Etap 5B

Grupa B6 – Loty

 Hanna Grodzicka

 Mateusz Najda

Rozszerzenia

3 tabele zamieniono z wierszowych na kolumnowe:

1. `Airline_col`

   Używana jest przez operacje: 2, 4, 5.

```
create table AIRLINE_COL (
  ID                number(10) not null,
  COMPANY           varchar2(255) not null,
  CODE              varchar2(255) not null unique,
  COUNTRY           varchar2(255) not null,
  ESTABLISHMENT_DATE date,
  primary key (ID)
) INMEMORY;
```

2. `Flight_col`

   Używana jest przez operacje: 1, 2, 4.

```
create table FLIGHT_COL (
  ID                  number(10) not null,
  FLIGHT_COL_NUMBER       varchar2(255) not null,
  DESTINATION_COUNTRY varchar2(255) not null,
  DESTINATION_CITY    varchar2(255) not null,
  DEPARTURE           date not null,
  ARRIVAL             date not null,
  PASSENGERS          number(10) not null,
  STATUS              varchar2(255) not null,
  DELAY               float(10),
  IS_NATIONAL         number(1) not null,
  PLANE_COL_ID            number(10) not null,
  AIRPORT_COL_ID          number(10) not null,
```

```
  primary key (ID)
) INMEMORY;
```

3. `Plane_col`

   Używana jest przez operacje: 2, 3, 4.

```
create table PLANE_COL (
  ID              number(10) not null,
  MANUFACTURER    varchar2(255) not null,
  MODEL           varchar2(255) not null,
  PRODUCTION_DATE date not null,
  SEAT_COUNT      number(10) not null,
  FUEL_CAPACITY   float(10) not null,
  CRUISING_RANGE  number(10) not null,
  AIRLINE_COL_ID      number(10) not null,
  primary key (ID)
) INMEMORY;
```

Operacje

1. **SELECT** Średni wiek pracowników

   Opis: Średni wiek wszystkich pracowników przypisanych do wszystkich lotów z opóźnieniem pomiędzy 5.0 a 10.0. Oczekiwany wynik: `average employee age = 49.7688022`

   Operacja SQL:

```
SELECT AVG(TRUNC((sysdate - e.birthdate) / 365)) AS "average employee age"
FROM employee_col e
  JOIN flight_employee_col fe ON e.id = fe.employee_col_id
  JOIN flight_col f ON fe.flight_col_id = f.id
WHERE f.delay > 5.0 AND f.delay < 10.0;
```

2. **SELECT** Firmy i długości odbytych w nich lotów

   Opis: Zestawienie wszystkich nazw firm linii lotniczych z sumą długości lotów (w dniach) odbytych w ciągu dnia najbliższego do teraźniejszej daty. Oczekiwany wynik: `company = Lectus Nullam Associates`

   Operacja SQL:

```
SELECT a.company, SUM(TO_NUMBER(f.arrival - f.departure))
FROM airline_col a
  JOIN plane_col p ON a.id = p.airline_col_id
  JOIN flight_col f ON p.id = f.plane_col_id
WHERE f.departure = (SELECT MAX(departure) FROM flight_col)
GROUP BY a.company;
```

▼ **CREATE**

Czas: 0.085 s

`4005 rows inserted.`

```
DROP SEQUENCE plane_id_seq;
CREATE SEQUENCE plane_id_seq START WITH 4001;
INSERT INTO plane (id,
                   manufacturer,
                   model,
                   production_date,
                   seat_count,
                   fuel_capacity,
                   cruising_range,
                   airline_id)
SELECT plane_id_seq.NEXTVAL, a, b, c, d, e, f, g
FROM (SELECT p.manufacturer a,
             p.model b,
             p.production_date c,
             p.seat_count d
             p.fuel_capacity e,
             p.cruising_range f,
             p.airline_id g
      FROM plane p
      ORDER BY TRUNC(p.fuel_capacity) / p.cruising_range);
```

Czas: 0.138 s

3. **CREATE** Dodanie wydajnych samolotów

   Opis: Znalezienie dla danej linii lotniczej samolotów, które mają najbardziej korzystny stosunek pojemności baku do zasięgu lotu i dodanie po jednym samolocie o tych samych atrybutach dla każdego z nich. Oczekiwany wynik: `4005 rows inserted.`

   Operacja SQL:

```
INSERT INTO plane_col (id,
                       manufacturer,
                       model,
                       production_date,
                       seat_count,
                       fuel_capacity,
                       cruising_range,
                       airline_col_id)
SELECT plane_id_seq.NEXTVAL, a, b, c, d, e, f, g
FROM (SELECT p.manufacturer a,
             p.model b,
             p.production_date c,
             p.seat_count d,
             p.fuel_capacity e,
             p.cruising_range f,
             p.airline_col_id g
      FROM plane_col p
      ORDER BY TRUNC(p.fuel_capacity) / p.cruising_range);
```

4. **UPDATE** Nowe daty stworzenia linii lotniczych

Opis: Uaktualnienie wszystkich dat stworzenia linii lotniczych wpisując datę wylotu pierwszego wykonanego dla nich lotu. Oczekiwany wynik: `3,000 rows updated`.

