

Labor dokumentáció – Adatbázisok Laboratórium

5. mérés: Relációs lekérdezések optimalizálása

Név:	Kurely Mózes
Neptun kód:	F0QEL1
Feladat kódja:	28-28-REND
Mérésvezető neve:	Barabás Martin
Mérés időpontja:	2022-11-18 10:15
Mérés helyszíne:	HSZK A
Megoldott feladatok:	1, 2, 3, (kivéve a 3/c)
Elérhető pontszám (plusz pontok nélkül):	13

Mérési feladatok megoldása

1. feladat: egyediség biztosítása

a,

A megoldáshoz használt SQL utasítás:

```
ALTER SESSION SET optimizer_adaptive_plans = false;
SELECT DISTINCT name, daily_cost FROM EDU_EVENT.items

WHERE purchase_date BETWEEN to_date('2015-01-01','yyyy-mm-dd') AND
to_date('2015-12-31','yyyy-mm-dd');
```

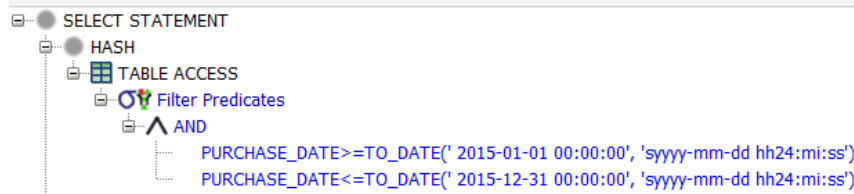
Magyarázat:

A SELECT-el kiválasztom mely oszlopok fognak megjelenülni a FROM-al kiválasztott táblából.

A DISTINCT-el jelzem hogy csak a különböző értékű neveket listázza ki.

A WHERE-el megadom a feladat kívánta feltételekt.

Végrehajtási terv:



Milyen módszerrel biztosítja az eredményhalmaz egyediségét az adatbáziskezelő?:

A HASH UNIQUE-al biztosítja az egyediséget

b,

A megoldáshoz használt SQL utasítás:

```
ALTER SESSION SET optimizer_adaptive_plans = false;

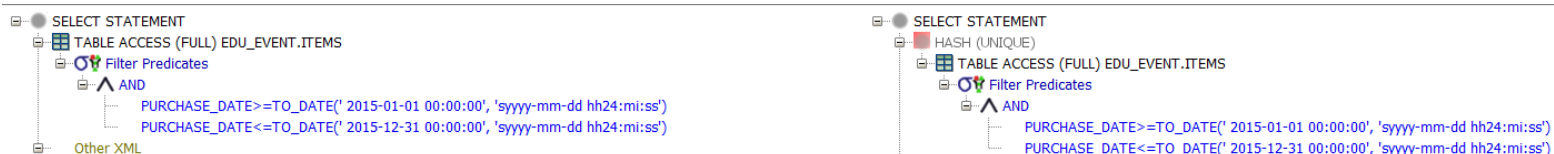
SELECT DISTINCT name, daily_cost, item_serial_code FROM
EDU_EVENT.items

WHERE purchase_date BETWEEN to_date('2015-01-01','yyyy-mm-dd') AND
to_date('2015-12-31','yyyy-mm-dd');
```

Magyarázat:

A SELECT utasítást kiegészítettem az item_serial_code-al.

Végrehajtási tervek összehasonlítása: (b – a)



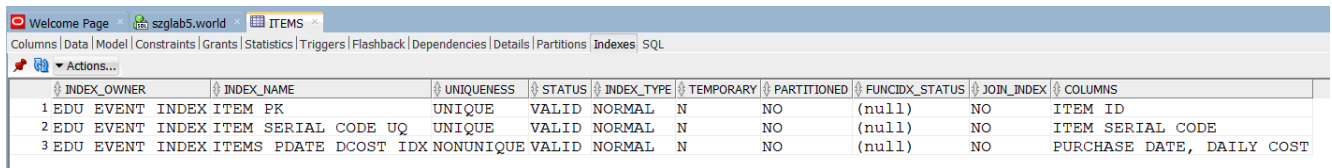
Mutasson rá és magyarázza meg a különbséget!

