Task1

```
-- Task1.1
ALTER TABLE actor ADD CONSTRAINT PK_ACTORID PRIMARY KEY (actor_id);

/*
Question: Which constraints in Table 1 have been created on these six tables?
*/
  primary key and foreign key


-- Task1.2


ALTER TABLE actor ADD CONSTRAINT PK_ACTORID PRIMARY KEY (actor_id);
ALTER TABLE film ADD CONSTRAINT PK_FILMID PRIMARY KEY (film_id);
ALTER TABLE film_actor ADD CONSTRAINT FK_FILMID1 FOREIGN KEY (film_id) REFERENCES
film;

ALTER TABLE category ADD CONSTRAINT PK_CATEGORYID PRIMARY KEY (category_id );

ALTER TABLE language ADD CONSTRAINT PK_LANGUAGEID PRIMARY KEY (language_id );

ALTER TABLE film ADD CONSTRAINT UN_DESCRIPTION    UNIQUE(description);

ALTER TABLE actor ADD CONSTRAINT CK_FNAME CHECK (first_name IS NOT NULL);

ALTER TABLE actor ADD CONSTRAINT CK_LNAME CHECK (last_name IS NOT NULL);

ALTER TABLE category ADD CONSTRAINT CK_CATNAME CHECK (name IS NOT NULL);

ALTER TABLE language ADD CONSTRAINT CK_LANNAME CHECK (name IS NOT NULL);

ALTER TABLE film ADD CONSTRAINT CK_TITLE CHECK (title IS NOT NULL);

ALTER TABLE film ADD CONSTRAINT CK_RELEASEYR CHECK (release_year<=2020);

ALTER TABLE film ADD CONSTRAINT CK_RATING CHECK (rating IN('G', 'PG', 'PG-13', 'R', 'NC-
17'));

ALTER TABLE film ADD CONSTRAINT CK_SPLFEATURES CHECK (special_features IN(null,
'Trailers',
'Commentaries', 'Deleted
```

Scenes', 'Behind the Scenes'));

ALTER TABLE film ADD CONSTRAINT FK_LANGUAGEID FOREIGN KEY (language_id) REFERENCES language(language_id);
ALTER TABLE film ADD CONSTRAINT FK_ORLANGUAGEID FOREIGN KEY (original_language_id) REFERENCES language(language_id);

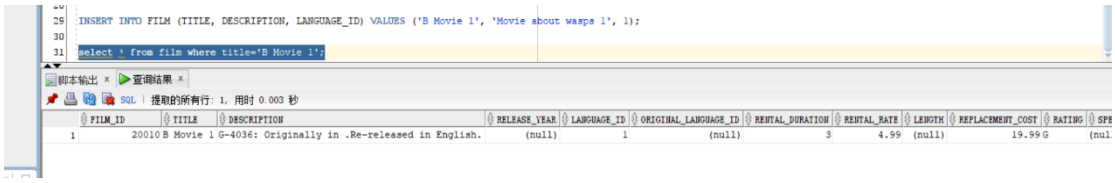
ALTER TABLE film_actor ADD CONSTRAINT FK_ACTORID FOREIGN KEY (actor_id) REFERENCES actor(actor_id);

ALTER TABLE film_category ADD CONSTRAINT FK_CATEGORYID FOREIGN KEY (category_id) REFERENCES category(category_id);

