72.41 - Bases de Datos II:
Tema 2: Cálculo de reputación de un sitio de e-commerce

Integrantes:

- Kevin Chidiac
- Daniel Lobo
- Romain Latron

Objetivos

- Realizar modificaciones para mejorar la performance de un script Oracle SQL de una base de datos de una empresa de e-commerce, puntualmente sobre el proceso de actualización de la reputación de vendedores.

- Para solucionar esto se realizaron mejoras de dos tipos:
  - Mejoras sobre el esquema
  - Mejoras sobre los procedures

- El script que hemos mejorado es el *calculate_seller_scoring.sql*

Lógica de negocio y explicación del proceso

- Cada venta es registrada en la tabla de SALE.
- Por cada venta se generan dos reviews.
- Al mismo tiempo cada review tiene las siguientes características:
  - Calidad: POSITIVA, NEUTRA, NEGATIVA
  - Estado: PUBLISHED, PENDING
  - Fechas: CREATED, MODIFIED
- El proceso calcula la reputación de cada vendedor y se ejecuta toda las noches.
- Los cambios se ven reflejados en la tabla SCORE.

Mejoras implementadas:
Mejoras sobre el esquema: Cambios en los tipos

- STATUS

```
AlTER TABLE Score
MODIFY Status varchar2(10);
/
```

- DATE

```
AlTER TABLE EUSER
MODIFY CREATED date;
/
AlTER TABLE EUSER
MODIFY MODIFIED date;
/

AlTER TABLE SCORE
MODIFY CREATED date;
/
AlTER TABLE SCORE
MODIFY MODIFIED date;
```

Mejoras implementadas:
Mejoras sobre el esquema: Cambios en los PCT FREE

- Para las tablas EUSER, REVIEW, SALE seteamos los valores en 0 porque solamente hacemos inserciones y no updates, por lo tanto no es necesario reservar espacio para posibles updates en los distintos bloques
  - Las modificaciones en EUSER no van a cambiar el tamaño del bloque.
  - Los campos de EUSER no tienden a variar.
- En el caso de REVIEW, solamente hacemos inserción.
- En el caso de SALE, es análogo a REVIEW.

```
ALTER TABLE EUSER PCTFREE 0;
ALTER TABLE REVIEW PCTFREE 0;
ALTER TABLE SALE PCTFREE 0;
```

Mejoras implementadas:
Mejoras sobre el esquema: Cambios en los PCT FREE

- En el caso de SCORE, el caso es distinto porque si va a tener actualizaciones de todas maneras pero no demasiadas, entonces decidimos setearlo en 10.

```
CREATE TABLE "BDII_TEAM3"."SCORE"
( "USER_ID" NUMBER(10,0),
  "LAST_WEEK_POSITIVE" NUMBER(10,0),
  "LAST_WEEK_NEUTRAL" NUMBER(10,0),
  "LAST_WEEK_NEGATIVE" NUMBER(10,0),
  "LAST_MONTH_POSITIVE" NUMBER(10,0),
  "LAST_MONTH_NEUTRAL" NUMBER(10,0),
  "LAST_MONTH_NEGATIVE" NUMBER(10,0),
  "LAST_6MONTH_POSITIVE" NUMBER(10,0),
  "LAST_6MONTH_NEUTRAL" NUMBER(10,0),
  "LAST_6MONTH_NEGATIVE" NUMBER(10,0),
  "SCORE" NUMBER(10,2),
  "STATUS" CHAR(10 BYTE),
  "CREATED" TIMESTAMP (6),
  "MODIFIED" TIMESTAMP (6)
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 10 INITRANS 1 MAXTRAN
STORAGE(INITIAL 10240 NEXT 10240 MINEXTE
PCTINCREASE 5 FREELISTS 1 FREELIST GROUP
TABLESPACE "BDII_TEAM3_DATA" ;
```

