5

Tables Index OF

Packages

Index

INDEX BY TABLE – INDEX BY Table of records

Définir des variables qui contiennent des collections

- Clé / valeur
- Clé / enregistrement

Déclaration

- Type

- Variable de ce type

- Parcourir

```
DECLARE

  TYPE list_of_names_t IS
          TABLE OF VARCHAR2(30)  INDEX BY BINARY_INTEGER;

  happy_persons list_of_names_t;

  l_row BINARY_INTEGER;

BEGIN
  happy_persons(202020) := 'Christine';
  happy_persons(-150) := 'Isabelle';
  happy_persons(-901) := 'Pierre';
  happy_persons(88) := 'John';
```

```
l_row := happy_persons.FIRST;

  WHILE (l_row IS NOT NULL)
  LOOP
    DBMS_OUTPUT.PUT_LINE(happy_persons(l_row));
    l_row := happy_persons.NEXT(l_row);
  END LOOP;

END;
/
```

Pierre
Isabelle
John
Christine

```
DECLARE
  TYPE t_tsh_name IS TABLE OF type_show_tsh.tsh_name%TYPE INDEX BY BINARY_INTEGER;
  v_tsh_name_tab t_tsh_name;
BEGIN

 FOR tsh_name_rec IN (SELECT tsh_id, tsh_name FROM type_show_tsh ) LOOP
      v_tsh_name_tab(tsh_name_rec.tsh_id) := tsh_name_rec.tsh_name;
  END LOOP;

FOR i IN v_tsh_name_tab.FIRST .. v_tsh_name_tab.LAST LOOP
    IF v_tsh_name_tab.EXISTS(i) THEN
       DBMS_OUTPUT.PUT_LINE(i || ' ' || v_tsh_name_tab(i));
    END IF;
  END LOOP;
END;
/
```

```
DECLARE
  CURSOR shw_cur IS SELECT shw_id, shw_title, shw_date
                                FROM show_shw
                                 ORDER BY shw_id;
  TYPE t_shw IS TABLE OF shw_cur%ROWTYPE INDEX BY BINARY_INTEGER;
   v_shw_tab t_shw;
BEGIN

  FOR shw_cur_rec IN shw_cur LOOP
      v_shw_tab(shw_cur_rec.shw_id) := shw_cur_rec;
  END LOOP;

 FOR i IN v_shw_tab.FIRST .. v_shw_tab.LAST LOOP
    IF v_shw_tab.EXISTS(i) THEN
       DBMS_OUTPUT.PUT_LINE(i || ' ' || v_shw_tab(i).shw_title);
    END IF;
  END LOOP;
END;
/
```

Package – Oracle – section 10

Exécution d’une requête

1 - SELECTION

```
SELECT bkg_id, bkg_date, bkg_cst_id
  FROM booking_bkg
  WHERE EXTRACT (YEAR FROM bkg_date) = 2016;
```

2 – JOIN + SELECTION

```
SELECT *
  FROM booking_bkg INNER JOIN customer_cst
    ON bkg_cst_id = cst_id
  WHERE cst_last_name LIKE 'M%';
```

1 - sélection

Table booking

- > 6000 lignes
- 5398 lignes correspondent

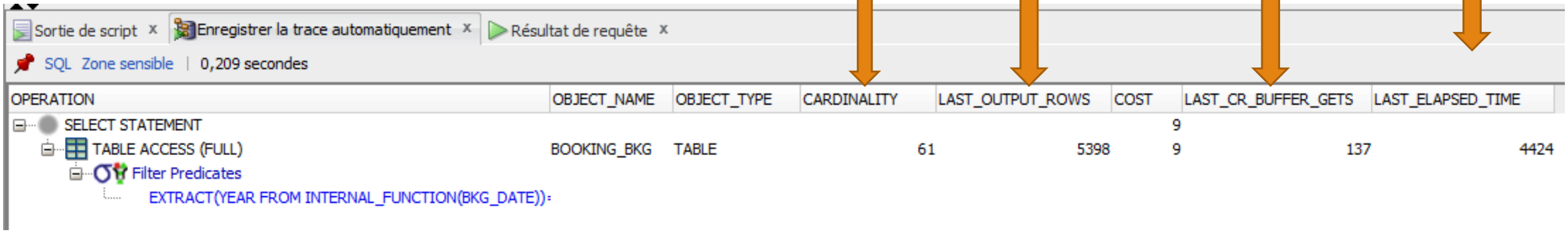
Oracle réalise un parcours complet (FULL TABLE SCAN)

How long the step took to execute – micro seconds

How many rows were produced

How many rows oracle think it would return

How much data was moved



OPERATION	OBJECT_NAME	OBJECT_TYPE	CARDINALITY	LAST_OUTPUT_ROWS	COST	LAST_CR_BUFFER_GETS	LAST_ELAPSED_TIME
SELECT STATEMENT					9		
TABLE ACCESS (FULL)	BOOKING_BKG	TABLE	61	5398	9	137	4424
Filter Predicates							
EXTRACT(YEAR FROM INTERNAL_FUNCTION(BKG_DATE))							

2 : JOIN + sélection

Operations ?