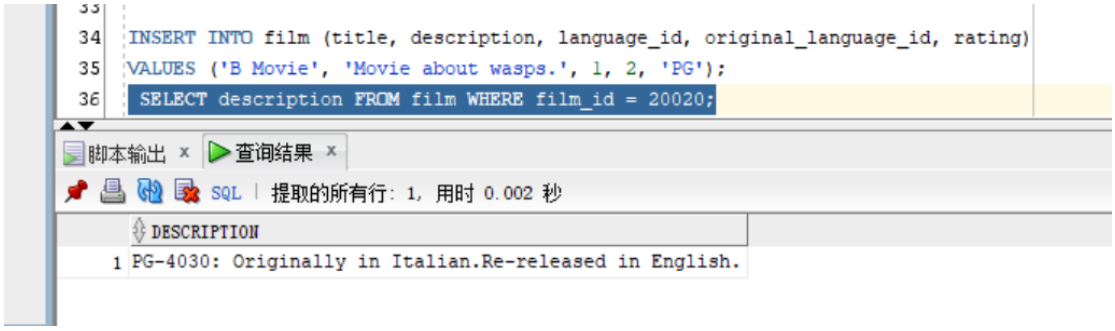
ALTER TABLE film_category ADD CONSTRAINT FK_FILMID2 FOREIGN KEY (film_id) REFERENCES film(film_id);

Task2

```
-- 1
create sequence FILM_ID_SEQ
start with 20010
increment by 10;

-- 2
create trigger BI_FILM_ID
before insert on film for each row
begin
select FILM_ID_SEQ.nextval into :new.film_id from dual;
end;

/*
INSERT INTO FILM (TITLE, DESCRIPTION, LANGUAGE_ID) VALUES ('B Movie 1', 'Movie about wasps 1', 1);

select * from film where title='B Movie 1';
*/
```



```
-- 3
-- drop trigger BI_FILM_DESP;

create trigger BI_FILM_DESP
before insert on film for each row
begin
select :new.rating||'-'||(select count(*)+1 from film where rating=:new.rating)||': '||'Originally in
'|| (select name
from language where language_id=:new.original_language_id)
||'.'||'Re-released in '||(select name
from language where language_id=:New.language_id) ||'.'
into :new.description from dual;
end;

/*

INSERT INTO film (title, description, language_id, original_language_id, rating)
VALUES ('B Movie', 'Movie about wasps.', 1, 2, 'PG');
  SELECT description FROM film WHERE film_id = 20020;
*/
```



Task3

```
-- 1
select f.title, f.length
from film f
join
(
select min(length) as length from film f join film_category fc on f.film_id=fc.film_id
join category c on c.category_id=fc.category_id
where c.name='Action'
) t on f.length=t.length;

-- 2
create or replace view MIN_ACTION_ACTORS
```

```
as
select distinct a.actor_id, a.first_name, a.last_name
from actor a join film_actor fa on a.actor_id=fa.actor_id
join film f on f.film_id=fa.film_id
join (
	select f.title, f.length
from film f
join
(
select min(length) as length from film f join film_category fc on f.film_id=fc.film_id
join category c on c.category_id=fc.category_id
where c.name='Action'
) t on f.length=t.length
) v1 on v1.title=f.title;

SELECT * FROM MIN_ACTION_ACTORS;

-- 3
create or replace view V_ACTION_ACTORS_2012
as
select distinct a.actor_id, a.first_name, a.last_name
from actor a join film_actor fa on a.actor_id=fa.actor_id
join film f on f.film_id=fa.film_id
join film_category fc on fc.film_id=f.film_id
join category c on c.category_id=fc.category_id
where c.name='Action' and f.release_year=2012;

SELECT * FROM V_ACTION_ACTORS_2012;

-- 4
create materialized view MV_ACTION_ACTORS_2012
as
select distinct a.actor_id, a.first_name, a.last_name
from actor a join film_actor fa on a.actor_id=fa.actor_id
join film f on f.film_id=fa.film_id
join film_category fc on fc.film_id=f.film_id
join category c on c.category_id=fc.category_id
where c.name='Action' and f.release_year=2012;

SELECT * FROM MV_ACTION_ACTORS_2012;
```

```sql
select f.title, f.length
from film f
join
(
select min(length) as length from film f join film_category fc on f.film_id=fc.film_id
join category c on c.category_id=fc.category_id
where c.name='Action'
) t on f.length=t.length;
```