Mejoras implementadas:
Mejoras sobre el esquema: Primary keys y foreign keys

- Se agregó una sola Foreign Key
  - Se relaciona un score con un user.
  - Se fuerza que exista un usuario referenciado con cada score

```
/
ALTER TABLE SCORE ADD CONSTRAINT "SCORE_EUSER_FK" FOREIGN KEY ("USER_ID")
    REFERENCES "BDII_TEAM2"."EUSER" ("ID") ENABLE;
/
```

Mejoras implementadas:
Mejoras sobre el esquema: Cambios en los índices

- Creación de índices

```
CREATE INDEX REVIEW_USER_MODIFIED_IDX
 ON REVIEW ("USER_ID","ROLE", "STATUS", "MODIFIED")
 tablespace team2_indexes;
 /

 CREATE INDEX REVIEW_USER_CREATED_IDX
 ON REVIEW ("USER_ID","CREATED")
 tablespace team2_indexes;
 /
```

Mejoras implementadas:

Mejoras sobre el esquema: Index organized table

- Tabla SCORE
- Originalmente se usaba solamente el USER_ID
- USER_ID es simplemente un identificador y que no tenía sentido este procesamiento en memoria al momento de ingresar los datos.

Mejoras implementadas:
Mejoras sobre los procedures: Variable vRATIO

- Error grave:
  - No se pueden representar ni el -1 ni el 1, solamente los valores que están entre ellos.
- Como la consulta devuelve valores que pueden llegar a ser -1 y 1, necesitamos poder representarlos.

```
-- Procedimiento que actualiza la reputacion de un vendedor
PROCEDURE update_seller_scoring( pUserId NUMBER,
    p7Positive NUMBER, p7Negative NUMBER, p7Neutral NUMBER,
    p30Positive NUMBER, p30Negative NUMBER, p30Neutral NUMBER,
    p180Positive NUMBER, p180Negative NUMBER, p180Neutral NUMBER,
    pScoreTotal NUMBER, pCountTotal NUMBER
    ) AS
    vRATIO NUMBER(3,2):=0;
BEGIN
```

Mejoras implementadas:
Mejoras sobre los procedures: Cursores

- Se modificó el cursor de USER_IDs con el objetivo de verificar esta condición de PUBLISHED antes de hacer los FOR internos que tenía el script.
- Al mismo tiempo, se sacó esa condición dentro de los FOR internos.

```
-- Cursor de vendedores cuyas calificaciones sufrieron modificaciones
-- o cuya reputacion hay que actualizar porque es "vieja" (de mas de 1 semana)
CURSOR SELLER_CUR( pInterval NUMBER ) IS
SELECT DISTINCT USER_ID
FROM REVIEW
WHERE ROLE = 'SELLER'
AND STATUS = 'PUBLISHED'
AND MODIFIED >= ( SYSDATE - pInterval )
UNION
SELECT USER_ID
FROM SCORE
WHERE MODIFIED >= ( SYSDATE - 7 )
;
```

