1. **Опишите принципы организации и функционирования Индекс-кластерных таблиц**

   Индекс-кластерные таблицы — это таблицы в Oracle, которые объединяют в себе ряд особенностей и принципов индексирования и кластеризации данных. Вот основные принципы организации и функционирования индекс-кластерных таблиц:

   1. Кластеризация данных: Индекс-кластерные таблицы группируются по физическому критерию, что означает, что данные с одинаковыми значениями индексного ключа будут физически храниться рядом друг с другом. Одна или несколько таблиц могут быть объединены в кластер, чтобы предоставить более эффективный доступ к данным, связанным между собой.

   2. Индексирование: Как и в обычных таблицах, в индекс-кластерных таблицах можно создавать индексы для оптимизации поиска и фильтрации данных. Эти индексы могут быть уникальными или неуникальными и могут использоваться для ускорения выполнения запросов.

   3. Совместное использование блоков данных: Индекс-кластерные таблицы могут использовать одинаковые блоки данных для хранения информации. Это означает, что блоки данных могут содержать несколько строк из разных таблиц, связанных с кластером, что сокращает объем занимаемого пространства на диске.

   4. Эффективный доступ к данным: Кластеризация данных и использование индексов позволяют эффективно выполнять операции поиска, объединения и сортировки данных в индекс-кластерных таблицах. Это особенно полезно для запросов, которые часто используют данные, связанные между собой.

   5. Управление и обслуживание: Индекс-кластерные таблицы могут быть созданы, изменены или удалены с помощью операций DDL (Data Definition Language) в Oracle, таких как CREATE, ALTER и DROP TABLE. Также можно добавлять, изменять или удалять индексы для улучшения производительности.

   Индекс-кластерные таблицы являются мощным инструментом для организации и работы с данными в Oracle, позволяя достичь оптимальной производительности при поиске и связи данных. Однако они должны быть использованы соответственно требованиям конкретных сценариев использования и поддерживаться правильным обслуживанием данных и индексов

2. **Средствами SQL*PLUS создайте индекс-кластер Emp_dept, содержащий таблицы Emp и Dept. Таблицы Emp и Dept – копии таблиц соответственно Employees и Departments, включающие все ограничения таблиц.**

Создадим отдельное табличное пространство:

```
SQL> conn sys\oracle@student as sysdba
Enter password:
Connected.
SQL>  CREATE SMALLFILE TABLESPACE "TBS_DOP_3"
  2  DATAFILE 'E:\APP\ADMINISTRATOR\ORADATA\Student\dop_2.dbf'
  3  size 100m LOGGING EXTENT MANAGEMENT LOCAL;

Tablespace created.
```

Создадим в нем кластер:

```
SQL> conn hr/hr@student
Connected.
SQL> CREATE CLUSTER Emp_dept (department_id NUMBER(4))
  2  TABLESPACE
  3  tbs_dop_3 size 1024;

Cluster created.
```

Создадим таблицы, копии таблиц Employees and Departments:

```
SQL> CREATE TABLE hr.Dept(
  2  department_id NUMBER(4) NOT NULL,
  3  department_name VARCHAR2(30) NOT NULL,
  4  manager_id NUMBER(6),
  5  location_id NUMBER(4),
  6  CONSTRAINT pk_cluster_dept PRIMARY KEY (department_id ))
  7  CLUSTER Emp_dept (department_id);

Table created.
```

```
Table dropped.

SQL> CREATE TABLE hr.Emp (
  2   employee_id NUMBER(6) NOT NULL,
  3   first_name VARCHAR2(20),
  4   last_name VARCHAR2(25) NOT NULL,
  5   email VARCHAR2(25) NOT NULL,
  6   phone_number VARCHAR2(20),
  7   hire_date DATE NOT NULL,
  8   job_id VARCHAR2(10) NOT NULL,
  9   salary NUMBER(8, 2),
 10   commission_pct NUMBER(2, 2),
 11   manager_id NUMBER(6),
 12   department_id NUMBER(4),
 13   CONSTRAINT pk_cluster_emp PRIMARY KEY (employee_id),
 14   CONSTRAINT fk_cluster_emp_dept_deptid FOREIGN KEY (department_id) REFERENCES hr.Dept (department_id),
 15   CONSTRAINT uk_cluster_emp_email UNIQUE (email),
 16   CONSTRAINT fk_cluster_emp_manager FOREIGN KEY (manager_id) REFERENCES HR.Emp (employee_id) DEFERRABLE,
 17   CONSTRAINT chk_cluster_emp_salry_min CHECK (salary > 0))
 18   CLUSTER Emp_dept  (department_id);

Table created.
```