查询结果

SQL | 已提取 50 行，用时 0.014 秒

	TITLE	LENGTH
1	ALIEN CENTER	46
2	IRON MOON	46
3	KWAI HOMEWARD	46
4	LABYRINTH LEAGUE	46
5	RIDGEMONT SUBMARINE	46
6	SANTA MILE	46
7	ROOF MUSSOLINI	46
8	PURPLE JAWS	46
9	KICK TIES	46
10	SHOOTIST LADYBUGS	46
11	STAGE STRANGER	46
12	GONE TRUMAN	46
13	SHAKESPEARE SECRETARY	46
14	SPIRIT DEVIL	46

```sql
select distinct a.actor_id, a.first_name, a.last_name
from actor a join film_actor fa on a.actor_id=fa.actor_id
join film f on f.film_id=fa.film_id
join (
    select f.title, f.length
from film f
join
(
select min(length) as length from film f join film_category fc on f.film_id=fc.film_id
join category c on c.category_id=fc.category_id
where c.name='Action'
) t on f.length=t.length
) vl on vl.title=f.title;

SELECT * FROM MIN_ACTION_ACTORS;
```

脚本输出 × 查询结果 ×

SQL | 已提取 50 行，用时 0.022 秒

	ACTOR_ID	FIRST_NAME	LAST_NAME
1	106	GROUCHO	DUNST
2	117	RENEE	TRACY
3	141	CATE	HARRIS
4	192	JOHN	SUVARI
5	120	PENELOPE	MONROE
6	56	DAN	HARRIS
7	75	BURT	POSEY
8	148	EMILY	DEE
9	12	KARL	BERRY
10	114	MORGAN	MCDORMAND
11	11	ZERO	CAGE
12	159	LAURA	BRODY
13	91	CHRISTOPHER	BERRY
14	165	AL	GARLAND

日志记录页 - 日志

```sql
30 create or replace view V_ACTION_ACTORS_2012
31 as
32 select distinct a.actor_id, a.first_name, a.last_name
33 from actor a join film_actor fa on a.actor_id=fa.actor_id
34 join film f on f.film_id=fa.film_id
35 join film_category fc on fc.film_id=f.film_id
36 join category c on c.category_id=fc.category_id
37 where c.name='Action' and f.release_year=2012;
38
39 SELECT * FROM V_ACTION_ACTORS_2012;
```

脚本输出 x　查询结果 x

SQL | 已提取 50 行，用时 0.014 秒

	ACTOR_ID	FIRST_NAME	LAST_NAME
1	75	BURT	POSEY
2	91	CHRISTOPHER	BERRY
3	114	MORGAN	MCDORMAND
4	148	EMILY	DEE
5	165	AL	GARLAND
6	166	NICK	DEGENERES
7	192	JOHN	SUVARI
8	194	MERYL	ALLEN
9	30	SANDRA	PECK
10	34	AUDREY	OLIVIER
11	46	PARKER	GOLDBERG
12	58	CHRISTIAN	AKROYD
13	69	KENNETH	PALTROW
14	77	CARY	MCCONAUGHEY

```
41
42 create materialized view MV_ACTION_ACTORS_2012
43 as
44 select distinct a.actor_id, a.first_name, a.last_name
45 from actor a join film_actor fa on a.actor_id=fa.actor_id
46 join film f on f.film_id=fa.film_id
47 join film_category fc on fc.film_id=f.film_id
48 join category c on c.category_id=fc.category_id
49 where c.name='Action' and f.release_year=2012;
50
51 SELECT * FROM MV_ACTION_ACTORS_2012;
```

脚本输出 × 查询结果 ×

SQL | 已提取 50 行，用时 0.005 秒

	ACTOR_ID	FIRST_NAME	LAST_NAME
1	75	BURT	POSEY
2	91	CHRISTOPHER	BERRY
3	114	MORGAN	MCDORMAND
4	148	EMILY	DEE
5	165	AL	GARLAND
6	166	NICK	DEGENERES
7	192	JOHN	SUVARI
8	194	MERYL	ALLEN
9	30	SANDRA	PECK
10	34	AUDREY	OLIVIER
11	46	PARKER	GOLDBERG
12	58	CHRISTIAN	AKROYD
13	69	KENNETH	PALTROW
14	77	CARY	MCCONAUGHEY

