



Benemérita Universidad Autónoma de Puebla

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

“DESCRIPTOR DE ARCHIVOS”

González Muñoz José María 201705405

jose.gonzalezutt@gmail.com

López Hernández Mark Anthony 201732531

sdwark0925@gmail.com

Patiño Martínez Alexis Yair 201735036

alex.mtz873@gmail.com

Pedraza Celón Ian Yael 201719516

ianpedrazacelon@gmail.com

Villalobos Vicente Franco Alberto 201746289

frankolian.fv@gmail.com

Profesora:

M.C. ALMA DELIA AMBROSIO VAZQUEZ

INTRODUCCIÓN

Un DBMS (Data Base Management System), son las siglas en inglés para los Sistemas de Gestión de Bases de Datos (SGBD).

Es un software que controla la organización, almacenamiento, recuperación, seguridad e integridad de los datos en una base de datos. Acepta solicitudes de la aplicación y ordena al sistema la adecuada transferencia de datos requeridos, para manejar un buen llamado a los datos se ocupa del algebra relacional.

El **algebra relacional** es un conjunto de operaciones que describen paso a paso cómo computar una respuesta sobre las relaciones, siendo de tipo declarativo. Estas operaciones se usan como una representación intermedia de una consulta a una base de datos.

Se asocia unívocamente los nombres de los campos de una relación con los valores de una instanciación de la misma, formando a si las tuplas, es decir una fila de una tabla relacional. Trabajando con operaciones como lo son:

- Selección.
- Proyección.
- Producto cartesiano.
- Unión.
- Intersección.

Los DBMS pueden trabajar con leguajes de programación tradicionales como COBOL, C, JAVA, etc.

Una pieza fundamental considerable es el Descriptor de Archivos, considerando que la información está organizada de la misma manera en los archivos, esta información reunida sobre temas particulares debe de estar estructurada de forma coherente. Cada campo del descriptor contiene el nombre del campo que lo describe.

El programa principal use el descriptor de archivos para acceder a la información por nombre, usando el nombre del campo al que se dirige, y no por la posición.

Cuando un archivo específico se abre (CLIENTES) la función Cargador-Descriptor intenta traer el tipo de archivo dentro de la memoria, verificando si el archivo existe y si puede ser abierto. Una vez abierto el archivo estará disponible para ser procesado: editar, eliminar, añadir información, etc.

Ejemplo: Suponiendo que queremos saber que clientes están activos y nos deben más de \$15,000, solicitamos una lista específica con el nombre, dirección y el monto en deuda.

Tabla 1: Ejemplo consulta

CONSULA CLIENTES	
* Archivo clientes *	Verificado que el Archivo existe, esperando entradas.
¿cuál es la pregunta?	
Clientes= activos con deudas>\$15,000	Solicitud escrita por el usuario.
¿Qué quieres saber?	
NOMBRE, DIRECCION, CANTIDAD	Propiedad que se imprimirán.
¿Hay algún orden?	
NOMBRE	Ordenamiento para seguir

Tabla 2: Resultado ejemplo consulta

NOMBRE	DIRECCIÓN	CANTIDAD
Álvarez Juan	5 de mayo 18, Ciudad de México	\$16,050
iguel Ángel	Reforma 21, Ciudad de México	\$18,500
...

Funcionamiento del Consultor

La petición es analizada por un pequeño analizador de texto y una tabla se forma en la memoria, la cual contiene la misma información que solicita la petición, pero de forma condensada. También una lista con las propiedades que cumplen con la petición es convertida en otra tabla. A cada registro almacenado se evalúa: si es cierto, el registro se imprime o guarda temporalmente, si es falso, no imprime o envía. El proceso termina cuando ya no hay más registros por evaluar.

