Отчет по индивидуальной задаче 5

выполнила

Новикова Дарья Романовна

5130203/10101

# Условие

Используя схему HR, вывести фамилии сотрудников, их оклад, идентификатор отдела и плотный ранг окладов в пределах отдела, в котором работает сотрудник. Ранг 1 должен быть у сотрудника с максимальной зарплатой. Сотрудник с зарплатой, величина которой не определена должен иметь максимальный ранг.

Задачу решить без использования аналитических функций и рекурсивных запросов.

Используя SQL Tuning Adviser и SQL Access Advisor без использования EM,  получить рекомендации по оптимизации запроса и реализовать полученные результаты.

Проанализировать планы выполнения запросов и оценить изменение производительности.

**Примечание:** Перед каждым выполнением SQL запроса необходимо выполнять очистку Buffer Cache и Shared Pool

# Решение

## План решения

1. Разблокировка схемы HR
2. Формирование SQL-запроса
3. Получение плана выполнения
4. Оптимизация с помощью SQL Tuning Advisor
5. Оптимизация с помощью SQL Access Advisor
6. Повторный анализ плана и времени выполнения

## Разблокировка пользователя HR

Разблокируем пользователя HR и задаем пароль, чтобы использовать данные из схемы HR.

ALTER USER hr IDENTIFIED BY hr ACCOUNT UNLOCK;

## Составление запроса

Set timing on

WITH MAXSAL AS (

SELECT DEPARTMENT\_ID, MAX(SALARY) MSALARY

FROM HR.EMPLOYEES

GROUP BY DEPARTMENT\_ID

),

BASE AS (

SELECT DEPARTMENT\_ID, SALARY, RTT

FROM (

SELECT DEPARTMENT\_ID, SALARY, ROWNUM RTT

FROM (

SELECT DISTINCT DEPARTMENT\_ID, SALARY

FROM HR.EMPLOYEES

ORDER BY DEPARTMENT\_ID, SALARY DESC NULLS LAST

)

)

),

RANKED\_SALARIES AS (

SELECT F.DEPARTMENT\_ID, F.SALARY, F.RTT, S.MIDX, F.RTT-S.MIDX+1 RANC

FROM BASE F

JOIN (

SELECT DEPARTMENT\_ID, MSALARY, MAX(IDXMAX) AS MIDX

FROM (

SELECT DEPARTMENT\_ID, SALARY, RTT, MSALARY,

CASE WHEN SALARY = MSALARY THEN RTT END AS IDXMAX

FROM (

SELECT BASE.DEPARTMENT\_ID, BASE.SALARY, BASE.RTT, MAXSAL.MSALARY

FROM BASE

JOIN MAXSAL ON BASE.DEPARTMENT\_ID = MAXSAL.DEPARTMENT\_ID

)

)

WHERE IDXMAX IS NOT NULL

GROUP BY DEPARTMENT\_ID, MSALARY

) S ON F.DEPARTMENT\_ID = S.DEPARTMENT\_ID

),

MAX\_RANK AS (

SELECT DEPARTMENT\_ID, MAX(RANC) AS MAX\_RANC

FROM RANKED\_SALARIES

GROUP BY DEPARTMENT\_ID

)

SELECT A.DEPARTMENT\_ID, A.LAST\_NAME, NVL(TO\_CHAR(A.SALARY), ' ') AS SALARY,

COALESCE(B.RANC, M.MAX\_RANC) AS SALARY\_RANC

FROM HR.EMPLOYEES A

LEFT JOIN RANKED\_SALARIES B

ON A.DEPARTMENT\_ID = B.DEPARTMENT\_ID

AND A.SALARY = B.SALARY

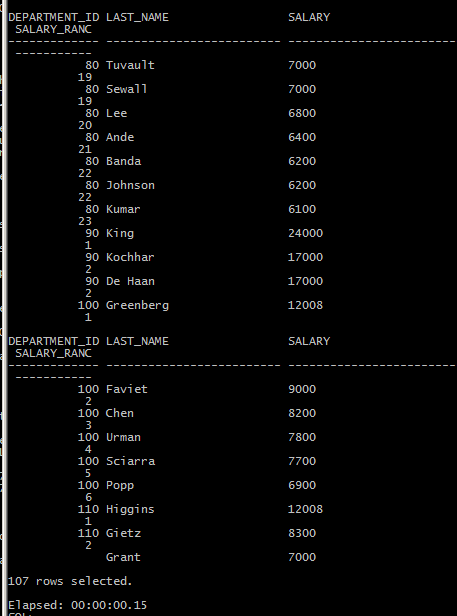
LEFT JOIN MAX\_RANK M

ON A.DEPARTMENT\_ID = M.DEPARTMENT\_ID

WHERE A.SALARY IS NOT NULL

OR (A.SALARY IS NULL AND B.SALARY IS NULL)

