Module 3

Optmizing Storage Structure

Created by Teerachai Laothong

Agenda

- Avoiding Row Chaining and Migration
- Using LOBs
- Using Partitioning

Avoiding Row Chaining and Migration

- Both Row Chaining and Row Migration hits performance

Example

```
CREATE TABLE HR.BIG_ROWS (
  id number NOT NULL,
  field1 char(2000) DEFAULT 'A' NOT NULL,
  field2 char(2000),
  field3 char(2000),
  field4 char(1000),
  constraint PK_BIG_ROWS primary key (ID))
TABLESPACE EXAMPLE PCTFREE 10;
```

General recommendation is to choose the proper data size and appropriate block size.

Using LOBs

LOBs (Large OBjects) are a particular data type, used to store large binary or character objects

- inside the database or
- outside the database when using `BFILEs`

Example

```
CREATE TABLESPACE ASSM_TS DATAFILE 'ASSM_TS.DBF' SIZE 100M EXTENT MANAGEM
```

```
CREATE TABLE MyCustomers AS SELECT * FROM Customers;
```

```
ALTER TABLE MyCustomers ADD (c_file BLOB)
 LOB(c_file) STORE AS SECUREFILE (
  tablespace ASSM_TS
  enable storage in row
  nocache
  logging
);
```

LOBs

- The data might not actually stored in rows, because when the size of the BLOB field is greater than 4000 bytes it is always stored off-line.
- We can even specify a CHUNK size when defining a BLOB field, the database engine accesses the data in pieces sized accordingly to the CHUNK parameter.
- It's more efficient to read large chunks of data than small.
- The drawback of having a large CHUNK parameter occurs when we want to update the field, because the database engine also writes data in logs.
- Sometimes we use `NOLOGGING` for BLOB fields because it's faster.

LOBs Caching

Caching LOB data in the database can be a nightmare, because this kind of data is usually very large in size, so caching a single field may prove costly in terms of the database buffers used to store the object. To accommodate enough space for a LOB—which ultimately won't be used anymore—there will be many database blocks which were once cached and have now been freed. For this reason, caching for LOB fields is often disabled, as we have done when defining the `c_file` field.

Using Paritioning

Partitioning is the way to improve performance in large tables (Oracle suggests to partition tables with more than 2 GB of data)

Example

```
CREATE TABLE SALES_RP (
 PROD_ID NUMBER NOT NULL,
 CUST_ID NUMBER NOT NULL,
 TIME_ID DATE NOT NULL,
 CHANNEL_ID NUMBER NOT NULL,
 PROMO_ID NUMBER NOT NULL,
 QUANTITY_SOLD NUMBER(10,2) NOT NULL,
 AMOUNT_SOLD NUMBER(10,2) NOT NULL)
PARTITION BY RANGE (TIME_ID)
(
  PARTITION SALES_BEFORE_2000 VALUES LESS THAN (TO_DATE('20000101','YYYYM
  PARTITION SALES_2000_2001_2002 VALUES LESS THAN (TO_DATE('20030101','Y
  PARTITION SALES_2003 VALUES LESS THAN (TO_DATE('20040101','YYYYMMDD'))
);
```

Partitioning

Can use to ease the configuration and care of these objects and to reduce downtime in case of failures or scheduled maintenance such as move the tables or take some data offline.

OLAP

Partitioning is often used in OLAP systems, because it allows us to perform parallel DML operations, such as massive UPDATEs and INSERTs with a degree of parallelism equal to the number of partitions of the table.

OLTP

In OLTP environments, there isn't a great performance gain—and if we implement a bad partitioning scheme we can have worse performance than without—but we will have easier maintenance tasks and increased availability, both as important as performance gains.

Other Types of Partitions

- List partitioning
- Hash partitioning
- Composite partitioning

THE END

- Source code & documentation
- Back to Course Outline