
3	8	8	3	4	5	118.00
3	8	10	3	4	10	236.00
3	9	5	3	3	6	9.90
4	2	9	1	4	1	1.55
4	3	5	3	3	2	3.50
4	7	10	1	4	1	35.20
5	3	4	5	1	2	3.50
5	3	10	4	5	4	7.00
5	5	4	5	2	3	24.50
5	9	3	5	2	5	8.25
6	1	3	2	1	6	19.50
6	2	4	1	5	3	4.65
6	5	3	5	1	7	52.50
6	8	1	5	1	2	47.20
6	10	6	1	2	3	39.75
7	2	8	1	4	2	3.10
7	3	1	2	2	1	1.75
7	3	4	3	4	3	5.25
7	7	2	4	1	2	70.20
7	10	3	5	1	1	13.25

   Table: store_dim

※store_id	store_code	city_id
1	IT-01	1
2	IT-02	4
3	AT-01	2
4	DE-01	3
5	UK-01	5

Table: city_dim

※city_id	city_name	city_country	city_state
1	Merano	Bolzano	Italy
2	Innsbruck	Tyrol	Austria
3	Munich	Bayern	Germany
4	Milano	Lombardia	Italy
5	London	London	United Kingdom

Query:

```
SELECT d.date, c.city_country, SUM(sales_receipts)
FROM date_dim d, sales_fact sa, store_dim st, city_dim c
WHERE sa.store_id = st.store_id AND st.city_id = c.city_id AND
      sa.date_id = d.date_id
Group By d.date, c.city_country
```

Result:

date	city_country	receipts
2010-01-21	Lombardia	8.25
2010-01-21	Tyrol	15.00
2010-02-12	Bayern	355.50
2010-02-12	Bolzano	10.00
2010-02-12	Tyrol	35.20
2011-01-02	Bayern	7.00
2011-01-02	London	36.25
2011-05-23	Bayern	13.00
2011-05-23	Bolzano	66.45
2011-05-23	Lombardia	16.25
2011-05-23	Tyrol	363.90
2011-08-11	Bolzano	36.75
2011-08-11	Tyrol	3.50
2012-04-04	Bayern	70.20
2012-04-04	Bolzano	3.10
2012-04-04	Lombardia	1.75
2012-04-04	London	13.25
2012-04-04	Tyrol	5.25
2012-11-10	Bolzano	44.40
2012-11-10	Lombardia	19.50
2012-11-10	London	99.70

b) What is the total amount of energy drinks ordered from a specific supplier every year?

Involved tables: date_dim, order_fact, supplier_dim, product_dim

Table: date_dim

※date_id	date	day	day_of_week	month	semester	year
1	2010-01-21	Thursday	4	January	1	2010
2	2010-02-12	Friday	5	February	1	2010
3	2011-05-23	Monday	1	May	1	2011
4	2011-08-11	Thursday	4	August	2	2011
5	2011-01-02	Tuesday	2	January	1	2011
6	2012-11-10	Saturday	6	November	2	2012
7	2012-04-04	Wednesday	3	April	1	2012

Table: order_fact

※product_id	※date_id	※supplier_id	order_quantity	order_total_costs
1	2	1	1000	3250.00
1	3	1	1000	3250.00
1	4	1	100	3250.00
2	1	2	50	77.50
2	3	5	100	155.00
2	6	2	50	77.50
3	2	4	200	465.00
3	5	4	200	465.00
4	1	5	250	625.00
4	4	5	200	500.00
4	6	5	300	750.00
4	7	5	100	250.00
5	3	4	40	3000.00
10	4	5	100	132.50

Table: supplier_dim

※supplier_id	supplier_name	city_id
1	Smith	5
2	Forst	1
3	Mueller	3
4	Agostini	4
5	Pfiff	2

Table: product_dim

※product_id	product_name	product_type	product_category	product_price
1	Fosters	Beer	alcoholic	3.25
2	Coca Cola	Soft Drink	non alcoholic	1.55
3	Forst	Beer	alcoholic	1.75
4	Red Bull	Energy Drink	non alcoholic	2.50
5	Lagrein	Wine	alcoholic	7.50
6	Fanta	Soft Drink	non alcoholic	1.55
7	Jack Daniels	Whiskey	super alcoholic	35.20
8	Barolo	Wine	alcoholic	23.60
9	Sprite	Soft Drink	non alcoholic	1.65
10	Bacardi	Rum	super alcholic	13.25