Nosotros como equipo tratamos de recrear la funcionalidad de todo este proceso mediante Excel, quedando de esta manera:

Usamos el archivo EMPLOYEES que contiene los campos de los trabajadores:

Tabla 3: Descripción tabla Employees

Identificador	EMPLOYEE_ID
nombre	FIRST_NAME
Apellido	LAST_NAME
Correo electrónico	EMAIL
Número de teléfono	PHONE_NUMBER
Fecha de contratación	HIRE_DATE
Identificador de trabajo	JOB_ID
Salario	SALARY
Comisiones	COMMISSIONS_PCT
Manager	MANAGER_ID

EMPLOYEES										
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17/06/1987	AD_PRES	24000			90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21/09/1989	AD_VP	17000		100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13/01/1993	AD_VP	17000		100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03/01/1990	IT_PROG	9000		102	60
104	Bruce	Ernst	BERNST	590.423.4568	21/05/1991	IT_PROG	6000		103	60
105	David	Austin	DAUSTIN	590.423.4569	25/06/1997	IT_PROG	4800		103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	05/02/1998	IT_PROG	4800		103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07/02/1999	IT_PROG	4200		103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17/08/1994	FI_MGR	12000		101	100
109	Daniel	Faviet	DFAVET	515.124.4169	16/08/1994	FI_ACCOUNT	9000		108	100
110	John	Chen	JCHEN	515.124.4269	28/09/1997	FI_ACCOUNT	8200		108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	30/09/1997	FI_ACCOUNT	7700		108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	07/03/1998	FI_ACCOUNT	7800		108	100
113	Luis	Popp	LPOPP	515.124.4567	07/12/1999	FI_ACCOUNT	6900		108	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	07/12/1994	PU_MAN	11000		100	30
115	Alexander	Khoo	AKHOO	515.127.4562	18/05/1995	PU_CLERK	3100		114	30
116	Shelli	Baida	SBAIDA	515.127.4563	24/12/1997	PU_CLERK	2900		114	30
117	Sigal	Tobias	STOBIAS	515.127.4564	24/07/1997	PU_CLERK	2800		114	30
118	Guy	Himuro	GHIMURO	515.127.4565	15/11/1998	PU_CLERK	2600		114	30
119	Karen	Colmenares	KCOLMENA	515.127.4566	10/08/1999	PU_CLERK	2500		114	30

Ilustración 1: Tabla employees

Se solicita que nuestro archivo EMPLOYEES se indique los trabajadores que tienen un salario entre \$10,000 y \$150,000

select *										
FROM EMPLOYEES										
where SALARY BETWEEN 10000 AND 15000;										
EMPLOYEES										
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
108	Nancy	Greenberg	NGREENBE	515.124.4569	17/08/1994	FI_MGR	12000		101	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	07/12/1994	PU_MAN	11000		100	30
145	John	Russell	JRUSSEL	011.44.1344.429268	01/10/1996	SA_MAN	14000	0.4	100	80
146	Karen	Partners	KPARTNER	011.44.1344.467268	05/01/1997	SA_MAN	13500	0.3	100	80
147	Alberto	Errazuriz	AERRAZUR	011.44.1344.429278	10/03/1997	SA_MAN	12000	0.3	100	80
148	Gerald	Cambraut	GCAMBRAU	011.44.1344.619268	15/10/1999	SA_MAN	11000	0.3	100	80
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29/01/2000	SA_MAN	10500	0.2	100	80
150	Peter	Tucker	PTUCKER	011.44.1344.129268	30/01/1997	SA_REP	10000	0.3	145	80
156	Janette	King	JKING	011.44.1345.429268	30/01/1996	SA_REP	10000	0.35	146	80
162	Clara	Vishney	CVISHNEY	011.44.1346.129268	11/11/1997	SA_REP	10500	0.25	147	80
168	Lisa	Ozer	LOZER	011.44.1343.929268	11/03/1997	SA_REP	11500	0.25	148	80
169	Harrison	Bloom	HBLOOM	011.44.1343.829268	23/03/1998	SA_REP	10000	0.2	148	80
174	Ellen	Abel	EABEL	011.44.1644.429267	11/02/1996	SA_REP	11000	0.3	149	80
201	Michael	Hartstein	MHARTSTE	515.123.5555	17/02/1996	MK_MAN	13000		100	20
204	Hermann	Baer	HBAER	515.123.8888	07/06/1994	PR_REP	10000		101	70
205	Shelley	Higgins	SHIGGINS	515.123.8080	07/06/1994	AC_MGR	12000		101	110

Ilustración 2: Tabla EMPLOYEES, con rango de salarios y datos completos de los empleados

