Iteración 3

Sebastián García 201630047, Nicolas Sotelo 201623026

Grupo D-05

Sistemas Transaccionales

Ingenieria de Sistemas y Computación

{js.garcial1, n.sotelo[}@uniandes.edu.co](mailto:%7d@uniandes.edu.co)

Mayo 20 de 2018

Tabla de contenido

[1 Introducción 1](#_Toc514573085)

[2 Diseño y Construcción de la Aplicación 1](#_Toc514573086)

[2.1 Diseño de la aplicación 1](#_Toc514573087)

[3 Diseño físico de la Aplicación 2](#_Toc514573088)

[3.1 Índices generados por Oracle 2](#_Toc514573089)

[3.2 RFC 10 Consultar consumo en Alohandes 3](#_Toc514573090)

[3.3 RFC 11 Consultar inversa de consumo en Alohandes 4](#_Toc514573091)

[3.4 RFC 12 Consultar de funcionamiento 5](#_Toc514573092)

[3.5 RFC 11 Consultar los buenos clientes 5](#_Toc514573093)

[4 Diseño de datos 5](#_Toc514573094)

[5 Consideraciones 6](#_Toc514573095)

[6 Resultados 6](#_Toc514573096)

# Introducción

El objetivo de la siguiente iteración es integrar requerimientos de eficiencia a una aplicación transaccional desarrollada en una arquitectura de tres niveles, con interfaz mediante servicios REST y manejo de persistencia en base de datos.

# Diseño y Construcción de la Aplicación

## Diseño de la aplicación

Para realizar los requerimientos funcionales de la presente iteración 3 no se hicieron modificaciones a las relaciones que se implementaron en la anterior iteración 2. De la misma forma no se implementaron nuevas relaciones a las ya establecidas en la anterior iteración.

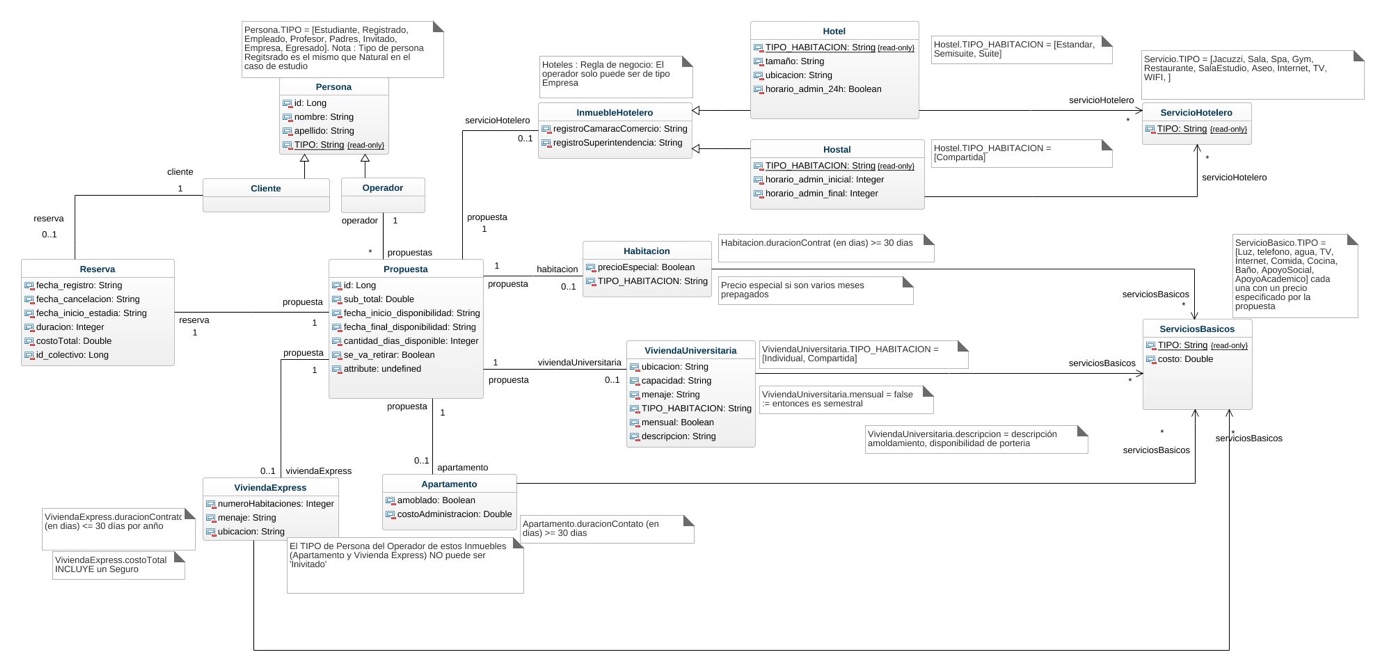


Figura 1. Modelo de clases UML. (Los modelos en formato PDF y original se encuentra en docs > Iteración 3 > modelos).