Query:

```
SELECT d.year, s.supplier_name, SUM(o.order_quantity) As Quantity
FROM date_dim d, order_fact o, supplier_dim s, product_dim p
WHERE d.date_id = o.date_id AND o.supplier_id = s.supplier_id AND
    p.product_id = o.product_id AND p.product_type = "Energy Drink"
Group By d.year, s.supplier_name
```

Result:

year	supplier_name	Quantity
2010	Pfiff	250
2011	Pfiff	200
2012	Pfiff	400

Data Warehouse
Name: Jürgen Tragust
Module 1 Task 4
Deadline: 16.11.2012

1. (2pts) Write an SQL script (name it name_surname_ex4.sql) that creates your data warehouse (fact and dimension tables). Name of each table should start with your initials, e.g. the sales table of Mateusz Pawlik should be named mp_SALES

   => Attached file

2. (2pts) Populate your data warehouse manually or using a data generator: minimum 100 tuples in the fact tables, minimum 10 tuples in the dimension tables (considering a star schema). Add instructions that populate your data warehouse to the script from point 1.

   => Attached file

3. (1pts)Write two queries for your data warehouse: first query using ROLLUP, CUBE or GROUPING SETS operator, second query using GROUPING ID and/or GROUP ID function. Queries, their SQL code, the results and explanation of the results save in name_surname_ex4.pdf file.

   Query 1: Show me the receipts by month, by year and the total receipts.

```
SELECT       d.year, d.month, SUM(sa.sales_receipts)
FROM         jt_store st,jt_sales sa, jt_date d
WHERE        st.store_id = sa.store_id
AND          sa.date_id  = d.date_id
GROUP BY Rollup(d.year, d.month)
ORDER BY     d.year, d.month;
```

	YEAR	MONTH	SUM(SA.SALES_RECEIPTS)
1	2010	February	564
2	2010	January	471,95
3	2010	September	173,7
4	2010	(null)	1209,65
5	2011	April	461,15
6	2011	August	663,95
7	2011	January	657,45
8	2011	September	230,1
9	2011	(null)	2012,65
10	2012	April	199,45
11	2012	August	233
12	2012	January	233,3
13	2012	November	1251,55
14	2012	(null)	1917,3
15	(null)	(null)	5139,6

Query 2: Show me the receipts by year and state with a subtotal for each year and each country and a total of all the receipts.

```
SELECT  DECODE(grouping_id(d.year, c.city_state),
                0, ' country',
                1, 'subtotal year',
                2, 'subtotal state',
                3, 'TOTAL',
               null) AS amount, d.year, c.city_state,
        SUM(sa.sales_receipts) As receipts

FROM          jt_sales sa, jt_date d, jt_store st, jt_city c
WHERE         sa.date_id  = d.date_id
AND           sa.store_id = st.store_id
AND           st.city_id  = c.city_id
GROUP BY Cube(d.year, c.city_state)
ORDER BY      d.year, c.city_state;
```

	AMOUNT	YEAR	CITY_STATE	RECEIPTS
1	country	2010	Austria	436,25
2	country	2010	Germany	375,5
3	country	2010	Italy	261,95
4	country	2010	United Kingdom	135,95
5	subtotal year	2010	(null)	1209,65
6	country	2011	Austria	469,55
7	country	2011	Germany	877,4
8	country	2011	Italy	503,25
9	country	2011	United Kingdom	162,45
10	subtotal year	2011	(null)	2012,65
11	country	2012	Austria	785,6
12	country	2012	Germany	522,9
13	country	2012	Italy	310,4
14	country	2012	United Kingdom	298,4
15	subtotal year	2012	(null)	1917,3
16	subtotal state	(null)	Austria	1691,4
17	subtotal state	(null)	Germany	1775,8
18	subtotal state	(null)	Italy	1075,6
19	subtotal state	(null)	United Kingdom	596,8
20	TOTAL	(null)	(null)	5139,6

Data Warehouse
Name: Jürgen Tragust
Module 1 Task 5
Deadline: 23.11.2012

1. (3pts) Write queries for each of the following points:
   a) one ranking query using NTILE, RANK or DENSE RANK functions,
   b) one windowing query using the windowing clause,
   c) one period-to-period comparison query (a query comparing values across time periods, e.g. compare sales for every week of the current year with the sales of the corresponding weeks in the past year).