Nincs szükség a HASH fv-re ugyanis az item_serial_code már biztosítja a mezők egyediségét, mert az egy kulcs

2. feladat: indexelt

a,

EDU_EVENT_INDEX → ITEMS → Indexes



	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCTION_STATUS	JOIN_INDEX	COLUMNS
1	EDU_EVENT	INDEX ITEM PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ITEM ID
2	EDU_EVENT	INDEX ITEM SERIAL CODE UQ	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ITEM SERIAL CODE
3	EDU_EVENT	INDEX ITEMS PDATE Dcost IDX	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	PURCHASE DATE, DAILY COST

Innen kigyűjtöttem az Indexek nevét és a COLUMNS-ból pedig az attribútumokat:

ITEM_PK(ITEM_ID)

ITEM_SERIAL_CODE_UQ(ITEM_SERIAL_CODE)

ITEMS_PDATE_Dcost_IDX(PURCHASE_DATE, DAILY_COST)

b,

A megoldáshoz használt SQL utasítás:

```
ALTER SESSION SET optimizer_adaptive_plans = false;
```

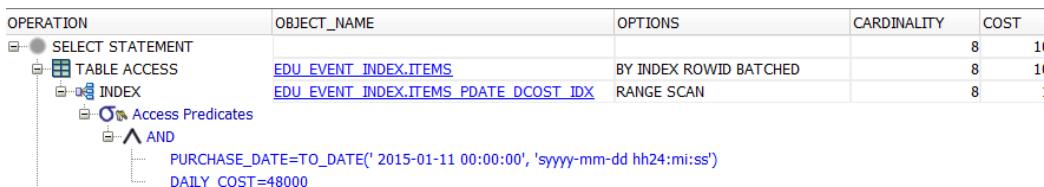
```
SELECT item_serial_code, name, daily_cost FROM EDU_EVENT_INDEX.items  
WHERE purchase_date = to_date('2015-01-11','yyyy-mm-dd') AND  
daily_cost = 48000 ;
```

Magyarázat:

A SELECT-el kiválasztom mely oszlopok fognak megjelenülni a FROM-al kiválasztott táblából.

A WHERE-el megadom a feladat kívánta feltételekt. A purchase_date a megfelelő legyen, valamint az összeg is a kért érték legyen.

Végrehajtási terv:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			8	10
TABLE ACCESS	EDU_EVENT_INDEX.ITEMS	BY INDEX ROWID BATCHED	8	10
INDEX	EDU_EVENT_INDEX.ITEMS_PDATE_Dcost_IDX	RANGE SCAN	8	1

Access Predicates:
AND
PURCHASE_DATE=TO_DATE(' 2015-01-11 00:00:00', 'yyyy-mm-dd hh24:mi:ss')
DAILY_COST=48000

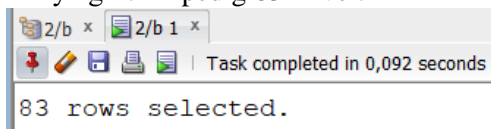
Milyen algoritmussal történik a keresés?:

A jobb oldalt lévő táblázat 2. sora a keresés, ami kiolvasható, hogy RANGE SCAN-el történik

Mennyi az eredmény becsült és tényleges rekordszáma?:

Becsült: Ez a CARDINALITY oszlopból látszik hogy 8

Tényleges: Ez pedig 83 mivel:



2/b x	2/b 1 x
Task completed in 0,092 seconds	
83 rows selected.	

c,

A megoldáshoz használt SQL utasítás:

```
ALTER SESSION SET optimizer_adaptive_plans = false;

SELECT item_serial_code, name, daily_cost FROM EDU_EVENT_INDEX.items

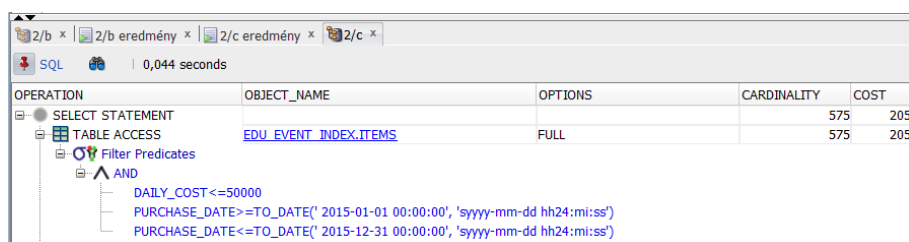
WHERE purchase_date BETWEEN to_date('2015-01-01','yyyy-mm-dd') AND
to_date('2015-12-31','yyyy-mm-dd')

AND daily_cost <= 50000 ;
```

Magyarázat:

A SELECT-el kiválasztom mely oszlopok fognak megjelenülni a FROM-al kiválasztott táblából.
A WHERE után megadom a feladat kívánta feltételeit.

