Iteración 2 Proyecto HotelAndes C6

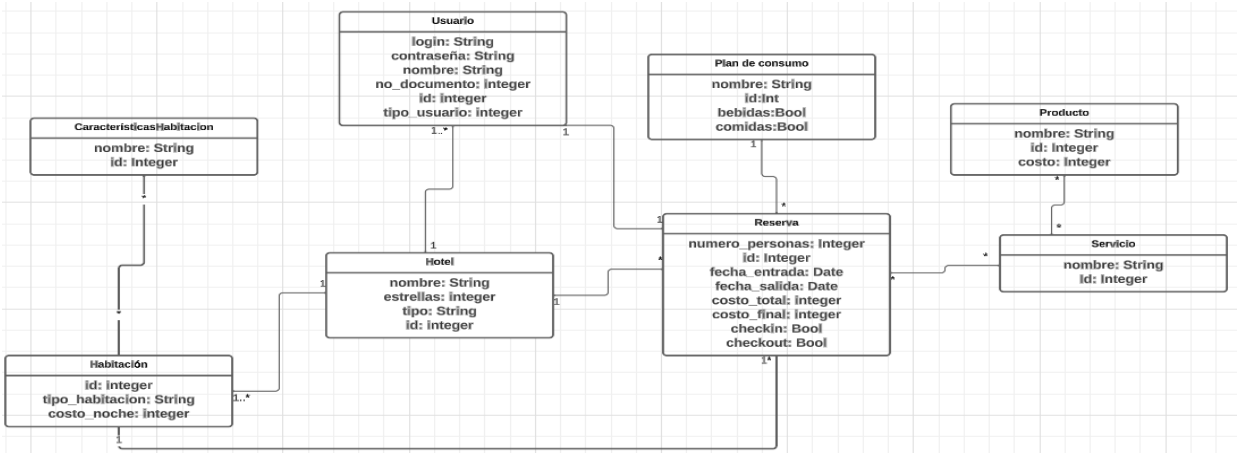
María Fernanda de la Hoz - 202214512

Juan David Obando - 202123148

Julián Contreras - 202211884

Análisis:

Teniendo en cuenta los problemas encontrados durante la implementación de nuestro primer diseño de aplicación, decidimos simplificar en gran medida el tamaño de esta. Para ello, eliminamos todas las herencias de la clase servicio y las reemplazamos por una mayor especificidad dentro de la tabla única “Servicio” asociada directamente a la tabla “Producto”, la cual se refiere a cualquier tipo de consumo posible, por ejemplo, un artículo de una tienda o un platillo del restaurante.



Por otra parte, para la correcta implementación de los nuevos requerimientos funcionales solicitados en esta entrega, tuvimos que añadir nuevos controladores y repositorios. Además, se incluyeron diferentes columnas a algunas tablas. Estas son:

Usuario: Se incluyo id como la llave primaria.

Hotel: Se implemento id como llave primaria.

Habitación: se añadió “cont_habitaciones” que rastrea cuántos días ha estado ocupada desde el primero de enero hasta la fecha en busca de resolver RFC3

Reserva_Servicio: Se añadió "fecha_consumo” para poder rastrear qué día se realizó un consumo de un servicio asociado a una reserva en particular en busca de resolver RFC5

Reserva: Se añadió “id_habitacion” que es una llave foránea que permite enlazar la reserva con la habitación que le fue asignada.

Diseño:

RF1: Para este requerimiento se solicita mostrar el dinero recolectado por servicios en cada habitación en el último año corrido. Por lo que la decisión de no utilizar indices está dada por los elementos que se necesitan en la consulta, es decir un filtrado por fechas del último año y el dinero recolectado por habitación. En el caso de un índice en datos por fechas, tendría baja selectividad un índice debido a que el rango de fechas cubre un año entero. Por otro lado, en la tabla "reservas_servicios" el considerar un número muy alto de filas con pocos servicios únicos (es decir, muchos servicios repetidos), entonces un índice en servicios_id puede no ser muy selectivo. Teniendo en cuenta que en la forma como se plantearon tiene un costo relativamente bajo (4), no se consideró necesario hacer una optimización.