3. **Создайте кластерный индекс на кластер Emp_dept**

```
SQL> CREATE INDEX emp_dept_idx ON CLUSTER Emp_dept;

Index created.
```

4. **Заполните кластер данными из таблиц Employees и Departments.**

Перед заполнением включим отложенную проверку ограничений, чтобы не вызывало ошибок:

```
SQL> ALTER SESSION SET CONSTRAINTS = DEFERRED;

Session altered.

SQL> BEGIN
  2  FOR x IN (SELECT * FROM hr.departments) LOOP
  3  INSERT INTO hr.Dept VALUES x;
  4  INSERT INTO hr.Emp SELECT * FROM hr.employees WHERE department_id = x.department_id;
  5  END LOOP;
  6  FOR x IN (select * from hr.employees where department_id IS NULL) LOOP
  7     insert into hr.Emp SELECT * FROM hr.employees WHERE employee_id= x.employee_id;
  8  END LOOP;
  9  END;
 10  /

PL/SQL procedure successfully completed.
```

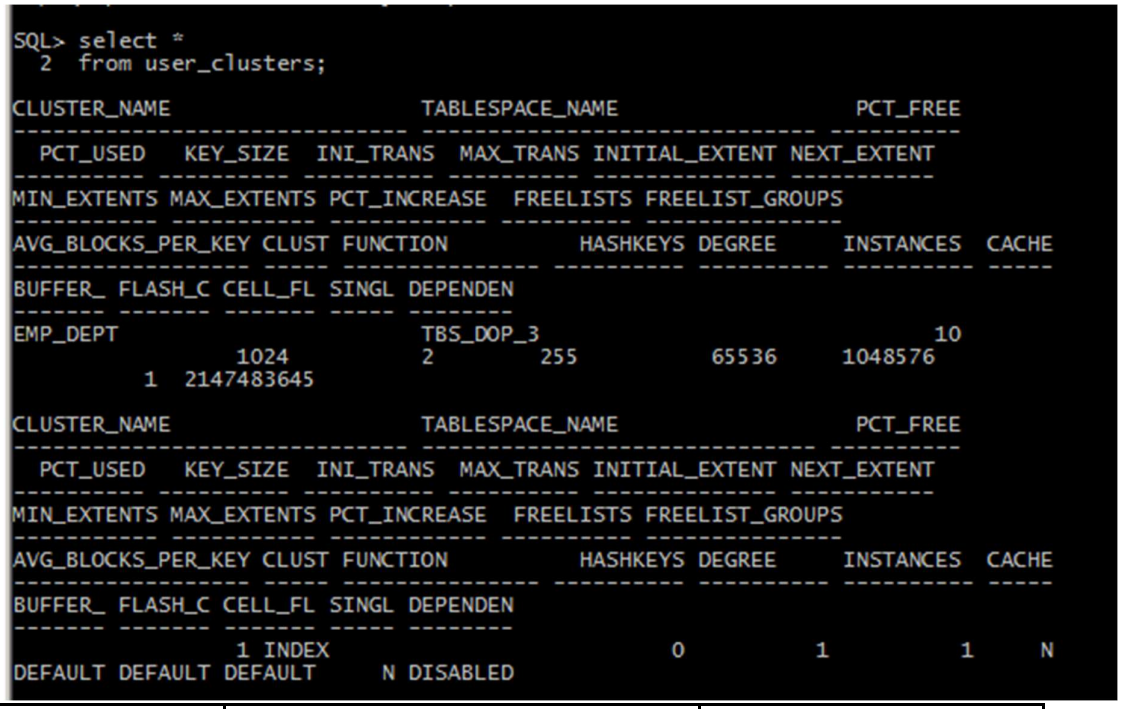
Заносим значения сначала о депортаменте, потом о всех кто в нем работает, чтобы данные по ключу кластера были в одном блоке.