Una vez obtenidos todos los trabajadores que tiene un rango de salario entre entre \$10,000 y \$150,000, queremos que haga la tabla donde se muestren los nombres y el salario correspondiente

select FIRST_NAME, SALARY	
FROM EMPLOYEES	
where SALARY BETWEEN 10000 AND 15000;	
EMPLOYEES	
FIRST_NAME	SALARY
Nancy	12000
Den	11000
John	14000
Karen	13500
Alberto	12000
Gerald	11000
Eleni	10500
Peter	10000
Janette	10000
Clara	10500
Lisa	11500
Harrison	10000
Ellen	11000
Michael	13000
Hermann	10000
Shelley	12000

Ilustración 3: Tabla EMPLOYEES, mostrando solo el nombre y el salario

Desarrollo del software.

El software fue desarrollado en java y consiste en cargar dos archivos, el descriptor de archivos y la tabla empleados, a partir de archivos de texto plano. Estos datos se mandan a una interfaz donde se muestran en forma de tabla, y a través de un panel podemos cambiar los rangos de la consulta **between** del campo **salary** de la tabla **employees**, así como los campos que se mostrarán en la proyección.

A continuación, se explica a detalle el funcionamiento del software.

1. Archivos

El **descriptor de archivos** debe contener la siguiente información con el formato indicado:

nombre_atributo, posición_inicial, longitud, tipo_dato, valor_inicial, valor_minimo, valor_maximo

nombre_tabla: El nombre del atributo que se describirá en la línea

posición_inicial: La posición del primer byte del atributo dentro del archivo de la tabla

longitud: La longitud que tiene el atributo en bytes.

tipo_dato: El tipo de dato que admite el atributo

valor_inicial: El valor inicial que puede tomar este atributo

valor_minimo: El valor mínimo posible que puede tomar este atributo

valor_maximo: El valor máximo que puede tomar el atributo

El descriptor de archivos para la tabla employees es el siguiente:

```

EMPLOYEE_ID,0,6,NUMBER,000001,000001,100000

FIRST_NAME,6,20,VARCHAR, , ,

LAST_NAME,26,25,VARCHAR, , ,

EMAIL,51,25,VARCHAR, , ,

PHONE_NUMBER,76,20,VARCHAR, , ,

HIRE_DATE,96,8,DATE,01/01/87,01/01/87,18/02/2020

JOB_ID,104,10,VARCHAR, , ,

SALARY,114,8,NUMBER,00000001,00000001,10000000

COMMISSION_PCT,122,3,NUMBER,0.0,0.0,1.0

MANAGER_ID,125,6,NUMBER,000001,000001,100000

DEPARTMENT_ID,131,6,NUMBER,000001,000001,100000

```

Ilustración 4: Descriptor de archivos

El archivo que contiene la **tabla** tiene que estar en función del descriptor. La tabla empleados dentro del archivo de texto plano, se ve de la siguiente forma:

000100	Steven	King	SKING	515.123.456717/06/87	AD_PRES000240000000000000000090
000101	Neena	Kochhar	NKOCHHAR	515.123.456821/09/89	AD_VP000170000000001000000090
000102	Lex	De Haan	LDEHAAN	515.123.456913/01/93	AD_VP000170000000001000000090
000103	Alexander	Hunold	AHUNOLD	590.423.456703/01/90	IT_PROG00009000000000102000060
000104	Bruce	Ernst	BERNST	590.423.456821/05/91	IT_PROG00006000000000103000060
000105	David	Austin	DAUSTIN	590.423.456925/06/97	IT_PROG00004800000000103000060
000106	Valli	Pataballa	VPATABAL	590.423.456005/02/98	IT_PROG00004800000000103000060
000107	Diana	Lorentz	DLORENTZ	590.423.556707/02/99	IT_PROG00004200000000103000060
000108	Nancy	Greenberg	NGREENBE	515.124.456917/08/94	FI_MGR00012000000000101000100
000109	Daniel	Faviet	DFAVIET	515.124.416916/08/94	FI_ACCOUNT00009000000000108000100
000110	John	Chen	JCHEN	515.124.426928/09/97	FI_ACCOUNT00008200000000108000100
000111	Ismael	Sciarra	ISCIARRA	515.124.436930/09/97	FI_ACCOUNT00007700000000108000100
000112	Jose Manuel	Urman	JMURMAN	515.124.446907/03/98	FI_ACCOUNT00007800000000108000100

Ilustración 5: Archivo tabla employees

Los archivos están contenidos en el directorio **data**, dentro de la raíz

2. Planeación: Diagrama de Clases

El diagrama de clase de la aplicación se muestra a continuación. En el Punto cuatro se explica a detalle el funcionamiento de cada clase a detalle y como contribuye al funcionamiento del software.