Operacja SQL:

```
UPDATE airline_col
SET establishment_date = (SELECT MIN(f.departure)
                          FROM flight_col f
                          JOIN plane_col p ON p.id = f.plane_col_id
                          JOIN airline_col a ON a.id = p.airline_col_id);
```

5. **DELETE** Usunięcie niektórych pracowników

   Opis: Usunięcie tych pracowników, którzy zostali zatrudnieni przez linię lotniczą przed jej utworzeniem. Oczekiwany wynik: `5,320 rows deleted.`

   Operacja SQL:

```
DELETE (SELECT *
        FROM employee_col e
          INNER JOIN airline_col a ON a.id = e.airline_col_id
        WHERE e.employment_date < a.establishment_date);
```

Pomiary

Wszystkie wyniki podane są w sekundach.

Operacja 1 (SELECT)

Aa Pomiar	# Wierszowe	# Kolumnowe
1	0.05	0.1
2	0.06	0.08
3	0.06	0.05
4	0.06	0.06
5	0.07	0.05

$$\sigma_{wierszowe} = 0.0063245553203368, \quad \sigma^2_{wierszowe} = 4.0E-5$$

$$\sigma_{kolumnowe} = 0.019390719429665, \quad \sigma^2_{kolumnowe} = 0.000376$$

Operacja 2 (SELECT)

Aa Pomiar	# Wierszowe	# Kolumnowe
1	0.09	0.08
2	0.07	0.11

Pomiar	Wierszowe	Kolumnowe
3	0.08	0.1
4	0.07	0.07
5	0.08	0.07

$$\sigma_{wierszowe} = 0.0074833147735479, \quad \sigma^2_{wierszowe} = 5.6E-5$$

$$\sigma_{kolumnowe} = 0.016248076809272, \quad \sigma^2_{kolumnowe} = 0.000264$$

Operacja 3 (INSERT)

Pomiar	Wierszowe	Kolumnowe
1	0.15	0.19
2	0.16	0.18
3	0.19	0.19
4	0.14	0.15
5	0.17	0.17

$$\sigma_{wierszowe} = 0.017204650534085, \quad \sigma^2_{wierszowe} = 0.000296$$

$$\sigma_{kolumnowe} = 0.014966629547096, \quad \sigma^2_{kolumnowe} = 0.000224$$

Operacja 4 (UPDATE)

Pomiar	Wierszowe	Kolumnowe
1	0.06	0.07
2	0.05	0.1
3	0.06	0.06
4	0.06	0.07
5	0.05	0.07

$$\sigma_{wierszowe} = 0.0048989794855664, \quad \sigma^2_{wierszowe} = 2.4E-5$$

$$\sigma_{kolumnowe} = 0.013564659966251, \quad \sigma^2_{kolumnowe} = 0.000184$$

Operacja 5 (DELETE)

Pomiar	Wierszowe	Kolumnowe
1	15.46	16.87

Aa Pomiar	# Wierszowe	# Kolumnowe
2	15.19	15.68
3	16.85	16.28
4	17.04	15.64
5	16.69	15.78

$$\sigma_{wierszowe} = 0.76489476400352, \quad \sigma^2_{wierszowe} = 0.585064$$

$$\sigma_{kolumnowe} = 0.46972332281887, \quad \sigma^2_{kolumnowe} = 0.22064$$

Wnioski

Z wyjątkiem operacji 5, wszystkie średnie **czasy były wyższe** (w sposób znaczący) w przypadku składowania kolumnowego.

Operacja 5 jest najdłuższym zapytaniem modyfikującym – usuwa pracowników korzystając z tabeli `Airline_col`, która była przedmiotem rozszerzeń. Sugeruje to, że w przypadku tej tabeli zmiana składowania na kolumnowe mogło przynieść pozytywny skutek – z pewnością nie stało się tak dla tabel `Plane_col` i `Flight_col`.

Eksperymenty

1. Porównanie dla różnych opcji składowania kolumny

Sprawdzenie różnicy w czasie wykonania wybranych operacji dla opcji składowania kolumn w pamięci:

- wyłączonej (składowanie wierszowe)
- włączonej dla jednej kolumny z wybranej tabeli
- włączonej dla całej wybranej tabeli

Dla operacji 1 wybrano tabelę `Employee_col` i kolumnę `name`.