   In the answer include queries in natural language, their SQL codes and the results.

   a) Show me the revenues of each state and rank them according to their receipts (from the highest amount to the lowest)

```
SELECT  c.city_state, SUM(sa.sales_receipts) As receipts,
     RANK() OVER (ORDER BY  SUM(sa.sales_receipts) DESC) as RANK
FROM          jt_sales sa, jt_date d, jt_store st, jt_city c
WHERE         sa.date_id  = d.date_id
AND           sa.store_id = st.store_id
AND           st.city_id  = c.city_id
GROUP BY      (c.city_state);
```

CITY_STATE	RECEIPTS	RANK
Germany	1775,8	1
Austria	1691,4	2
Italy	1075,6	3
United Kingdom	596,8	4

   b) Show me the sales per month and the accumulated sales of this year.

```
SELECT  d.year,d.month, d.month_of_year, SUM (sa.sales_receipts) as receipts,
        SUM(SUM(sa.sales_receipts)) OVER (Partition by d.year ORDER BY
        d.month_of_year) as accumulated
FROM    jt_sales sa, jt_date d, jt_store st, jt_city c
WHERE   sa.date_id  = d.date_id
AND     sa.store_id = st.store_id
AND     st.city_id  = c.city_id
GROUP BY d.year, d.month, d.month_of_year;
```

	YEAR	MONTH	MONTH_OF_YEAR	RECEIPTS	ACCUMULATED
1	2010	January	1	471,95	471,95
2	2010	February	2	564	1035,95
3	2010	September	9	173,7	1209,65
4	2011	January	1	657,45	657,45
5	2011	April	4	461,15	1118,6
6	2011	August	8	663,95	1782,55
7	2011	September	9	230,1	2012,65
8	2012	January	1	233,3	233,3
9	2012	April	4	199,45	432,75
10	2012	August	8	233	665,75
11	2012	November	11	1251,55	1917,3

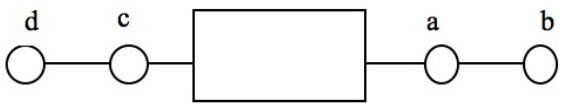
c) Compare each sales date with the date before and the date after.

```
SELECT  d.date_sale, SUM (sa.sales_receipts) as receipts,
      LAG(SUM (sa.sales_receipts),1) OVER (Order By d.date_sale) as LAG,
      LEAD(SUM (sa.sales_receipts),1) OVER (Order By d.date_sale) as LEAD
FROM    jt_sales sa, jt_date d, jt_store st, jt_city c
WHERE   sa.date_id  = d.date_id
AND     sa.store_id = st.store_id
AND     st.city_id  = c.city_id
GROUP BY d.date_sale;
```

	DATE_SALE	RECEIPTS	LAG	LEAD
1	21.01.10	471,95	(null)	93,4
2	01.02.10	93,4	471,95	470,6
3	12.02.10	470,6	93,4	173,7
4	02.09.10	173,7	470,6	657,45
5	02.02.11	657,45	173,7	461,15
6	11.04.11	461,15	657,45	156,55
7	11.08.11	156,55	461,15	507,4
8	23.08.11	507,4	156,55	230,1
9	23.09.11	230,1	507,4	127,2
10	02.01.12	127,2	230,1	106,1
11	30.01.12	106,1	127,2	199,45
12	04.04.12	199,45	106,1	233
13	06.08.12	233	199,45	836,9
14	02.11.12	836,9	233	414,65
15	10.11.12	414,65	836,9	(null)

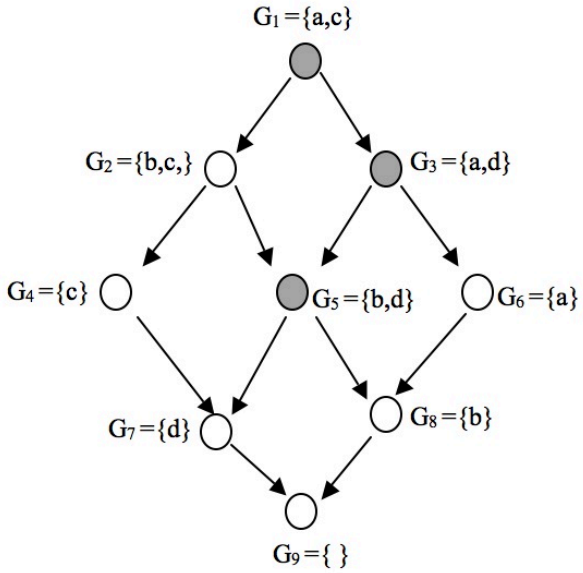
2. (2pts) Choose three queries over the same fact that you think would be frequently executed on your data warehouse. Using the concept of Multidimensional lattice write, which materialized views you should create in order to optimize the query execution. The queries can be simple, but should consider one or more dimensions at different granularity level. You can reduce the dimensions in order to draw the multidimensional lattice (e.g. instead of the entire date hierarchy use only day and year). In the answer include three queries written in natural language and which grouping sets do they consider, simplified fact schema, lattice drawing, what are the candidate views and which of them should be created.

a. Total revenue grouped by store and date
   Candidate view: $G_1 = \{a,c\}$
b. Total number of sold products grouped by year and state.
   Candidate view: $G_5 = \{b,d\}$
c. Total revenue grouped by store and year.
   Candidate view: $G_3 = \{a,d\}$



a => store
b => state
c => date
d => year

Multidimensional lattice:



The candidate views are highlighted in gray

As we see G_1 is the root in this lattice and therefore it can compute also G_3 and G_5. Vice versa it is not possible, because we cannot go for example from year to day. So G_1 has to be created.