Es por esto que se optó por la no utilización de índices para este requerimiento.

RF2: Para este requerimiento se solicita mostrar los 20 servicios más populares. Dado que la consulta requiere agrupar y ordenar los resultados, el motor de la base de datos necesitará acceso a todas las filas de todas formas para calcular la suma y ordenar los resultados. Un índice podría no ser beneficioso si el proceso de agrupación domina el costo de la consulta. También es importante resaltar que los índices adicionales requieren mantenimiento y pueden afectar el rendimiento de las operaciones de escritura (INSERT, UPDATE, DELETE). Dado que la tabla reservas es altamente volátil con muchas escrituras, los índices pueden resultar en un costo de rendimiento para las operaciones de escritura.

Además, por la manera como está implementada la lógica de este requerimiento, el costo de la consulta es muy bajo (5). Por lo que la creación de índices mejoraría muy poco el costo de consulta.

Es por esto que se optó por la no implementación de índices.

RF3: Para este requerimiento se pide mostrar el índice de ocupación de cada una de las habitaciones del hotel. Dado que la consulta selecciona y calcula valores para todas las filas de la tabla habitaciones. No hay una cláusula WHERE que filtre las filas basándose en criterios específicos donde un índice pudiera ser útil para reducir el número de filas a examinar. En este caso, la consulta no es selectiva en absoluto; recupera todas las filas, por lo que un índice no mejoraría el rendimiento.

Es por esto que se optó por la no implementación de índices.

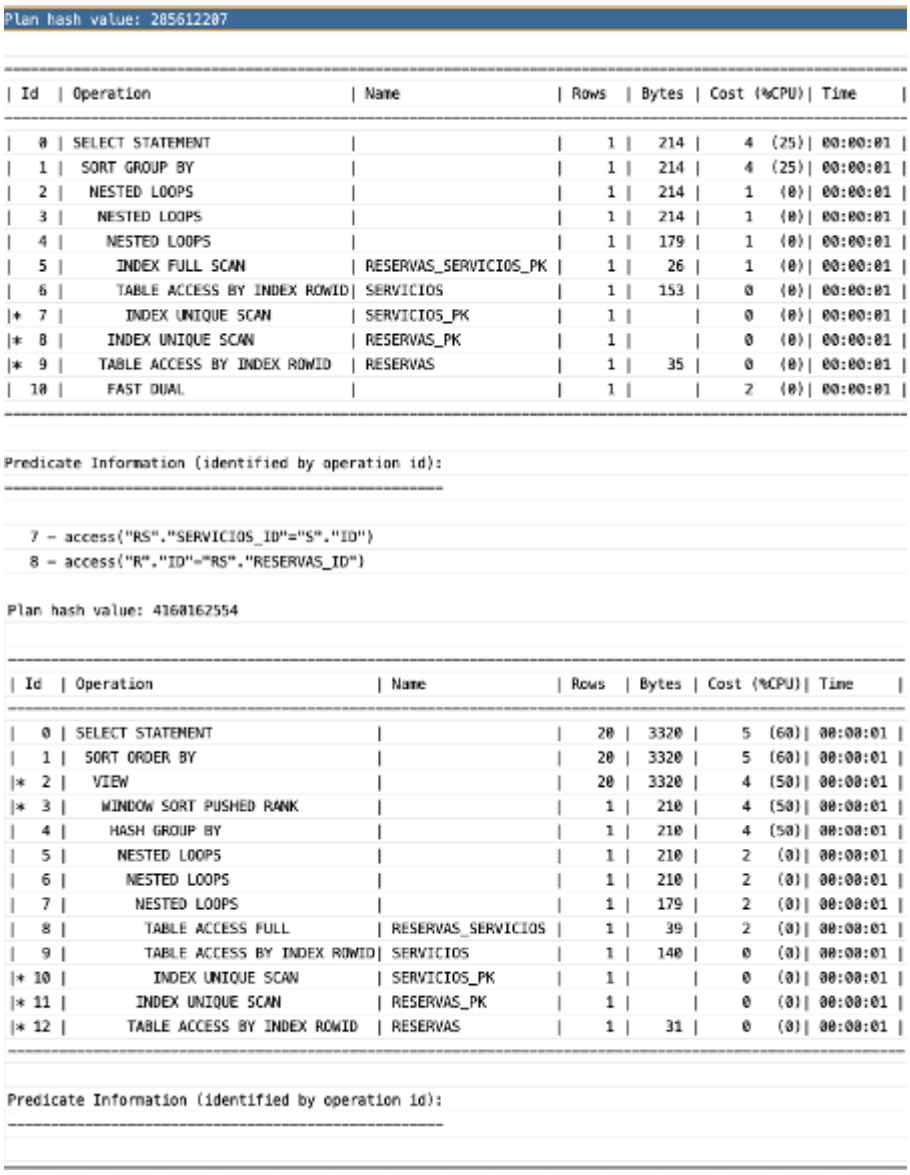
RF5: Para este requerimiento se solicita mostrar el consumo en hotelandes por un usuario dado, en un rango de fechas indicado en la consulta proporcionada, estamos filtrando por el campo login de la tabla usuarios, que podría tener una alta selectividad si cada usuario tiene un login único. Por lo tanto, un índice en usuarios(login) sería beneficioso. También estamos filtrando por un rango de fechas en la tabla reservas, lo que podría justificar un índice en reservas (fecha_entrada, fecha_salida) para acelerar esta parte de la consulta.