Végrehajtási terv:



The screenshot shows the SQL Developer interface with the execution plan for query 'c'. The plan is as follows:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			575	205
TABLE ACCESS	EDU_EVENT_INDEX.ITEMS	FULL	575	205

The diagram on the left shows the query structure: a SELECT statement with a table access to EDU_EVENT_INDEX.ITEMS, followed by an AND predicate containing three conditions: DAILY_COST <= 50000, PURCHASE_DATE >= TO_DATE('2015-01-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss'), and PURCHASE_DATE <= TO_DATE('2015-12-31 00:00:00', 'yyyy-mm-dd hh24:mi:ss').

Hogyan változott a végrehajtási terv?:

Nem használja az indexes keresést, mert a daily_costra nem egyenlőségi keresés van megadva. Így végigmegy az összes rekordon. Azért van FULL a táblázatban.

d,

A megoldáshoz használt SQL utasítás:

```
ALTER SESSION SET optimizer_adaptive_plans = false;

SELECT item_serial_code, name, daily_cost FROM EDU_EVENT_INDEX.items

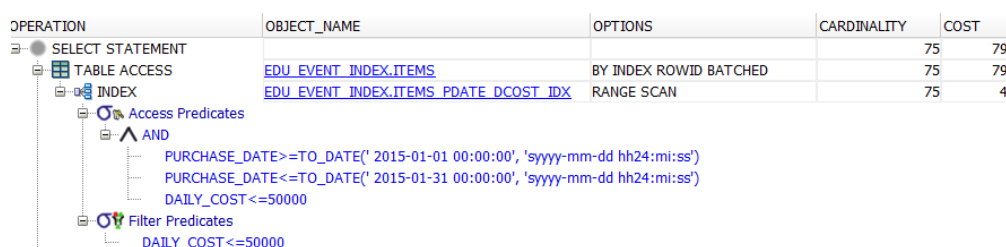
WHERE purchase_date BETWEEN to_date('2015-01-01','yyyy-mm-dd') AND
to_date('2015-01-31','yyyy-mm-dd')

AND daily_cost <= 50000 ;
```

Magyarázat:

Módosítottam a dátumot 2015. 01. 31.-re

Végrehajtási terv:



The screenshot shows the SQL Developer interface with the execution plan for query 'd'. The plan is as follows:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			75	79
TABLE ACCESS	EDU_EVENT_INDEX.ITEMS	BY INDEX ROWID BATCHED	75	79
INDEX	EDU_EVENT_INDEX.ITEMS_PDATE_DCost_IDX	RANGE SCAN	75	4

The diagram on the left shows the query structure: a SELECT statement with a table access to EDU_EVENT_INDEX.ITEMS using an index (EDU_EVENT_INDEX.ITEMS_PDATE_DCost_IDX) via a range scan. The index is used for the PURCHASE_DATE >= TO_DATE('2015-01-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and PURCHASE_DATE <= TO_DATE('2015-01-31 00:00:00', 'yyyy-mm-dd hh24:mi:ss') predicates. The DAILY_COST <= 50000 predicate is a filter predicate.

Hogyan változott a végrehajtási terv?:

Már használ INDEX RANGE SCAN-t az EDU_EVENT_INDEX.ITEMS_PDATE_DCost_IDX alapján

e,

Az eredményhalmaz becsült számossága		Az items tábla tárolására használt blokkok számá
c, feladat	d, feladat	
575	75	748

Miért változott a keresési algoritmus?:

Mivel a kardinalitás alapján a c, részben majdnem az egész items tábla bele fog esni ezért nem volt érdemes INDEX RANGE SCANT használni. Hanem végigment az egész táblán (FULL). Míg d,-ben jóval kevesebb volt a becsült számosság, ezért itt már érdemes volt azt használni.

3. feladat: JOIN-ok

a,

A megoldáshoz használt SQL utasítás:

```
ALTER SESSION SET optimizer_adaptive_plans = false;

SELECT edu_event.events.event_id ,edu_event.events.customer_name
FROM EDU_EVENT.events, EDU_EVENT.orders

WHERE edu_event.events.customer_name LIKE 'T%'
AND edu_event.events.time_of_day = 'd'
AND edu_event.events.event_date < to_date('2017-02-01','yyyy-mm-dd')

AND edu_event.orders.event_id = edu_event.events.event_id;
```

Magyarázat:

A SELECT-el kiválasztom a kívánt paramétereket a FROM-mal megadott events táblából valamint ott megadom az orders táblát is mivel a szövegben megadott feltételeken kívül a WHERE-ben illeszttem az events és az orders táblákat.