ORDER BY A.DEPARTMENT\_ID, SALARY\_RANC NULLS LAST;



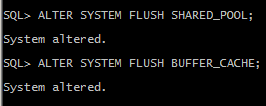
Врем запроса 2:11

## Очистка памяти перед выполнением

Перед выполнением запроса очищаем Shared Pool и Buffer Cache, чтобы исключить влияние кэширования на время выполнения:

ALTER SYSTEM FLUSH SHARED\_POOL;

ALTER SYSTEM FLUSH BUFFER\_CACHE;



## Получение плана выполнения

Составим план выполнения запроса для анализа его структуры и оценки затрат:

explain plan for

WITH MAXSAL AS (

SELECT DEPARTMENT\_ID, MAX(SALARY) MSALARY

FROM HR.EMPLOYEES

GROUP BY DEPARTMENT\_ID

),

BASE AS (

SELECT DEPARTMENT\_ID, SALARY, RTT

FROM (

SELECT DEPARTMENT\_ID, SALARY, ROWNUM RTT

FROM (

SELECT DISTINCT DEPARTMENT\_ID, SALARY

FROM HR.EMPLOYEES

ORDER BY DEPARTMENT\_ID, SALARY DESC NULLS LAST

)

)

),

RANKED\_SALARIES AS (

SELECT F.DEPARTMENT\_ID, F.SALARY, F.RTT, S.MIDX, F.RTT-S.MIDX+1 RANC

FROM BASE F

JOIN (

SELECT DEPARTMENT\_ID, MSALARY, MAX(IDXMAX) AS MIDX

FROM (

SELECT DEPARTMENT\_ID, SALARY, RTT, MSALARY,

CASE WHEN SALARY = MSALARY THEN RTT END AS IDXMAX

FROM (

SELECT BASE.DEPARTMENT\_ID, BASE.SALARY, BASE.RTT, MAXSAL.MSALARY

FROM BASE

JOIN MAXSAL ON BASE.DEPARTMENT\_ID = MAXSAL.DEPARTMENT\_ID

)

)

WHERE IDXMAX IS NOT NULL

GROUP BY DEPARTMENT\_ID, MSALARY

) S ON F.DEPARTMENT\_ID = S.DEPARTMENT\_ID

),

MAX\_RANK AS (

SELECT DEPARTMENT\_ID, MAX(RANC) AS MAX\_RANC

FROM RANKED\_SALARIES

GROUP BY DEPARTMENT\_ID

)

SELECT A.DEPARTMENT\_ID, A.LAST\_NAME, NVL(TO\_CHAR(A.SALARY), ' ') AS SALARY,

COALESCE(B.RANC, M.MAX\_RANC) AS SALARY\_RANC

FROM HR.EMPLOYEES A

LEFT JOIN RANKED\_SALARIES B

ON A.DEPARTMENT\_ID = B.DEPARTMENT\_ID

AND A.SALARY = B.SALARY

LEFT JOIN MAX\_RANK M

ON A.DEPARTMENT\_ID = M.DEPARTMENT\_ID

WHERE A.SALARY IS NOT NULL

OR (A.SALARY IS NULL AND B.SALARY IS NULL)

ORDER BY A.DEPARTMENT\_ID, SALARY\_RANC NULLS LAST;