Проверим что данные в одном блоке:

```
SQL> SELECT DBMS_ROWID.ROWID_BLOCK_NUMBER(dept.rowid) dept_rid, DBMS_ROWID.ROWID_BLOCK_NUMBER (emp.rowid) emp_rid, dept.depa
  2  FROM hr.emp, hr.dept
  3  WHERE emp.department_id = dept.department_id;

 DEPT_RID    EMP_RID DEPARTMENT_ID
--------- ---------- -------------
      134        134            10
      134        134            20
      134        134            20
      134        134            30
      134        134            30
      134        134            30
      134        134            30
      134        134            30
      134        134            30
      134        134            40
      134        134            50

 DEPT_RID    EMP_RID DEPARTMENT_ID
--------- ---------- -------------
      134        134            50
      134        134            50
      134        134            50
      134        134            50
      134        134            50
      134        134            50
      134        134            50
      134        134            50
      134        134            50
      134        134            50
      134        134            50

 DEPT_RID    EMP_RID DEPARTMENT_ID
--------- ---------- -------------
```

5. **Получите и опишите информацию о кластере из словаря данных. Учтите, что некоторые столбцы представлений заполняются при помощи пакета DBMS_STATS и команды ANALYZE**

```
SQL> ANALYZE CLUSTER emp_dept COMPUTE STATISTICS;

Cluster analyzed.

SQL> ANALYZE TABLE hr.emp COMPUTE STATISTICS;

Table analyzed.

SQL> ANALYZE TABLE hr.dept COMPUTE STATISTICS;

Table analyzed.

SQL> EXEC DBMS_STATS.gather_table_stats('HR', 'EMP');

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_STATS.gather_table_stats('HR', 'DEPT');

PL/SQL procedure successfully completed.

SQL>
```



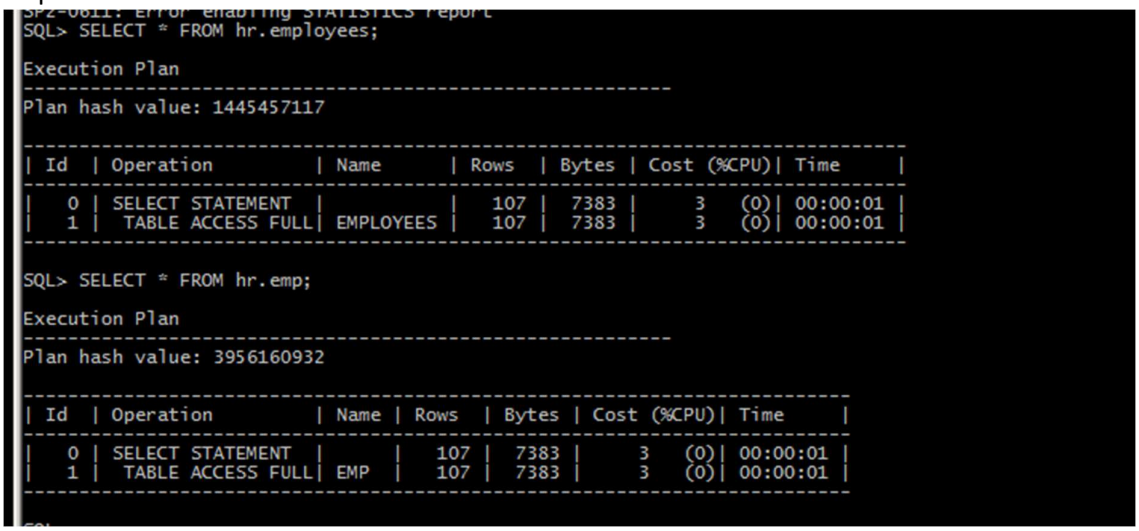
```
SQL> select *
  2  from user_clusters;

CLUSTER_NAME                   TABLESPACE_NAME                 PCT_FREE
------------------------------ ------------------------------ ----------
  PCT_USED   KEY_SIZE  INI_TRANS  MAX_TRANS INITIAL_EXTENT NEXT_EXTENT
---------- ---------- ---------- ---------- -------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  FREELISTS FREELIST_GROUPS
----------- ----------- ------------ ---------- ---------------
AVG_BLOCKS_PER_KEY CLUST FUNCTION          HASHKEYS DEGREE     INSTANCES  CACHE
------------------ ----- --------------- ---------- ---------- ---------- -----
BUFFER_ FLASH_C CELL_FL SINGL DEPENDEN
------- ------- ------- ----- --------
EMP_DEPT                       TBS_DOP_3                              10
                 1024          2         255          65536     1048576
           1  2147483645

CLUSTER_NAME                   TABLESPACE_NAME                 PCT_FREE
------------------------------ ------------------------------ ----------
  PCT_USED   KEY_SIZE  INI_TRANS  MAX_TRANS INITIAL_EXTENT NEXT_EXTENT
---------- ---------- ---------- ---------- -------------- -----------
MIN_EXTENTS MAX_EXTENTS PCT_INCREASE  FREELISTS FREELIST_GROUPS
----------- ----------- ------------ ---------- ---------------
AVG_BLOCKS_PER_KEY CLUST FUNCTION          HASHKEYS DEGREE     INSTANCES  CACHE
------------------ ----- --------------- ---------- ---------- ---------- -----
BUFFER_ FLASH_C CELL_FL SINGL DEPENDEN
------- ------- ------- ----- --------
                 1 INDEX                          0          1          1     N
DEFAULT DEFAULT DEFAULT     N DISABLED
```

