POLITECHNIKA GDAŃSKA

Optymalizacja baz danych

Report 3

Oskar Kwidziński

S156013

Informatyka Stosowana

Wydział Fizyki Technicznej i Matematyki Stosowanej

In accordance with commands to Report 3 a database performance tests have been created. The project was created in Oracle database managment system (version 11g) with Oracle SQLDeveloper as a data modeler. It consists of 13 entities and its main purpose is to help manage an online shop with collecting and processing data about employees, sales, supplieres, customers etc. The ER diagram of database is presented in Fig. 1.

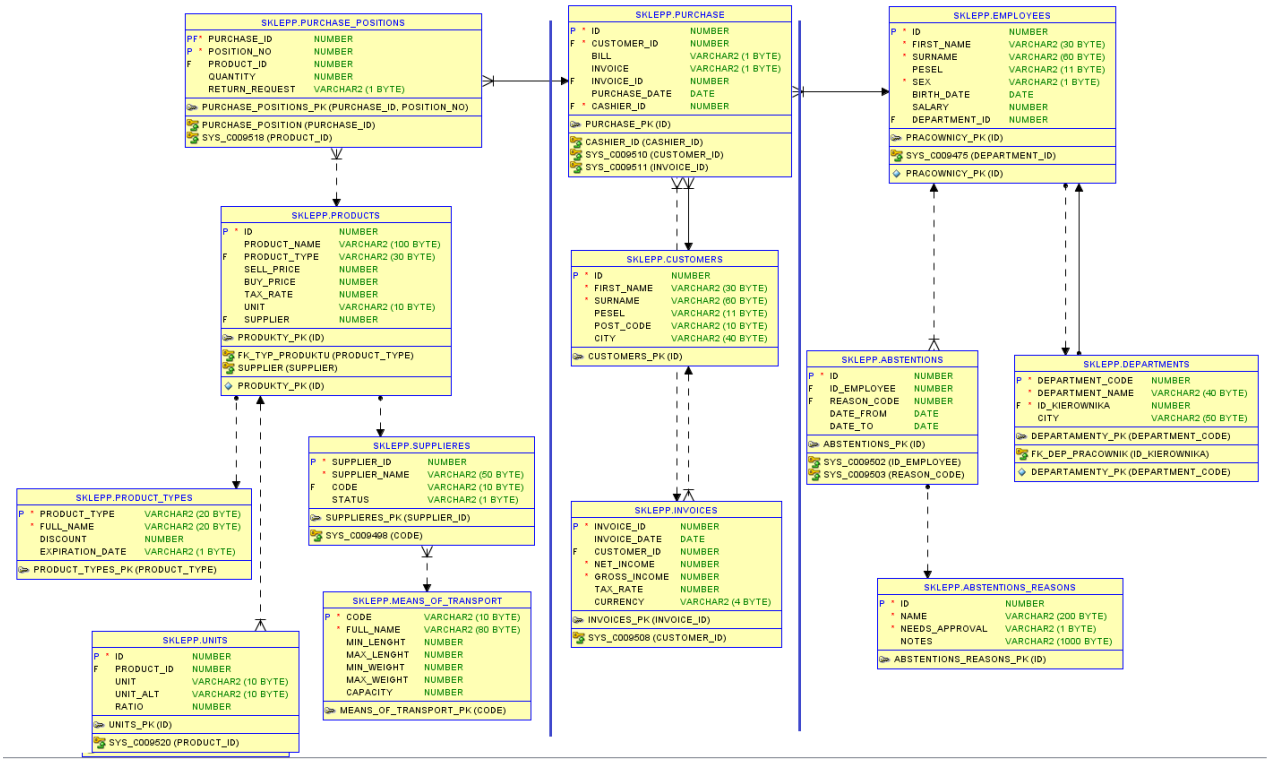

Fig. 1. Schema of the database.