Materialized view:

```
CREATE MATERIALIZED VIEW JT_DAY_STORE
ENABLE QUERY REWRITE
AS
SELECT d.date_sale, sa.store_id, SUM(sa.sales_receipts) as Receipts
FROM jt_SALES sa, jt_date d
WHERE sa.date_id = d.date_id
GROUP BY d.date_sale, sa.store_id
ORDER BY Receipts desc;
```

```
CREATE MATERIALIZED VIEW JT_YEAR_STORE
ENABLE QUERY REWRITE
AS
SELECT d.year, sa.store_id, SUM(sa.sales_receipts) as Receipts
FROM jt_SALES sa, jt_date d
WHERE sa.date_id = d.date_id
GROUP BY d.year, sa.store_id
ORDER BY Receipts desc;
```

```
CREATE MATERIALIZED VIEW JT_YEAR_STATE_Quantity
ENABLE QUERY REWRITE
AS
SELECT d.year, c.city_state, SUM(sa.sales_quantity) as Quantity
FROM jt_SALES sa, jt_date d, jt_store st,jt_city c
WHERE sa.date_id = d.date_id AND sa.store_id = st.store_id AND st.city_id =
c.city_id
GROUP BY d.year, c.city_state
ORDER BY Quantity desc;
```

(+1pt) Explain what would you gain/loose (in the query cost, used space and query execution time) by using the queries pointed by the multidimensional lattice. Be specific. Run some experiments to confirm the theoretical results.

3. (2pts) Write a query, which you could speed up by using a bitmap index. Show how will the index look like. In the answer include the query in both, natural language and SQL, description on which attributes should the index be built, drawing of the bitmap index for several rows, SQL of how to build the desired index and SQL of the rewritten query that uses the index.

   a) Show me all non alcoholic products

```
SELECT product_name
FROM   jt_product
WHERE  product_category = "non alcoholic"
```

The index should be built on the product_category attribute since it has low cardinality (values: *alcoholic* and non *alcoholic*)

product_id	product_category	
	alcoholic	non alcoholic
1	1	0
2	0	1
3	1	0
4	0	1
5	1	0
6	0	1
7	1	0
8	1	0
9	0	1
10	1	0
11	1	0
12	0	1

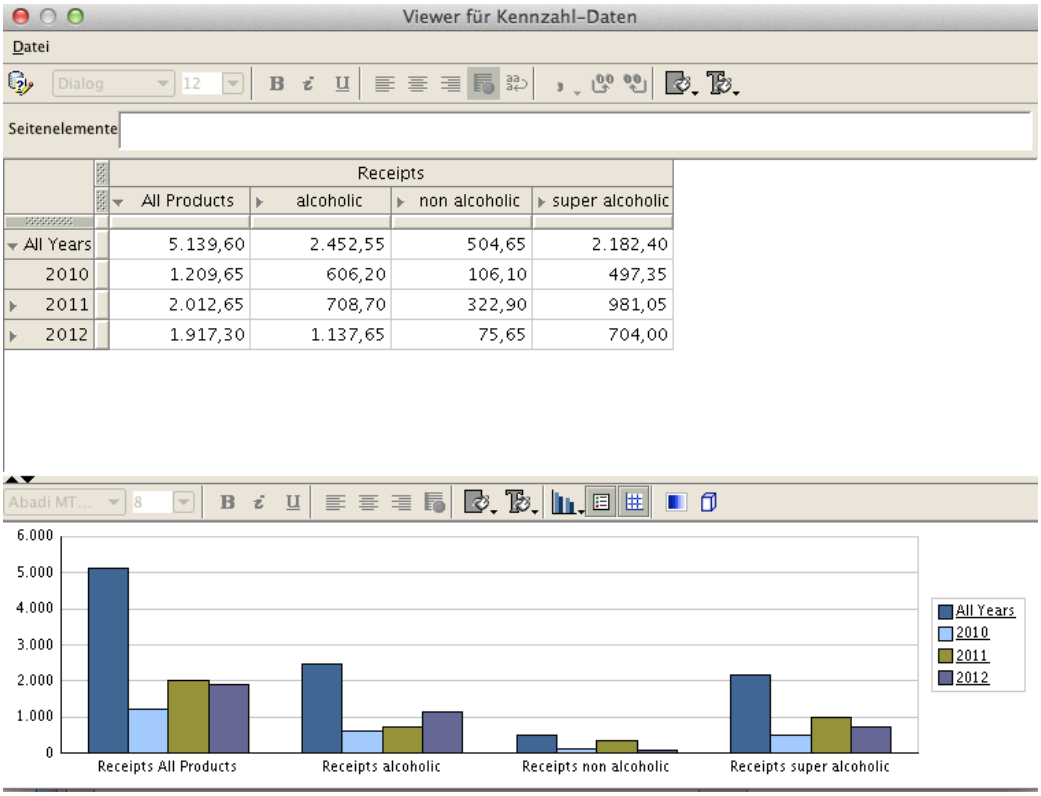
```
CREATE BITMAP INDEX product_category_bi
ON jt_product(product_category);
```