Dla operacji 3 wybrano tabelę `Plane_col` i kolumnę `model`.

```
alter table employee_col nomemory;
alter table employee_col inmemory;
alter table employee_col inmemory no inmemory
(SURNAME, BIRTHDATE, GENDER, OCCUPATION, EMPLOYMENT_DATE, FLOWN_HOURS, AIRLINE_COL_ID);

alter table plane_col nomemory;
alter table plane_col inmemory;
alter table plane_col inmemory no inmemory
(ID, MANUFACTURER, PRODUCTION_DATE, SEAT_COUNT, FUEL_CAPACITY, CRUISING_RANGE, AIRLINE_COL_ID);
```

Operacja 1 (SELECT)

Aa Pomiar	# Wyłączona	# Kolumna Name	# Tabela Employee
1	0.05	0.05	0.05
2	0.06	0.05	0.06
3	0.06	0.06	0.05
4	0.06	0.05	0.05
5	0.07	0.05	0.06

Operacja 3 (INSERT)

Aa Pomiar	# Wyłączona	# Kolumna Model	# Tabela Plane
1	0.15	0.18	0.16
2	0.16	0.14	0.15
3	0.19	0.16	0.14
4	0.14	0.19	0.13
5	0.17	0.2	0.18

Wnioski

Dla operacji niemodyfikującej (select) czasy wykonania zapytania się nieznacznie poprawiły po zastosowaniu składowania kolumnowego.

W przypadku operacji modyfikującej (insert) czas wykonania był nieco wyższy dla składowania jednej kolumny i nieco niższy dla składowania całej tabeli.

---

2. Porównanie algorytmów kompresji

Sprawdzenie czasu wykonania (i/lub wielkości skompresowanej tabeli `Plane_col`) **dla trzeciej operacji (INSERT)** przy wykorzystaniu kilku różnych algorytmów kompresji danych:

- Query Low (domyślna kompresja)

```
ALTER TABLE PLANE_COL INMEMORY MEMCOMPRESS FOR QUERY LOW;
```

- Query High

```
ALTER TABLE PLANE_COL INMEMORY MEMCOMPRESS FOR QUERY HIGH;
```

- Archive Low

```
ALTER TABLE PLANE_COL INMEMORY MEMCOMPRESS FOR CAPACITY LOW;
```

- Archive High

```
ALTER TABLE PLANE_COL INMEMORY MEMCOMPRESS FOR CAPACITY HIGH;
```

Porównanie czasów (w sekundach)

Aa Pomiar	# Query Low	# Query high	# Capacity Low	# Capacity High
1	0.191	0.177	0.18	0.183
2	0.139	0.143	0.164	0.173
3	0.133	0.149	0.141	0.19
4	0.134	0.171	0.146	0.143
5	0.185	0.172	0.133	0.145

W tabeli poniżej podano wielkości po wykonaniu następujących zapytań:

```
select bytes, blocks, extents
from user_segments
where segment_type='TABLE' and segment_name='PLANE_COL';

select b.tablespace_name, tbs_size SizeMb, a.free_space FreeMb
from  (select tablespace_name, round(sum(bytes)/1024/1024 ,2) as free_space
       from dba_free_space
       group by tablespace_name) a,
      (select tablespace_name, sum(bytes)/1024/1024 as tbs_size
       from dba_data_files
       group by tablespace_name) b
where a.tablespace_name=b.tablespace_name;
```

Aby odzyskać wolną przestrzeń w tabeli stosowano:

```
ALTER TABLE plane_col ENABLE ROW MOVEMENT;
ALTER TABLE plane_col SHRINK SPACE;
```

Porównanie rozmiarów dla tabeli Plane_col

Aa Kolumna	# Initial (before query)	# Query Low	# Query High	# Capacity Low	# Capacity High
Bytes [B]	786,432	2,097,152	2,097,152	2,097,152	2,097,152
Blocks	96	256	256	256	256
Extents	12	17	17	17	17
USERS tablespace					
size [MB]	450	450	450	450	450
free [MB]	60.94	59.5	59.5	59.5	59.5

Pomiar rozmiaru dla tabel dla różnych typów składowania

Tabela	Składowanie wierszowe [B]	Składowanie kolumnowe [B]
Airline_col	262,144	262,144
Flight_col	3,145,728	3,145,728
Plane_col	786,432	786,432

Wnioski

Nie zauważono znaczącej różnicy w czasach wykonania operacji 3 dla różnych typów kompresji.

Przy porównaniu wielkości wolumenu danych w tabeli sprawdzano wiele różnych atrybutów, które miały wskazywać na rozmiar tabeli m.in. jej rozmiar w bajtach, *free space* dla przestrzeni tabeli (ang. *tablespace*) `users`. Jednak niezależnie od wybranego typu kompresji, rozmiar tabeli przed i zwiększeniu był identyczny.

Składowanie kolumnowe również nie wpłynęło na rozmiar tabel w stosunku do wierszowego.

Dla porównania, wybór kompresji w Oracle 11g miał skutkować w mniejszym rozmiarze tabeli:

Type of Compression	Compressed Table Size
Uncompressed	100.00%
Query Low	14.70%
Query High	8.82%
Archive Low	6.62%
Archive High	4.41%

Źródło: https://www.oracle.com/technical-resources/articles/enterprise-manager/11g-compression.html