Параметр	Назначение	Значение
CLUSTER_NAME	Название кластера	EMP_DEPT
TABLESPACE_NAME	Название табличного пространства, в котором расположен кластер	TBS_dop_3
PCT_FREE	Параметр pctfree для кластера (пространство, которое должно гарантированно остаться пустым в блоках кластера)	10
PCT_USED	Параметр pctused для кластера (максимально возможный процент занятости блока перед повторной записью в него)	null
KEY_SIZE	Предполагаемый размер ключа кластера и связанных строк	1024
INI_TRANS	Начальное количество транзакций (на блок)	2
MAX_TRANS	Максимальное количество транзакций (на блок)	255
INITIAL_EXTENT	Размер начального экстента в байтах	65536

NEXT_EXTENT	Размер следующего экстента в байтах	1048576
MIN_EXTENTS	Минимальное количество экстентов в сегменте	1
MAX_EXTENTS	Максимальное количество экстентов в сегменте	2147483645
PCT_INCREASE	Процент, на который будет увеличиваться размер добавляемых экстентов	null
FREELISTS	Количество списков свободных блоков, выделенных для кластера	null
FREELIST_GROUPS	Количество групп списков свободных блоков, выделенных для кластера	null
AVG_BLOCKS_PER_KEY	Среднее количество блоков на каждый ключ кластера	1
CLUSTER_TYPE	Тип кластера	INDEX
FUNCTION	Используемая хэш-функция	null
HASHKEYS	Количество хэш-ключей (для хэш-кластера)	0
DEGREE	Степень параллелизма для кластера (количество потоков на экземпляр для сканирования кластера)	1
INSTANCES	Количество экземпляров, используемых для кластера	1
CACHE	Индикатор необходимости кэширования кластера в buffer cache	N
BUFFER_POOL	Buffer pool для кластера по умолчанию	DEFAULT
FLASH_CACHE	Подсказка Database Smart Cache Flash, механизм кэширования часто используемых данных	DEFAULT