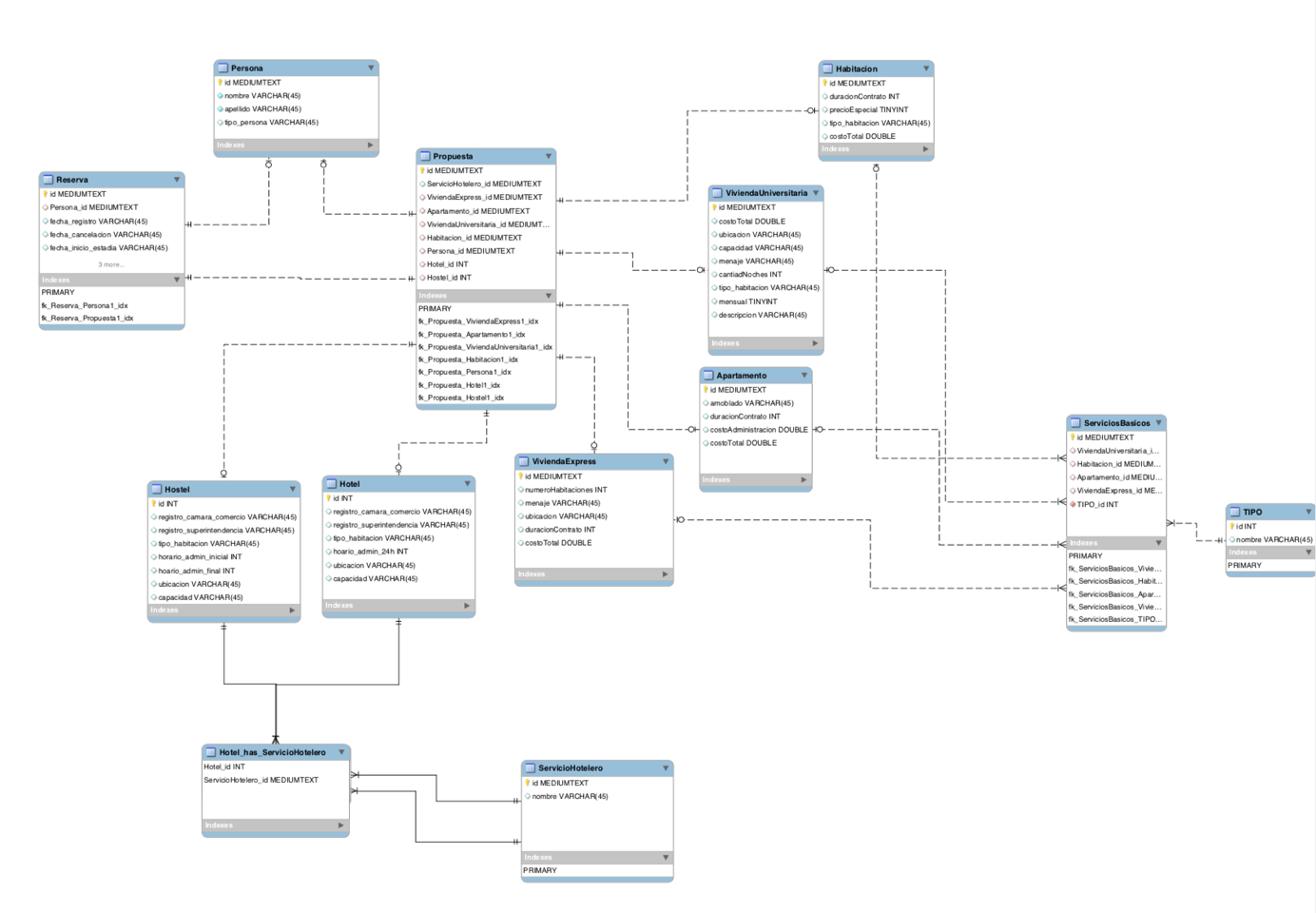


Figura 2. Modelo relacional. (Los modelos en formato PDF y original se encuentra en docs > Iteración 3 > modelos).