Test has been conducted via implementing five complexed queries (with subqueries, joins etc.). Those queries are associated with managing enterpise database has been created to. Those queries were created twice (original and alternative) and their explaing plans are as follows:

1. Displaying data about bills from cashiers whose salary is higher then cashier no. 14:

1.1.

```
explain plan for
select p.customer_id, p.bill, c.surname
from   purchase p
join   customers  c on (p.customer_id = c.id)
join   employees  e on (e.id = p.cashier_id)
where e.salary >= (select salary from employees where id = 14);
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	81	10 (10)	00:00:01
1	NESTED LOOPS		3	81	9 (12)	00:00:01
2	NESTED LOOPS		3	81	9 (12)	00:00:01
3	MERGE JOIN		3	45	6 (17)	00:00:01
* 4	TABLE ACCESS BY INDEX ROWID	EMPLOYEES	1	7	2 (0)	00:00:01
5	INDEX FULL SCAN	PRACOWNICY_PK	22		1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	EMPLOYEES	1	7	1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	PRACOWNICY_PK	1		0 (0)	00:00:01
* 8	SORT JOIN		6	48	4 (25)	00:00:01
9	TABLE ACCESS FULL	PURCHASE	6	48	3 (0)	00:00:01
* 10	INDEX UNIQUE SCAN	SYS_C009506	1		0 (0)	00:00:01
11	TABLE ACCESS BY INDEX ROWID	CUSTOMERS	1	12	1 (0)	00:00:01

1.2.

```
explain plan for
select p.customer_id, p.bill, c.surname
from   purchase p, customers  c, employees e
where p.cashier_id = e.id and p.customer_id = c.id and e.salary > 4444;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		6	162	9 (12)	00:00:01
* 1	HASH JOIN		6	162	9 (12)	00:00:01
2	MERGE JOIN		6	90	6 (17)	00:00:01
* 3	TABLE ACCESS BY INDEX ROWID	EMPLOYEES	12	84	2 (0)	00:00:01
4	INDEX FULL SCAN	PRACOWNICY_PK	22		1 (0)	00:00:01
* 5	SORT JOIN		6	48	4 (25)	00:00:01
6	TABLE ACCESS FULL	PURCHASE	6	48	3 (0)	00:00:01
7	TABLE ACCESS FULL	CUSTOMERS	9	108	3 (0)	00:00:01

2. Displaying absensions of employees form certain departments:

2.1.

```
explain plan for
select  e.surname, d.department_code, a.date_from
from    employees  e
join    abstentions   a on (a.id_employee = e.id)
join    departments d on (d.department_code = e.department_id)
where     d.head_id <= (select head_id from departments where department_name = 'Zarzad');
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		6	204	19 (6)	00:00:01
1	NESTED LOOPS		6	204	12 (9)	00:00:01
2	NESTED LOOPS		6	204	12 (9)	00:00:01
3	MERGE JOIN		6	162	6 (17)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	EMPLOYEES	22	352	2 (0)	00:00:01
5	INDEX FULL SCAN	PRACOWNICY_PK	22		1 (0)	00:00:01
* 6	SORT JOIN		6	66	4 (25)	00:00:01
7	TABLE ACCESS FULL	ABSTENTIONS	6	66	3 (0)	00:00:01
* 8	INDEX UNIQUE SCAN	DEPARTAMENTY_PK	1		0 (0)	00:00:01
* 9	TABLE ACCESS BY INDEX ROWID	DEPARTMENTS	1	7	1 (0)	00:00:01
* 10	TABLE ACCESS FULL	DEPARTMENTS	1	10	7 (0)	00:00:01

2.2.

```
select  e.surname, d.department_code, a.date_from
from    employees  e, abstentions   a, departments d
where       d.head_id  <=  16  and  a.id_employee  =  e.id  and  d.department_code  =
e.department_id;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		6	204	12 (9)	00:00:01
1	NESTED LOOPS		6	204	12 (9)	00:00:01
2	NESTED LOOPS		6	204	12 (9)	00:00:01
3	MERGE JOIN		6	162	6 (17)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	EMPLOYEES	22	352	2 (0)	00:00:01
5	INDEX FULL SCAN	PRACOWNICY_PK	22		1 (0)	00:00:01
* 6	SORT JOIN		6	66	4 (25)	00:00:01
7	TABLE ACCESS FULL	ABSTENTIONS	6	66	3 (0)	00:00:01
* 8	INDEX UNIQUE SCAN	DEPARTAMENTY_PK	1		0 (0)	00:00:01
* 9	TABLE ACCESS BY INDEX ROWID	DEPARTMENTS	1	7	1 (0)	00:00:01

