Задача: Создать табличное пространство DB_4 с размером блока 4 KB.
Block size данных 8к, поэтому для создания табличного пространства в с размером блока 4к необходимо выделить место в соответствующим кэше: db_4k_cashe_size;
После обновления системы, необходимо перезагрузить базу данных
И создаем необходимое табличное пространство

```
SQL> alter system set db_4k_cache_size=50M scope=both;

System altered.

SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup;
ORACLE instance started.

Total System Global Area  855982080 bytes
Fixed Size                  2180544 bytes
Variable Size             545262144 bytes
Database Buffers          306184192 bytes
Redo Buffers                2355200 bytes
Database mounted.
Database opened.
SQL> create tablespace db_4 datafile 'db_4_datafile.dbf' size 100M blocksize 4K;


Tablespace created.

SQL>
```

1. Вывести информацию о параметрах созданного табличного пространства. Объяснить смысл всех параметров.

```
SQL> select *
  2  from dba_tablespaces
  3  where tablespace_name='DB_4';

TABLESPACE_NAME                BLOCK_SIZE INITIAL_EXTENT NEXT_EXTENT MIN_EXTENTS

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

MAX_EXTENTS   MAX_SIZE PCT_INCREASE MIN_EXTLEN STATUS    CONTENTS  LOGGING   FOR

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

EXTENT_MAN ALLOCATIO PLU SEGMEN DEF_TAB_ RETENTION   BIG PREDICA ENC
---------- --------- --- ------ -------- ----------- --- ------- ---
COMPRESS_FOR
------------
DB_4                                4096          65536                          1

 2147483645 2147483645                    65536 ONLINE    PERMANENT LOGGING   NO
LOCAL      SYSTEM    NO  AUTO   DISABLED NOT APPLY   NO  HOST    NO



SQL>
```

1. TABLESPACE_NAME: Название табличного пространства.
2. BLOCK_SIZE: Размер блока в байтах используемого для хранения данных в табличном пространстве.
3. INITIAL_EXTENT: Начальный размер выделенного пространства для новых сегментов в табличном пространстве.
4. NEXT_EXTENT: Размер следующего выделенного пространства для новых сегментов в табличном пространстве.

5. MIN_EXTENTS: Минимальное количество сегментов, выделенных для объектов в табличном пространстве.
6. MAX_EXTENTS: Максимальное количество сегментов, выделенных для объектов в табличном пространстве.
7. MAX_SIZE: Максимальный размер табличного пространства.
8. PCT_INCREASE: Процент увеличения размера выделенного пространства при его расширении.
9. MIN_EXTLEN: Минимальный размер для данных в табличном пространстве.
10. STATUS: Статус табличного пространства (активное, выключенное, восстановление и т.д.).
11. CONTENTS: Тип контента табличного пространства (постоянное, временное, UNDO).
12. LOGGING: Определяет, будет ли операции в табличном пространстве регистрироваться в журнале протоколирования.
13. FORCE_LOGGING: Определяет, требуется ли принудительное регистрирование операций в журнале протоколирования для табличного пространства.
14. EXTENT_MANAGEMENT: Метод управления выделением и освобождением пространства в табличном пространстве (локальное или глобальное).
15. ALLOCATION_TYPE: Тип выделения пространства в табличном пространстве (AUTOALLOCATE или UNIFORM).
16. PLUGGED_IN: Показывает, является ли табличное пространство разъемным (PLUGGABLE) или нет.
17. SEGMENT_SPACE_MANAGEMENT: Метод управления пространством сегментов в табличном пространстве (MANUAL или AUTO).
18. DEF_TAB_COMPRESSION: Уровень сжатия данных по умолчанию для объектов в табличном пространстве.
19. RETENTION: Время (в днях), в течение которого данные остаются в табличном пространстве после удаления объектов.
20. BIGFILE: Показывает, является ли табличное пространство BIGFILE (размер каждого файла в пространстве может быть больше 4 ГБ).
21. PREDICATE_EVALUATION: Определяет, как оцениваются предикаты в табличном пространстве (IMMEDIATE или DEFERRED).
22. ENCRYPTED: Показывает, зашифровано ли табличное пространство или нет.
23. COMPRESS_FOR: Тип компрессии данных, используемый для объектов в табличном пространстве.