Для удобства увеличим размеры:

set long 65536

set longchunksize 65536

set linesize 100

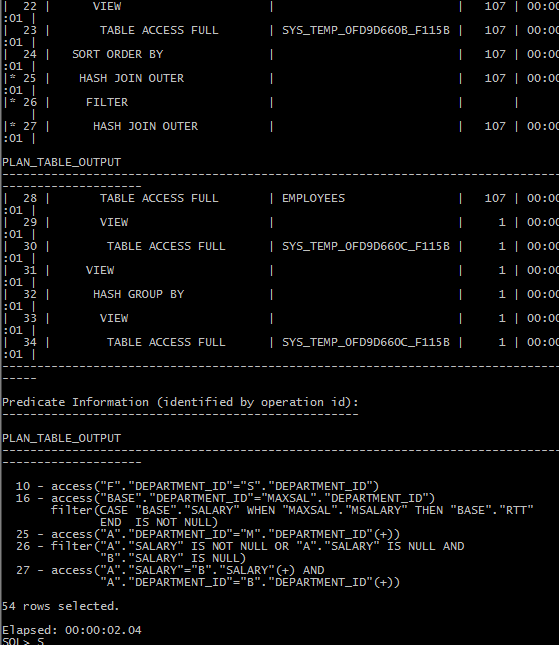
Выведем план:

select plan\_table\_output from table(dbms\_xplan.display('plan\_table',null,'typical -cost -bytes'));

'plan\_table' — имя таблицы, в которой хранится план (по умолчанию PLAN\_TABLE);

NULL — означает, что выводится последний план без указания идентификатора;

'typical -cost -bytes' — тип отображения: стандартный набор информации без показа стоимости (COST) и объема данных (BYTES), чтобы план был компактнее.



Основные операции в плане:

SELECT STATEMENT: Основная операция выборки данных.

TEMP TABLE TRANSFORMATION: Временные таблицы, используемые для промежуточных вычислений.

LOAD AS SELECT: Загрузка данных в временные таблицы.

VIEW: Операции с представлениями.

COUNT: Подсчет строк.

SORT UNIQUE: Уникальная сортировка.

TABLE ACCESS FULL: Полный доступ к таблице.

HASH JOIN: Соединение таблиц с использованием хеш-таблиц.

SORT GROUP BY: Группировка данных.

FILTER: Фильтрация данных.

## Оптимизация с помощью SQL Tuning Advisor

Создаем задачу SQL Tuning Advisor для получения рекомендаций по улучшению запроса:

SET SERVEROUTPUT ON;

DECLARE

l\_task\_id VARCHAR2(100);

BEGIN

l\_task\_id := DBMS\_SQLTUNE.CREATE\_TUNING\_TASK (

sql\_text => 'WITH MAXSAL AS (

SELECT DEPARTMENT\_ID, MAX(SALARY) MSALARY

FROM HR.EMPLOYEES

GROUP BY DEPARTMENT\_ID

),

BASE AS (

SELECT DEPARTMENT\_ID, SALARY, RTT

FROM (

SELECT DEPARTMENT\_ID, SALARY, ROWNUM RTT

FROM (

SELECT DISTINCT DEPARTMENT\_ID, SALARY

FROM HR.EMPLOYEES

ORDER BY DEPARTMENT\_ID, SALARY DESC NULLS LAST

)

)

),

RANKED\_SALARIES AS (

SELECT F.DEPARTMENT\_ID, F.SALARY, F.RTT, S.MIDX, F.RTT-S.MIDX+1 RANC

FROM BASE F

JOIN (

SELECT DEPARTMENT\_ID, MSALARY, MAX(IDXMAX) AS MIDX

FROM (

SELECT DEPARTMENT\_ID, SALARY, RTT, MSALARY,

CASE WHEN SALARY = MSALARY THEN RTT END AS IDXMAX

FROM (

SELECT BASE.DEPARTMENT\_ID, BASE.SALARY, BASE.RTT, MAXSAL.MSALARY

FROM BASE

JOIN MAXSAL ON BASE.DEPARTMENT\_ID = MAXSAL.DEPARTMENT\_ID

)

)

WHERE IDXMAX IS NOT NULL

GROUP BY DEPARTMENT\_ID, MSALARY

) S ON F.DEPARTMENT\_ID = S.DEPARTMENT\_ID

),

MAX\_RANK AS (

SELECT DEPARTMENT\_ID, MAX(RANC) AS MAX\_RANC

FROM RANKED\_SALARIES

GROUP BY DEPARTMENT\_ID

)