- JOIN
- SELECTION

Laquelle est exécutée d'abord ?

- Combien de lignes ?
  - env 6000 lignes dans booking_bkg
  - env 100 lignes dans customer_cst
  - 10 clients avec M…
- JOINTURE d'abord ?
- SELECTION d'abord ?

JOIN

Nested loop

- Table booking : full table scan – Pour chaque ligne
  - Contrôle la table customer_cst (full scan => plus de 100 tests)
- Jusqu'à la fin de la table booking_bkg (6000 rows)

Produira 6000 lignes

Puis exécution de la sélection (WHERE)

Sélection

Parcours de la table customer

- Full scan 100 lignes

Résultat = 10 lignes

Jointure de booking (6000 lignes) avec le résultat précédent (10 rows)

Qui décide ?

```
SELECT *
FROM booking_bkg INNER JOIN customer_cst
  ON bkg_cst_id = cst_id
WHERE cst_last_name LIKE 'M%';
```

Oracle

- Optimiseur
- Plans d’exécution
- Evaluer les coûts

Index

Permet d’accélérer les requêtes (éviter le FULL TABLE SCAN)

Fournit un accès direct et rapide aux lignes de la table

- ROWID : permet de localiser la ligne (fichier, bloc, position dans le bloc)

Réduit les opérations de lecture sur les supports physiques

Est utilisé et maintenu automatiquement par oracle

Est créé ou supprimé indépendamment de la table

Types d’index

UNIQUE

- Créé automatiquement
  - UNIQUE
  - PRIMARY KEY

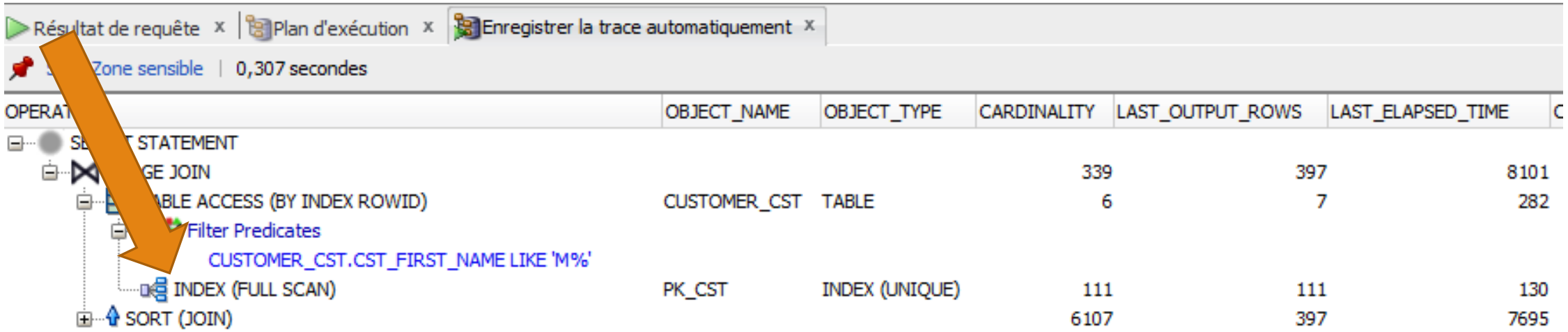
NONUNIQUE

- Créé pour accélérer les requêtes
- Par exemple sur la clé étrangère

Indexes créés par Oracle

PRIMARY KEY, UNIQUE

```
SELECT *
FROM booking_bkg
INNER JOIN customer_cst
ON bkg_cst_id = cst_id
WHERE cst_first_name LIKE 'M%'
ORDER BY cst_id;
```



OPERAT	OBJECT_NAME	OBJECT_TYPE	CARDINALITY	LAST_OUTPUT_ROWS	LAST_ELAPSED_TIME	C
SE T STATEMENT						
GE JOIN			339	397	8101	
ABLE ACCESS (BY INDEX ROWID)	CUSTOMER_CST	TABLE	6	7	282	
Filter Predicates						
CUSTOMER_CST.CST_FIRST_NAME LIKE 'M%'						
INDEX (FULL SCAN)	PK_CST	INDEX (UNIQUE)	111	111	130	
SORT (JOIN)			6107	397	7695	

Pour créer un index

```
CREATE INDEX index_name
    ON table_name(column_name);

DROP INDEX index_name;
```

Quand ?

Seulement si :

- La colonne contient un grand nombre de valeur NULL
- La colonne contient des valeurs très différentes
- Une (ou plusieurs) colonne est utilisée fréquemment dans un WHERE ou une condition de jointure
- La plupart des requêtes sur une table retourne entre 2 et 4% des lignes.

Chaque operation LMD (INSERT, UPDATE, DELETE) réalisée sur une table avec des indexes va nécessiter la mise à jour de l'index