

**Instituto Tecnológico de Costa Rica**

**Sede Central**

**Escuela de Ingeniería en Computación**

**Base de Datos I**

**Grupo 01**

**Proyecto programado #1**

**Interpretador Algebra Relacional**

**Profesor: William Mata Rodríguez**

**Estudiantes:**

**Kristin Alvarado González 2015096407**

**Gerardo Monge Corella 2015145533**

**II Semestre 2017**

# Enunciado del proyecto

# PROYECTO 1 INTÉRPRETE DE ÁLGEBRA RELACIONAL

Un componente fundamental en los modelos de datos es el lenguaje de manipulación, o lenguaje de consulta, para la extracción y actualización de los datos.

Entre estos lenguajes de consulta están el álgebra relacional y el cálculo relacional definidos por Codd (1971). Ambos lenguajes son teóricos y formales. Se han utilizado como base para desarrollar otros lenguajes de manipulación de datos en el modelo relacional como el SQL.

Este proyecto consiste en desarrollar un intérprete de algunas operaciones del lenguaje álgebra relacional, ofreciendo también las instrucciones de SQL equivalentes y los datos obtenidos con la operación.

Durante el curso estudiamos conceptos de SQL desde el punto de vista del estándar de la ISO. Al usar un manejador de base de datos (DBMS o SABD) de un fabricante particular, podemos encontrar algunas diferencias pues estos han modificado parte del estándar y han agregado sus propias funcionalidades. En tales casos hay que consultar la documentación respectiva para adaptarse a un DBMS específico.

El DBMS a utilizar en el proyecto es Microsoft SQL Server o MySQL. Las demás herramientas que necesiten para desarrollar esta aplicación Ustedes las pueden seleccionar.

Para interactuar con el usuario la aplicación debe usar una GUI (Graphical User Interface) diseñada por Ustedes que cumpla con los requerimientos solicitados.

El interpretador solo hace una operación del álgebra a la vez, no hay operaciones anidadas.

El desarrollo del proyecto es en equipos de 3 estudiantes máximo, uno de ellos lo deben nombrar como el coordinador. Importante: las experiencias han demostrado que los proyectos en equipos que no han sido administrados adecuadamente van a fallar, así que en cuanto noten que se presentan problemas al respecto de inmediato trátenlo primeramente con los miembros del equipo, y de no resolver lo comunican al profesor.

Cualquier comunicación al profesor que vaya copiada a todos los miembros del equipo.

Buenas prácticas de la ingeniería de software: deben usar un software para administrar el desarrollo de proyectos en equipo el cual incluye un control de versiones (Git, otros).

## REQUERIMIENTOS DEL PROYECTO

Las operaciones del álgebra relacional que debe manejar el interpretador son:

* Selección
* Proyección generalizada
* Unión
* Diferencia de conjuntos
* Producto cartesiano
* Intersección
* División
* Renombrar un relación y sus atributos
* Concatenación (join)
* Concatenación natural (natural join)
* Agregación
* Agrupación

El usuario debe tener un mecanismo para seleccionar la operación que necesite. El programa debe quedarse pidiendo funciones al usuario hasta que él pida finalizar el programa.

En general para cada operación Usted va a encontrar en esta especificación los datos que se deben solicitar al usuario, las validaciones que se deben hacer y los resultados esperados.

En las validaciones algunos errores deben tener un manejo de errores personalizado, es decir, se envían al usuario los mensajes tales como se especifican aquí. Cuando no se indiquen mensajes personalizados el manejo de los errores puede hacerlo como Ustedes decidan.

## Operación: Selección

Datos solicitados: Tabla

Predicado (string)

Tabla resultante (se explica más adelante)

Validaciones:

* Tabla debe existir. Mensaje personalizado cuando no exista: ERROR: NO EXISTE LA TABLA nombre\_de\_tabla
* Otros errores Usted los maneja.