SELECT A.DEPARTMENT\_ID, A.LAST\_NAME, NVL(TO\_CHAR(A.SALARY), '' '') AS SALARY,

COALESCE(B.RANC, M.MAX\_RANC) AS SALARY\_RANC

FROM HR.EMPLOYEES A

LEFT JOIN RANKED\_SALARIES B

ON A.DEPARTMENT\_ID = B.DEPARTMENT\_ID

AND A.SALARY = B.SALARY

LEFT JOIN MAX\_RANK M

ON A.DEPARTMENT\_ID = M.DEPARTMENT\_ID

WHERE A.SALARY IS NOT NULL

OR (A.SALARY IS NULL AND B.SALARY IS NULL)

ORDER BY A.DEPARTMENT\_ID, SALARY\_RANC NULLS LAST',

scope => DBMS\_SQLTUNE.scope\_comprehensive,

time\_limit => 60,

task\_name => 'tune\_task',

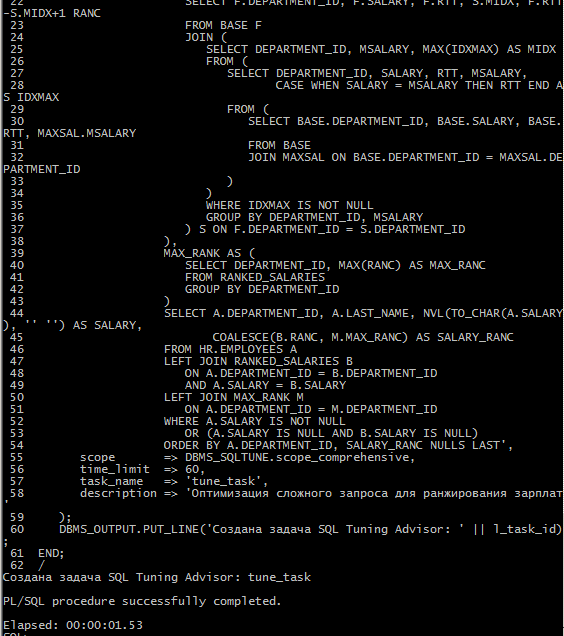
description => 'Оптимизация сложного запроса для ранжирования зарплат'

);

DBMS\_OUTPUT.PUT\_LINE('Создана задача SQL Tuning Advisor: ' || l\_task\_id);

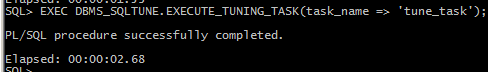
END;

/



Запускаем задачу SQL Tuning Advisor

EXEC DBMS\_SQLTUNE.EXECUTE\_TUNING\_TASK(task\_name => 'tune\_task');

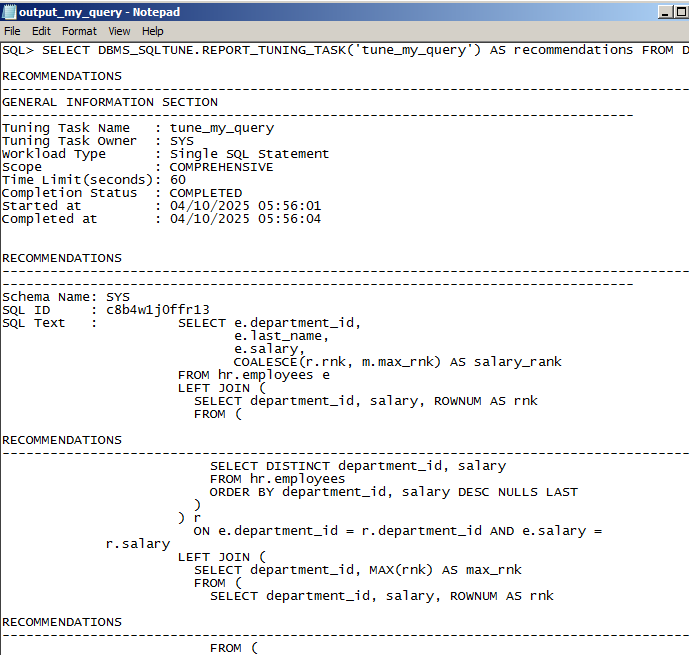


Advisor анализирует план и предлагает оптимизации. Включим вывод в файл:

SPOOL E:\output1.txt

SELECT DBMS\_SQLTUNE.REPORT\_TUNING\_TASK('tune\_task') AS recommendations FROM DUAL;

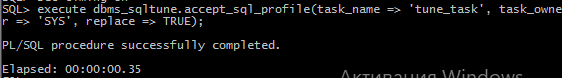
Посмотрим текст вывода рекомендаций в файле



Advisor обнаружил альтернативный план выполнения с ожидаемой выгодой до 10%. В разделе Findings указано, что новый план отличается методом соединения и порядком операций. Oracle рекомендует принять SQL профиль для использования более эффективного плана. Это может немного сократить затраты и ускорить выполнение запроса.

