CONFIDENTIAL

RENDU TD-8 : ACHEHBOUNE Mohammed-Amine

Étape 1 : Création de la table et vérification que la table est vide :

```
SQL>
SQL> CREATE TABLE CLIENTS_TEST (
  2      ID_CLIENT NUMBER,
  3      NOM VARCHAR2(100),
  4      PRENOM VARCHAR2(100),
  5      EMAIL VARCHAR2(150),
  6      DATE_INSCRIPTION DATE,
  7      COMMENTS VARCHAR2(4000) -- Colonne "lourde" pour augmenter la taille des blocs
  8  );

Table created.

SQL> SELECT COUNT(*) FROM CLIENTS_TEST;

  COUNT(*)
----------
         0
```

Étape 2 : Injection massive de données (Palier 1) :

```
SQL> DECLARE
  2      v_nb_rows NUMBER := 50000;
  3  BEGIN
  4      FOR i IN 1..v_nb_rows LOOP
  5          INSERT INTO CLIENTS_TEST (ID_CLIENT, NOM, PRENOM, EMAIL, DATE_INSCRIPTION, COMMENTS)
  6          VALUES (
  7              i,
  8              'Nom_' || DBMS_RANDOM.STRING('U', 5),
  9              'Prenom_' || DBMS_RANDOM.STRING('U', 5),
 10              'user_' || i || '@test.com',
 11              SYSDATE - DBMS_RANDOM.VALUE(0, 3650),
 12              RPAD('A', 200, 'A')
 13          );
 14          IF MOD(i, 5000) = 0 THEN
 15              COMMIT;
 16          END IF;
 17      END LOOP;
 18      COMMIT;
 19  END;
 20  /

PL/SQL procedure successfully completed.
```

Vérifie le nombre de lignes :

```
SQL>
SQL> SELECT COUNT(*) FROM CLIENTS_TEST;

  COUNT(*)
----------
     50000

SQL>
```

Étape 3 : Test sans index

3.1 Analyse du plan

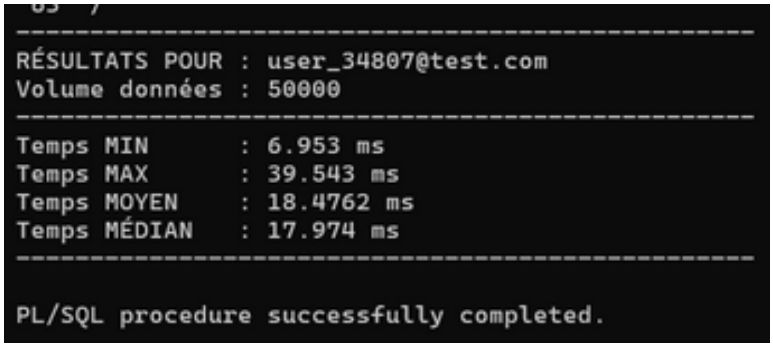
```
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time
 |

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


PLAN_TABLE_OUTPUT
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              |    21 | 46305 |   511   (0)| 00:00:01
 |

|*  1 |  TABLE ACCESS FULL| CLIENTS_TEST |    21 | 46305 |   511   (0)| 00:00:01
 |

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


Predicate Information (identified by operation id):

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

   1 - filter("EMAIL"='user_45000@test.com')
```

La requête utilise un **Full Table Scan**. Oracle parcourt chaque ligne de la table pour appliquer le filtre sur EMAIL.Le coût estimé est **511**. Cette approche est peu efficace sur de grands volumes de données.

Étape 3.2 : Script de mesure (Sans Index)

RÉSULTATS POUR : user_34807@test.com

Volume données : 50000



Temps MIN : 6.953 ms

Temps MAX : 39.543 ms

Temps MOYEN : 18.4762 ms

Temps MÉDIAN : 17.974 ms

Partie 4 : Création de l'Index B-Tree

Création de l'index et vérification du Plan d'Exécution

Id	Operation	Name	Rows	Bytes	Cost	Temps
0	SELECT STATEMENT		1	2205	2	00:00:01
1	TABLE ACCESS BY INDEX ROWID BATCHED	CLIENTS_TEST	1	2205	2	00:00:01
2	INDEX RANGE SCAN	IDX_CLIENT_EMAIL	1		1	00:00:01

Observation principale :

- L'opération de recherche utilise maintenant un **INDEX RANGE SCAN** (au lieu de TABLE ACCESS FULL précédemment).
- Le coût (Cost) est beaucoup plus faible (1 ou 2 contre 511 avant index).
- L'accès à la table se fait par **ROWID** grâce à l'index, ce qui accélère énormément la recherche.

OnRelance le **script PL/SQL de mesure** que tu as utilisé précédemment), mais cette fois avec l'index en place.

