Índice

Bibliografía

- Fundamentals of Database Systems, 6th edition
  R. Elmasri and S. B. Navathe
  Capítulo 20

- Beginning Oracle Database 11g Administration. From Novice to Professional
  I. Fernández. Springer, 2009
  Capítulo 7

- OCA Oracle Database 11g: Administration I Exam Guide (Exam 1Z0-052)
  J. Watson, McGraw-Hill. 2008
  Capítulo 7

4.2.1 Diseño físico de datos

Tarea 3: Afinar el rendimiento de la base de datos

Tarea 4: Mejorar el rendimiento de las consultas

En contraposición con el modelo lógico de datos:

- se pretende maximizar el rendimiento, principalmente minimizando los tiempos de ejecución de consultas.
- es un diseño más dinámico, que evoluciona a lo largo de la vida de la BBDD según varían las necesidades de información.

Elementos de la capa física de datos son:

1. Índices: acceso rápido a datos.
2. Particiones y *Clusters*: organización de los datos.
3. Vistas materializadas: mejora dramáticamente las consultas que involucran la reunión (*join*).

4.2.2 Índices

- Usualmente mejoran las consultas, empeoran actualizaciones
- Mejor no usarlos para recuperar la mayor parte de la tabla indexada

```
select * from empleando where fecha_nacimiento>1900
```

- Tipos:
  - Con valores únicos o con duplicados

```
CREATE UNIQUE INDEX empleado_idx ON empleados(id)
```

  - Cocatenados

```
CREATE INDEX nombre_idx ON empleados (apellido_1,apellido_2)
```

  - Funcionales

```
CREATE INDEX nota_idx ON alumnos (nota_t+nota_p)
```

  - Tablas organizadas por índice (IOT, *Index Organized tables*)

```
SQL> CREATE TABLE my_iot (id INTEGER PRIMARY KEY, value VARCHAR2(50),
  2     comments varchar2(1000))
  3     ORGANIZATION INDEX
  4     INCLUDING value OVERFLOW;
```

- Dónde usarlos
  - Claves principales (se crean por defecto)
  - Claves externas
  - `DBMS_ADVISOR.QUICK_TUNE` (requiere permisos `ADVISOR`)

4.2.3 Particiones

- El particionado de una tabla permite distribuir la tabla entre diferentes segmentos según algún tipo de criterio. Cada segmento puede estar en un espacio de tablas diferente.
  - Mejora el rendimiento de consultas SQL al reducir bloques de datos necesarios
  - Copias de respaldo (*backup*) más rápido
  - Permite planificar la distribuición de la tabla sobre diversos dispositivos físicos
  - Cuando usarlas varía según el servidor, el uso de la tabla, etc. pero generalmente a partir del millón de filas puede ser una buena opción
  - Oracle: disponible solo para la version Enteprise

```
CREATE TABLE sales
( prod_id       NUMBER(6)
, cust_id       NUMBER
, time_id       DATE
, channel_id    CHAR(1)
, promo_id      NUMBER(6)
, quantity_sold NUMBER(3)
, amount_sold   NUMBER(10,2)
)
PARTITION BY RANGE (time_id)
( PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy'))
TABLESPACE tsa
, PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy'))
TABLESPACE tsb
, PARTITION sales_q3_2006 VALUES LESS THAN (TO_DATE('01-OCT-2006','dd-MON-yyyy'))
TABLESPACE tsc
, PARTITION sales_q4_2006 VALUES LESS THAN (TO_DATE('01-JAN-2007','dd-MON-yyyy'))
TABLESPACE tsd );
```

4.2.3 Particiones: tipos (i)

- Particionado por rango
- Particionado Hash

```
CREATE TABLE dept (deptno NUMBER, deptname VARCHAR(32))
PARTITION BY HASH(deptno) PARTITIONS 16;
```

- Particionado List

```
CREATE TABLE sales_list (
         salesman_id NUMBER(5),
         salesman_name VARCHAR2(30),
         sales_state VARCHAR2(20),
         sales_amount NUMBER(10),
         sales_date DATE
)
PARTITION BY LIST(sales_state)
(
 PARTITION sales_west VALUES('California', 'Hawaii'),
 PARTITION sales_east VALUES ('New York', 'Virginia', 'Florida'),
 PARTITION sales_central VALUES('Texas', 'Illinois')
 PARTITION sales_other VALUES(DEFAULT)
);
```