Mejoras implementadas:
Mejoras sobre los procedures: Cursores

- **La mejora más sustancial en cuanto performance de toda la mejora del script:** se agregó un nuevo cursor:
  - Trabaja mejor en conjunto con el otro cursor.
  - Elimina la repetición de código.
  - Elimina la repetición de lógica.

```
-- Por cada seller,
FOR SELLER_REC IN SELLER_CUR(ndays) LOOP
  -- Buscar el puntaje recibido en las ventas de la ultima semana
  SELECT COUNT(1)
  INTO v7Positive
  FROM REVIEW
  WHERE USER_ID = SELLER_REC.USER_ID
  AND ROLE = 'SELLER'
  AND CREATED > SYSDATE - 7
  AND STATUS = 'PUBLISHED'
  AND SCORE = 'POSITIVE'
  ;
  SELECT COUNT(1)
  INTO v7Negative
  FROM REVIEW
  WHERE USER_ID = SELLER_REC.USER_ID
  AND ROLE = 'SELLER'
  AND CREATED > SYSDATE - 7
  AND STATUS = 'PUBLISHED'
  AND SCORE = 'NEGATIVE'
  ;
  SELECT COUNT(1)
  INTO v7Neutral
  FROM REVIEW
  WHERE USER_ID = SELLER_REC.USER_ID
  AND ROLE = 'SELLER'
  AND CREATED > SYSDATE - 7
  AND STATUS = 'PUBLISHED'
  AND SCORE = 'NEUTRAL'
  ;
```

```
;
-- Buscar el puntaje recibido en las ventas del ultimo mes
SELECT COUNT(1)
INTO v30Positive
FROM REVIEW
WHERE USER_ID = SELLER_REC.USER_ID
AND ROLE = 'SELLER'
AND CREATED > SYSDATE - 30
AND STATUS = 'PUBLISHED'
AND SCORE = 'POSITIVE'
;
SELECT COUNT(1)
INTO v30Negative
FROM REVIEW
WHERE USER_ID = SELLER_REC.USER_ID
AND CREATED > SYSDATE - 30
AND ROLE = 'SELLER'
AND STATUS = 'PUBLISHED'
AND SCORE = 'NEGATIVE'
;
SELECT COUNT(1)
INTO v30Neutral
FROM REVIEW
WHERE USER_ID = SELLER_REC.USER_ID
AND CREATED > SYSDATE - 30
AND ROLE = 'SELLER'
AND STATUS = 'PUBLISHED'
AND SCORE = 'NEUTRAL'
;
```

```
;
-- Buscar el puntaje recibido en las ventas de los ultimos seis meses
SELECT COUNT(1)
INTO v180Positive
FROM REVIEW
WHERE USER_ID = SELLER_REC.USER_ID
AND CREATED > SYSDATE - 180
AND ROLE = 'SELLER'
AND STATUS = 'PUBLISHED'
AND SCORE = 'POSITIVE'
;
SELECT COUNT(1)
INTO v180Negative
FROM REVIEW
WHERE USER_ID = SELLER_REC.USER_ID
AND CREATED > SYSDATE - 180
AND ROLE = 'SELLER'
AND STATUS = 'PUBLISHED'
AND SCORE = 'NEGATIVE'
;
SELECT COUNT(1)
INTO v180Neutral
FROM REVIEW
WHERE USER_ID = SELLER_REC.USER_ID
AND CREATED > SYSDATE - 180
AND ROLE = 'SELLER'
AND STATUS = 'PUBLISHED'
AND SCORE = 'NEUTRAL'
;
```

Mejoras implementadas:
Mejoras sobre los procedures: Cursores

- Terminó siendo...

- El nuevo cursor agregado es de REVIEWs:

```
CURSOR LAST_MONTHS_REVIEWS(pUserId NUMBER ) IS
        SELECT CREATED, SCORE
        FROM REVIEW
        WHERE USER_ID = pUserId
        AND CREATED > SYSDATE - 180;
        FOR SELLER_REC IN SELLER_CUR(ndays)
```

```
LOOP
 -- Buscar el puntaje recibido en las ventas de la ultima semana
    FOR REVIEW_REC IN LAST_MONTHS_REVIEWS(SELLER_REC.USER_ID) LOOP
  IF REVIEW_REC.SCORE = 'POSITIVE'  THEN
        v180Positive := v180Positive + 1;
        IF REVIEW_REC.CREATED >  SYSDATE - 30  THEN
            v30Positive := v30Positive + 1;
            IF REVIEW_REC.CREATED >  SYSDATE - 7  THEN
                v7Positive := v7Positive + 1;
            END IF;
        END IF;

    ELSIF REVIEW_REC.SCORE = 'NEUTRAL'  THEN
        v180Neutral := v180Neutral + 1;
        IF REVIEW_REC.CREATED >  SYSDATE - 30  THEN
            v30Neutral := v30Neutral + 1;
            IF REVIEW_REC.CREATED >  SYSDATE - 7  THEN
                v7Neutral := v7Neutral + 1;
            END IF;
        END IF;

    ELSIF REVIEW_REC.SCORE = 'NEGATIVE'  THEN
        v180Negative := v180Negative + 1;
        IF REVIEW_REC.CREATED >  SYSDATE - 30  THEN
            v30Negative := v30Negative + 1;
            IF REVIEW_REC.CREATED >  SYSDATE - 7  THEN
                v7Negative := v7Negative + 1;
            END IF;
        END IF;
    END IF;
END LOOP;
```