2. Создать в табличном пространстве 3 сегмента отката: UNDO_MY1, UNDO_MY2, UNDO_MY3.

   Сначала необходимо было изменить параметр undo_management с автоматического, на ручное

   Потом необходимо было пересоздать табличное пространство, с ручным segment space management

```
SQL> alter system set undo_management=manual scope=spfile;

System altered.

SQL>
```

```
SQL> conn sys/oracle@student as sysdba
Connected.
SQL> create rollback segment undo_my1
  2  tablespace db_4;
create rollback segment undo_my1
*
ERROR at line 1:
ORA-30574: Cannot create rollback segment in tablespace with AUTO segment space
management
```

```
SQL> create tablespace db_4 datafile 'db_4_df.dbf' size 100M blocksize 4K segmen
t space management manual;

Tablespace created.

SQL>
```

```
SQL> create rollback segment undo_my1
  2  tablespace db_4;

Rollback segment created.
```

```
SQL> create rollback segment undo_my2
  2  tablespace db_4;

Rollback segment created.

SQL> create rollback segment undo_my3
  2  tablespace db_4;

Rollback segment created.

SQL> select segment_name, tablespace_name
  2  from dba_rollback_segs;

SEGMENT_NAME                   TABLESPACE_NAME
------------------------------ ------------------------------
SYSTEM                         SYSTEM
_SYSSMU10_378818850$           UNDOTBS1
_SYSSMU9_3186340089$           UNDOTBS1
_SYSSMU8_1682283174$           UNDOTBS1
_SYSSMU7_1101470402$           UNDOTBS1
_SYSSMU6_1439239625$           UNDOTBS1
_SYSSMU5_2520346804$           UNDOTBS1
_SYSSMU4_1451910634$           UNDOTBS1
_SYSSMU3_478608968$            UNDOTBS1
_SYSSMU2_1531987058$           UNDOTBS1
_SYSSMU1_3086899707$           UNDOTBS1

SEGMENT_NAME                   TABLESPACE_NAME
------------------------------ ------------------------------
UNDO_MY3                       DB_4
UNDO_MY2                       DB_4
UNDO_MY1                       DB_4

14 rows selected.

SQL>
```

3. Вывести информацию о параметрах сегментов (начальный размер, минимальное и максимальное количество экстентов и др.). Объясните смысл всех параметров.

UNDO_MY2 находится онлайн, потому что в дальнейшем нам необходимо его использовать

```
SQL> select
  2   SEGMENT_NAME, TABLESPACE_NAME, BLOCK_ID, INITIAL_EXTENt,NEXT_EXTENT,
  3   MIN_EXTENTS, MAX_EXTENTS,  STATUS
  4  from dba_rollback_segs
  5   where segment_name like 'UNDO%';

SEGMENT_NAME                   TABLESPACE_NAME                 BLOCK_ID
------------------------------ ------------------------------ ----------
INITIAL_EXTENT NEXT_EXTENT MIN_EXTENTS MAX_EXTENTS STATUS
-------------- ----------- ----------- ----------- ----------------
UNDO_MY1                       DB_4                               256
       2097152     1048576           1       32765 OFFLINE

UNDO_MY2                       DB_4                               768
       2097152     1048576           1       32765 ONLINE

UNDO_MY3                       DB_4                              1280
       2097152     1048576           1       32765 OFFLINE
```

1. SEGMENT_NAME: имя сегмента. Сегмент - это логическая структура, которая содержит данные (таблицы, индексы, процедуры и т. Д.) в базе данных Oracle.
2. OWNER: владелец сегмента. Это имя пользователя или схемы, которая создала и владеет данным сегментом.
3. TABLESPACE_NAME: имя табличного пространства, в котором размещается данный сегмент. Табличное пространство - это физическое место хранения данных в базе данных Oracle.
4. SEGMENT_ID: уникальный идентификатор сегмента.
5. FILE_ID: уникальный идентификатор файла данных, в котором хранится данный сегмент.
6. BLOCK_ID: уникальный идентификатор блока данных, в котором хранится начало сегмента.
7. INITIAL_EXTENT: начальное количество блоков, выделенных для данного сегмента при его создании.
8. NEXT_EXTENT: количество блоков, которые будут выделены для данного сегмента после использования всех блоков начального выделения.
9. MIN_EXTENTS: минимальное количество блоков, которые могут быть выделены для данного сегмента.
10. MAX_EXTENTS: максимальное количество блоков, которые могут быть выделены для данного сегмента.
11. PCT_INCREASE: процент увеличения размера сегмента при автоматическом расширении.
12. STATUS: текущий статус сегмента (активен, отключен, удален и т. д.).
13. INSTANCE_NUM: номер экземпляра базы данных, на котором размещен данный сегмент (в случае использования кластерной базы данных).
14. RELATIVE_FNO: относительный номер файла данных, в котором размещен данный сегмент (в случае использования кластерной базы данных).

