Бази даних та інформаційні системи

Тема 16. Послідовності. Індекси. Синоніми

СумДУ, каф. КН

2020

Задачи занятия

Після завершення заняття ви повинні вміти і знати наступне:

- Властивості і використання об'єктів бази даних;
- Створювати, підтримувати і використовувати послідовності;
- Створювати і підтримувати індекси;
- Створювати приватні і загальнодоступні синоніми.

Объекты в БД

Об'єкт	Опис
Table	Елемент зберігання; складається з рядків і стовпців
View	Логічно представляє (подає) підмножину
Sequence	**Генерує значення первинних ключів**
Index	**Збільшує швидкість роботи**
Synonym	**Альтернативне ім'я об'єкта**

Що таке послідовність?

- Автоматично генерує унікальні числа;
- Є об'єктом із загальним доступом;
- Зазвичай використовується для формування значень первинних ключів;
- Замінює додатковий код;
- Збільшує ефективність доступу до елементів послідовності при кешуванні послідовності в пам'яті.

Створення послідовності

`CREATE SEQUENCE` – Визначає послідовність, автоматично генерує послідовні числа

```
CREATE SEQUENCE sequence
    [INCREMENT BY n]
    [START WITH n]
    [{MAXVALUE n | NOMAXVALUE}]
    [{MINVALUE n | NOMINVALUE}]
    [{CYCLE | NOCYCLE}]
    [{CACHE n | NOCACHE}];
```

Створення послідовності

Приклад створення послідовності **seq_dept_deptno** від 91 до 100 з кроком 1, яка буде використовуватися для формування первинних ключів таблиці **DEPT**.
Параметр **CYCLE** не використовується.

```
SQL> CREATE SEQUENCE seq_dept_deptno
2    INCREMENT BY 1
3    START WITH 91
4    MAXVALUE 100
5    NOCACHE
6    NOCYCLE;

Sequence created
```

Псевдостовпці NEXTVAL и CURRVAL

- **`NEXTVAL`** повертає наступне число послідовності. Повертає унікальне значення при кожному виклику, навіть якщо викликають різні користувачі.
- **`CURRVAL`** зберігає поточне число послідовності.

`NEXTVAL` *повинен бути викликаний хоча б раз, щоб в* ***`CURRVAL`*** *зберігалося значення..*

Використання послідовностей

- Додавання нового підрозділу “MARKETING” у Сан-Дієго.

```
SQL> INSERT INTO dept(deptno, dname, loc)
   2 VALUES (seq_dept_deptno.NEXTVAL,
   3 'MARKETING', 'SAN-DIEGO');

1 row created.
```

- Перегляд поточного значення послідовності **seq_dept_deptno:**

```
SQL> SELECT seq_dept_deptno.CURRVAL
   2 FROM dual;
```

Використання послідовностей

Кешування значень послідовності в пам'яті дозволяє збільшити швидкість доступу до цих значень.

Пропуски в послідовності виникають при:

- відкатах транзакцій;
- системних збоях;
- Використанні послідовності в декількох таблицях.

Якщо послідовність була створена з параметром **`NOCACHE`**, її наступне значення можна побачити, сформувавши запит до таблиці **`USER_SEQUENCES`**.

Зміна послідовності

Зміна значення інкремента, максимального і мінімального значень, параметрів циклічності і кешування.

```
SQL> ALTER SEQUENCE seq_dept_deptno
  2  INCREMENT BY 1
  3  MAXVALUE 999999
  4  NOCACHE
  5  NOCYCLE;

Sequence altered
```

Зміна послідовності

- Ви повинні бути власником або мати право **`ALTER`** для даної послідовності.
- Зміни вступають у силу тільки для майбутніх значень послідовності.
- Для зміни початкового числа послідовності її слід знищити і створити знову.

Видалення послідовності

- Для видалення послідовності з каталогу даних використовуйте вираз **DROP SEQUENCE**.
- До віддаленої послідовності звертатися не можна.

```
SQL> DROP SEQUENCE seq_dept_deptno;

Sequence dropped
```

Які послідовності є в БД?

- Ви можете перевірити параметри послідовності в таблиці **USER_SEQUENCES** каталогу даних:

```
SQL> SELECT sequence_name, min_value, max_value,
2    increment_by, last_number
3    FROM USER_SEQUENCES;
```

- Стовпець **LAST_NUMBER** показує наступне число в послідовності.

Що таке індекс?