## Operación: proyección generalizada

Datos solicitados: Tabla

Expresión de la proyección generalizada (string) Tabla resultante

Validaciones:

* Tabla debe existir. Mensaje personalizado cuando no exista: ERROR: NO EXISTE LA TABLA nombre\_de\_tabla
* Otros errores Usted los maneja.

## Operación: Unión

Datos solicitados: Tabla 1

Tabla 2

Tabla resultante

Validaciones:

* Tablas deben existir. Mensaje personalizado cuando no existan: ERROR: NO EXISTE LA TABLA nombre\_de\_tabla
* Las tablas deben tener igual aridad. Mensaje personalizado: ERROR: TABLAS CON DIFERENTE ARIDAD. LA TABLA 1 TIENE ARIDAD “X” Y LA TABLA 2 TIENE ARIDAD “Y”
* Cada atributo respectivo debe tener el mismo dominio. Mensaje personalizado:

ERROR: DOMINIOS DIFERENTES. EL ATRIBUTO “X” TIENE “DOMINIO X” Y EL ATRIBUTO “Y” TIENE “DOMINIO “Y”.

* Otros errores Usted los maneja.

## Operación: Diferencia de conjuntos

Datos solicitados: Tabla 1

Tabla 2

Tabla resultante

Validaciones:

* Igual a la operación de unión.

## Operación: Producto cartesiano

Datos solicitados: Tabla 1

Tabla 2

Tabla resultante

Validaciones:

* Tablas deben existir. Mensaje personalizado cuando no existan: ERROR: NO EXISTE LA TABLA nombre\_de\_tabla
* Otros errores Usted los maneja.

## Operación: Intersección

Datos solicitados: Tabla 1

Tabla 2

Tabla resultante

Validaciones:

* Igual a la operación de unión.

## Operación: División

Datos solicitados: Tabla 1

Tabla 2

Tabla resultante

Validaciones:

* Tablas deben existir. Mensaje personalizado cuando no existan: ERROR: NO EXISTE LA TABLA nombre\_de\_tabla
* Todos los atributos de Tabla 2 deben estar en Tabla 1. Mensaje personalizado:

ERROR: EL ATRIBUTO “X” DE LA TABLA nombre\_de\_tabla1 NO ESTA EN LA TABLA nombre\_de\_tabla2.

* Otros errores Usted los maneja.

## Operación: Renombrar una relación y sus atributos

Físicamente en la base de datos tanto la tabla como los atributos siguen manteniendo el nombre de origen. El renombramiento es solo para efectos de conocer esos objetos con otro nombre en la corrida actual de esta aplicación.

Datos solicitados: Tabla

Nombres de atributos (string): Cada uno de los nombres nuevos de los atributos: el i-ésimo atributo dado aquí corresponde al i-ésimo atributo de la tabla.

Tabla resultante.

Validaciones:

* Tabla debe existir. Mensaje personalizado cuando no exista: ERROR: NO EXISTE LA TABLA nombre\_de\_tabla
* Por cada atributo en la tabla debe existir un nombre de atributo.

Mensaje de error personalizado:

ERROR: NO HAY CORRESPONDENCIA EN LA CANTIDAD DE

ATRIBUTOS. LA TABLA nombre\_de\_tabla TIENE “X” ATRIBUTOS Y SE ESTAN DANDO “Y” ATRIBUTOS.

* Otros errores Usted los maneja.

## Operación: concatenación (join)

Datos solicitados: Tabla 1.

Tabla 2.

Predicado (string): para unir las tablas. Tabla resultante.

Validaciones:

* Tablas deben existir. Mensaje personalizado cuando no existan: ERROR: NO EXISTE LA TABLA nombre\_de\_tabla
* Otros errores Usted los maneja.

## Operación: concatenación natural(natural join)

Datos solicitados: Tabla 1.

Tabla 2.

Tabla resultante.