4. Создайте индексно-организованную таблицу Employees_IOT. Приведите информацию о сегменте для хранения таблицы.

```
SQL> create table employees_iot
  2  (employee_id number(6) primary key,
  3  first_name varchar2(20),
  4  last_name varchar2(25),
  5  hire_date date,
  6  job_id varchar(10),
  7  salary number(8,2),
  8  commission_pct number(2,2),
  9  email varchar2(25),
 10  phone_number varchar2(20),
 11  manager_id number(6),
 12  department_id number(4)
 13  )
 14  organization index;

Table created.

SQL> _
```

Чтобы узнать название сегмент для этой таблицы, выводит хозяина, табличное пространство и название сегмента:

```
SQL> select owner, segment_name, tablespace_name
  2  from dba_segments
  3  where tablespace_name='DB_4';

OWNER
------------------------------
SEGMENT_NAME
--------------------------------------------------------------------------------

TABLESPACE_NAME
------------------------------
HR
SYS_IOT_TOP_74596
DB_4
```

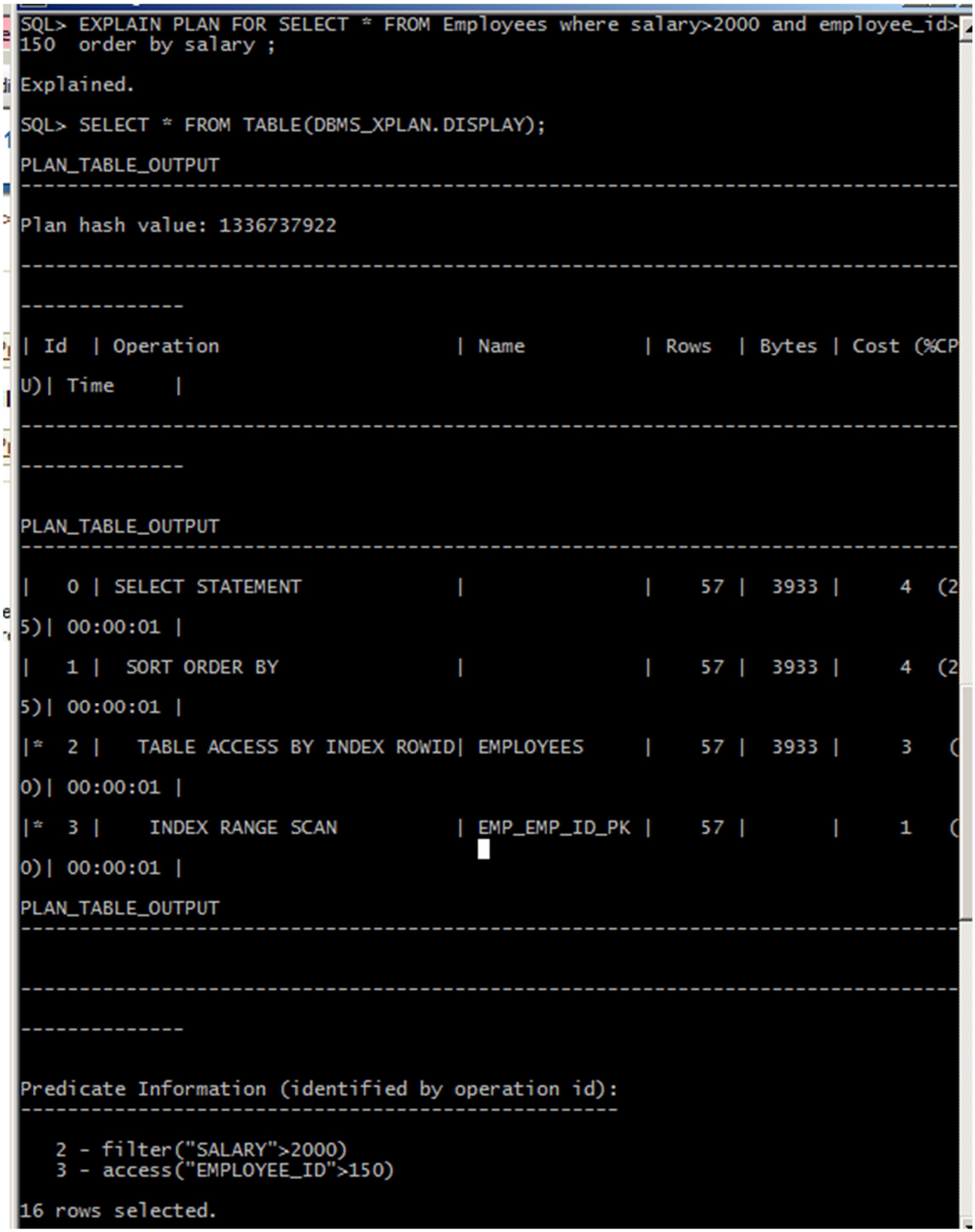
```
SQL> SELECT SEGMENT_NAME, SEGMENT_TYPE, SEGMENT_SUBTYPE, TABLESPACE_NAME,
  2   BYTES, EXTENTS, INITIAL_EXTENT, NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS,
  3   MAX_SIZE
  4  FROM DBA_SEGMENTS
  5  where SEGMENT_name='SYS_IOT_TOP_74596';

SEGMENT_NAME
--------------------------------------------------------------------------------

SEGMENT_TYPE       SEGMENT_SU TABLESPACE_NAME                    BYTES
------------------ ---------- ------------------------------ ----------
   EXTENTS INITIAL_EXTENT NEXT_EXTENT MIN_EXTENTS MAX_EXTENTS   MAX_SIZE
---------- -------------- ----------- ----------- ----------- ----------
SYS_IOT_TOP_74596
INDEX              MSSM       DB_4                                65536
         1         65536     1048576           1  2147483645 2147483645
```