RF6: Para este requerimiento se pide indicar cuáles fueron las fechas de los días de mayor ocupación (mayor cantidad habitaciones ocupadas), las fechas de mayores ingresos (mayor cantidad de consumos realizados) y también las fechas de menor demanda (menor ocupación). Estamos filtrando por un rango de fechas en la tabla reservas, lo que podría justificar un índice en reservas (fecha_entrada, fecha_salida) para acelerar esta parte de la consulta.

RF7: Para este requerimiento se solicita encontrar clientes que han estado en el hotel por lo menos dos semanas (no necesariamente en una sola estadía) o si ha consumido más de $15’000.000.oo, durante el último año de operación de HotelAndes. Estamos filtrando por un rango de fechas en la tabla reservas, lo que podría justificar un índice en reservas (fecha_entrada, fecha_salida) para acelerar esta parte de la consulta.

RF8: Para este requerimiento se pide encontrar los servicios que hayan sido solicitados menos de 3 veces semanales, durante el último año de operación de HotelAndes. Estamos filtrando por un rango de fechas en la tabla reservas, lo que podría justificar un índice en reservas (fecha_entrada, fecha_salida) para acelerar esta parte de la consulta.

Índices Generados por Oracle:



```
Plan hash value: 285612207

-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                      |     1 |   214 |     4  (25)| 00:00:01 |
|   1 |  SORT GROUP BY                |                      |     1 |   214 |     4  (25)| 00:00:01 |
|   2 |   NESTED LOOPS                |                      |     1 |   214 |     1   (0)| 00:00:01 |
|   3 |    NESTED LOOPS               |                      |     1 |   214 |     1   (0)| 00:00:01 |
|   4 |     NESTED LOOPS              |                      |     1 |   179 |     1   (0)| 00:00:01 |
|   5 |      INDEX FULL SCAN          | RESERVAS_SERVICIOS_PK |     1 |    26 |     1   (0)| 00:00:01 |
|   6 |      TABLE ACCESS BY INDEX ROWID| SERVICIOS          |     1 |   153 |     0   (0)| 00:00:01 |
|*  7 |       INDEX UNIQUE SCAN       | SERVICIOS_PK         |     1 |       |     0   (0)| 00:00:01 |
|*  8 |     INDEX UNIQUE SCAN         | RESERVAS_PK          |     1 |       |     0   (0)| 00:00:01 |
|*  9 |    TABLE ACCESS BY INDEX ROWID| RESERVAS             |     1 |    35 |     0   (0)| 00:00:01 |
|  10 |     FAST DUAL                 |                      |     1 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   7 - access("RS"."SERVICIOS_ID"="S"."ID")
   8 - access("R"."ID"="RS"."RESERVAS_ID")

Plan hash value: 4160162554

-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                      |    20 |  3320 |     5  (60)| 00:00:01 |
|   1 |  SORT ORDER BY                |                      |    20 |  3320 |     5  (60)| 00:00:01 |