Применяем предложенный SQL профиль для улучшения выполнения запроса. Это позволяет оптимизатору использовать рекомендованный план. Параметр replace => TRUE указывает, что при наличии ранее созданного профиля он будет заменён.

execute dbms\_sqltune.accept\_sql\_profile(task\_name => 'tune\_task', task\_owner => 'SYS', replace => TRUE);



## Оптимизация с помощью SQL Access Advisor

Запускаем SQL Access Advisor, чтобы получить рекомендации по структурам доступа:

BEGIN  
DBMS\_ADVISOR.quick\_tune(  
advisor\_name => DBMS\_ADVISOR.SQLACCESS\_ADVISOR,  
task\_name => 'TEST\_sql\_access\_task1',  
attr1 => q'[

WITH MAXSAL AS (

SELECT DEPARTMENT\_ID, MAX(SALARY) MSALARY

FROM HR.EMPLOYEES

GROUP BY DEPARTMENT\_ID

),

BASE AS (

SELECT DEPARTMENT\_ID, SALARY, RTT

FROM (

SELECT DEPARTMENT\_ID, SALARY, ROWNUM RTT

FROM (

SELECT DISTINCT DEPARTMENT\_ID, SALARY

FROM HR.EMPLOYEES

ORDER BY DEPARTMENT\_ID, SALARY DESC NULLS LAST

)

)

),

RANKED\_SALARIES AS (

SELECT F.DEPARTMENT\_ID, F.SALARY, F.RTT, S.MIDX, F.RTT-S.MIDX+1 RANC

FROM BASE F

JOIN (

SELECT DEPARTMENT\_ID, MSALARY, MAX(IDXMAX) AS MIDX

FROM (

SELECT DEPARTMENT\_ID, SALARY, RTT, MSALARY,

CASE WHEN SALARY = MSALARY THEN RTT END AS IDXMAX

FROM (

SELECT BASE.DEPARTMENT\_ID, BASE.SALARY, BASE.RTT, MAXSAL.MSALARY

FROM BASE

JOIN MAXSAL ON BASE.DEPARTMENT\_ID = MAXSAL.DEPARTMENT\_ID

)

)

WHERE IDXMAX IS NOT NULL

GROUP BY DEPARTMENT\_ID, MSALARY

) S ON F.DEPARTMENT\_ID = S.DEPARTMENT\_ID

),

MAX\_RANK AS (

SELECT DEPARTMENT\_ID, MAX(RANC) AS MAX\_RANC

FROM RANKED\_SALARIES

GROUP BY DEPARTMENT\_ID

)