5. Заполните таблицу данными из таблицы Employees, задав для использования транзакцией сегмент UNDO_MY2. Подтвердите использование транзакцией заданного сегмента отмены.

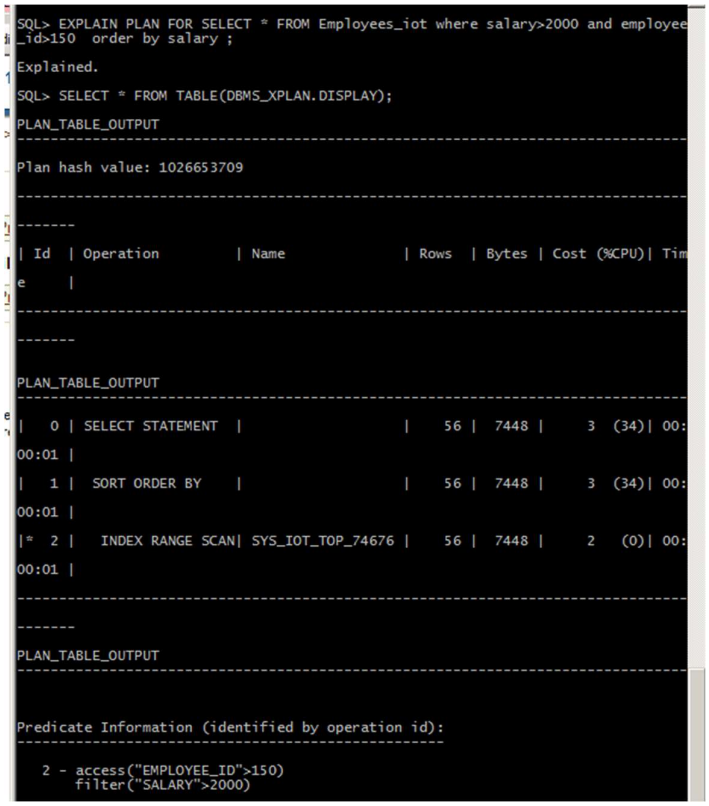
   Сначала необходимо установить rollback segment для транзакции:

```
SQL> set transaction use rollback segment UNDO_MY2;

Transaction set.

SQL> insert into employees_iot
  2  select *
  3  from employees;

107 rows created.

SQL> commit;

Commit complete.

SQL>
```