4.2.3 Particiones: tipos (ii)

- Particionado compuesto (*composite*)
  - Range-hash disponible desde Oracle 8i
  - Range-list disponible desde Oracle 9i
  - Range-range disponible desde Oracle 11g
  - List-range disponible desde Oracle 11g
  - List-hash disponible desde Oracle 11g
  - List-list disponible desde Oracle 11g
  - Interval-range disponible desde Oracle 11g
  - Interval-list disponible desde Oracle 11g
  - Interval-hash disponible desde Oracle 11g
  - Hash-hash disponible desde Oracle 11gR2

```
 CREATE TABLE TAB2
(ord_id     NUMBER(10)
,     ord_day    NUMBER(2),
     ord_month  NUMBER(2),
     ord_year   NUMBER(4)
)
PARTITION BY RANGE(ord_year)
SUBPARTITION BY HASH(ord_id)
SUBPARTITIONS 8
( PARTITION q1 VALUES LESS THAN(2001)
    ( SUBPARTITION q1_h1 TABLESPACE TBS1,
      SUBPARTITION q1_h2 TABLESPACE TBS2,
      SUBPARTITION q1_h3 TABLESPACE TBS3,
      SUBPARTITION q1_h4 TABLESPACE TBS4
   ),
    PARTITION q2 VALUES LESS THAN(2002)
     ( SUBPARTITION q2_h5 TABLESPACE TBS5,
       SUBPARTITION q2_h6 TABLESPACE TBS6,
       SUBPARTITION q2_h7 TABLESPACE TBS7,
       SUBPARTITION q2_h8 TABLESPACE TBS8
      ),
    PARTITION q3 VALUES LESS THAN(2003)
     ( SUBPARTITION q3_h1 TABLESPACE TBS1,
       SUBPARTITION q3_h2 TABLESPACE TBS2,
       SUBPARTITION q3_h3 TABLESPACE TBS3,
       SUBPARTITION q3_h4 TABLESPACE TBS4
      ),
    PARTITION q4 VALUES LESS THAN(2004)
     ( SUBPARTITION q4_h5 TABLESPACE TBS5,
       SUBPARTITION q4_h6 TABLESPACE TBS6,
       SUBPARTITION q4_h7 TABLESPACE TBS7,
       SUBPARTITION q4_h8 TABLESPACE TBS8
     )
 )
```

- Particionado manual

```
create table t (
 c1 int,
 c2 varchar2(10),
 c3 date
)
partition by system        (
  partition p1,
  partition p2,
  partition p3
);
insert into t partition (p3) values (1,'A',sysdate);
```

- Particionado por rango *interval*

```
CREATE TABLE T_11G
(
 C1 NUMBER(38,0),
 C2 VARCHAR2(10),
 C3 DATE
)
PARTITION BY RANGE (C3) INTERVAL (NUMTOYMINTERVAL(1,'MONTH'))
(PARTITION P0902 VALUES LESS THAN (TO_DATE('2009-03-01 00:00:00','YYYY-MM-DD
HH24:MI:SS')));
```

- Los *clusters* son grupos de una o más tablas almacenadas físicamente juntas porque comparten columnas comunes y a menudo se utilizan juntas. Dado que las filas relacionadas son almacenadas físicamente juntas, mejora el tiempo de acceso a disco.

Index cluster:

```
CREATE CLUSTER personnel
    ( department_number  NUMBER(2) )
    SIZE 512;
CREATE INDEX idx_personnel ON CLUSTER personnel;
```

Hash cluster:

```
CREATE CLUSTER personnel
( department_number  NUMBER )
    SIZE 512  HASHKEYS 500 HASH IS departmen_number
    STORAGE (INITIAL 100K  NEXT 50K);
```

Tablas que conforman el cluster:

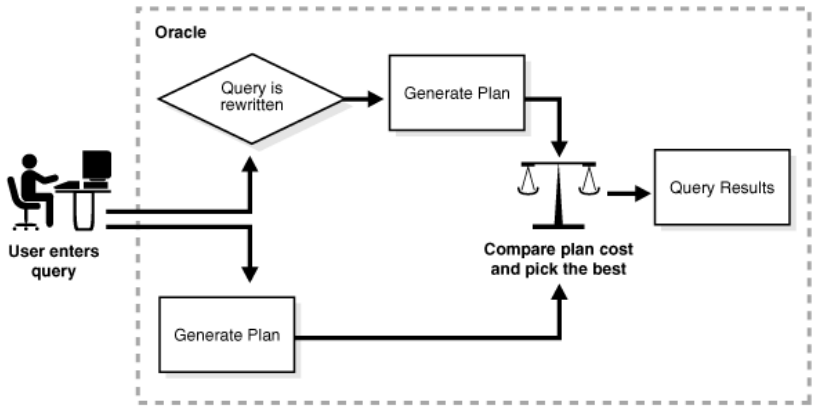
```
CREATE TABLE emp
    (empno     NUMBER         PRIMARY KEY,
     ename     VARCHAR2(10)   NOT NULL
                              CHECK (ename = UPPER(ename)),
     job       VARCHAR2(9),
     mgr       NUMBER         REFERENCES scott.emp(empno),
     hiredate  DATE           CHECK (hiredate < TO_DATE ('08-14-1998', 'MM-DD-YYYY')),
     sal       NUMBER(10,2)   CHECK (sal > 500),
     comm      NUMBER(9,0)    DEFAULT NULL,
     deptno    NUMBER(2)      NOT NULL )
     CLUSTER personnel (deptno);
CREATE TABLE dept
    (deptno  NUMBER(2),
     dname   VARCHAR2(9),
     loc     VARCHAR2(9))
     CLUSTER personnel (deptno);
```

- Vista materializada: vista + datos
- Muy útil para evitar la reunion natural (*natural join*) en tablas que se actualizan poco, datawarehouse, etc.

```
CREATE MATERIALIZED VIEW mi_vista_materializada
 [TABLESPACE mi_tablespace]
 [BUILD {IMMEDIATE | DEFERRED}]
 [REFRESH {ON COMMIT | ON DEMAND | [START WITH fecha_inicio] NEXT fecha_intervalo } |
          {COMPLETE | FAST | FORCE} ]
 [{ENABLE|DISABLE} QUERY REWRITE] AS
     SELECT t1.campo1, t2.campo2
     FROM mi_tabla1 t1 , mi_tabla2 t2
     WHERE t1.campo_fk = t2.campo_pk AND …
```

- `BUILD`: cuándo debe hacerse la carga inicial
- `REFRESH`: cómo debe actualizarse la vista
- `QUERY REWRITE`: usa la vista materializada implicitamente en el plan de ejecución de otras consultas

4.2.5 Vistas materializadas
Tema 4: Optimización de Bases de datos Parte II: Modelo físico de datos



Vista	Objeto
DBA_INDEXES ALL_INDEXES USER_INDEXES	indices de la base de datos
DBA_IND_COLUMNS ALL_INDEXES USER_INDEXES	columnas indexadas en la base de datos
DBA_CLUSTERS ALL_CLUSTERS USER_CLUSTERS	Todos los *clusters* *Clusters* accesibles por el usuario *clusters* propiedad del usuario
DBA_CLU_COLUMNS ALL_CLU_COLUMNS USER_CLU_COLUMNS	Columnas que definen *clusters*
DBA_PART_TABLES ALL_PART_TABLES USER_PART_TABLES	Todas las tablas particionadas Particiones accesibles por el usuario Particiones propiedad del usuario
USER_MVIEWS	Vistas materializadas del usuario

`select *`
`from user_objects`
`where object_type='MATERIALIZED VIEW'`
	Vistas materializadas del usuario

- Logs del sistema
  - El más importate es el log de eventos.
- `Statspack`
  - Permite obtener “instantaneas” de la base de datos
  - Está incluido en todas las versiones de Oracle
- `Automatic Workload Repository` (`AWR`)
  - Alternativa a statspack, solo disponible en la versión *enterprise*
  - Detallado analisis del rendimiento de la base de datos (CPU, memoria, E/S, PL/SQL, RAC, configuración del sistema…)
- `Automated Database Diagnostics Monitor` (`ADDM`)
  - Realiza análisis del sistema, identifica los posibles problemas y sus causas potenciales, y por último plantea recomendaciones para solucionarlos.
  - `DBMS_ADVISOR` es parte de este paquete
  - Se puede ejecutar desde `SQL*PLUS` mediante el script `addmrpt.sql`
  - Internamente utiliza `AWR`
- `TKProf`
  - Permite generar informes a partir del seguimiento de la ejecución de una consulta SQL