Validaciones:

* Tablas deben existir. Mensaje personalizado cuando no existan: ERROR: NO EXISTE LA TABLA nombre\_de\_tabla
* Para hacer este tipo de join deben existir al menos dos nombres de atributos comunes (y dominios comunes) entre las dos tablas.

Mensaje personalizado de error:

ERROR: NO HAY ATRIBUTOS COMUNES

* Otros errores Usted los maneja.

## Operación: Agregación

Datos solicitados: Tabla.

Lista de operaciones de agregación (string). Tabla resultante.

Se permiten al menos las funciones básicas: sum, count, mix, max, avg. Validaciones:

* Tablas deben existir. Mensaje personalizado cuando no existan: ERROR: NO EXISTE LA TABLA nombre\_de\_tabla
* Otros errores Usted los maneja.

## Operación: Agrupación

Datos solicitados: Tabla.

Lista de atributos que indican las agrupaciones a realizar (string).

Lista de operaciones de agregación que se van a aplicar a los grupos (string).

Tabla resultante.

Se permiten al menos las funciones básicas: sum, count, mix, max, avg. Validaciones:

* Tablas deben existir. Mensaje personalizado cuando no existan: ERROR: NO EXISTE LA TABLA nombre\_de\_tabla
* Otros errores Usted los maneja.

-

## Tabla resultante.

El usuario puede dejar en blanco este dato solicitado o dar un nombre de tabla. En caso de dar un nombre significa que necesita guardar los resultados en una tabla temporal para usos posteriores en otras operaciones. Note que este guardado es la operación de asignación.

Estas tablas donde se guardan los resultados son temporales, van a existir solamente cuando esté corriendo la aplicación. Cuando la aplicación termina las tablas temporales deben eliminarse de la base de datos.

No se permite dar el nombre de una tabla que sea permanente de la base de datos. Validar esta condición, mensaje personalizado de error:

ERROR: NO SE PUEDE DEJAR EL RESULTADO EN UNA TABLA PERMANENTE DE LA BASE DE DATOS.

## Resultados.

Después de validar los datos solicitados en cada operación se deben mostrar tres resultados:

* La instrucción del álgebra relacional: siempre debe estar visible al usuario.
* La instrucción equivalente de SQL: siempre debe estar visible al usuario.
* La tabla resultante: el programa debe ofrecer una forma para que el usuario pueda desplazarse por los resultados: se despliega una línea por cada tupla.

Ejemplo usando la base de datos Banco: determinar los nombres de los clientes del banco excepto los que vivan en la ciudad León.

Esta consulta se debe hacer en dos operaciones: una selección y una proyección.

Primera operación: selección: Tabla: cliente

Predicado: ciudad\_cliente <> ‘León’ Tabla resultante: r1

Instrucción de álgebra relacional:

r1  σ ciudad\_cliente <> ‘León’ (cliente) instrucción SQL:

select \* from cliente where ciudad\_cliente <> ‘León’ Tabla resultante: r1

Se manejan barras de scroll horizontal y vertical para navegar por el resultado

|  |  |  |
| --- | --- | --- |
| **nombre\_cliente** | **calle\_cliente** | **ciudad\_cliente** |
| Abril | Preciados | Valsaín |
| Amo | Embajadores | Arganzuela |

Segunda operación: proyección generalizada Tabla: r1

Expresión: nombre\_cliente Tabla resultante:

Instrucción de álgebra relacional:

**∏** nombre\_cliente (r1) instrucción SQL:

select distinct nombre\_cliente from r1

Tabla resultante:

… datos …

## Funciones adicionales del intérprete:

**Ver tablas de la base de datos**

Nombre de la tabla1

Atributo1 Dominio Propiedades (PK, FK, etc.) (un atributo por línea)

…

AtributoN Dominio Propiedades (PK, FK, etc.)

…

Nombre de la tablaN

Atributo1 Dominio Propiedades (PK, FK, etc.)

…

AtributoN Dominio Propiedades (PK, FK, etc.)