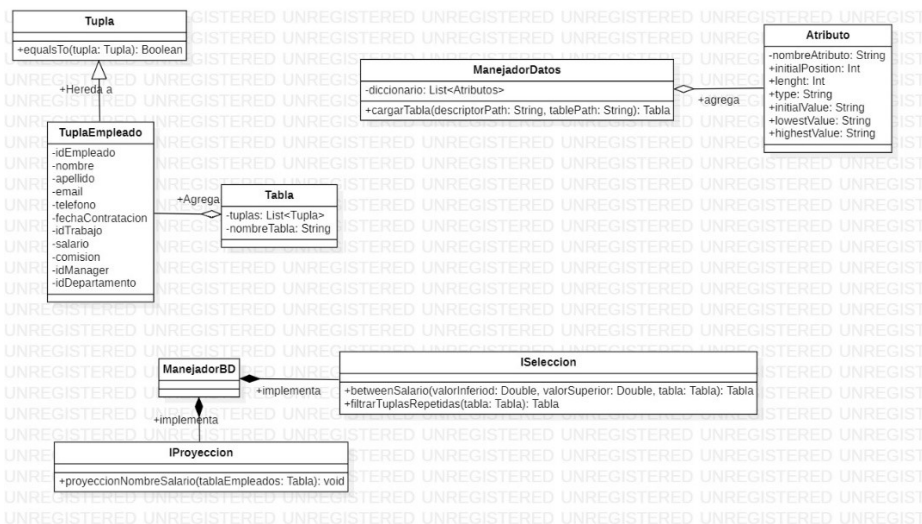


Ilustración 6: Diagrama de clases

3. Arquitectura

La arquitectura de la aplicación está basada en un modelo Model View View Model (MVVM), separando por un lado la capa de la vista, la capa del modelado de datos y la capa de procesamiento de datos, modularizando cada parte del software y haciéndola trabajar en conjunto.

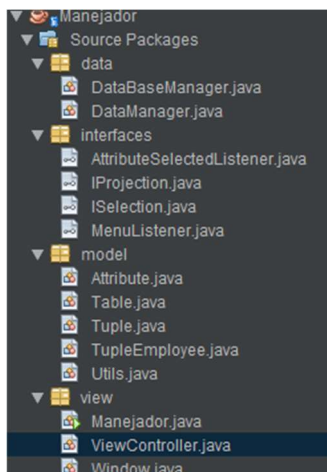


Ilustración 7: Arquitectura de la aplicación

4. Descripción de clases

Tuple:

Se tiene una clase ***Tuple***, que ayuda a hacer más modular la codificación. Esta tupla heredarà a todos los distintos tipos de tupla que se quieran crear, y funcionará de igual manera con el resto de los métodos, porque todos serán objetos de tipo tupla.

Tupla
+equalsTo(tupla: Tupla): Boolean +getAttrFromString(String attr): String +toArrayString(): List<String>

Ilustración 8: Clase Tuple

Contiene tres métodos:

equalsTo:

Compara la tupla con otra, para regresar si son iguales o no. Es utilizado cuando filtramos que no haya tuplas repetidas.

getAttrFromString:

Obtiene el atributo de la tupla a partir de un *String* con el nombre del atributo. Utilizado para obtener los datos para la proyección a partir de la lista de atributos seleccionados en la vista.

toArrayString:

Convierte toda la tupla a un arreglo de *String* para mandar los datos a la vista.

TupleEmployee:

La clase ***TupleEmployee***, modela al empleado dentro del programa y hereda de la clase ***Tuple***.