6. Оцените эффективность выполнения запросов на таблице Employees и таблице Employees_IOT. Приведите и объясните планы выполнения запросов.



```
SQL> EXPLAIN PLAN FOR SELECT * FROM Employees where salary>2000 and employee_id>
150  order by salary ;

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

Plan hash value: 1336737922

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

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

| Id  | Operation                    | Name          | Rows  | Bytes | Cost (%CP

U)| Time     |

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|   0 | SELECT STATEMENT             |               |    57 |  3933 |     4  (2

5)| 00:00:01 |

|   1 |  SORT ORDER BY               |               |    57 |  3933 |     4  (2

5)| 00:00:01 |

|*  2 |   TABLE ACCESS BY INDEX ROWID| EMPLOYEES     |    57 |  3933 |     3   (

0)| 00:00:01 |

|*  3 |    INDEX RANGE SCAN          | EMP_EMP_ID_PK |    57 |       |     1   (

0)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------


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

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


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

   2 - filter("SALARY">2000)
   3 - access("EMPLOYEE_ID">150)

16 rows selected.
```



```
SQL> EXPLAIN PLAN FOR SELECT * FROM Employees_iot where salary>2000 and employee
_id>150  order by salary ;

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

Plan hash value: 1026653709

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

-------

| Id  | Operation         | Name              | Rows  | Bytes | Cost (%CPU)| Tim
e     |

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

-------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|   0 | SELECT STATEMENT  |                   |    56 |  7448 |     3  (34)| 00:
00:01 |

|   1 |  SORT ORDER BY    |                   |    56 |  7448 |     3  (34)| 00:
00:01 |

|*  2 |   INDEX RANGE SCAN| SYS_IOT_TOP_74676 |    56 |  7448 |     2   (0)| 00:
00:01 |

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

-------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------


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

   2 - access("EMPLOYEE_ID">150)
       filter("SALARY">2000)
```

Перенесу для наглядности всё в таблицу:

Id	Operation	Name	Rows	Bytes	%CPU	Time	
0	Select statement		57	3933	25	1sec	
1	Sort order by		57	3933	25	1sec	
2	Table access by index rowid	employees	57	3933	0	1sec	
3	Index range scan	Emp_emp_id_pk	57		1	1sec	
0	Select statement		56	7448	34	1sec	
1	Sort order by		56	7448	34	1sec	
2	Index range scan	Sys_iot_top_74676	56	7448	1	1sec	

1. Таблица Employees:

В случае с таблицей Employees, который не является IOT-таблицей, база данных, вероятно, будет использовать следующий план выполнения:

- Проверка условий в предложении WHERE для определения соответствующих записей.

- Использование существующих индексов для оптимизации процесса поиска и сортировки, если они существуют.

- Обход таблицы для извлечения данных, соответствующих запросу.

- Объединение и сортировка данных, если необходимо.

При выполнении запросов на таблице Employees может потребоваться больше времени, поскольку системе приходится просматривать все строки таблицы и выбирать только те, которые удовлетворяют условиям запроса.

2. Таблица Employees_IOT:

В случае с IOT-таблицей Employees_IOT, план выполнения будет отличаться:

- Проверка условий в предложении WHERE для определения соответствующих записей.

- Использование индекса для поиска непосредственно в физическом порядке, в котором данные хранятся на диске, обеспечивая более эффективное выполнение запроса.

- Обход индекса и извлечение данных, удовлетворяющих условиям запроса.

- Объединение и сортировка данных, если необходимо.

Запросы на таблице Employees_IOT могут выполняться быстрее благодаря более эффективному использованию индексов и меньшему количеству операций для доступа к данным.

При большом blocksize (например, 8192 байт), чтение данных будет более эффективным, поскольку больше данных будет попадать в один блок чтения. При меньшем blocksize (например, 4096 байт), на чтение данных потребуется больше блоков, что может замедлить выполнение запроса.

Список литературы:

1. https://docs.oracle.com/cd//B19306_01/server.102/b14237/initparams037.htm#REFRN10027
2. https://use-the-index-luke.com/sql/explain-plan/oracle/getting-an-execution-plan