SELECT A.DEPARTMENT\_ID, A.LAST\_NAME, NVL(TO\_CHAR(A.SALARY), ' ') AS SALARY,

COALESCE(B.RANC, M.MAX\_RANC) AS SALARY\_RANC

FROM HR.EMPLOYEES A

LEFT JOIN RANKED\_SALARIES B

ON A.DEPARTMENT\_ID = B.DEPARTMENT\_ID

AND A.SALARY = B.SALARY

LEFT JOIN MAX\_RANK M

ON A.DEPARTMENT\_ID = M.DEPARTMENT\_ID

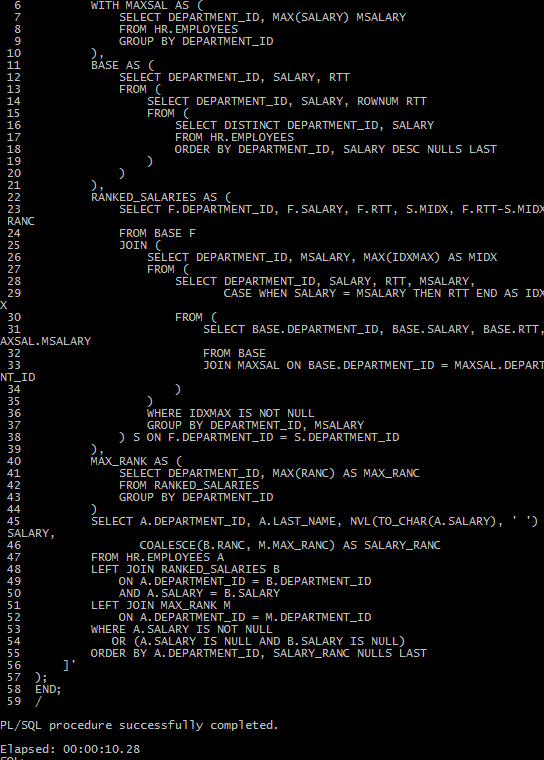
WHERE A.SALARY IS NOT NULL

OR (A.SALARY IS NULL AND B.SALARY IS NULL)

ORDER BY A.DEPARTMENT\_ID, SALARY\_RANC NULLS LAST

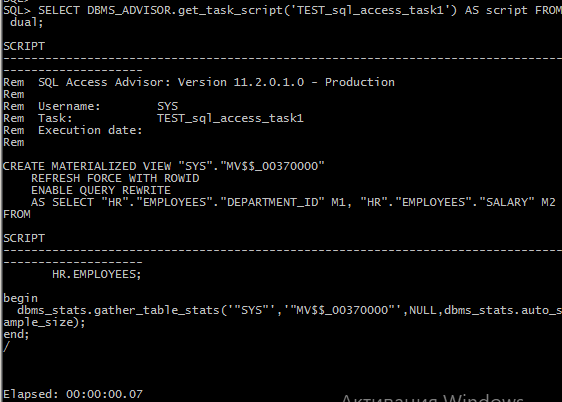
]'

);  
END;  
/



Выводим рекомендации:

SELECT DBMS\_ADVISOR.get\_task\_script('TEST\_sql\_access\_task1') AS script FROM dual;



Oracle предлагает следующее:  
– создать материализованное представление, которое будет содержать DEPARTMENT\_ID и SALARY из таблицы HR.EMPLOYEES;  
– после этого обязательно собрать статистику, чтобы оптимизатор мог учитывать это представление при построении плана.

В параметрах указывается REFRESH FORCE WITH ROWID — значит, обновление будет идти по ROWID, а ENABLE QUERY REWRITE позволяет Oracle автоматически подменять обычный запрос на обращение к материализованному представлению, если это ускоряет выполнение.

**Реализация рекомендаций SQL Access Advisor**

Создаём материализованное представление, как рекомендовано Advisor’ом, и собираем статистику:

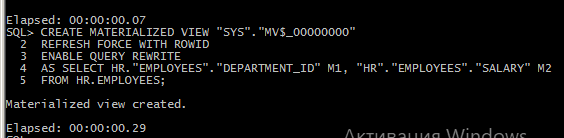
CREATE MATERIALIZED VIEW "SYS"."MV$\_00000000"

REFRESH FORCE WITH ROWID

ENABLE QUERY REWRITE

AS SELECT HR."EMPLOYEES"."DEPARTMENT\_ID" M1, "HR"."EMPLOYEES"."SALARY" M2

FROM HR.EMPLOYEES;

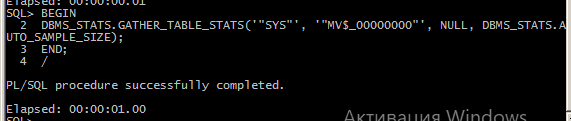


BEGIN

DBMS\_STATS.GATHER\_TABLE\_STATS('"SYS"', '"MV$\_00000000"', NULL, DBMS\_STATS.AUTO\_SAMPLE\_SIZE);

END;