Resultados: Naturaleza de los datos proveídos

- Los resultados obtenidos en cuanto a performance podrían ser mejores si los datos de la tabla REVIEW en el campo MODIFIED no tuvieran todos el mismo valor.
- Se tomó la decisión de no tocar los datos que la empresa
- Cuando hay una mala función hash o una mala distribución de valores a lo largo de una estructura, esto causa problemas de performance.

Resultados: Resultados experimentales

- Todos los resultados son medidos con un script inicial.

```
DECLARE
  NDAYS NUMBER;
BEGIN
  NDAYS := 1000;

  SCORING_PKG.CALCULATE_SELLER_SCORING(
    NDAYS => NDAYS
  );
--rollback;
END;
/
```

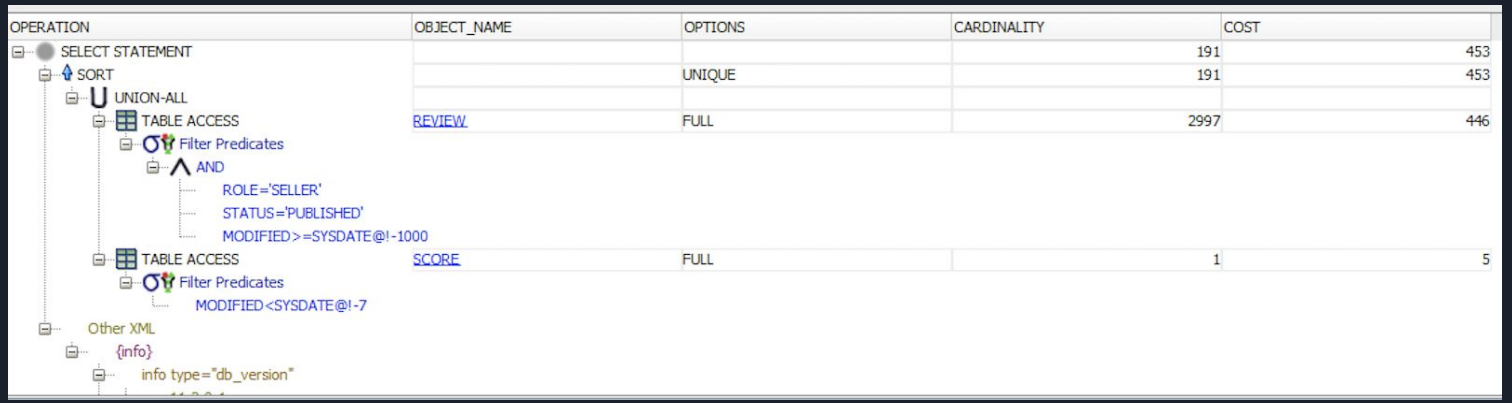
Resultados: Resultados experimentales

- Los valores experimentales obtenidos fueron realizaron sobre las siguientes modificaciones:

Modificacion	Tiempo (s)
Sin modification	3.918
Nuevo cursor	0.825
Con indices	0.153

