Iteración 2 – Proyecto JDO

Diego A. Párraga González, Juan D. Vásquez Hernández Iteración 2 JDO Universidad de los Andes, Bogotá, Colombia

{d.parraga, jd.vasquezh[}@uniandes.edu.co](mailto:%7d@uniandes.edu.co) Fecha de presentación: Marzo 21 de 2023

**Tabla de contenido**

[1 Consulta 1: información general estadística de tablas 1](#_Toc130133298)

[2 Consulta 2: información de las llaves primarias 2](#_Toc130133299)

[3 Consulta 3: caracterización de los tipos de datos de las tablas 2](#_Toc130133300)

[4 Consulta 4: listado general de las columnas y sus restricciones 3](#_Toc130133301)

[5 Consulta 5: consultas sobre el catálogo y sobre los datos 5](#_Toc130133302)

# Consulta 1: información general estadística de tablas

Para esta consulta se pedía hallar el nombre de las tablas disponibles en la base de datos Parranderos, junto con su número de columnas general, el número de columnas que permiten valores tipo null y la fecha del último análisis realizado por Oracle sobre las tablas. La siguiente figura muestra el código desarrollado:

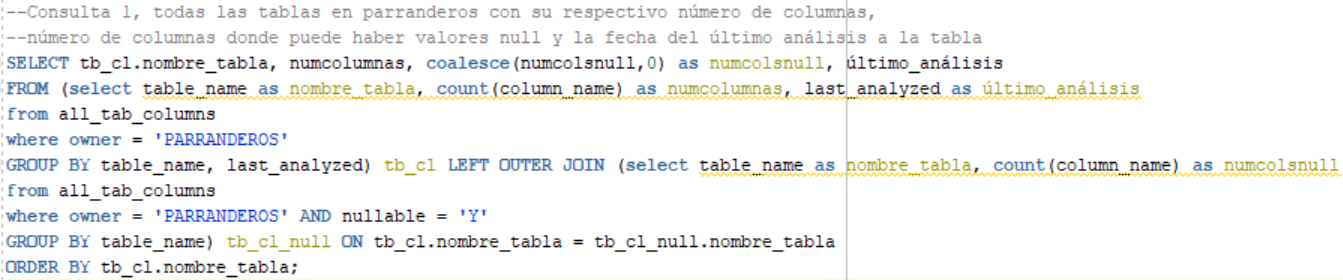


Figura 1. Solución planteada en SQL para la solución de la consulta 1.

Como se puede observar, se optó por realizar dos **select** internos, el primero encargado de encontrar el número de columnas general para cada una de las tablas y el segundo de encontrar aquellas tablas con columnas que contienen la propiedad *nullable.* Se realiza un **left outer join** dado a que existen tablas que no poseen ninguna columna de tipo *nullable* para obtener la respuesta final que está organizada por el nombre de cada una de las tablas. Para aquellas tuplas que no contienen columnas tipo *nullable,* se usa la función *coalesce()* para reemplazar el valor *null* por 0. Los resultados obtenidos por la consulta se muestran en la siguiente imagen:

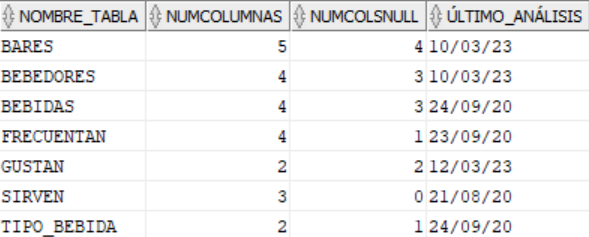


Figura 2. Resultado obtenido a partir de la solución implementada en el ejercicio 1.