-- 5

SQL> explain plan for SELECT * FROM V_ACTION_ACTORS_2012;

Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT

-

Plan hash value: 1938178838

```
-
------------------

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

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


PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------
-
|   0 | SELECT STATEMENT             |                             |   121 |  7381 |   223
   (1)| 00:00:01 |

|   1 |  VIEW                        | V_ACTION_ACTORS_2012 |   121 |  7381 |   223
   (1)| 00:00:01 |

|   2 |   HASH UNIQUE                |                             |   121 |  5929 |   223
   (1)| 00:00:01 |

|*  3 |    HASH JOIN                 |                             |   121 |  5929 |   222
   (1)| 00:00:01 |

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

|*  4 |     HASH JOIN                |                             |   121 |  3993 |   219
   (1)| 00:00:01 |

|*  5 |      HASH JOIN SEMI          |                             |    19 |   494 |   150
   (0)| 00:00:01 |

|   6 |       MERGE JOIN CARTESIAN|                             |   303 |  5757 |   139
   (0)| 00:00:01 |

|*  7 |        TABLE ACCESS FULL  | CATEGORY                    |     1 |    10 |     3

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------
-
```

```
    (0)| 00:00:01 |

|     8 |              BUFFER SORT              |                                |     303 |    2727 |     136
    (0)| 00:00:01 |

|*    9 |               TABLE ACCESS FULL | FILM                           |     303 |    2727 |     136
    (0)| 00:00:01 |

|    10 |            TABLE ACCESS FULL      | FILM_CATEGORY              | 20000 |     136K|       11
    (0)| 00:00:01 |


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-
|    11 |           TABLE ACCESS FULL       | FILM_ACTOR                   |     128K|     879K|       69
    (2)| 00:00:01 |

|    12 |         TABLE ACCESS FULL         | ACTOR                          |     200 |    3200 |        3
    (0)| 00:00:01 |

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


Predicate Information (identified by operation id):

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

    3 - access("A"."ACTOR_ID"="FA"."ACTOR_ID")
    4 - access("F"."FILM_ID"="FA"."FILM_ID")
    5 - access("C"."CATEGORY_ID"="FC"."CATEGORY_ID" AND "FC"."FILM_ID"="F"."FILM_
ID")

    7 - filter("C"."NAME"='Action')
    9 - filter("F"."RELEASE_YEAR"=2012)

28 rows selected.

SQL> explain plan for SELECT * FROM MV_ACTION_ACTORS_2012;
```