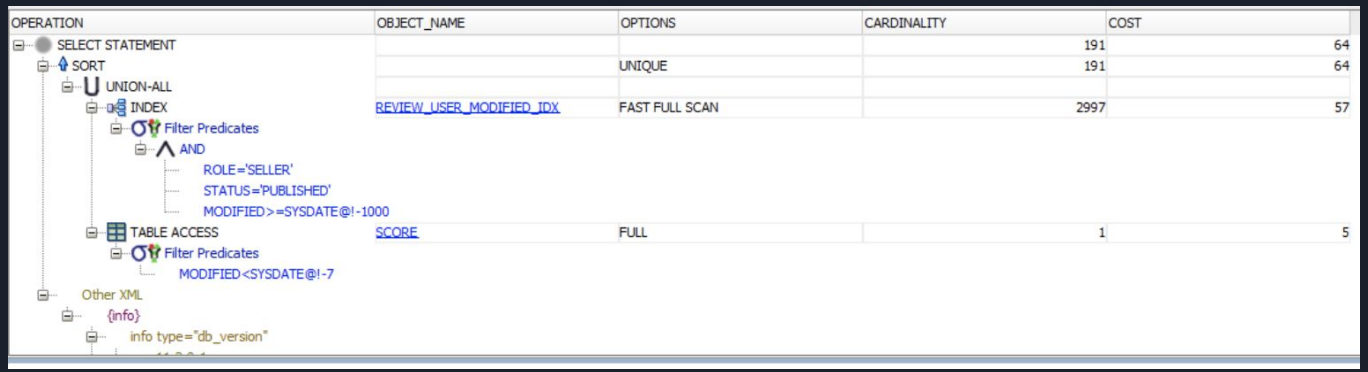
Resultados: Resultados experimentales

- **Sin uso de índices para el cursor de USER_ID**, se puede observar:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			191	453
SORT		UNIQUE	191	453
UNION-ALL				
TABLE ACCESS	REVIEW	FULL	2997	446
Filter Predicates				
AND				
ROLE='SELLER'				
STATUS='PUBLISHED'				
MODIFIED>=SYSDATE@!-1000				
TABLE ACCESS	SCORE	FULL	1	5
Filter Predicates				
MODIFIED<SYSDATE@!-7				
Other XML				
{info}				
info type="db_version"				

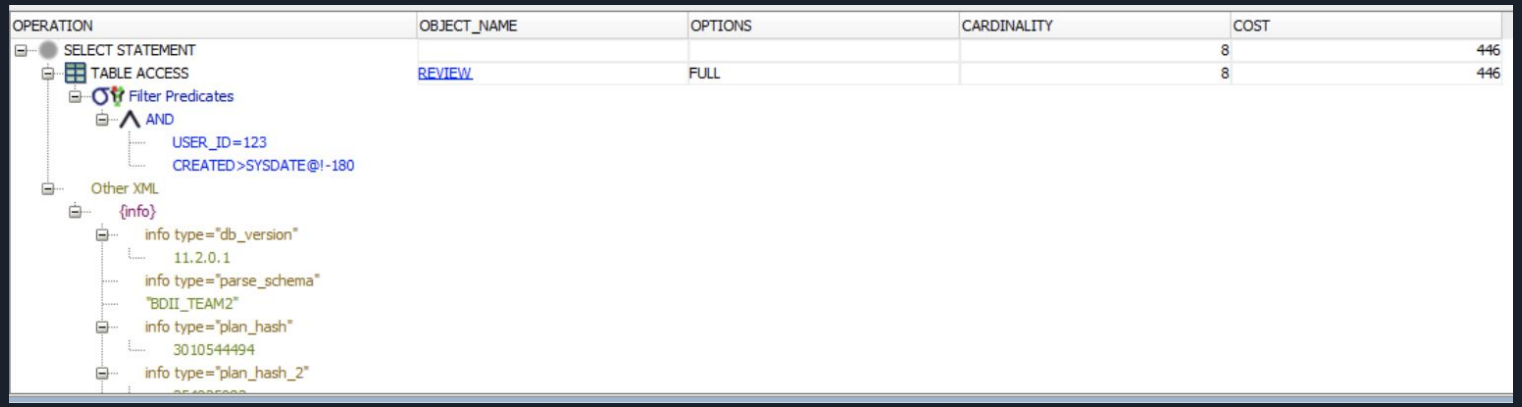
- **Con uso de índices para el cursor de USER_ID**, se puede observar:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			191	64
SORT		UNIQUE	191	64
UNION-ALL				
INDEX	REVIEW_USER_MODIFIED_IDX	FAST FULL SCAN	2997	57
Filter Predicates				
AND				
ROLE='SELLER'				
STATUS='PUBLISHED'				
MODIFIED>=SYSDATE@!-1000				
TABLE ACCESS	SCORE	FULL	1	5
Filter Predicates				
MODIFIED<SYSDATE@!-7				
Other XML				
{info}				
info type="db_version"				