## Ver tablas temporales

Nombre de la tabla1

Atributo1 Propiedades (dominio, PK, FK, etc.) (un atributo por línea)

…

AtributoN Propiedades (dominio, PK, FK, etc.)

…

Nombre de la tablaN

Atributo1 Propiedades (dominio, PK, FK, etc.)

…

AtributoN Propiedades (dominio, PK, FK, etc.)

## Ver referencia cruzada atributos / tablas

Esta consulta permite ver las tablas de la base de datos en donde aparece cada atributo según el diccionario de datos.

Atributo1 Propiedades (dominio, PK, FK, etc.) Tabla1

… TablaN

…

AtributoN Propiedades (dominio, PK, FK, etc.) Tabla1

… TablaN

## Ayuda

Esta opción la usaremos para que el usuario pueda ver el Manual de Usuario (explicado más adelante) preparado en la documentación.

## Acerca de

Esta opción la usaremos para desplegar una ventana con información “Acerca de la aplicación” donde colocaremos al menos los datos del nombre de la aplicación, la versión, la fecha de creación y los autores.

## Salir

Esta opción se usa para salir del programa.

En este momento las tablas temporales se deben borrar.

Puede agregar cualquier funcionalidad que Usted considere va a mejorar el producto.

Al nombre de la base de datos asígnele: bdproy1 Al nombre del esquema asígnele: proy1

Crear dos usuarios:

**dbaproy1**: va a tener la responsabilidad de DBA. Como DBA tiene todos los privilegios. Con este usuario se van a crear los objetos de la base de datos. Para pruebas puede usar la base de datos Banco que se ha trabajado en clases.

**usproy1**: va a ser un usuario regular de la base de datos. Debe tener estos privilegios de acceso:

* Lectura para los datos de las tablas permanentes.
* CRUD para las tablas temporales.

# DOCUMENTACIÓN DEL PROYECTO

## REQUISITO PARA REVISAR EL PROYECTO

**El requisito consiste en presentar la documentación del proyecto indicada en esta sección.**

**La nota de la documentación del proyecto sirve para aceptar o rechazar el proyecto: se revisan los proyectos que cumplan con este requisito en un 90% o más**.

Enviar vía tecDigital, sección EVALUACIONES / PROYECTOS, una carpeta comprimida (.rar, .zip, etc.) de nombre **proyecto1** que contenga las siguientes partes:

* Parte 1: Documentación del proyecto (**nombre: documentación\_interpretador\_álgebra\_relacional.PDF**).
  + Portada.
  + Contenido. (3 p)
  + Enunciado completo del proyecto. (2 p)
  + Temas investigados para desarrollar el proyecto: material no estudiado en el curso. Por cada uno de estos temas debe poner el marco teórico incluyendo de qué trata y el detalle de cómo lo usaron en el proyecto.
    - Temas investigados. (15 p): de estos temas excluya el lenguaje de programación pero incluya las metodologías de programación de bases de datos.
    - Software para manejo de versiones. (15 p)
  + Conclusiones del trabajo: (15 p)
    - Problemas encontrados y soluciones a los mismos.
    - Aprendizajes obtenidos.
  + Rúbrica de evaluación y análisis de resultados (PONGA LA HOJA DE LA RÚBRICA EN PÁGINA NUEVA DE TAL FORMA QUE LOS CONCEPTOS QUEDEN EN UNA MISMA PÁGINA). (15 p)
    - Tome la rúbrica de evaluación y por cada concepto calificado Usted debe indicar el % de avance y el análisis de resultados
      * 100: Totalmente desarrollado. No hace falta análisis excepto que requiera hacer alguna observación.
      * 80: Desarrollado parcialmente, un 80% (el % que corresponda). En el análisis indicar: ¿qué hace?, ¿qué falta?, ¿ por qué no se completó

?