(+1pt) Explain exactly what and how do you gain by using the bitmap index. What is the overhead of creating the index, especially in its size? Be specific. Run some experiments to confirm the theoretical results.

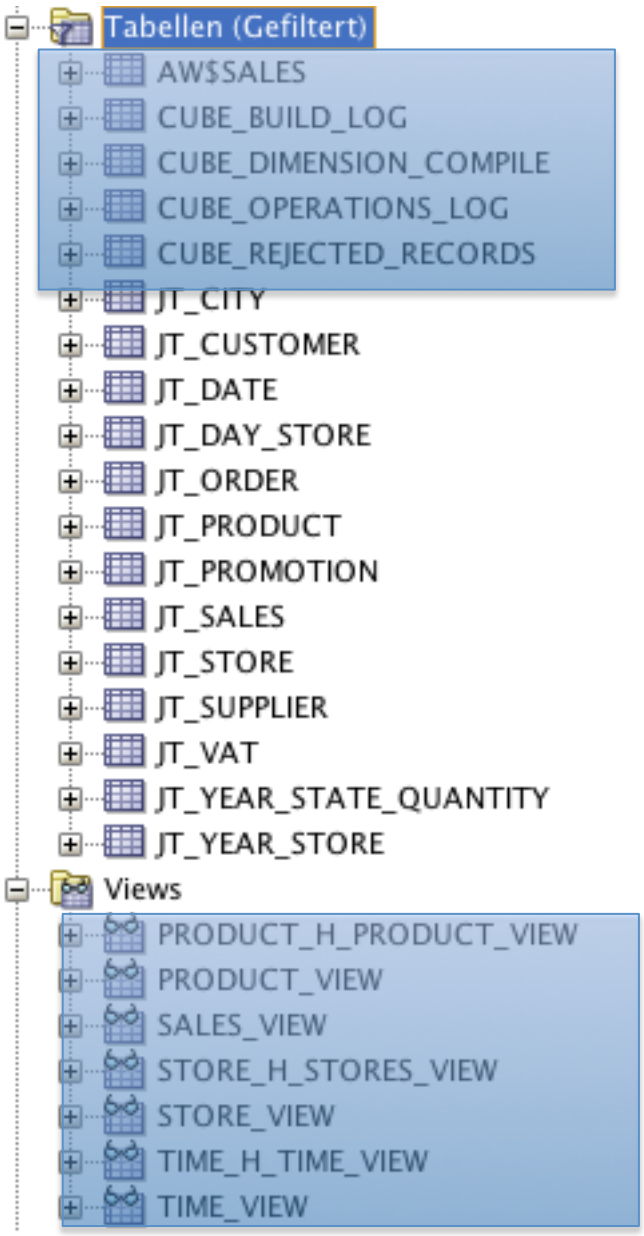
Data Warehouse
Name: Jürgen Tragust
Module 1 Task 6
Deadline: 30.11.2012

1. (3pts) Using AWM create cubes, dimensions, hierarchies, attributes and measures of your data warehouse. Load the cube data. Show the result of the ROLLUP/CUBE query from the Task 4.3 in AWM.

This is the cube, which was possible to create. It shows the receipts by Time and Product



2. (2pts) Write two queries that use OLAP objects created in the Task 5.1. Show the query in natural language, SQL code and the results. You can rewrite the queries from the Task 5.1.



The highlighted tables and views are generated automatically by the AWM and the cube generation.

Now I can use the views instead of joining a lot of views, but I didn't know exactly how to use those for my queries.