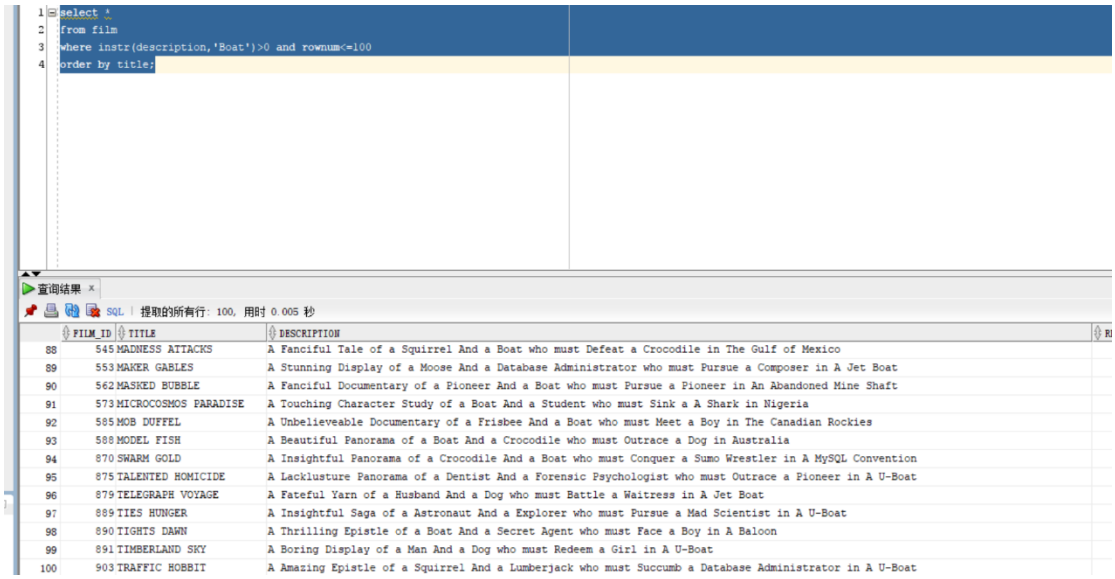
```
Explained.

SQL>    select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-
Plan hash value: 1015139828

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

| Id    | Operation                    | Name                            | Rows    | Bytes | Cost (%CP
U)| Time         |

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-
|     0 | SELECT STATEMENT         |                                 |     109 |    1744 |         3     (
0)| 00:00:01 |

|     1 |    MAT_VIEW ACCESS FULL| MV_ACTION_ACTORS_2012 |     109 |    1744 |         3     (
0)| 00:00:01 |

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

8 rows selected.
```

based on the execution plans above, the MV_ACTION_ACTORS_2012 indeed speed up the query processing.

Task4

-- 1

select *
from film
where instr(description,'Boat')>0 and rownum<=100
order by title;



```
select *
from film
where instr(description,'Boat')>0 and rownum<=100
order by title;
```

查询结果

SQL | 提取的所有行: 100, 用时 0.005 秒

	FILM_ID	TITLE	DESCRIPTION
88	545	MADNESS ATTACKS	A Fanciful Tale of a Squirrel And a Boat who must Defeat a Crocodile in The Gulf of Mexico
89	553	MAKER GABLES	A Stunning Display of a Moose And a Database Administrator who must Pursue a Composer in A Jet Boat
90	562	MASKED BUBBLE	A Fanciful Documentary of a Pioneer And a Boat who must Pursue a Pioneer in An Abandoned Mine Shaft
91	573	MICROCOSMOS PARADISE	A Touching Character Study of a Boat And a Student who must Sink a A Shark in Nigeria
92	585	MOB DUFFEL	A Unbelieveable Documentary of a Frisbee And a Boat who must Meet a Boy in The Canadian Rockies
93	588	MODEL FISH	A Beautiful Panorama of a Boat And a Crocodile who must Outrace a Dog in Australia
94	870	SWARM GOLD	A Insightful Panorama of a Crocodile And a Boat who must Conquer a Sumo Wrestler in A MySQL Convention
95	875	TALENTED HOMICIDE	A Lacklusture Panorama of a Dentist And a Forensic Psychologist who must Outrace a Pioneer in A U-Boat
96	879	TELEGRAPH VOYAGE	A Fateful Yarn of a Husband And a Dog who must Battle a Waitress in A Jet Boat
97	889	TIES HUNGER	A Insightful Saga of a Astronaut And a Explorer who must Pursue a Mad Scientist in A U-Boat
98	890	TIGHTS DAWN	A Thrilling Epistle of a Boat And a Secret Agent who must Face a Boy in A Baloon
99	891	TIMBERLAND SKY	A Boring Display of a Man And a Dog who must Redeem a Girl in A U-Boat
100	903	TRAFFIC HOBBIT	A Amazing Epistle of a Squirrel And a Lumberjack who must Succumb a Database Administrator in A U-Boat