# Consulta 2: información de las llaves primarias

Para esta consulta se pedía hallar la información de las columnas que conformaban la llave primaria de cada tabla junto con su tipo de dato. La siguiente figura muestra el código desarrollado:

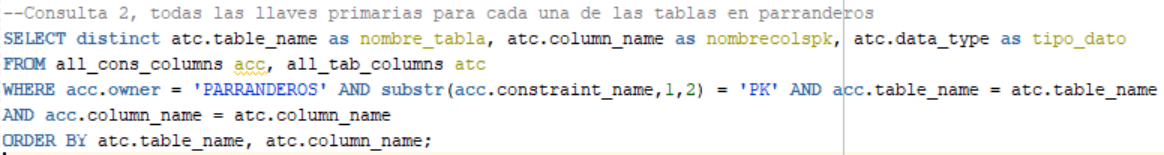


Figura 3. Solución planteada en SQL para la solución de la consulta 2.

Como se puede observar, se realiza un único **select** donde se buscan aquellos datos dentro de la información de todas las restricciones para las columnas que poseen en su nombre ‘PK’, lo cual es un indicativo de que la columna es parte de la llave primaria y se muestra la información de las tuplas de cada columna que cumplen con esta condición, además de que se organizan los datos según el nombre de la tabla y de la columna. Los resultados obtenidos por la consulta se muestran en la siguiente imagen:

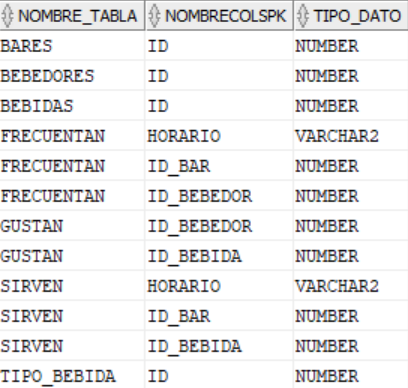


Figura 4. Resultado obtenido a partir de la solución implementada en el ejercicio 2.

# Consulta 3: caracterización de los tipos de datos de las tablas

Para cada una de las tablas, se pedía hallar los tipos de datos de las columnas, el número de datos de cada tipo y el promedio de la longitud de los datos al interior de las tuplas. La siguiente figura muestra el código desarrollado:

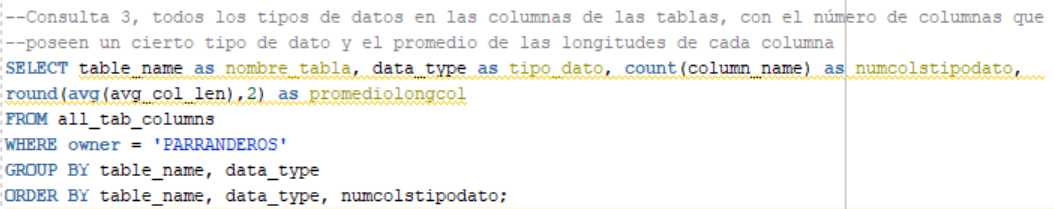


Figura 5. Solución planteada en SQL para la solución de la consulta 3.

Como se puede observar, se realiza un único **select** donde se utilizan todos los datos de las columnas en las diferentes tablas, se realiza un **group by** con el objetivo de aplicar las dos funciones de agregación necesarias – *count* para el número de columnas, *avg* para el promedio de la longitud de columna – y se organizan las tuplas de la respuesta basados en el nombre de la tabla, tipo de dato y número de columnas para un tipo de dato dado. Los resultados obtenidos por la consulta se muestran en la siguiente imagen:

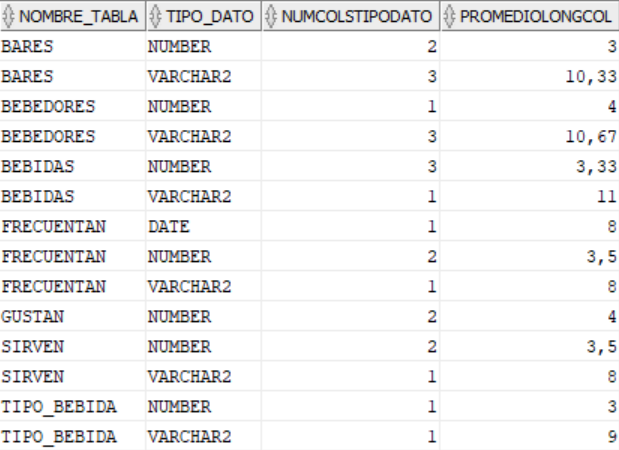


Figura 6. Resultado obtenido a partir de la solución implementada en el ejercicio 3.