TuplaEmpleado
-idEmpleado -nombre -apellido -email -telefono -fechaContratacion -idTrabajo -salario -comision -idManager -idDepartamento

Ilustración 9: Clase TupleEmployee

Table:

Modela una tabla de datos, continuando con la modularidad contiene una **lista de tuplas**, que pueden ser de cualquier tipo que herede de la clase tupla. Y también contiene el **nombre** de la tabla.

Tabla
-tuplas: List<Tupla> -nombreTabla: String

Ilustración 10: Clase Table

Attribute:

Se encarga de modelar los atributos leídos a partir del descriptor de archivos y que serán almacenados en el diccionario de datos. Como atributos de la clase tenemos lo mencionado del formato del descriptor de archivos.

Atributo
-nombreAtributo: String +initialPosition: Int +length: Int +type: String +initialValue: String +lowestValue: String +highestValue: String

Ilustración 11: Clase Attribute

DataManager:

Para el uso la lectura y almacenamiento de datos utilizamos la clase *DataManager*, que contine un diccionario de datos, y método que carga los datos del descriptor de archivos al diccionario y regresa una tabla con los datos de la tabla que se le manda por parámetro.

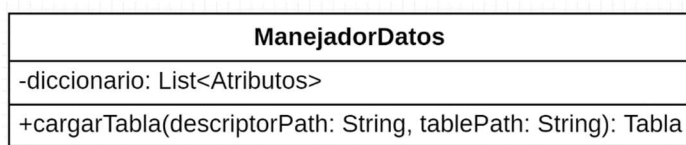


Ilustración 12: Clase DataManager

IProyeccion

Provee a la clase manejadora de la base de datos lo métodos necesarios para realizar la proyección.

projection:

Recibe una tabla, y una lista de atributos a la cual se hará la proyección de los datos de la tabla. Devuelve una tabla en forma de lista de listas de String que contiene la proyección.

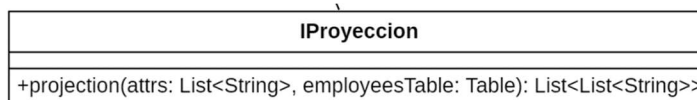


Ilustración 13: Interface IProjection

ISelection

Es una interface que proporcionará los métodos necesarios para hacer la selección a la clase manejador de la base de datos.

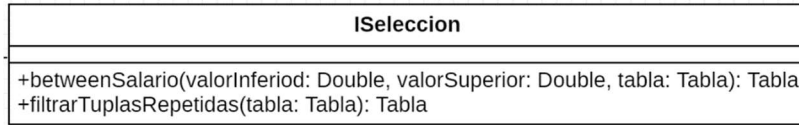


Ilustración 14: Interface ISelection

Contiene dos métodos:

betweenSalary:

Recibe el valor inicial del rango, el valor final del rango y la tabla sobre la cual se hará el between.

filtrarTuplasRepetidas:

Recibe una lista de listas de String, y devuelve la misma lista, eliminando los campos repetidos. La lista de listas es lo que devuelve la proyección. En otras palabras, recibe una proyección y devuelve otra proyección eliminando las tuplas repetidas.

DataBaseManager

Esta clase es la que maneja la base de datos, y extiende de las interfaces *ISelection* y *IProyeccion*, que le brindan los métodos para hacer funciones de proyección y selección.

Esta es la clase principal que se encarga de toda la lógica de las consultas.