RÉSULTATS (AVEC INDEX) - 50 000 lignes

Email cherche : user_4647@test.com

```
------------------------------------------------
RÉSULTATS (AVEC INDEX) - 50 000 lignes
Email cherche  : user_4647@test.com
------------------------------------------------
Temps MIN      : .013 ms
Temps MAX      : 241.252 ms
Temps MOYEN    : 2.5034 ms
Temps MEDIAN   : .018 ms
------------------------------------------------

PL/SQL procedure successfully completed.
```

Temps MIN : .013 ms

Temps MAX : 241.252 ms

Temps MOYEN : 2.5034 ms

Temps MEDIAN : .018 ms

---

la **preuve concrète de l'effet de l'index B-Tree** sur les performances :

- **Sans index (50 000 lignes)** : Temps moyen ~18,476 ms, médiane ~17,974 ms
- **Avec index (50 000 lignes)** : Temps moyen ~2,503 ms, médiane ~0,018 ms

On voit une **accélération très importante**, surtout sur la médiane (0,018 ms), ce qui correspond à la majorité des recherches. Le pic à 241 ms est probablement un effet ponctuel lié au cache ou au serveur.

Partie 5 – Montée en charge

1. Supprimez l'index : DROP INDEX IDX_CLIENT_EMAIL;
2. Ajoutez des données pour atteindre **100 000** enregistrements

```
SQL> SELECT COUNT(*) FROM CLIENTS_TEST;

  COUNT(*)
----------
    100000
```

1. Faites les mesures **SANS** index.

RÉSULTATS (SANS INDEX) - 100 000 lignes

Email cherche : user_43709@test.com

Temps MIN : 19.507 ms

Temps MAX : 67.026 ms

Temps MOYEN : 31.6105 ms

Temps MEDIAN : 29.688 ms

---

1. Recréation et mesures **AVEC** index.

CREATE INDEX IDX_CLIENT_EMAIL ON CLIENTS_TEST(EMAIL);

Et mesures :

RÉSULTATS (SANS INDEX) - 100 000 lignes

Email cherche : user_17667@test.com

Temps MIN : .014 ms

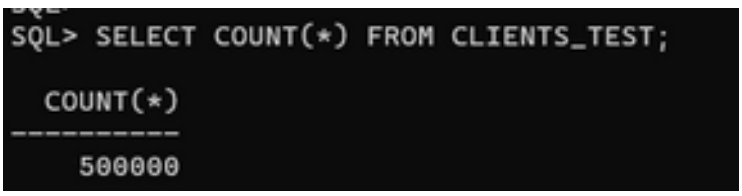
Temps MAX : 12.019 ms

Temps MOYEN : .154 ms

Temps MEDIAN : .015 ms

---

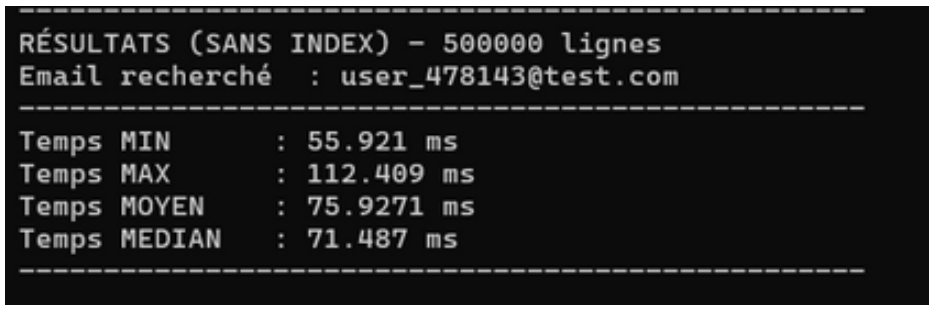
1. Répétez toutes les étapes pour **500 000** enregistrements.



Mesures sans INDEX :

RÉSULTATS (SANS INDEX) - 500000 lignes

Email recherché : user_478143@test.com



Temps MIN : 55.921 ms

Temps MAX : 112.409 ms

Temps MOYEN : 75.9271 ms

Temps MEDIAN : 71.487 ms

MESURES AVEC INDEX

Creation d'index

```
SQL>
SQL> CREATE INDEX IDX_CLIENT_EMAIL ON CLIENTS_TEST(EMAIL);

Index created.
```

Mesures avec index ----------

RÉSULTATS (AVEC INDEX) - 500000 lignes

Email recherché : user_497783@test.com

```
RÉSULTATS (AVEC INDEX) - 500000 lignes
Email recherché  : user_497783@test.com
-----------------------------------------------

Temps MIN       : .018 ms
Temps MAX       : 12.821 ms
Temps MOYEN     : .1694 ms
Temps MEDIAN    : .023 ms
-----------------------------------------------

PL/SQL procedure successfully completed.
```