3. Displaying certain information about products in store:

3.1.

```
explain plan for
select  s.code, p.product_name, pt.product_type
from    supplieres s
join products p on (s.supplier_id = p.supplier)
join product_types pt on (pt.product_type = p.product_type)
where p.product_name = 'whiskey';
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	21	8 (0)	00:00:01
1	NESTED LOOPS		1	21	8 (0)	00:00:01
2	NESTED LOOPS		1	21	8 (0)	00:00:01
* 3	TABLE ACCESS FULL	PRODUCTS	1	14	7 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	SYS_C009394	1		0 (0)	00:00:01
5	TABLE ACCESS BY INDEX ROWID	SUPPLIERES	1	7	1 (0)	00:00:01

3.2.

```
explain plan for
select  s.code, p.product_name, pt.product_type
from    products p
join supplieres s on (s.supplier_id = p.supplier)
join product_types pt on (pt.product_type = p.product_type)
where p.product_name = (select product_name from products where buy_price = 40);
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		82	1722	17 (6)	00:00:01
1	MERGE JOIN		82	1722	10 (10)	00:00:01
2	TABLE ACCESS BY INDEX ROWID	SUPPLIERES	20	140	2 (0)	00:00:01
3	INDEX FULL SCAN	SYS_C009394	20		1 (0)	00:00:01
* 4	SORT JOIN		82	1148	8 (13)	00:00:01
* 5	TABLE ACCESS FULL	PRODUCTS	82	1148	7 (0)	00:00:01
* 6	TABLE ACCESS FULL	PRODUCTS	1	11	7 (0)	00:00:01

4. Displaying information about products sold with lowest product_id:

4.1.

```
explain plan for
select * from
   (
   select * from
   purchase_positions order by product_id ASC
   )
where rownum <4
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	162	4 (25)	00:00:01
* 1	COUNT STOPKEY					
2	VIEW		16	864	4 (25)	00:00:01
* 3	SORT ORDER BY STOPKEY		16	224	4 (25)	00:00:01
4	TABLE ACCESS FULL	PURCHASE_POSITIONS	16	224	3 (0)	00:00:01

4.2.

```
explain plan for
select * from purchase_positions where product_id <5
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	42	3 (0)	00:00:01
* 1	TABLE ACCESS FULL	PURCHASE_POSITIONS	3	42	3 (0)	00:00:01

5. Displaying margin of some products:

5.1.

```
explain plan for
select product_name, (1-(tax_rate/100))*(sell_price - buy_price) marza, full_name from
products
JOIN product_types on products.product_type = product_types.product_type
WHERE product_types.product_type='OWO' and products.sell_price > 14;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	33	8 (0)	00:00:01
1	NESTED LOOPS		1	33	8 (0)	00:00:01
2	TABLE ACCESS BY INDEX ROWID	PRODUCT_TYPES	1	12	1 (0)	00:00:01
* 3	INDEX UNIQUE SCAN	SYS_C009382	1		0 (0)	00:00:01
* 4	TABLE ACCESS FULL	PRODUCTS	1	21	7 (0)	00:00:01

5.2.

```
explain plan for
select product_name, (1-(tax_rate/100))*(sell_price - buy_price) marza, full_name from
products, product_types
WHERE product_types.product_type='OWO' and products.sell_price > 15 and
product_types.product_type = products.product_type;
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	33	8 (0)	00:00:01
1	NESTED LOOPS		1	33	8 (0)	00:00:01
2	TABLE ACCESS BY INDEX ROWID	PRODUCT TYPES	1	12	1 (0)	00:00:01
* 3	INDEX UNIQUE SCAN	SYS_C009382	1		0 (0)	00:00:01
* 4	TABLE ACCESS FULL	PRODUCTS	1	21	7 (0)	00:00:01

All the displaying of plan table has been achieved via:

```
SELECT *
FROM table(DBMS_XPLAN.DISPLAY (FORMAT=>'ALL +OUTLINE'));
```

There were no large differences in the perfomance of the database due to factors such as the load (cost) or the execution time of querries. In overall database is performing good since both multithreaded and performance tests have been passed. However, presented database is rather of small size so issues can occur while expanding.