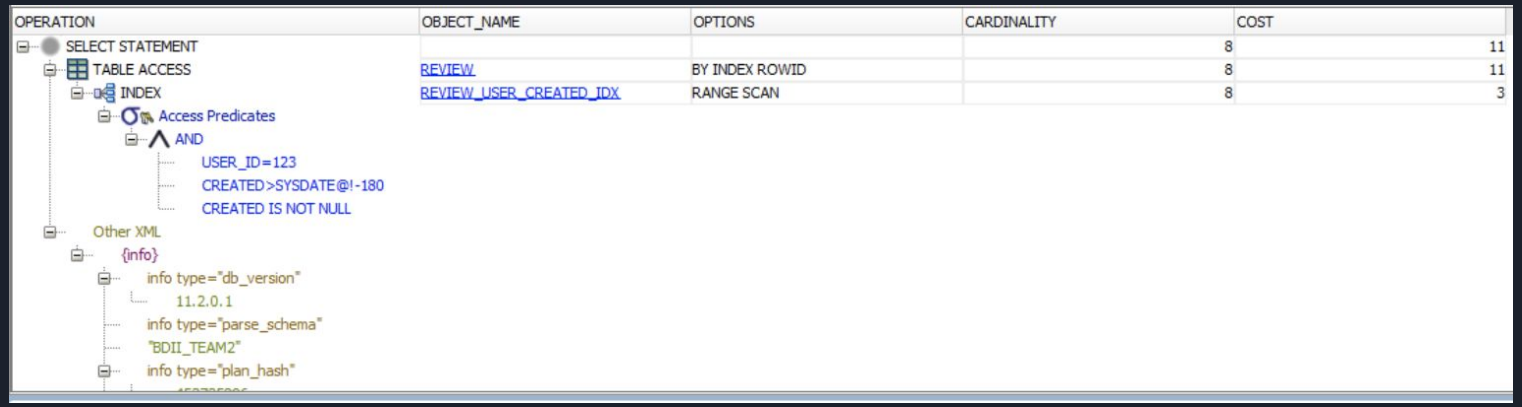
Resultados: Resultados experimentales

- **Sin uso de índices para el cursor de REVIEW**, se puede observar:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			8	446
TABLE ACCESS	REVIEW	FULL	8	446
Filter Predicates				
AND				
USER_ID=123				
CREATED>SYSDATE@!-180				
Other XML				
{info}				
info type="db_version"				
11.2.0.1				
info type="parse_schema"				
"BDII_TEAM2"				
info type="plan_hash"				
3010544494				
info type="plan_hash_2"				

- **Con uso de índices para el cursor de REVIEW**, se puede observar:



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			8	11
TABLE ACCESS	REVIEW	BY INDEX ROWID	8	11
INDEX	REVIEW_USER_CREATED_IDX	RANGE SCAN	8	3
Access Predicates				
AND				
USER_ID=123				
CREATED>SYSDATE@!-180				
CREATED IS NOT NULL				
Other XML				
{info}				
info type="db_version"				
11.2.0.1				
info type="parse_schema"				
"BDII_TEAM2"				
info type="plan_hash"				

Conclusiones

1. Mejoras sobre el esquema
    a. Mejor uso de espacio
    b. Mejor semántica
2. Mejoras sobre los procedures
    a. Mejoras de performance
    b. Mejoras de velocidad