- об'єкт схеми;
- Використовується Oracle Server для збільшення швидкості отримання значень рядків шляхом використання покажчика;
- Знижує кількість звернень до жорстких дисків за рахунок швидкого знаходження даних;
- Не залежить від таблиці, для якої створений. Вони можуть бути створені або видалені в будь-який час і не впливають на базові таблиці або інші індекси;
- Автоматично використовується і підтримується Oracle Server.

Як створюються індекси?

- **Автоматично** - індекс створюється автоматично, якщо ви вказуєте обмеження **`PRIMARY KEY`** або **`UNIQUE`** для таблиці.
- **Вручну** - користувач може створювати індекси для збільшення швидкості доступу до даних. Наприклад, ви можете створити індекс стовпчика "FOREIGN KEY" для об'єднання в запиті, щоб збільшити швидкість пошуку.

Створення індексу

– Синтаксис:

```
CREATE INDEX index
ON table (column[, column]...);
```

– Збільшення швидкості доступу до стовпця ENAME таблиці EMP:

```
SQL> CREATE INDEX emp_ename_idx
  2  ON emp(ename);
Index created.
```

– Створення індексу для одного або більше стовпців:

```
SQL> CREATE INDEX emp_ename_idx2
  2  ON emp(ename,depno);
Index created.
```

Коли створювати індекси?

- Стовпець часто використовується у фразі `WHERE` або умовах з'єднання;
- Стовпець містить широкий спектр значень;
- У стовпці велика кількість значень `null`;
- Два або більше стовпців часто використовуються спільно у фразі `WHERE` або умовах з'єднання;
- Розміри таблиці великі і більшість запитів імовірно буде повертати менше 2-4% рядків.

Поради щодо створення індексів

Не створюйте індекси, якщо:

- Таблиця невелика;
- Стовпці рідко використовуються в умовах запитів;
- Більшість запитів імовірно будуть повертати більше 2-4% рядків;
- Таблиця часто змінюється.

Видалення індексів

– Видалення індексу з каталогу даних:

```
SQL> DROP INDEX index;
```

– Видалення індексу `EMP_ENAME_IDX` з каталогу даних:

```
SQL> DROP INDEX emp_ename_idx;
Index dropped.
```

– Для знищення індексу, ви повинні бути його власником або мати дозвіл **`DROP ANY INDEX`**.

Які індекси є в БД?

- Представлення **USER_INDEXES** каталогу даних містить імена індексів і їх унікальність.
- Представлення **USER_IND_COLUMNS** містить ім'я індексу, ім'я таблиці та ім'я стовпця.

```
SQL> SELECT   ic.index_name, ic.column_name,
  2           ic.column_position col_pos,ix.uniqueness
  3  FROM     user_indexes ix, user_ind_columns ic
  4  WHERE    ic.index_name = ix.index_name
  5  AND      ic.table_name = 'EMP';
```

Типи індексів

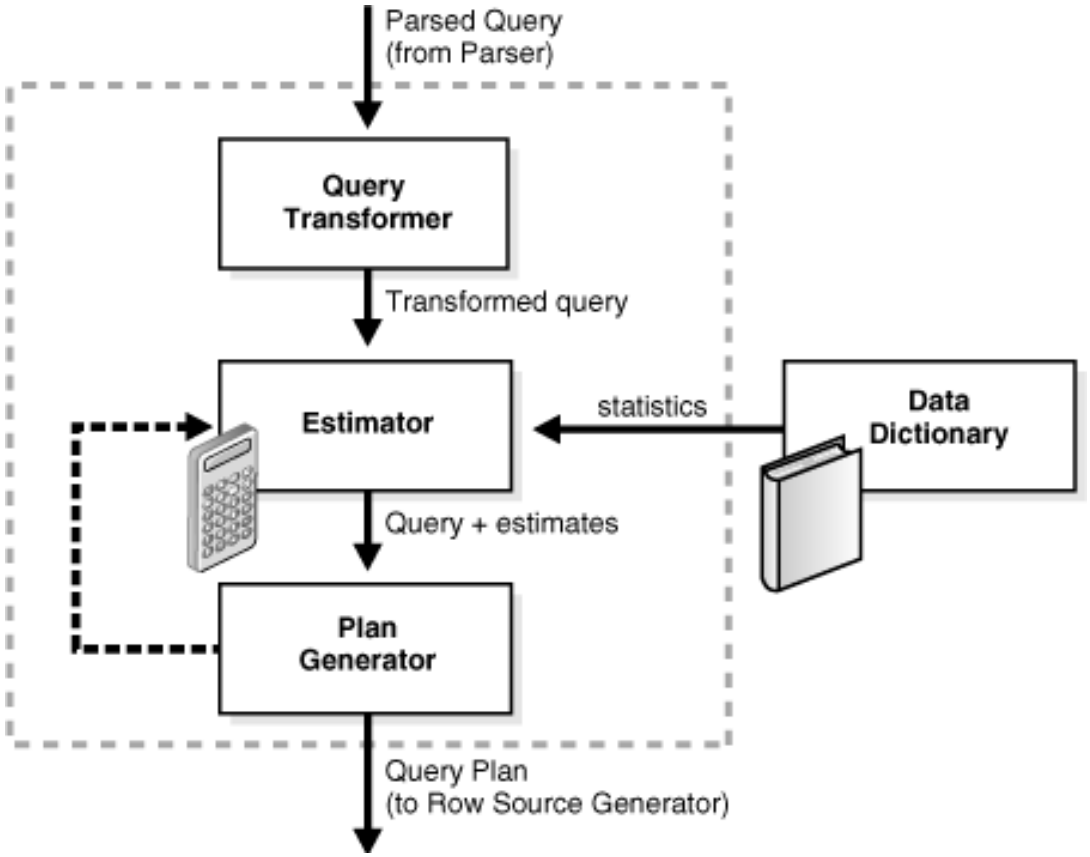
- B-tree indexes - створюється за замовчуванням.
- B-tree cluster indexes - посилання не на рядок, а блок, де зберігається рядок.
- Hash cluster indexes - як ключ використовується не саме значення, а його хеш.
- Bitmap indexes - бітовий вектор присутності ознак (добре працює коли діапазон значень малий).
- Function-based indexes - містить попередньо обчислені значення функцій для рядків таблиці.
- Domain indexes - призначена для користувача реалізація індексу.