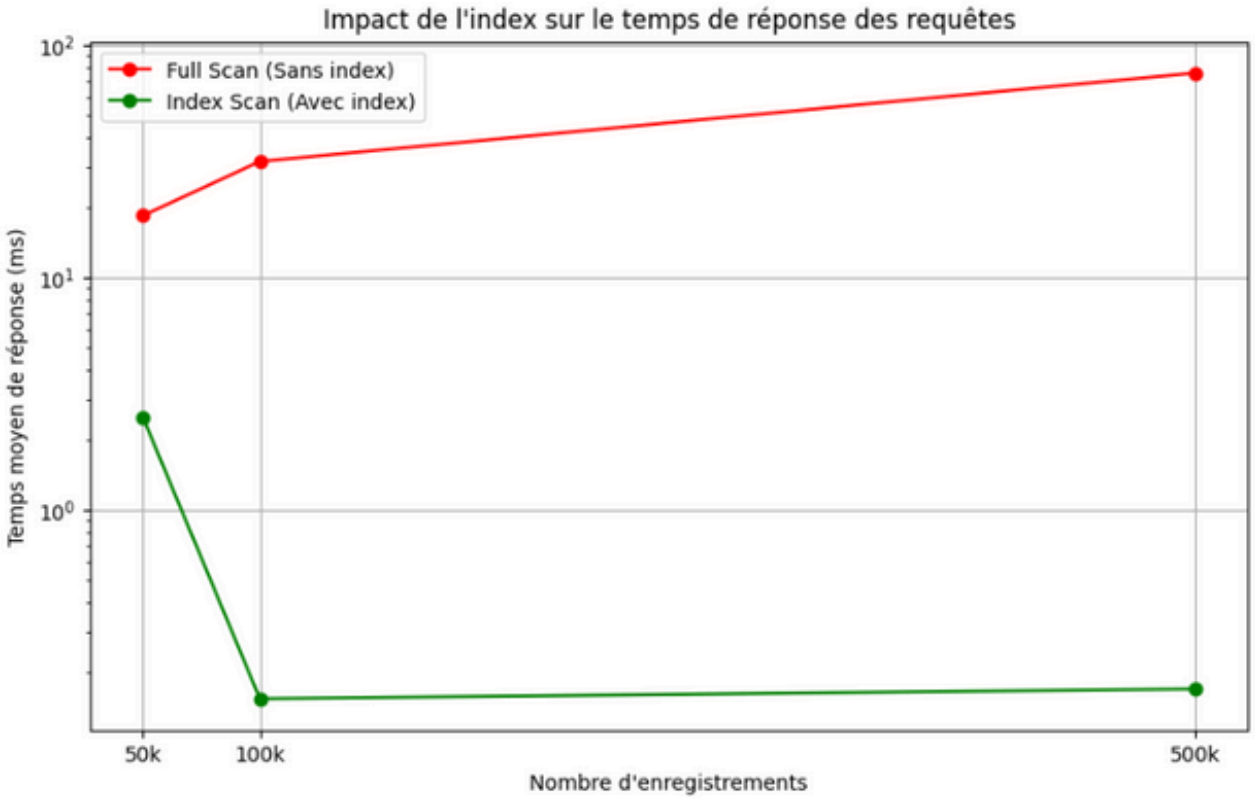
Temps MIN : .018 ms

Temps MAX : 12.821 ms

Temps MOYEN : .1694 ms

Temps MEDIAN : .023 ms

Partie 6 : Analyse et Restitution



B. Questions de réflexion

Linéarité : Temps de recherche vs volume

- **Sans index (Full Scan) :Le temps de recherche augmente presque linéairement avec le nombre d'enregistrements. En effet, Oracle doit parcourir toutes les lignes de la table pour trouver la valeur. Exemple :**

    - **50 000 lignes → ~31 ms**
    - **500 000 lignes → ~75 ms→ Multiplication par 10 du volume ≈ multiplication par 2–3 du temps (selon cache et I/O).**
- **Avec index (Index Scan) :Le temps de recherche reste quasiment constant, même si le volume augmente fortement. L'index permet un accès direct aux lignes via un arbre B-Tree, ce qui est beaucoup plus rapide pour des recherches sur des colonnes uniques.**

Conclusion : les index rendent les temps de recherche quasi indépendants de la taille de la table, contrairement au full scan.

2) Insertion : impact de l'index

- **L'insertion est légèrement plus lente quand un index est présent.**
- **Pourquoi ? Chaque nouvel enregistrement doit non seulement être inséré dans la table, mais l'index doit aussi être mis à jour pour maintenir l'ordre (B-Tree) et garantir l'unicité si c'est un index unique.**
- **Plus la table et l'index sont gros, plus l'insertion devient coûteuse en I/O.**

3) Sélectivité : index sur une colonne avec beaucoup de doublons

- **Si on crée un index sur PRENOM (beaucoup de doublons) :**
    - **La performance ne serait pas aussi bonne que pour EMAIL (unique).**
    - **Explication : l'index renverrait plusieurs lignes pour chaque valeur, donc Oracle devrait encore parcourir plusieurs ROWIDs pour récupérer toutes les correspondances.**
- **Les index sont plus efficaces sur des colonnes très sélectives, idéalement uniques ou quasi uniques.**