-- 2

```
create index IDX_BOAT on  film (instr(description,'Boat'));
```

-- 3

```
SQL> explain plan for select *
from film
where instr(description,'Boat')>0 and rownum<=100
order by title;

Explained.
```

```
SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------
-----------
Plan hash value: 2306941308

--------------------------------------------------------------------------
-------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
--------------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT    |      |   100 | 15200 |    16   (7)|
00:00:01 |
|   1 |  SORT ORDER BY      |      |   100 | 15200 |    16   (7)|
00:00:01 |
|*  2 |   COUNT STOPKEY     |      |       |       |            |
|
|*  3 |    TABLE ACCESS FULL| FILM |   100 | 15200 |    15   (0)|
00:00:01 |
--------------------------------------------------------------------------
-------


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

   2 - filter(ROWNUM<=100)
   3 - filter(INSTR("DESCRIPTION",'Boat')>0)

16 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2137789089

--------------------------------------------------------------------------
-----------
```

```
-------------

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

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

|   0 | SELECT STATEMENT                   |          |  8168 | 16336 |
29   (0
)| 00:00:01 |

|   1 |  COLLECTION ITERATOR PICKLER FETCH| DISPLAY |  8168 | 16336 |
29   (0
)| 00:00:01 |

-----------------------------------------------------------------------
```

```
SQL> explain plan for select *
from film
where instr(description,'Boat')>0 and rownum<=100
order by title;  2    3    4

Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2306941308

-----------------------------------------------------------------------------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |      |   100 | 15200 |    16   (7)| 00:00:01 |
|   1 |  SORT ORDER BY      |      |   100 | 15200 |    16   (7)| 00:00:01 |
|*  2 |   COUNT STOPKEY     |      |       |       |            |          |
|*  3 |    TABLE ACCESS FULL| FILM |   100 | 15200 |    15   (0)| 00:00:01 |
-----------------------------------------------------------------------------


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

   2 - filter(ROWNUM<=100)
   3 - filter(INSTR("DESCRIPTION",'Boat')>0)

16 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2137789089

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

| Id  | Operation                          | Name     | Rows  | Bytes | Cost (%CPU
```

```
SQL> explain plan for select *
from film
where instr(description,'Boat')>0 and rownum<=100
order by title;  2    3    4

Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
-----------
Plan hash value: 2306941308

--------------------------------------------------------------------
-------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
--------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT    |      |   100 | 15200 |    16   (7)|
00:00:01 |
|   1 |  SORT ORDER BY      |      |   100 | 15200 |    16   (7)|
00:00:01 |
|*  2 |   COUNT STOPKEY     |      |       |       |            |
|
|*  3 |    TABLE ACCESS FULL| FILM |   100 | 15200 |    15   (0)|
00:00:01 |
--------------------------------------------------------------------
-------


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

   2 - filter(ROWNUM<=100)
   3 - filter(INSTR("DESCRIPTION",'Boat')>0)
```

```
16 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2137789089

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

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

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

|   0 | SELECT STATEMENT                   |          |  8168 | 16336 |
29   (0
)| 00:00:01 |

|   1 |  COLLECTION ITERATOR PICKLER FETCH| DISPLAY |  8168 | 16336 |
29   (0
)| 00:00:01 |

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


SQL> create index IDX_BOAT on  film (instr(description,'Boat'));

Index created.

SQL> explain plan for select *
from film
where instr(description,'Boat')>0 and rownum<=100
order by title;   2    3    4

Explained.

SQL> select * from table(dbms_xplan.display());
```

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------
-----------
Plan hash value: 1934079693

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

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------
-----------
|   0 | SELECT STATEMENT                       |          |   100 |
15200 |     5
  (20)| 00:00:01 |

|   1 |  SORT ORDER BY                         |          |   100 |
15200 |     5
  (20)| 00:00:01 |

|*  2 |   COUNT STOPKEY                        |          |       |
|
      |          |

|   3 |    TABLE ACCESS BY INDEX ROWID BATCHED| FILM     |   101 |
15352 |     4
   (0)| 00:00:01 |

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

|*  4 |     INDEX RANGE SCAN                   | IDX_BOAT |   180 |
|     2
```

```
   (0)| 00:00:01 |

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


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


PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------
-----------
   2 - filter(ROWNUM<=100)
   4 - access(INSTR("DESCRIPTION",'Boat')>0)