|*  2 |   VIEW                        |                      |    20 |  3320 |     4  (50)| 00:00:01 |
|*  3 |    WINDOW SORT PUSHED RANK    |                      |     1 |   210 |     4  (50)| 00:00:01 |
|   4 |     HASH GROUP BY             |                      |     1 |   210 |     4  (50)| 00:00:01 |
|   5 |      NESTED LOOPS             |                      |     1 |   210 |     2   (0)| 00:00:01 |
|   6 |       NESTED LOOPS            |                      |     1 |   210 |     2   (0)| 00:00:01 |
|   7 |        NESTED LOOPS           |                      |     1 |   179 |     2   (0)| 00:00:01 |
|   8 |         TABLE ACCESS FULL     | RESERVAS_SERVICIOS   |     1 |    39 |     2   (0)| 00:00:01 |
|   9 |         TABLE ACCESS BY INDEX ROWID| SERVICIOS       |     1 |   140 |     0   (0)| 00:00:01 |
|* 10 |          INDEX UNIQUE SCAN    | SERVICIOS_PK         |     1 |       |     0   (0)| 00:00:01 |
|* 11 |        INDEX UNIQUE SCAN      | RESERVAS_PK          |     1 |       |     0   (0)| 00:00:01 |
|* 12 |       TABLE ACCESS BY INDEX ROWID| RESERVAS          |     1 |    31 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
```

```
Plan hash value: 2883760253

-------------------------------------------------------------------------------------
| Id  | Operation          | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |               |     1 |    26 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| HABITACIONES |     1 |    26 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

Plan hash value: 2304871597

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            |    50 |  2400 |     9  (34)| 00:00:01 |
|   1 |  SORT ORDER BY               |            |    50 |  2400 |     9  (34)| 00:00:01 |
|*  2 |   VIEW                       |            |    50 |  2400 |     8  (25)| 00:00:01 |
|*  3 |    WINDOW SORT PUSHED RANK|            |     1 |    22 |     8  (25)| 00:00:01 |
|   4 |     HASH GROUP BY            |            |     1 |    22 |     8  (25)| 00:00:01 |
|   5 |      VIEW                    | VM_NWVW_1 |     1 |    22 |     7  (15)| 00:00:01 |
|   6 |       HASH GROUP BY          |            |     1 |    25 |     7  (15)| 00:00:01 |
|*  7 |        TABLE ACCESS FULL     | RESERVAS   |     1 |    25 |     2   (0)| 00:00:01 |
|   8 |         SORT AGGREGATE       |            |     1 |     9 |            |          |
|   9 |          TABLE ACCESS FULL| RESERVAS   |     1 |     9 |     2   (0)| 00:00:01 |
|  10 |         SORT AGGREGATE       |            |     1 |     9 |            |          |
|  11 |          TABLE ACCESS FULL| RESERVAS   |     1 |     9 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
```