# Consulta 4: listado general de las columnas y sus restricciones

Para esta consulta se pedía hallar para todas las columnas en todas las tablas el nombre de su tabla, el nombre de la columna, su tipo de dato, el nombre de las restricciones que posee y si permite nulos o no en sus valores. La siguiente figura muestra el código desarrollado:

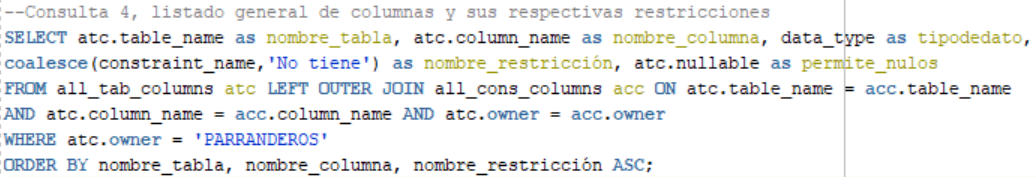


Figura 7. Solución planteada en SQL para la solución de la consulta 4.

Como se puede observar, se realiza un único **select** donde internamente se realiza un **left outer join** entre los datos de las columnas y las restricciones de cada columna (lo anterior debido a que existen columnas que no poseen ninguna restricción, y estas también deben hacer parte de la respuesta). Se entrega la información que se pide en la descripción del ejercicio, realizando un cambio sobre el nombre de la restricción si es que el dato es de tipo *null* por el String *‘No tiene’* y se organizan los datos de la respuesta en base al nombre de la tabla, el nombre de la columna y el nombre de la restricción. Una parte de los resultados obtenidos por la consulta se muestran en la siguiente imagen:



Figura 8. Resultado obtenido a partir de la solución implementada en el ejercicio 4.

# Consulta 5: consultas sobre el catálogo y sobre los datos

*Etapa 1 de la solución*

Para la primera parte de este ejercicio se pedía hallar la información de aquellas columnas dentro de la base de datos que no contenían una ‘R’ en su interior y que contenían entre 2 a menos vocales. La respuesta generada por esta parte de la consulta, debía generar una tabla cuyas filas contuviesen las sentencias SQL que se usarían en la siguiente parte del ejercicio, en el cual se pedía hallar todas aquellas restricciones que afectan de una u otra forma sobre una columna dada (es decir, las *foreign key* de otras tablas sobre una columna dada, también cuentan como restricciones que afectan sobre la columna). La siguiente figura muestra el código desarrollado para la primera parte del ejercicio:

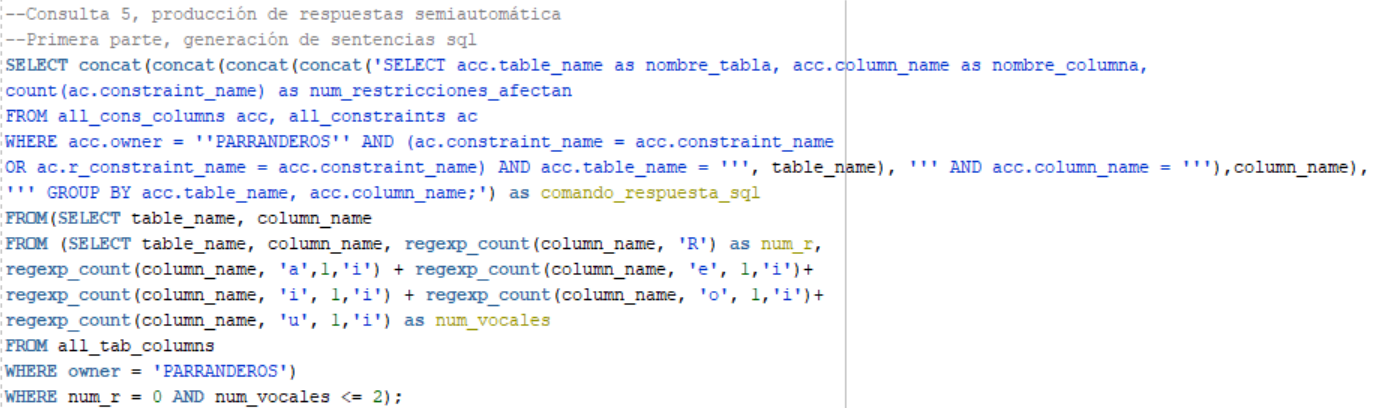


Figura 9. Solución planteada en SQL para la primera parte de la solución de la consulta 5.

Como se puede observar, se realizan 3 **select** al interior de la solución, donde el **select** más interno se encarga de calcular la cantidad de ‘R’s al interior del nombre de una columna, así como la cantidad de vocales que posee. Para esto, se emplea la función *regexp\_count()* cuyo primer parámetro corresponde al nombre de la columna – una cadena de caracteres – y el segundo parámetro, al carácter o patrón de caracteres que se debe buscar cuántas veces aparece al interior de la cadena. Posteriormente, el siguiente **select** se encarga de obtener solo la información de aquellas columnas que cumplen con la condición de no tener ninguna ‘R’ en su interior y no superar un total de 2 vocales en su nombre. Al finalizar, el **select** externo tiene la tarea de construir las sentencias SQL que se entregarán en una tabla para realizar el Script de la parte 2. Los resultados obtenidos por la consulta se muestran en la siguiente imagen:

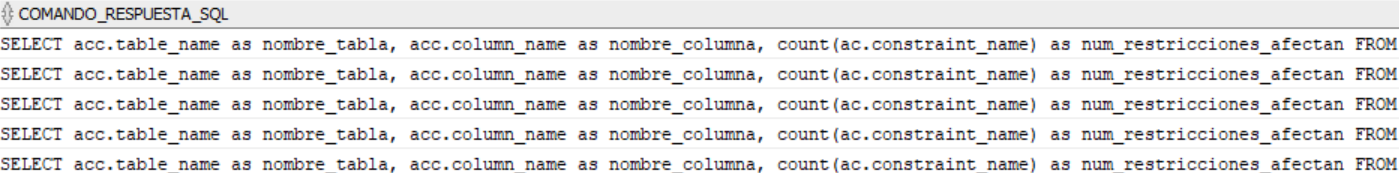


Figura 10. Resultado obtenido a partir de la solución implementada para la primera parte del ejercicio 5.

# 

Figura 11. Ejemplo de los comandos SQL arrojados por la primera parte.

Cada sentencia SQL desarrollada se encarga de calcular para el nombre de la columna y el nombre de una tabla dada, las restricciones que afectan sobre dicha columna. Lo que hace internamente es obtener todas las restricciones sobre cada una de las columnas y todas las restricciones definidas en la base de datos Parranderos. Al final, se retorna la información que corresponde al nombre y la tabla dada, así como las restricciones que afectan sobre la columna (bien sea porque es una restricción propiamente definida sobre la columna o porque existe alguna otra columna con una restricción que la implica, como es el caso de las *foreign key*). Para obtener el número de restricciones se emplea el comando de agrupación con **group by** en base al nombre de la tabla y de la columna y se realiza la función de agregación *count* sobre los nombres de las restricciones relacionados a la columna.