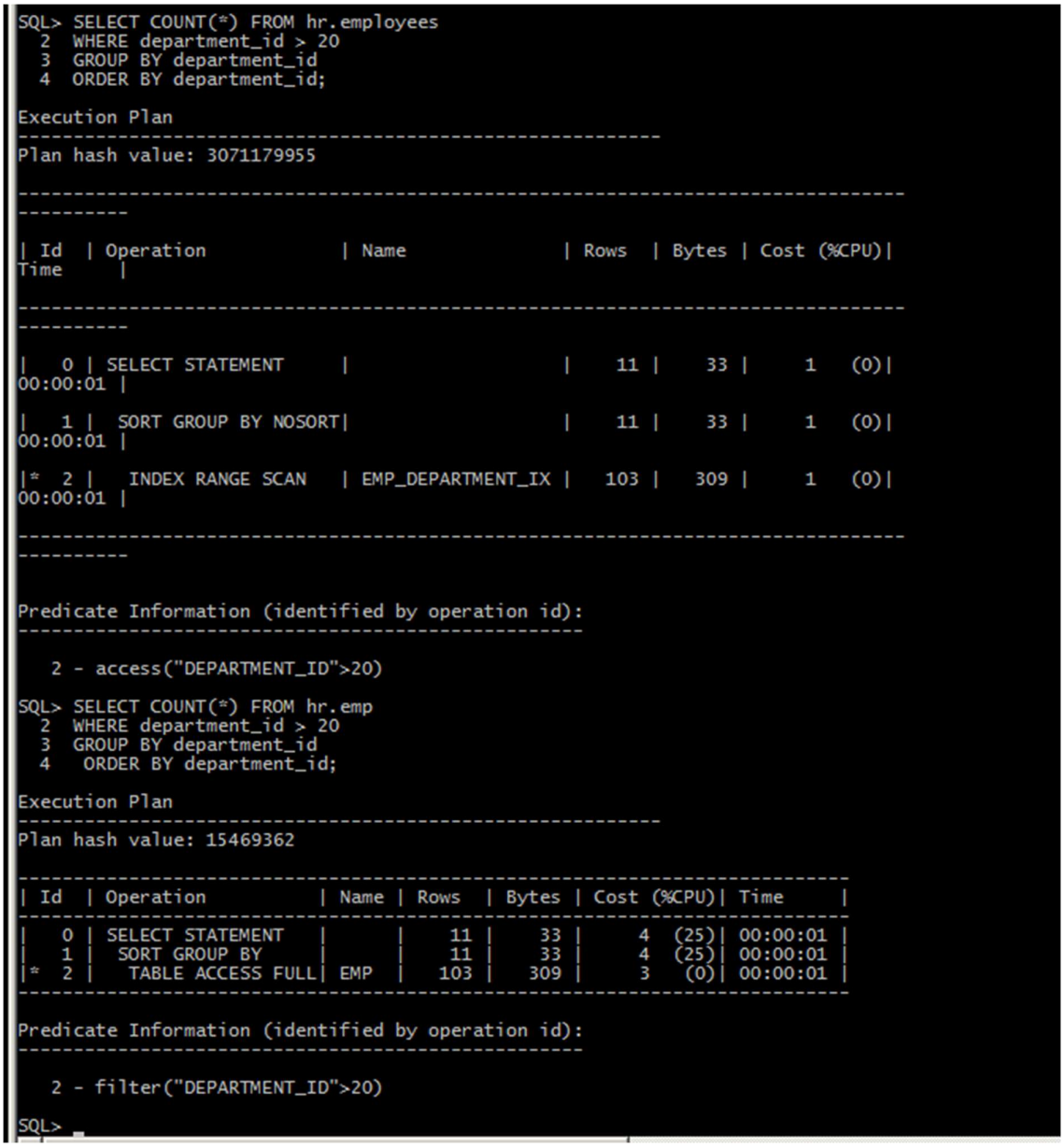
CELL_FLASH_CACHE	Управляет приоритетом блоков в ESFC, а также обработкой блоков Smart Scan	DEFAULT (действует механизм автоматического кэширования)
SINGLE_TABLE	Индикатор того, что данный кластер выделен на одну таблицу	N
DEPENDENCIES	Указывает, включено ли отслеживание зависимостей на уровне строк	DISABLED

6. Выполните одинаковые запросы на кластере и на таблицах Employees и Departments.