# Diseño físico de la Aplicación

## Índices generados por Oracle

Todos los índices encontrados son creados de forma automática por Oracle debido a que son o hacen parte de una llave primaria esto pasa porque la llave primaria tiene un valor único que me permite identificar cada uno de los registros y encontrarlos en disco de manera más rápida y eficiente. Estos índices facilitan la realización de *joins* e inserciones en las tablas para los requerimientos anteriores.

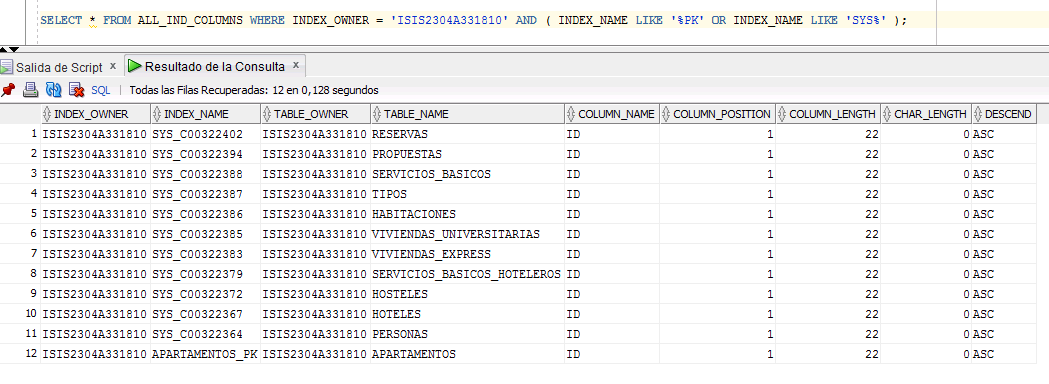


Figura 3. Índices creados por Oracle.

## RFC 10 Consultar consumo en Alohandes

* **Selección de índices**

Para este requerimiento se implementaron dos índices. El primero, TIPO\_INMUEBLE\_INDEX, sobre la relación PROPUESTAS en los atributos ID y TIPO\_INMUEBLE, esto con el fin de poder obtener una propuesta por identificador y su tipo de inmueble de una forma más eficiente. De la misma manera, se implementó el índice P\_FIE\_INDEX sobre la relación de RESERVAS que involucra a los atributos ID\_PROPUESTA y FECHA\_INICIO\_ESTADIA con el propósito de obtener de las reservas, la propuesta sobre la cual se reservó y su fecha en la que inicia la estadía en el inmueble. Estos dos índices son de tipo **B+** en donde el costo es proporcional al número de hojas y la cantidad de operaciones *input-output*. Adicionalmente, los índices TIPO\_INMUEBLE\_INDEX y P\_FIE\_INDEX son secundarios, o no integrados, lo que significa que se mantienen en un archivo aparte, al archivo que contiene la relación.

* **Sentencia SQL utilizada**

SELECT PER.\*, PP.TIPO

FROM PERSONAS PER

INNER JOIN (

SELECT R.ID\_PERSONA AS PERSO, P.TIPO\_INMUEBLE AS TIPO

FROM RESERVAS R

INNER JOIN PROPUESTAS P

ON P.ID = R.ID\_PROPUESTA

WHERE P.ID = 12

AND R.FECHA\_INICIO\_ESTADIA BETWEEN '2018-01-18' AND '2018-01-28'

)

PP ON PP.PERSO = PER.ID

ORDER BY PP.TIPO

;

* **Valores de los parámetros utilizados**

Identificador de la oferta de alojamiento sobre la cual se va a hacer el análisis. En este caso, el identificador de la oferta es 12.

Rango de fechas. En este caso es '2018-01-18' AND '2018-01-28'.