1. Partiendo del siguiente modelo lógico de datos, escribir un modelo físico de datos adecuado a los siguientes supuestos. Justifica la la respuesta

```
create table al;
cod number(5,0) primary key,
dni varchar2(9) unique not null,
nombre varchar2(20) not null,
apellidos varchar2 (40) not null ,
email varchar2(20),
movil number(9),
direccion_postal varchar2(50) not null,
poblacion varchar2(50) not null
);

create table asignatura (
  cod number(5,0) primary key,
  nombre varchar2(40) not null,
  titulacion varchar2(40) not null,
  ct number(2,1) not null check (ct>=3 AND ct<=7.5),
  cp number(2,1) not null check (cp>=3 AND cp<=7.5),
  constraint ck_asignatura check (ct+cp<=9)
  );

create table alas (
    cod number(7,0) primary key,
    cod_al number not null references al,
    cod_as number not null references asignatura,
    fecha date,
    constraint ck_alas unique(cod_al,cod_as,fecha)
);
```

```
create table dpto (
 cod number(3,0) primary key,
 nombre varchar2(40) unique not null
);
create table prof (
 cod number(4,0) primary key,
 dni varchar2(9) unique not null,
 nombre varchar2(20) not null,
 apellidos varchar2 (40) not null ,
 email varchar2(20),
 telefono number(9),
 cod_dpto number references dpto
);


create table area_conocimiento (
 cod number(4,0) primary key,
 nombre varchar2(20) not null,
 cod_dpto number(3,0) references dpto
);

create table profas (
  cod_prof number(4,0) references prof,
  cod_as number(5,0) references asignatura,
  constraint pk_profas primary key(cod_prof, cod_as)
);
```

4.2.8 Ejercicios

Tema 4: Diseño físico de la base de datos

a. Frecuentemente se realizan consultas atendiendo al numero total de créditos de la asignatura

b. La tabla `alas` puede tener algunos millones de entradas y generalmente solo se recuperan las asignaturas de un curso concreto

c. La tabla `area_conocimiento` prácticamente siempre aparece combinada con la tabla `dpto` mediante una reunión natural.

d. El centro de prospectiva de estudios universitarios realiza análisis trimestrales atendiendo a la edad de los alumnos, su procedencia, su sexo, las asignaturas que está matriculado, las titulaciones a las que pertenecen estos alumnos. Por ejemplo: ¿en qué porcentaje aprueban los alumnos por localidad “Gestión y administración de bases de datos”? ¿qué grados presentan mayor diferencia de matriculados atendiendo al sexo? ¿qué grados presentan un mayor índice de suspensos? etc.

e. Suponiendo un uso normal de la base de datos, ¿qué otras mejoras se te ocurren?

2. Sobre el diseño físico de la base de datos:
   a) Es consecuencia directa del diseño lógico de datos
   b) Su principal objetivo es mejorar la integridad de los datos
   c) Su principal objetivo es mejorar el rendimiento de la base de datos
   d) Cualquier modificación del diseño físico de la base de datos conlleva modificar el modelo lógico de datos

3. Si creamos un índice asociado a una columna de una determinada tabla
   a) Mejorará los tiempos de acceso a esa tabla a través de la columna indexada
   b) Mejorará las inserciones de nuevas filas en esa tabla
   c) Conlleva que la columna no podrá almacenar valores nulos
   d) Conlleva que la columna necesariamente es clave principal o parte de una clave externa

4. Una tabla organizada por índice
   a) Es aquella donde almacenamos toda la tupla en el índice asociado a la clave principal
   b) Es aquella cuya clave está indexada
   c) Optimiza el acceso a tablas con pocos atributos no primos
   d) Es aquella donde todos los atributos son primos

4.2.8 Ejercicios

Tema 4: Optimización de Bases de datos Parte II: Modelo físico de datos

5. En el diseño físico de la base de datos, sobre un cluster se puede afirmar que:
   a) Se almacenan físicamente juntas tuplas de un conjunto de tablas que comparten una o más columnas
   b) Optimiza operaciones de tipo reunión natural atendiendo a los atributos que definen el cluster
   c) Es un tipo de organización física de tabla donde almacenamos toda la tupla en el índice asociado a la clave principal
   d) Una tabla puede pertenecer a varios clusters simultáneamente

6. En el diseño físico de la base de datos, sobre una partición se puede afirmar que:
   a) Se almacenan físicamente juntas tuplas de un conjunto de tablas que comparten una o más columnas
   b) Todas las particiones de una misma tabla se almacenan en el mismo espacio de tablas
   c) Distribuye una tabla entre diferentes segmentos. Cada segmento almacena solo algunas de las columnas de la tabla
   d) Su finalidad principal es optimizar las operaciones de actualización de datos