17 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2137789089

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

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

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

|   0 | SELECT STATEMENT                   |         |  8168 | 16336 |
29   (0
)| 00:00:01 |

|   1 |  COLLECTION ITERATOR PICKLER FETCH| DISPLAY |  8168 | 16336 |
29   (0
)| 00:00:01 |
```

```
SQL> create index IDX_BOAT on  film (instr(description,'Boat'));

Index created.

SQL> explain plan for select *
from film
where instr(description,'Boat')>0 and rownum<=100
order by title;   2    3    4

Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1934079693

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

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |          |   100 | 15200 |     5
  (20)| 00:00:01 |

|   1 |  SORT ORDER BY                      |          |   100 | 15200 |     5
  (20)| 00:00:01 |

|*  2 |   COUNT STOPKEY                     |          |       |       |
      |          |

|   3 |    TABLE ACCESS BY INDEX ROWID BATCHED| FILM     |   101 | 15352 |     4
   (0)| 00:00:01 |

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

|*  4 |     INDEX RANGE SCAN                | IDX_BOAT |   180 |       |     2
   (0)| 00:00:01 |

--------------------------------------------------------------------------------
```

conclusion: the index indeed speed up the query processing.

(1) the first one which doesn’t have an index, use “TABLE ACCESS FULL” plan

(2) the second one which created an index on column description, use “TABLE ACCESS BY INDEX ROWID BATCHED’ plan

(3) the second one consumes time less than first one.

-- 4

```
-- 4
select count(f1.film_id)
from film f1
where exists (select count(film_id) from film f2 where f1.release_year=f2.release_year
 and f1.rating=f2.rating and f1.special_features=f2.special_features
 and f1.film_id!=f2.film_id
 having count(film_id)>=40
 );
```

脚本输出 × 查询结果 ×

SQL | 提取的所有行: 1, 用时 17.282 秒

	COUNT(F1.FILM_ID)
1	265

-- 5

I think in the three columns release_year, rating, and special_features, the column rating and special_features are suitable for using Hash indexes, and the release_year is suitable for B+ tree index.

because the rating column, it only has 'G', 'PG', 'PG-13', 'R', 'NC-17' five values and usually use “=” to query records.
and the special_features column, it only has null, 'Trailers', 'Commentaries', 'Deleted
Scenes', 'Behind the Scenes' values, and also use “=” to find records.
release_year column can use “>”,”<” and “>=”, “<=” to query records. so use B+ tree is more suitable.

Task5

-- 1

```
SQL>  select
index_name, blevel, leaf_blocks, num_rows, distinct_keys, clustering_factor
from user_ind_statistics
where table_name=UPPER('film');  2    3    4

INDEX_NAME
--------------------------------------------------------------------------------
    BLEVEL LEAF_BLOCKS   NUM_ROWS DISTINCT_KEYS CLUSTERING_FACTOR
---------- ----------- ---------- ------------- -----------------
PK_FILMID
         1          37      20002         20002               447