* **Plan de consulta**

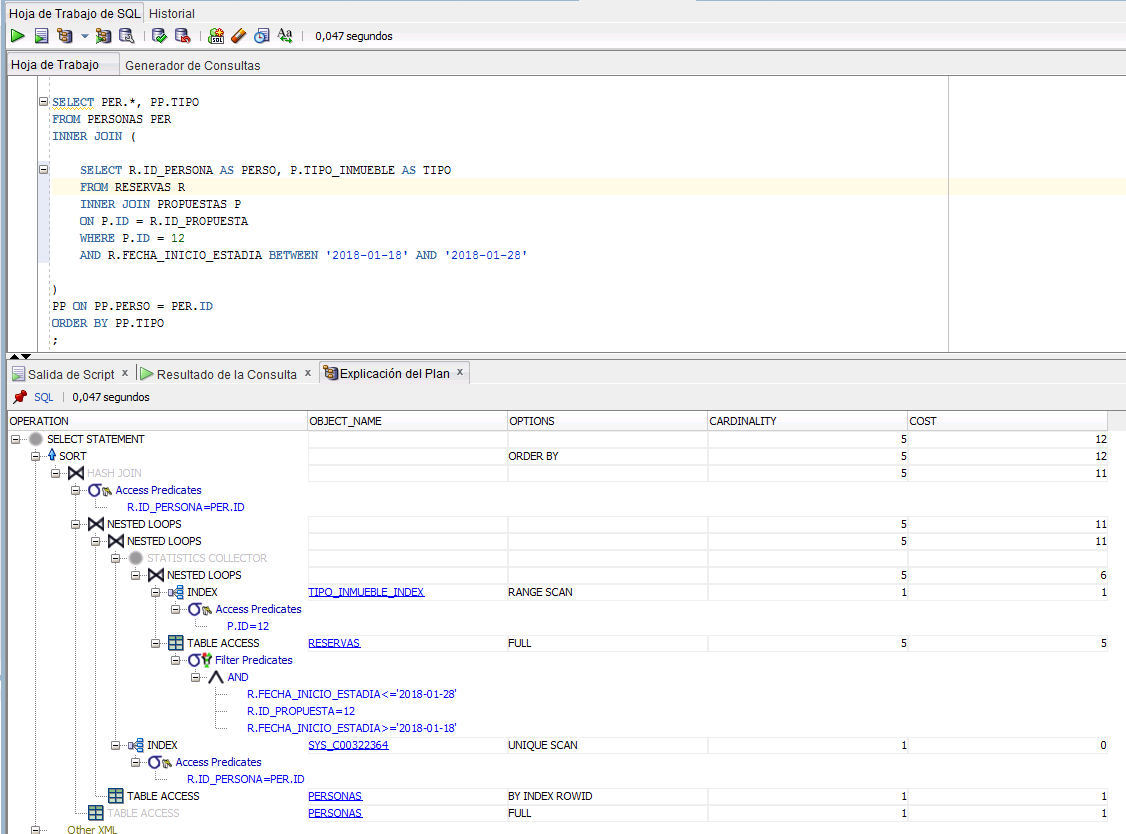


Figura 4. Plan de consulta para el requerimiento RFC 10.

* **Análisis de eficiencia**

## RFC 11 Consultar inversa de consumo en Alohandes

* **Selección de índices**

Para este requerimiento se implementaron los mismo dos índices del requerimiento anterior RFC 10.

* **Sentencia SQL utilizada**

SELECT PER.\*, PP.TIPO

FROM PERSONAS PER

INNER JOIN (

SELECT R.ID\_PERSONA AS PERSO, P.TIPO\_INMUEBLE AS TIPO

FROM RESERVAS R

INNER JOIN PROPUESTAS P

ON P.ID = R.ID\_PROPUESTA

WHERE P.ID = 12

AND R.FECHA\_INICIO\_ESTADIA NOT BETWEEN '2018-01-18' AND '2018-01-28'

)

PP ON PP.PERSO = PER.ID

ORDER BY PP.TIPO

;

* **Valores de los parámetros utilizados**

Identificador de la oferta de alojamiento sobre la cual se va a hacer el análisis. En este caso, el identificador de la oferta es 12.

Rango de fechas. En este caso es '2018-01-18' AND '2018-01-28'.