Végrehajtási terv:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			54	444
HASH JOIN			54	444
Access Predicates				
ORDERS.EVENT_ID=EVENTS.EVENT_ID				
TABLE ACCESS	EDU_EVENT.EVENTS	FULL	13	136
Filter Predicates				
AND				
EVENTS.EVENT_DATE<TO_DATE(' 2017-02-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
EVENTS.CUSTOMER_NAME LIKE 'T%'				
EVENTS.TIME_OF_DAY='d'				
TABLE ACCESS	EDU_EVENT.ORDERS	FULL	120000	308

Milyen algoritmussal történik a join kiszámítása?:

A join a HASH JOIN algoritmussal van számítva

Melyik reláció és milyen rekordszámmal kerül a külső és melyik a belső ciklusba?:

Külső: Az EVENTS reláció kerül 13 rekordszámmal

Belső: Az ORDERS reláció 120000-as rekordszámmal

b,

A megoldáshoz használt SQL utasítás:

```
ALTER SESSION SET optimizer_adaptive_plans = false;

SELECT edu_event_index.events.event_id,
edu_event_index.events.customer_name
FROM EDU_EVENT_index.events, EDU_EVENT_index.orders

WHERE edu_event_index.events.customer_name LIKE 'T%' AND
edu_event_index.events.time_of_day = 'd' AND
edu_event_index.events.event_date < to_date('2017-02-01','yyyy-mm-dd')

AND
edu_event_index.orders.event_id = edu_event_index.events.event_id;
```

Magyarázat:

Mindent átírtam, hogy az index-el tábában fusson le.

Milyen indexet használ a végrehajtáshoz az Oracle?:

RANGE SCAN indexet használ az EDU_EVENT_INDEX.ORDERS_EVENT_IDX alapján valamint érdekes, hogy NESTED LOOPS-t használ HASH JOIN helyett

Végrehajtási terv:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			53	149
NESTED LOOPS			53	149
TABLE ACCESS	EDU_EVENT_INDEX.EVENTS	FULL	13	136
Filter Predicates				
AND				
	EVENTS.EVENT_DATE<TO_DATE(' 2017-02-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss')			
	EVENTS.CUSTOMER_NAME LIKE U'T%			
	EVENTS.TIME_OF_DAY=U'd'			
INDEX	EDU_EVENT_INDEX.ORDERS_EVENT_IDX	RANGE SCAN	4	1
Access Predicates				
	ORDERS.EVENT_ID=EVENTS.EVENT_ID			

d,

A megoldáshoz használt SQL utasítás:

```
ALTER SESSION SET optimizer_adaptive_plans = false;

SELECT edu_event_index.events.event_id,
edu_event_index.events.customer_name,
edu_event_index.orders.order_id,
edu_event_index.orders.item_id,
edu_event_index.orders.liable_person

FROM EDU_EVENT_index.events, EDU_EVENT_index.orders

WHERE edu_event_index.events.customer_name LIKE 'T%'
AND edu_event_index.events.time_of_day = 'd'
AND edu_event_index.events.event_date < to_date('2017-02-01','yyyy-
mm-dd')

AND
edu_event_index.orders.event_id = edu_event_index.events.event_id;
```

Magyarázat:

Kiegészítettem a kért megjelenítendő adatokkal

Végrehajtási terv:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			53	213
NESTED LOOPS			53	213
NESTED LOOPS			53	213
TABLE ACCESS	EDU_EVENT_INDEX.EVENTS	FULL	13	136
Filter Predicates				
AND				
EVENTS.EVENT_DATE<TO_DATE(' 2017-02-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
EVENTS.CUSTOMER_NAME LIKE U'T%'				
EVENTS.TIME_OF_DAY=U'd'				
INDEX	EDU_EVENT_INDEX.ORDERS_EVENT_IDX	RANGE SCAN	4	1
Access Predicates				
ORDERS.EVENT_ID=EVENTS.EVENT_ID				
TABLE ACCESS	EDU_EVENT_INDEX.ORDERS	BY INDEX ROWID	4	6

Hogyan és miért változott a végrehajtási terv?:

Egymásba ágyazott NESTED LOOPS-ok jötte létre. Az elsőben először szűr a megadott feltételekre az EVENTS táblában és a másodikban illeszti össze az ORDERS táblával hiszen ez a leghatékonyabb megoldás. Azért változott meg mert most több adatot kell az EVENTS táblához illeszteni mint a b-ben így már egyszerűbb ha előbb szűrünk és csak aztán illesztünk.