*Etapa 2 de la solución*

Con los resultados obtenidos, se procede a producir el Script de forma semi automática copiando la información de las tuplas y haciendo algunos cambios sobre los comandos recibidos. Estos cambios se consideran dentro de lo razonable para un cálculo semiautomático de la información como el que se exige, siendo que son pocos y de carácter sencillo. Los cambios realizados sobre el script se muestran en la siguiente imagen:

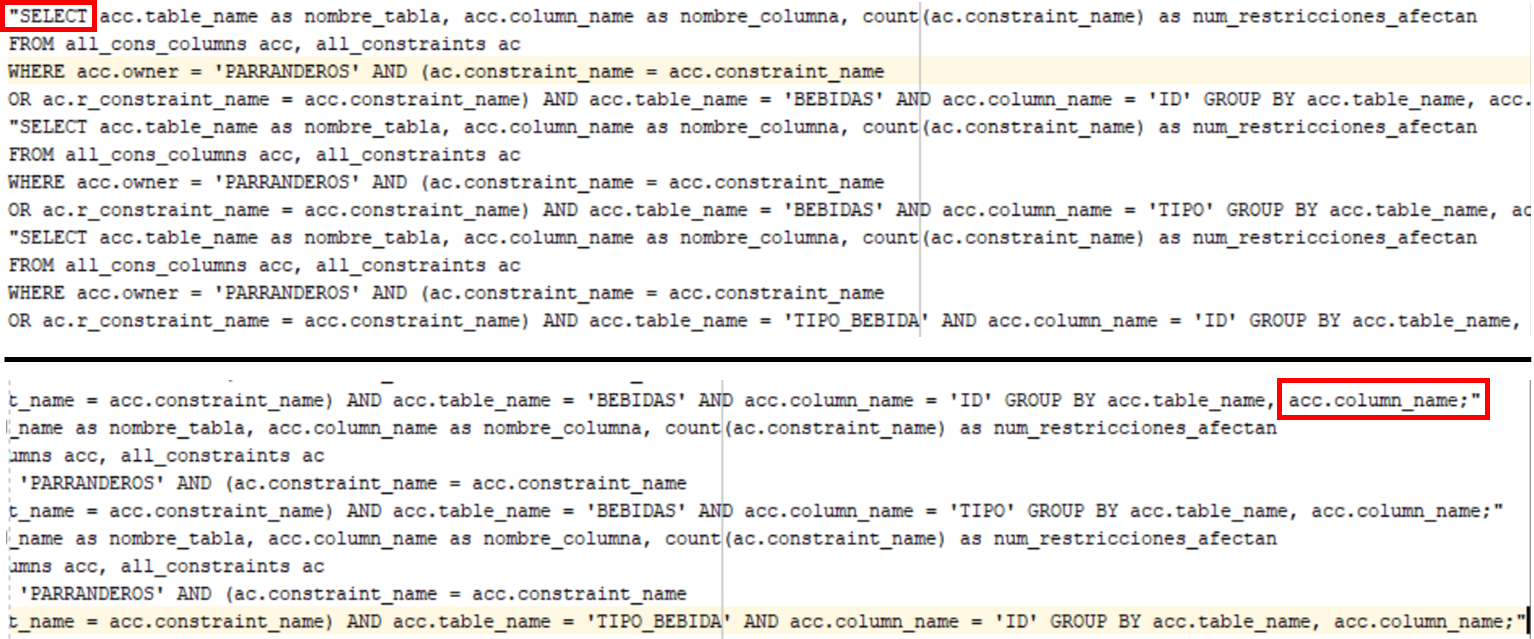


Figura 12. Script original sin cambios. Recibe todas las instrucciones como Strings y con un punto y coma (;) al final de cada instrucción que se encarga de encontrar la información para una columna dada.