* **Plan de consulta**

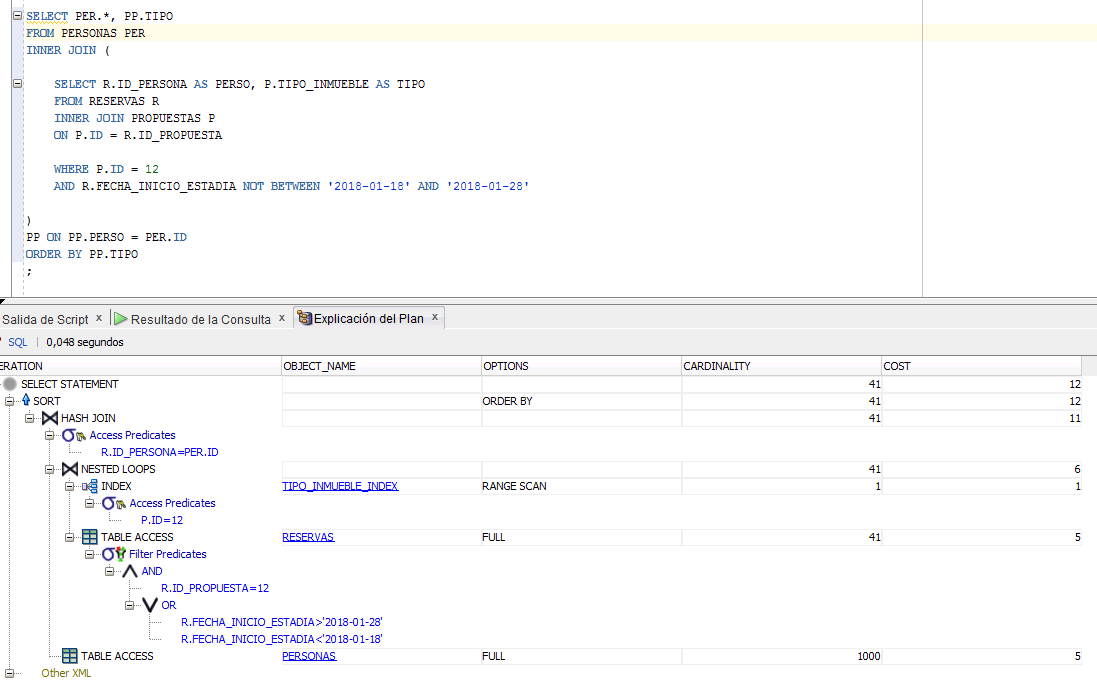


Figura 5. Plan de consulta para el requerimiento RFC 11.

* **Análisis de eficiencia**

## RFC 12 Consultar de funcionamiento

## RFC 13 Consultar los buenos clientes

# Diseño de datos

Se implementó un *loop* en SQL Developer para realizar la carga masiva de datos. Un ejemplo de esto, es con la relación APARTAMENTOS como se muestra a continuación.

DECLARE

ID NUMBER(20) :=1;

MAXIMO NUMBER(20) := 250000;

MUEBLE NUMBER(1) := 0;

COSTO NUMBER(20) := 120;

CAP NUMBER(20) := 2;

BEGIN

WHILE ID <= MAXIMO

LOOP

INSERT INTO

APARTAMENTOS( ID, AMOBLADO, COSTO\_ADMIN, CAPACIDAD\_MAXIMA )

VALUES ( ID, mueble, COSTO, cap );

ID := ID + 1;

IF(MUEBLE = 0 ) THEN

MUEBLE := 1;

ELSE

MUEBLE := 0;

END IF;

IF ( COSTO >= 340 ) THEN

COSTO := COSTO - 97;

ELSE

COSTO := COSTO + 26;

END IF;

IF ( CAP >= 20 ) THEN

CAP := CAP - 3;

ELSE

CAP := CAP + 1;

END IF;

END LOOP;

END;

En la anterior sentencia se empiezan declarando variables que sirven para generar valores distintos en cada atributo y de esta forma evitar la repetición de datos. Luego se implementa la función WHILE LOOP en donde en cada recorrido se inserta una nueva tupla en la relación y luego se modifican algunos valores para evitar la repetición de los mismos en distintas tuplas. Un ejemplo de lo anterior es incrementar en 1 el valor del identificador.

Para las demás relaciones en la base de datos se implementó la misma metodología. Las sentencias completas para cada relación se pueden encontrar en la carpeta *insert* que se encuentra en docs > Iteracion 3 > SQL.

# Análisis del proceso de optimización y el modelo de ejecución de consultas

# Escenarios de prueba

# Consideraciones

En los documentos anexos se encuentran la colección de pruebas Postman (docs > Iteracion 3 > Pruebas).

En la carpeta de SQL (docs > Iteracion 3 > SQL) se encentran las sentencias SQL implementadas para: crear las tablas y sus atributos, poblar las tablas, realizar los requerimientos funcionales, y se encuentran las sentencias utilizadas para completar los requerimientos funcionales de consulta.

# Resultados