* + - * 0: No desarrollado. En el análisis indicar el motivo.
    - Partes que desarrolló adicionales a los requerimientos.
  + Manual de usuario

(**nombre: manual\_de\_usuario\_interpretador\_álgebra\_relacional. PDF**). (35 p)

Es un documento de comunicación técnica utilizado para guiar a las personas que usan el software. Explica paso a paso cómo usar cada una de las funcionalidades del programa. Apóyese en imágenes, capturas de pantallas, menús, diagramas y los aspectos que considere van a servir como una guía útil para que el usuario pueda usar el programa.

Este manual será usado en la revisión del proyecto para hacer las pruebas desde el punto de vista funcional.

* Parte 2: Fuentes y otros objetos necesarios para ejecutar la aplicación.

En la semana 10 se revisará un prototipo de la GUI: modelos de pantallas.

**IMPORTANTE: CONOCIMIENTO DE LA SOLUCIÓN PRESENTADA**. En la

revisión del trabajo, los estudiantes deben demostrar un completo dominio de la solución que implementaron, tanto desde el punto de vista técnico (uso de herramientas) como de la funcionalidad del proyecto. La revisión se puede hacer individualmente o en grupos, examinando la solución o temas específicos aplicados en el proyecto.

# Temas Investigados:

## Procesar sentencias de SQL en JDBC

Para esto generalmente se siguen los siguientes pasos:

1. Estableciendo una conexión.
2. Crear una instrucción.
3. Ejecute la consulta.
4. Procese el objeto de resultado.
5. Cierra la conexión.
6. Estableciendo conexiones

En primer lugar, establezca una conexión con el origen de datos que desea utilizar. Un origen de datos puede ser un DBMS, un sistema de archivos heredado o alguna otra fuente de datos con un controlador JDBC correspondiente. Esta conexión está representada por un objeto *Connection*.

Primero se necesita establecer una conexión con la fuente de datos que se desea usar. Un origen de los datos puede ser un BDMS. Normalmente, una aplicación JDBC se conecta a un origen de datos de destino mediante una de dos clases:

* **DriverManager**: esta clase totalmente implementada conecta una aplicación a un origen de datos, que se especifica mediante una URL de base de datos.
* **DataSource**: esta interfaz permite que los detalles sobre el origen de datos subyacente sean transparentes a su aplicación. Las propiedades de un objeto *DataSource* se establecen de manera que representa un origen de datos determinado.

1. Crear instrucciones

Una instrucción es una interfaz que representa una instrucción SQL. Se ejecutan objetos de instrucción y generan objetos de resultado, que es una tabla de datos que representa un conjunto de resultados de base de datos. Se necesita un objeto Connection para crear un objeto Statement.

Existen tres tipos diferentes de declaraciones:

* **Instrucción**: se utiliza para implementar instrucciones SQL simples sin parámetros.
* **PreparedStatement**: Se utiliza para precompilar sentencias SQL que pueden contener parámetros de entrada.
* **CallableStatement**: Se utiliza para ejecutar procedimientos almacenados que pueden contener parámetros de entrada y salida.

1. Procesamiento de objetos de resultado

Se accede a los datos en un objeto de resultado mediante un cursor. Note que este cursor no es un cursor de base de datos. Este cursor es un puntero que apunta a una fila de datos en el objeto de resultado. Inicialmente, el cursor se coloca antes de la primera fila. Se llaman varios métodos definidos en el objeto de conjunto para mover el cursor.

1. Cerrar conexiones

Cuando haya terminado de utilizar una instrucción, llame a la instrucción Method.close para liberar inmediatamente los recursos que está utilizando. Cuando se llama a este método, sus objetos de resultado se cierran.

## Información sobre java.sql

interfaz pública SQLData

El objeto Class de una clase que implementa la interfaz SQLData se introducirá en el mapa de tipo del objeto de conexión apropiado junto con el nombre SQL del definidor para el que se trata de una asignación personalizada.