UN_DESCRIPTION
         2         296      20002         20002             19934

IDX_BOAT
         1          37      20002            84              1997
```

-- 2

SQL> explain plan for SELECT * FROM FILM WHERE FILM_ID > 100;

Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT

-
Plan hash value: 1232367652

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

| 0 | SELECT STATEMENT | | 19903 | 2993K| 136 (0)| 00:00:01 |
|* 1 | TABLE ACCESS FULL| FILM | 19903 | 2993K| 136 (0)| 00:00:01 |

Predicate Information (identified by operation id):

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-

   1 - filter("FILM_ID">100)

13 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2137789089

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

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

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

|   0 | SELECT STATEMENT                         |            |   8168 | 16336 |    29   (0
)| 00:00:01 |

|   1 |   COLLECTION ITERATOR PICKLER FETCH| DISPLAY |   8168 | 16336 |    29   (0
)| 00:00:01 |
```

```
SQL> explain plan for SELECT * FROM FILM WHERE FILM_ID > 100;

Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1232367652

---------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      | 19903 |  2993K|   136   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| FILM | 19903 |  2993K|   136   (0)| 00:00:01 |
---------------------------------------------------------------------------

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

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

   1 - filter("FILM_ID">100)

13 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2137789089

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

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

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

|   0 | SELECT STATEMENT                 |         |  8168 | 16336 |    29   (0
)| 00:00:01 |

|   1 |  COLLECTION ITERATOR PICKLER FETCH| DISPLAY |  8168 | 16336 |    29   (0
)| 00:00:01 |

--------------------------------------------------------------------------------
-------------
```

use full access plan

-- 3

rows number decreased to 8168 compared to the full table records.

-- 4

SQL> explain plan for SELECT * FROM FILM WHERE FILM_ID > 19990;

```
Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-
Plan hash value: 1620599584

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

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
-
|   0 | SELECT STATEMENT                            |               |    30 |  4620 |     3
  (0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| FILM          |    30 |  4620 |     3
  (0)| 00:00:01 |

|*  2 |   INDEX RANGE SCAN                          | PK_FILMID |    30 |       |     2
  (0)| 00:00:01 |

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

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


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

```
   2 - access("FILM_ID">19990)

14 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2137789089

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

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

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

|   0 | SELECT STATEMENT                         |            |   8168 | 16336 |    29   (0
)| 00:00:01 |

|   1 |  COLLECTION ITERATOR PICKLER FETCH| DISPLAY |   8168 | 16336 |    29   (0
)| 00:00:01 |
```

```
SQL> explain plan for SELECT * FROM FILM WHERE FILM_ID > 19990;

Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1620599584

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

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |           |    30 |  4620 |     3
  (0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| FILM      |    30 |  4620 |     3
  (0)| 00:00:01 |

|*  2 |   INDEX RANGE SCAN                  | PK_FILMID |    30 |       |     2
  (0)| 00:00:01 |

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

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


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

   2 - access("FILM_ID">19990)

14 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2137789089
```

use index range scan. row number is smaller than the task 5.3.

-- 5

SQL> explain plan for SELECT * FROM FILM WHERE FILM_ID = 100;

Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT

-

```
Plan hash value: 2104374699

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

| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| T
ime     |

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


PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------
-
|   0 | SELECT STATEMENT             |             |     1 |   154 |     2   (0)| 0
0:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID| FILM        |     1 |   154 |     2   (0)| 0
0:00:01 |

|*  2 |   INDEX UNIQUE SCAN          | PK_FILMID |     1 |       |     1   (0)| 0
0:00:01 |

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

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


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

   2 - access("FILM_ID"=100)

14 rows selected.


Execution Plan
```

```
-----------------------------------------------------------
Plan hash value: 2137789089

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

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

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

|   0 | SELECT STATEMENT                         |            |   8168 | 16336 |    29   (0
)| 00:00:01 |

|   1 |  COLLECTION ITERATOR PICKLER FETCH| DISPLAY |   8168 | 16336 |    29   (0
)| 00:00:01 |

-----------------------------------------------------------------------------------
-
-------------
```

```
SQL> explain plan for SELECT * FROM FILM WHERE FILM_ID = 100;

Explained.

SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2104374699

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

| Id  | Operation                   | Name      | Rows  | Bytes | Cost (%CPU)| T
ime     |

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |           |     1 |   154 |     2   (0)| 0
0:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID| FILM      |     1 |   154 |     2   (0)| 0
0:00:01 |

|*  2 |   INDEX UNIQUE SCAN         | PK_FILMID |     1 |       |     1   (0)| 0
0:00:01 |

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

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


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

   2 - access("FILM_ID"=100)

14 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2137789089
```

use index unique scan, the row number is only 1.