{} PLAN_TABLE_OUTPUT

```
Plan hash value: 2513089040

-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name                  | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                       |     1 |   398 |     3   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                 |                       |     1 |   398 |     3   (0)| 00:00:01 |
|   2 |   NESTED LOOPS                |                       |     1 |   398 |     3   (0)| 00:00:01 |
|   3 |    NESTED LOOPS               |                       |     1 |   258 |     3   (0)| 00:00:01 |
|   4 |     MERGE JOIN CARTESIAN      |                       |     1 |   223 |     3   (0)| 00:00:01 |
|   5 |      NESTED LOOPS             |                       |     1 |   179 |     1   (0)| 00:00:01 |
|   6 |       NESTED LOOPS            |                       |     1 |   179 |     1   (0)| 00:00:01 |
|   7 |        INDEX FULL SCAN        | SERVICIOS_PRODUCTOS_PK |     1 |    26 |     1   (0)| 00:00:01 |
|*  8 |        INDEX UNIQUE SCAN      | PRODUCTOS_PK          |     1 |       |     0   (0)| 00:00:01 |
|   9 |       TABLE ACCESS BY INDEX ROWID| PRODUCTOS          |     1 |   153 |     0   (0)| 00:00:01 |
|  10 |      BUFFER SORT              |                       |     1 |    44 |     3   (0)| 00:00:01 |
|* 11 |       TABLE ACCESS FULL       | RESERVAS              |     1 |    44 |     2   (0)| 00:00:01 |
|  12 |     TABLE ACCESS BY INDEX ROWID | RESERVAS_SERVICIOS  |     1 |    35 |     0   (0)| 00:00:01 |
|* 13 |      INDEX UNIQUE SCAN        | RESERVAS_SERVICIOS_PK |     1 |       |     0   (0)| 00:00:01 |
|* 14 |    INDEX UNIQUE SCAN          | USUARIOS_PK           |     1 |       |     0   (0)| 00:00:01 |
|* 15 |   TABLE ACCESS BY INDEX ROWID | USUARIOS              |     1 |   140 |     0   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	179	8 (25)	00:00:01
* 1	HASH JOIN		1	179	8 (25)	00:00:01
2	VIEW		1	153	3 (34)	00:00:01
* 3	FILTER					
4	HASH GROUP BY		1	175	3 (34)	00:00:01
5	NESTED LOOPS		1	175	2 (0)	00:00:01
6	NESTED LOOPS		1	175	2 (0)	00:00:01
* 7	TABLE ACCESS FULL	RESERVAS	1	35	2 (0)	00:00:01
* 8	INDEX UNIQUE SCAN	USUARIOS_PK	1		0 (0)	00:00:01
9	TABLE ACCESS BY INDEX ROWID	USUARIOS	1	140	0 (0)	00:00:01
10	VIEW		1	26	5 (20)	00:00:01
* 11	FILTER					
12	HASH GROUP BY		1	171	5 (20)	00:00:01
* 13	FILTER					
14	NESTED LOOPS		1	171	2 (0)	00:00:01
15	NESTED LOOPS		1	171	2 (0)	00:00:01
* 16	TABLE ACCESS FULL	RESERVAS	1	31	2 (0)	00:00:01
17	FAST DUAL		1		2 (0)	00:00:01
* 18	INDEX UNIQUE SCAN	USUARIOS_PK	1		0 (0)	00:00:01
19	TABLE ACCESS BY INDEX ROWID	USUARIOS	1	140	0 (0)	00:00:01

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	179	8 (25)	00:00:01
* 1	HASH JOIN		1	179	8 (25)	00:00:01
2	VIEW		1	153	3 (34)	00:00:01
* 3	FILTER					
4	HASH GROUP BY		1	175	3 (34)	00:00:01
5	NESTED LOOPS		1	175	2 (0)	00:00:01
6	NESTED LOOPS		1	175	2 (0)	00:00:01
* 7	TABLE ACCESS FULL	RESERVAS	1	35	2 (0)	00:00:01
* 8	INDEX UNIQUE SCAN	USUARIOS_PK	1		0 (0)	00:00:01
9	TABLE ACCESS BY INDEX ROWID	USUARIOS	1	140	0 (0)	00:00:01
10	VIEW		1	26	5 (20)	00:00:01
* 11	FILTER					
12	HASH GROUP BY		1	171	5 (20)	00:00:01
* 13	FILTER					
14	NESTED LOOPS		1	171	2 (0)	00:00:01
15	NESTED LOOPS		1	171	2 (0)	00:00:01
* 16	TABLE ACCESS FULL	RESERVAS	1	31	2 (0)	00:00:01
17	FAST DUAL		1		2 (0)	00:00:01
* 18	INDEX UNIQUE SCAN	USUARIOS_PK	1		0 (0)	00:00:01
19	TABLE ACCESS BY INDEX ROWID	USUARIOS	1	140	0 (0)	00:00:01