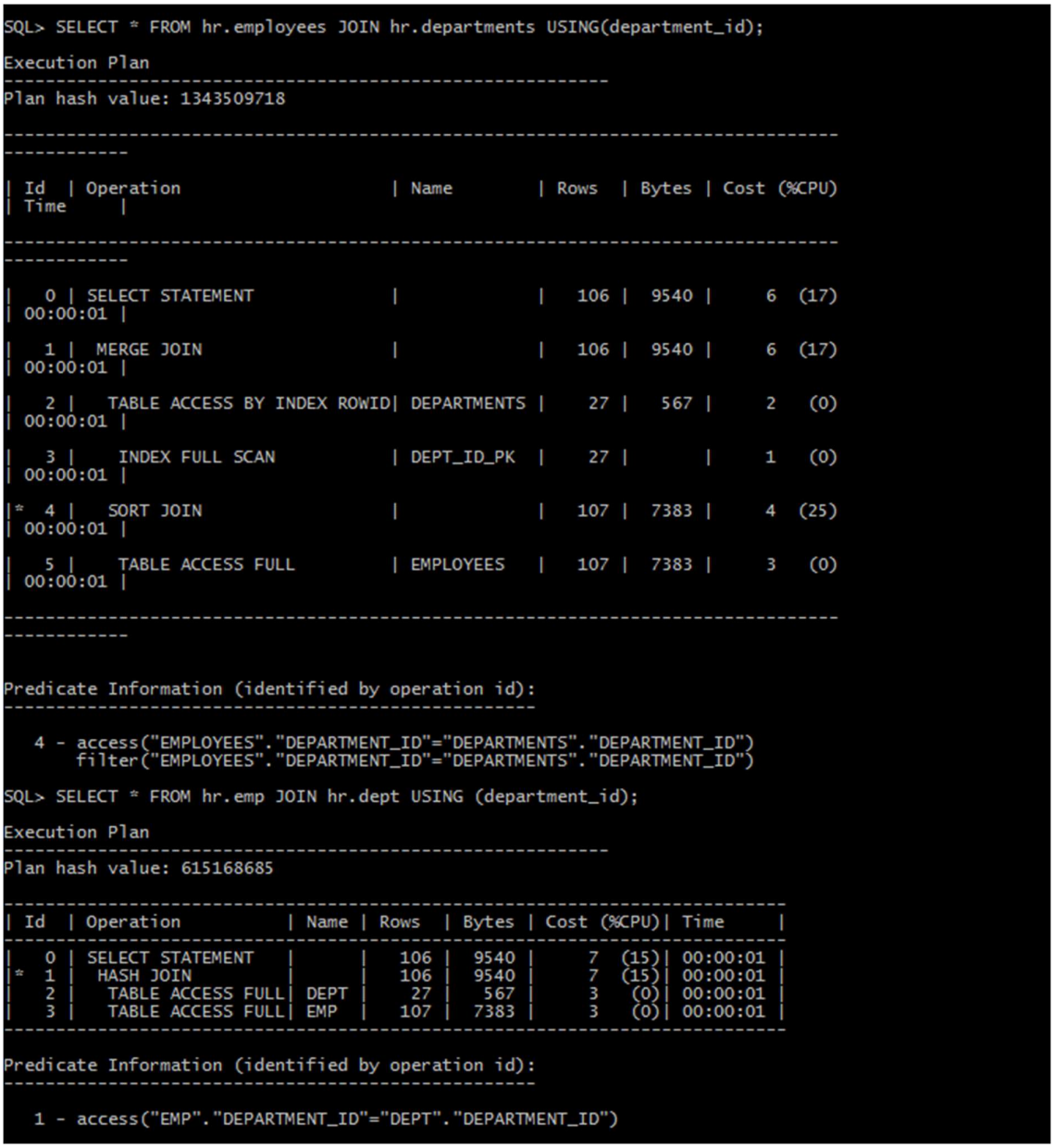
Для простой выборке видно, что и время и планы совершенно одинаковые.