Por lo general, una implementación de SQLData definirá un campo para cada atributo de un tipo estructurado SQL o un único campo para un tipo distinto de SQL. Cuando se recupera el definido de un origen de datos con el método GetObject, se asignará como instancia de esta clase. Un programador puede operar en esta instancia de clase igual que en cualquier otro objeto del lenguaje de programación Java y, a continuación, almacenar los cambios realizados en él llamando al método PreparedStatement setObject, que lo mapeará de nuevo al tipo SQL.

Se espera que la implementación de la clase para un mapeo personalizado será realizada por una herramienta. En una implementación típica, el programador simplemente proporcionaría el nombre del SQL definido, el nombre de la clase a la que se está asignando, y los nombres de los campos a los que se va a mapear cada uno de los atributos del definidor. La herramienta usará esta información para implementar los métodos SQLData readSQL y SQLData. writeSQL. El método readSQL llama a los métodos SQLInput apropiados para leer cada atributo de un objeto SQLInput, y el método writeSQL llama a métodos SQLOutput para escribir cada atributo de nuevo en el origen de datos a través de un objeto SQLOutput.

Un programador de aplicaciones normalmente no llamará a los métodos SQLData directamente, y los métodos SQLInput y SQLOutput se denominan internamente por métodos SQLData, no por código de aplicación.

## Patrón vista controlador

El modelo vista controlador consiste en un modelo con varias vistas y varios controladores, las vistas se relacionan con los controladores, estas vistas son la interfaz gráfica.

El modelo vista controlador con Java Swing.

El modelo lo hace el desarrollador, las vistas son las interfaces graficas con las que interactúa el usuario y el controlador, es como se manejan los eventos que el usuario selecciono en la interfaz, haciendo click o escribiendo en algún área de texto, etc.

En el modelo están las operaciones que se van a efectuar.

## Software controlador de versiones

Git es un sistema de control de revisiones rápido, escalable y distribuido con un conjunto de comandos inusualmente enriquecido que proporciona operaciones de alto nivel y acceso completo a internos.

1. Primero se debe configurar el usuario y el correo de la persona que utiliza git.
2. Luego se debe crear el repositorio en la computadora con el comando *git init*
3. Después de agregan los archivos que estén en esa carpeta con el comando *git add .* para agregar a git todo lo que este en la carpeta.
4. Despues se debe hacer el commit, esto serían los comentarios de los cambios realizados a los archivos del repositorio, esto se hace con el comando *git commit –m “<mensaje de actualización>”*
5. Al usar github se puede crear un remote, que es un acceso remoto a un proyecto en github y editar los archivos desde git en la computadora, esto se hace con el comando *git remote add <nombre del remote> <url del proyecto en github u otra plataforma de git>*
6. Por ultimo se hace el push para agregar a github los archivos de git, esto es con el comando *git push*
7. En caso de tener archivos que no estén en el repositorio de la computadora se debe hacer primero un pull con el comando *git pull*

Para poder ver el estado del proyecto, si hay archivos que no se han obtenido antes de hacer el push, se pude usar el comando *git status*, este muestra si hay archivos que necesitan ser agregados o ya se agregaron pero no se han hecho el commit.

# Conclusiones del trabajo:

## Problemas encontrados y soluciones

En primera instancia el problema que se presento fue el aprender a utilizar el software controlador de versiones, para este proyecto se utiliza git y el proyecto se sube a github.

Se presentaron algunos problemas con los pull y push ya que habían dos personas trabajando en el mismo proyecto al mismo tiempo, lo que hacía que hubieran errores en el repositorio y tener que cambiar el orden de los comandos ya estipulados, tener más cuidado con los push y pull, usando el *git status*.

Otro problema encontrado fue con el manejo de la base de datos, que ambas computadoras tuvieran la misma base de datos y que esta se encontrara en el mismo puerto, al inicio no sabía el problema pero se descubrió que escuchaban puertos diferentes, para este proyecto se usa el puerto 1433.

