Fachhochschule Nordwestschweiz
Hochschule für Technik

dbarc Lab 5:
Indexierung mit B-Tree Indexes

In diesem Lab geht es um die Erstellung und Verwendung von Indexes in der Oracle-Datenbank. Ein Index kann die Performance einer Abfrage beschleunigen, aber nicht jede Abfrage ist für die Verwendung eines Indexes geeignet

Die nachfolgenden Übungen bearbeitet Ihr am besten in Zweierteams, damit Ihr Eure Überlegungen austauschen und miteinander diskutieren könnt.

1. Index oder Full Table Scan?

Vorbereitung

Erstellt auf der Tabelle ADDRESSES im Demo-Schema (dbarc*n*) folgende Non-Unique B-Tree Indexes:

```
CREATE INDEX ADR_CUST_ID ON ADDRESSES(CUST_ID);
CREATE INDEX ADR_CTR_CODE ON ADDRESSES(CTR_CODE);
CREATE INDEX ADR_CITY ON ADDRESSES(CITY);
CREATE INDEX ADR_ZIP_CITY ON ADDRESSES(ZIP_CODE, CITY);
```

Überprüft, welche Indexes auf der Tabelle ADDRESSES vorhanden sind. Dies kann auf verschiedene Arten ermittelt werden:

- Mit der grafischen Oberfläche von SQL Developer (bzw. DBeaver oder Data Grid)
- Mit folgendem Befehl in SQL Developer:

  ```
  info ADDRESSES
  ```

- Mit eine SQL-Abfrage auf die Data Dictionary Views:

  ```
  SELECT i.index_name, i.index_type, i.uniqueness,
         LISTAGG(c.column_name, ', ') WITHIN GROUP (ORDER BY c.column_position)
  column_list
    FROM user_indexes i
    JOIN user_ind_columns c ON (c.index_name = i.index_name)
   WHERE i.table_name = 'ADDRESSES'
  GROUP BY i.index_name, i.index_type, i.uniqueness;
  ```

Abfragen auf ADDRESSES

Das Script index_usage.sql enthält 15 Abfragen auf die Tabelle ADDRESSES. Überlegt Euch für jede der Abfragen die korrekten Antworten zu folgenden Fragen:

- Kann für die Abfrage ein Index verwendet werden, abhängig von der WHERE-Bedingung?
- Falls ja, ist es auch sinnvoll, den Index zu benutzen?
- Falls nein, warum ist ein Full Table Scan die bessere Wahl?
- Kommt der Optimizer zur gleichen Entscheidung? Wenn nicht, was könnte der Grund sein?

Um Eure Antworten zu überprüfen, führt die Aufrufe EXPLAIN PLAN und dbms_xplan.display aus, die im Script bereitgestellt werden.

2. Wann ist ein Index sinnvoll?

Versucht, die Performance für die folgenden Abfragen durch die Erstellung geeigneter Indizes zu verbessern. Alle Abfragen sind bereits schnell, aber mit einem geeigneten Index können die Kosten für den Execution Plan reduziert werden. Ist dies für alle Abfragen möglich?

Die SQL-Abfragen stehen im Script **index_creation.sql** zur Verfügung.

```
SELECT * FROM customers
 WHERE date_of_birth = DATE'1990-07-10';

SELECT * FROM customers
 WHERE last_name = 'Zweig' AND first_name = 'Sandra';

SELECT * FROM customers
 WHERE marital_status = 'married';

SELECT * FROM customers
 WHERE language_code = 'de';

SELECT * FROM customers
 WHERE language_code = 'nl';
```

3. Struktur von B-Tree Indexes

Ein B-Tree Index besteht aus „Branch Blocks“ und „Leaf Blocks“. Um zu den Leaf Blocks zu gelangen, müssen ein oder mehrere Branch Blocks durchlaufen werden. In den Index-Statistiken von Oracle stehen dazu folgende Informationen zur Verfügung:

- BLEVEL: Anzahl Branch Blocks, die gelesen werden müssen, um zu einem Leaf Block zu gelangen („Höhe des B-Baums“)
- LEAF_BLOCKS: Anzahl der Leaf Blocks auf der untersten Hierarchiestufe des Index („Breite des B-Baums“)

Überprüft mit Hilfe folgender Abfrage auf den Data Dictionary, wie gross die Indexes in Eurem Demo-Schema sind. Welche Faktoren beeinflussen die Grösse eine Index?

```
SELECT table_name, index_name, blevel, leaf_blocks, distinct_keys
  FROM user_indexes
ORDER BY table_name, index_name;
```

Indexes werden bei jedem DML-Befehl (INSERT, UPDATE, DELETE) automatisch nachgeführt. Versucht zu verstehen, wie das Einfügen und Löschen eines Indexeintrags in einem B-Tree Index funktioniert. Dazu könnt Ihr folgende Quellen verwenden:

- **Wikipedia-Eintrag zu „B-Baum“**: https://de.wikipedia.org/wiki/B-Baum
- **Lernvideo auf studyflix.de**: https://studyflix.de/informatik/b-baum-1435

Bei der von Oracle (und auch anderen Datenbankherstellern) verwendeten Implementierung von Indexes handelt es sich um eine Spezialvariante von B-Bäumen. Um welche Variante es sich dabei handelt, wird in beiden Quellen erwähnt.

4. Funktionsaufrufe in WHERE-Bedingungen

Eine häufige Ursache für Performanceprobleme in SQL-Abfragen sind Funktionsaufrufe in der WHERE-Bedingung, wie beispielsweise diese Query:

```
SELECT * FROM customers
 WHERE UPPER(last_name) = 'GRANGER';
```

In Oracle-Datenbanken gibt es verschiedene Möglichkeiten, um diese zu lösen. Weitere Informationen sind dazu sind hier beschrieben:

https://danischnider.wordpress.com/2022/02/28/performance-tips-function-calls-in-where-conditions/

Im Hinblick auf das nächste Lab lohnt es sich, den Blog Post zu lesen.