Додаткова інформація

- Огляд типів індексів Oracle, MySQL, PostgreSQL, MS_SQL
  - http://habrahabr.ru/post/102785

- Документація Oracle за індексами:
  - http://docs.oracle.com/cd/E11882_01/server.112/e10713/indexiot.htm#BABHJAJF

План виконання запиту




```
set autotrace [on | off |traceonly] [explain] [statistic]
```

План виконання запиту

```
SQL> set autotrace traceonly statistics
SQL> SELECT * from emp where empno > 7500;
```

```
15 rows selected.

Statistics
----------------------------------------------------------
   0  recursive calls
   0  db block gets
   3  consistent gets
   0  physical reads
   0  redo size
1437  bytes sent via SQL*Net to client
 419  bytes received via SQL*Net from client
   2  SQL*Net roundtrips to/from client
   0  sorts (memory)
   0  sorts (disk)
  15  rows processed
```

План виконання запиту

```
SQL> set autotrace on explain
SQL> select ename from emp where empno < 7500;
```

```
Execution Plan
----------------------------------------------------------
Plan hash value: 3956160932

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |    13 |   507 |     2   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| EMP  |    13 |   507 |     2   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
1 - filter("EMPNO">7500)
```

Оптимізатор

- **`Full table scans`** - з БД прочитані всі рядки, для кожної перевірені умови
- **`Rowid scans`** - БД знає, де знаходяться рядки-кандидати і вибирає тільки частину рядків по номерах, після чого перевіряє їх.
- **`Index scans`** - поиск нужных срок ведется по индексу. В некотрых случаев данные из таблицы даже не читаются.
- **`Cluster scans`** - читается не вся таблица, а только ее часть (блок). Номер блока выясняется по номеру строки.
- **`Hash scans`** - читается не вся таблица, а только ее часть. Номер части узнается из хеш-функции ключа.

Підказки

```
SELECT /*+ full(t) */ t.name
FROM tbl1 t WHERE t.DATE = SYSDATE;

SELECT /*+ index(t ind_date) */ t.name
 FROM tbl1 t WHERE t.DATE = SYSDATE;
```

- http://iusoltsev.wordpress.com/profile/individual-sql-and-cbo/cbo-hints/

Синонім

Ви можете спростити доступ до об'єктів, створивши синонім (додаткове ім'я об'єкта):

- Для доступу до таблиць інших користувачів.
- Для скорочення довгих імен.

```
CREATE [PUBLIC] SYNONYM synonym FOR object;
```

Створення та видалення синонімів

Створення скороченого імені для представлення DEPT_SUM_VU:

```
SQL> CREATE SYNONYM d_sum
    2 FOR dept_sum_vu;

Synonym Created.
```

Видалення синоніму:

```
SQL> CREATE SYNONYM d_sum
    2 FOR dept_sum_vu;

Synonym Created.
```