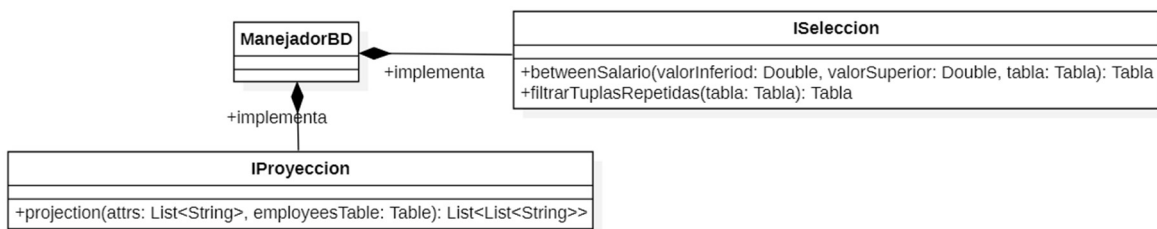


Ilustración 15: Clase DataBaseManager

Pruebas

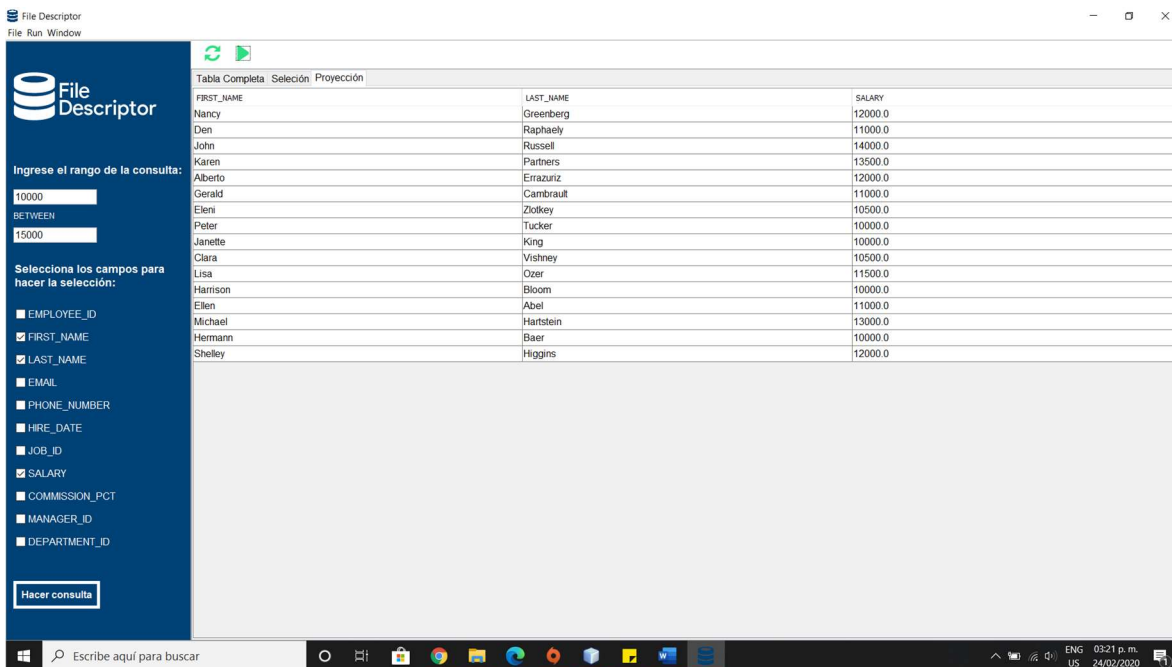
Pruebas de Caja Negra.

Las pruebas fueron realizadas a través de la técnica de partición equivalente.

Tipo de entrada	Clase válida	Clase inválida
rango inicial > 0	10000	-5
rango final rango inicial	15000	-10000
{EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID, DEPARTMENT_ID }	EMPLOYEE_ID, FIRST_NAME, SALARY	Sin seleccionar opción.

Ilustración 16: Pruebas caja negra

Pantalla con datos válidos.