Un problema que se presenta constantemente en los trabajos en grupo es el orden y la congruencia, por esto se optó por el uso de patrones como el patrón vista controlador y trabajar el proyecto con modularidad, se creó una clase abstracta de expresión racional, luego se crearon clases distintas para cada uno de las operaciones:

* Selección
* Proyección generalizada
* Unión
* Diferencia de conjuntos
* Producto cartesiano
* Intersección
* División
* Renombrar un relación y sus atributos
* Concatenación
* Concatenación natural
* Agregación
* Agrupación

Estas clases heredan de la clase ExpresionRacional y usan sus métodos abstractos de manera diferente para cada uno.

Aparte de estas clases, esta las de Controller, DataModel y Factory. Controller controla la base de datos, el flujo de información de esta, la conexión, etc. DataModel se encarga de contrlar las tablas del programa.

Factory se encarga de controlar que operación el usuario desea y llamar su respectiva clase. Por ultimo esta cada interfaz, este es la vista del modelo.

## Lecciones aprendidas

Con este proyecto aprendimos a usar mas la clase SQL de Java y el driver JDBC, tanto como sintaxis de transact SQL.

También la importancia del software controlador de versiones cuando se trabaja en grupo.

Otra lección aprendida es dividir el trabajo en trabajos más pequeños para qué es más fácil resolver varios problemas pequeños que un problema muy grande y difícil.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Concepto** | **Puntos** | **Puntos obte- nidos** | **% de Avance**  **100/x/0** | **Análisis de resultados** |
| Resultado de selección | 2 |  |  |  |
| Validación | 1 |  |  |  |
| Resultado de proyección generalizada | 2 |  |  |  |
| Validación | 1 |  |  |  |
| Resultado de unión | 2 |  |  |  |
| Validación | 4 |  |  |  |
| Resultado diferencia de conjuntos | 2 |  |  |  |
| Validación | 4 |  |  |  |
| Resultado de producto cartesiano | 2 |  |  |  |
| Validación | 1 |  |  |  |
| Resultado de Intersección | 2 |  |  |  |
| Validación | 4 |  |  |  |
| Resultado de división | 2 |  |  |  |
| Validación | 4 |  |  |  |
| Resultado de renombrar una relación y atributos | 7 |  |  |  |
| Validación | 7 |  |  |  |
| Resultado de concatenación (join) | 2 |  |  |  |
| Validación | 1 |  |  |  |
| Resultado de concatenación natural (natural join) | 5 |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Validación | 5 |  |  |  |
| Resultado de agregación | 2 |  |  |  |
| Validación | 1 |  |  |  |
| Resultado de agrupación | 2 |  |  |  |
| Validación | 1 |  |  |  |
| Resultado mostrar expresión de álgebra relacional | 2 |  |  |  |
| Resultado mostrar instrucción SQL | 3 |  |  |  |
| Manejo de la tabla de datos resultante (incluye scroll) | 5 |  |  |  |
| Ver tablas de la base de datos | 5 |  |  |  |
| Ver tablas temporales | 5 |  |  |  |
| Ver referencia cruzada atributos / tablas | 5 |  |  |  |
| Borrar tablas temporales | 2 |  |  |  |
| Creación de usuarios (DBA, usuario) | 2 |  |  |  |
| Ayuda (manual de usuario) | 5 |  |  |  |
| Acerca de / Salir | 0 |  |  |  |
| **TOTAL** | **100** |  |  |  |
| Partes desarrolladas adicionalmente |  |  |  |  |

# Bibliografía:

1. Package java.sql, encontrado en: https://docs.oracle.com/javase/7/docs/api/java/sql/package-summary.html
2. Processing SQL Statements with JDBC, encontrado en: https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html#establishing\_connections
3. Estructura de las Aplicaciones Orientadas a Objetos El patrón Modelo-Vista-Controlador (MVC), Juan Pavón Mestras, España, 2008.
4. Git Reference, encontrado en: https://git-scm.com/docs