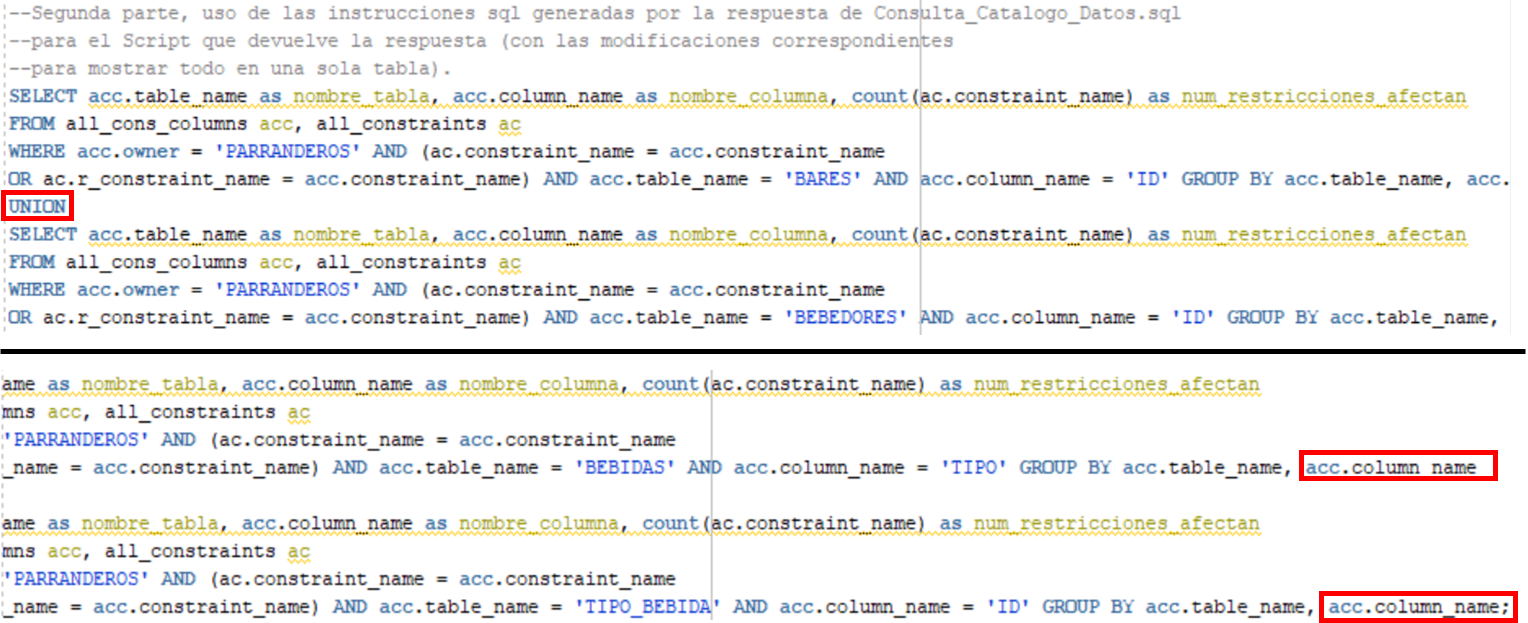


Figura 13. Script final con los cambios. Se eliminan todas las comillas para convertir los Strings en fragmentos de código, se agrega el comando **union** para presentar todos los resultados en una única tabla, y se elimina el punto y coma (;) de cada instrucción exceptuando por la última instrucción obtenida de la parte 1.

Al final, una vez se han realizado todos los cambios se obtiene la información exigida para la consulta. Los resultados obtenidos por el Script se muestran en la siguiente imagen:

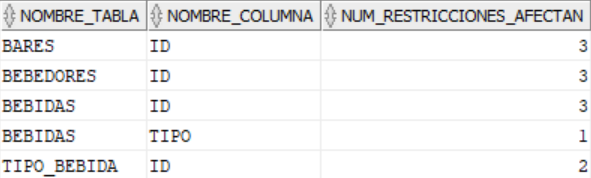


Figura 14. Resultado obtenido a partir de la solución implementada para la segunda parte del ejercicio 5.