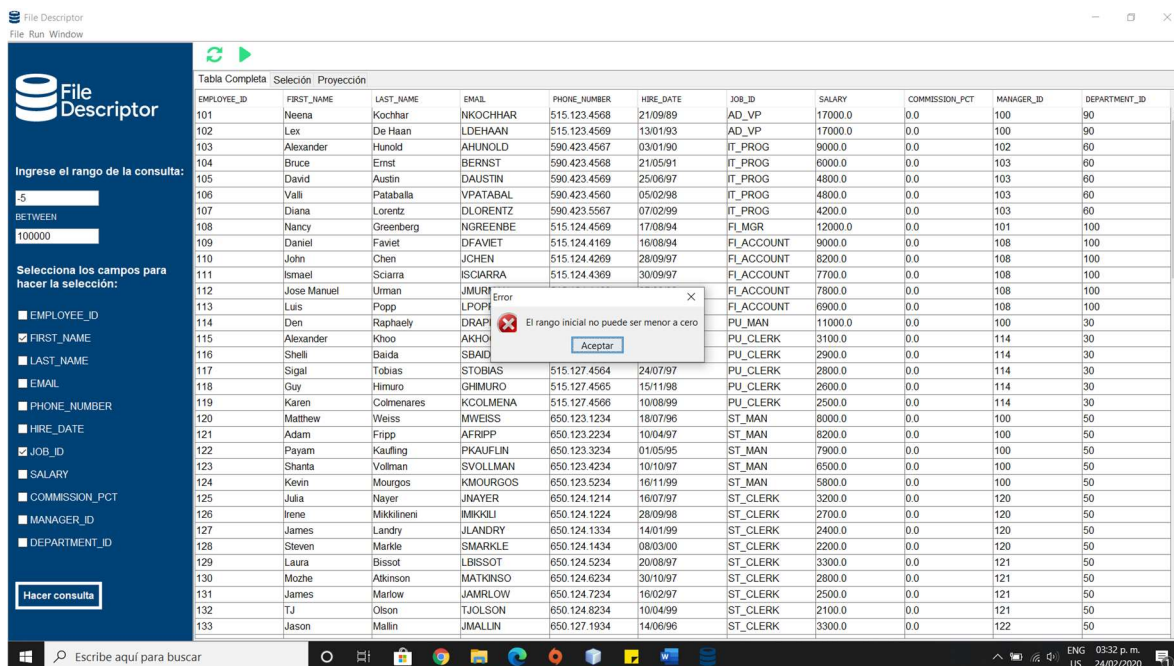


The screenshot shows the File Descriptor application window. On the left, there is a sidebar with the File Descriptor logo and a section titled "Ingrese el rango de la consulta:" with input fields for "10000" and "15000" and a "BETWEEN" dropdown. Below this is a section titled "Selecciona los campos para hacer la selección:" with a list of database fields and checkboxes. The "FIRST_NAME", "LAST_NAME", and "SALARY" fields are selected. At the bottom of the sidebar is a "Hacer consulta" button. The main area of the window displays a table with the following data:

FIRST_NAME	LAST_NAME	SALARY
Nancy	Greenberg	12000.0
Oen	Raphaely	11000.0
John	Russell	14000.0
Karen	Partners	13500.0
Alberto	Errazuriz	12000.0
Gerald	Cambrault	11000.0
Eleni	Zlotkey	10500.0
Peter	Tucker	10000.0
Janette	King	10000.0
Clara	Vishney	10500.0
Lisa	Ozer	11500.0
Harrison	Bloom	10000.0
Ellen	Abel	11000.0
Michael	Hartstein	13000.0
Hermann	Baer	10000.0
Shelley	Higgins	12000.0

Ilustración 17: Resultado de prueba

Pantalla con rango inicial negativo y final positivo.