/

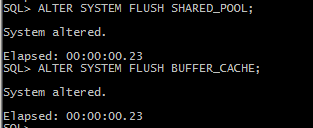


## Повторное выполнение запроса и анализ результата

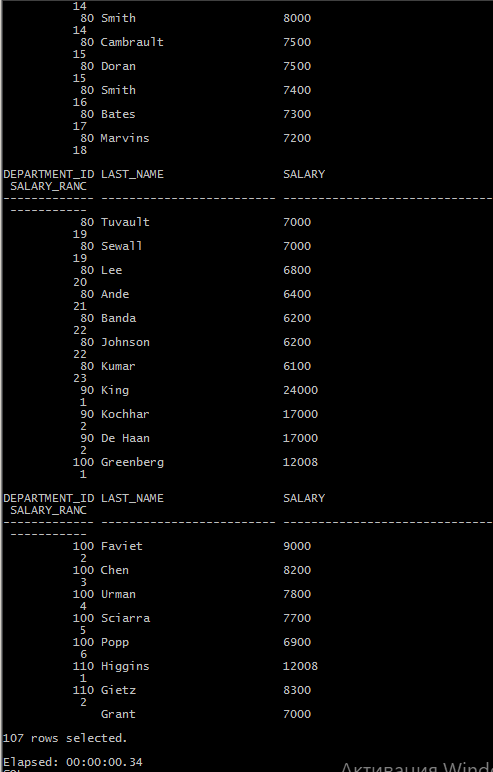
Очищаем кэш

ALTER SYSTEM FLUSH SHARED\_POOL;

ALTER SYSTEM FLUSH BUFFER\_CACHE;



Выполним запрос еще раз и посмотрим на результат



Измеряем время выполнения и сравниваем с первоначальным (до оптимизации).  
В нашем случае:  
было — 2:11  
стало — 0:34

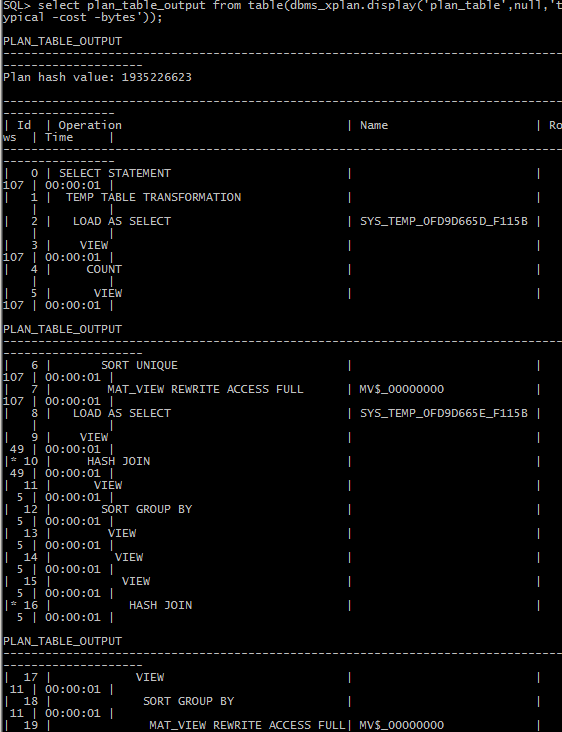
Это подтверждает, что применённые рекомендации улучшили производительность.

Посмотрим, как изменился план:

ALTER SYSTEM FLUSH SHARED\_POOL;

ALTER SYSTEM FLUSH BUFFER\_CACHE;

select plan\_table\_output from table(dbms\_xplan.display('plan\_table',null,'typical -cost -bytes'));



В запросе используется созданное материализованное представление и профиль, время запроса уменьшилось, поэтому оптимизация прошла успешно.

# Список источников

1. Oracle Corporation. *SQL Tuning Advisor*. Oracle® Database SQL Tuning Guide 21c [Электронный ресурс]. – Режим доступа: <https://docs.oracle.com/en/database/oracle/oracle-database/21/tgsql/sql-tuning-advisor.html> (дата обращения: 10.04.2025).
2. Oracle Corporation. *SQL Access Advisor*. Oracle® Database SQL Tuning Guide 19c [Электронный ресурс]. – Режим доступа: <https://docs.oracle.com/en/database/oracle/oracle-database/19/tgsql/sql-access-advisor.html> (дата обращения: 10.04.2025).
3. Hawthorne D.F. *Oracle 11g OCM – Performance Management: Using SQL Tuning Advisor* [Электронный ресурс]. – Режим доступа: <http://dfhawthorne.github.io/home/11g-ocm/11g-ocm-performance-management/11g-ocm-use-sql-tuning-advisor.html> (дата обращения: 10.04.2025)