7. Оцените эффективность выполнения запросов на таблице Employees и Departments и Индекс-кластере.



Выполнили более сложные запросы с использованием уже ключа по кластеру: для не кластерной таблицы группировка происходит без сортировки и используется индекс для доступа к данным. Время планов совпадает, но цена каждого действия разная, для не кластерной все 1, а для кластерной таблицы цена действий 4,4,3

8. Приведите и объясните планы выполнения запросов.
   Выполняем запросы по ключу кластера в данном случаем соединение таблиц:



```
SQL> SELECT * FROM hr.employees JOIN hr.departments USING(department_id);

Execution Plan
----------------------------------------------------------
Plan hash value: 1343509718

--------------------------------------------------------------------------------
------------

| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)
| Time     |

--------------------------------------------------------------------------------
------------

|   0 | SELECT STATEMENT             |             |   106 |  9540 |     6  (17)
| 00:00:01 |

|   1 |  MERGE JOIN                  |             |   106 |  9540 |     6  (17)
| 00:00:01 |

|   2 |   TABLE ACCESS BY INDEX ROWID| DEPARTMENTS |    27 |   567 |     2   (0)
| 00:00:01 |

|   3 |    INDEX FULL SCAN           | DEPT_ID_PK  |    27 |       |     1   (0)
| 00:00:01 |

|*  4 |   SORT JOIN                  |             |   107 |  7383 |     4  (25)
| 00:00:01 |

|   5 |    TABLE ACCESS FULL         | EMPLOYEES   |   107 |  7383 |     3   (0)
| 00:00:01 |

--------------------------------------------------------------------------------
------------


Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("EMPLOYEES"."DEPARTMENT_ID"="DEPARTMENTS"."DEPARTMENT_ID")
       filter("EMPLOYEES"."DEPARTMENT_ID"="DEPARTMENTS"."DEPARTMENT_ID")

SQL> SELECT * FROM hr.emp JOIN hr.dept USING (department_id);

Execution Plan
----------------------------------------------------------
Plan hash value: 615168685

-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |   106 |  9540 |     7  (15)| 00:00:01 |
|*  1 |  HASH JOIN         |      |   106 |  9540 |     7  (15)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| DEPT |    27 |   567 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| EMP  |   107 |  7383 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("EMP"."DEPARTMENT_ID"="DEPT"."DEPARTMENT_ID")
```

Планы выполнения получились совершенно разные: общее время выполнение индекс-кластерной таблицы меньше,

Цены каждого действия также отличается.
План обычной таблицы:
- SELECT STATEMENT: это оператор, с помощью которого выбираются необходимые данные из базы данных.

- MERGE JOIN: это метод объединения двух таблиц на основе соответствующих значений в их столбцах.

- TABLE ACCESS BY INDEX ROWID: это оператор, который используется для получения доступа к данным в таблице через индексованный ROWID.

- INDEX FULL SCAN: это оператор, который сканирует все записи в индексе таблицы, чтобы выполнить поиск или фильтрацию данных.

- SORT JOIN: это оператор, который сортирует данные перед их объединением с помощью операции объединения.

- TABLE ACCESS FULL: это оператор, который сканирует все записи в таблице при доступе к данным без использования индекса.

План кластерной таблицы:
- SELECT STATEMENT: оператор, используемый для выборки данных из базы данных.

- HASH JOIN: метод объединения двух таблиц с использованием хеш-функции для сопоставления строк.

- TABLE ACCESS FULL: оператор, который сканирует все записи в таблице для доступа к данным без использования индекса.

Список литературы

1. Кайт, Том Oracle для профессионалов / Том Кайт. – СПб : ДиаСофтЮП, 2003. – 1831 с. – ISBN 5-93772-072-5.
2. ALL_CLUSTERS // Oracle Help Center : сайт. – URL: https://docs.oracle.com/cd//B19306_01/server.102/b14237/statviews_1029.htm#i1575202 (дата обращения: 27.10.2023)