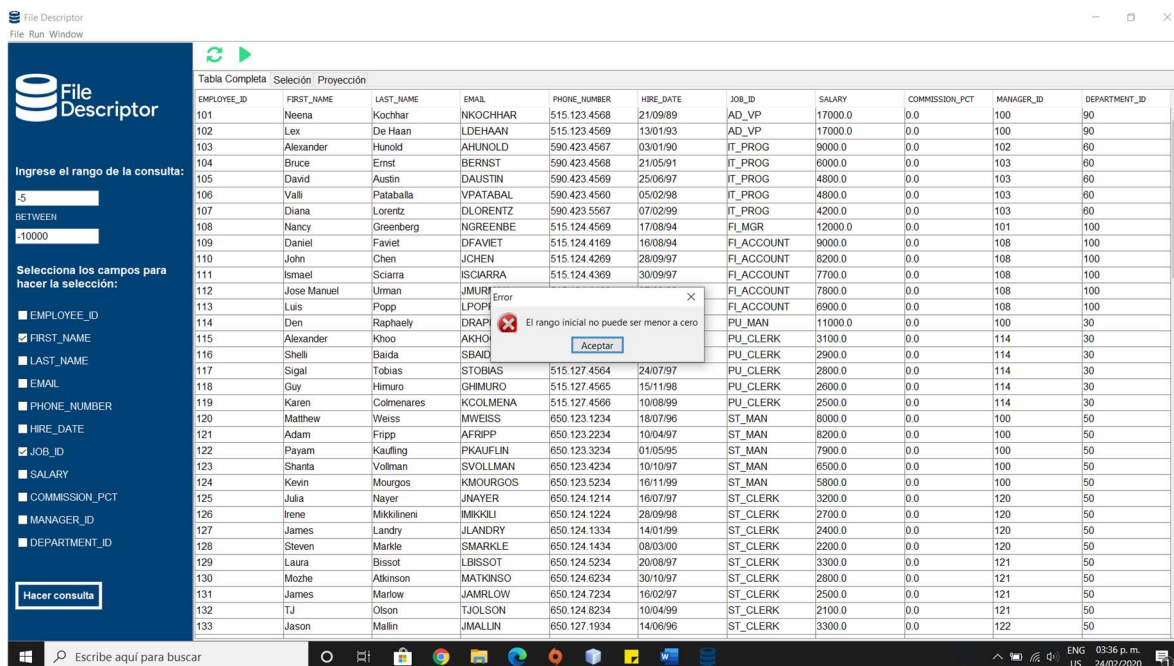
The screenshot shows the File Descriptor application interface. On the left, there's a sidebar with the File Descriptor logo and a section titled "Ingrese el rango de la consulta:" (Enter the range of the query:). Below this, there are input fields for "BETWEEN" with values "-5" and "10000". A section titled "Selecciona los campos para hacer la selección:" (Select the fields to make the selection:) lists various fields with checkboxes. The "Hacer consulta" (Make query) button is at the bottom of this section.

The main area displays a table titled "Tabla Completa Selección Proyección". The table has columns: EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID, and DEPARTMENT_ID. The table contains 20 rows of employee data. An error message box is overlaid on the table, stating "Error El rango inicial no puede ser menor a cero" (Error The initial range cannot be less than zero) with an "Aceptar" (Accept) button.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
101	Neena	Kochhar	NKOCHHAR	515 123 4568	21/09/89	AD_VP	17000.0	0.0	100	90
102	Lex	De Haan	LDEHAAN	515 123 4569	13/01/93	AD_VP	17000.0	0.0	100	90
103	Alexander	Hunold	AHUNOLD	590 423 4567	03/01/90	IT_PROG	9000.0	0.0	102	60
104	Bruce	Ernst	BERNST	590 423 4568	21/05/91	IT_PROG	6000.0	0.0	103	60
105	David	Austin	DAUSTIN	590 423 4569	25/06/97	IT_PROG	4800.0	0.0	103	60
106	Valli	Pataballa	VPATABAL	590 423 4560	05/02/98	IT_PROG	4800.0	0.0	103	60
107	Diana	Lorentz	DLORENTZ	590 423 5567	07/02/99	IT_PROG	4200.0	0.0	103	60
108	Nancy	Greenberg	NGREENBE	515 124 4569	17/08/94	FL_MGR	12000.0	0.0	101	100
109	Daniel	Faviet	DFAVIET	515 124 4169	16/08/94	FL_ACCOUNT	9000.0	0.0	108	100
110	John	Chen	JCHEN	515 124 4269	28/09/97	FL_ACCOUNT	8200.0	0.0	108	100
111	Ismael	Sciarra	ISCIARRA	515 124 4369	30/09/97	FL_ACCOUNT	7700.0	0.0	108	100
112	Jose Manuel	Uman	JMUR	Error		FL_ACCOUNT	7800.0	0.0	108	100
113	Luis	Popp	LPOPP	Error		FL_ACCOUNT	6900.0	0.0	108	100
114	Den	Raphaely	DRAP	Error		PU_MAN	11000.0	0.0	100	30
115	Alexander	Khoo	AKHO	Error		PU_CLERK	3100.0	0.0	114	30
116	Shell	Baida	SBAID	Error		PU_CLERK	2900.0	0.0	114	30
117	Sigal	Tobias	STOBIAS	515 127 4564	24/07/97	PU_CLERK	2800.0	0.0	114	30
118	Guy	Himuro	GHIMURO	515 127 4565	15/11/98	PU_CLERK	2600.0	0.0	114	30
119	Karen	Coimnare	KCOLMENA	515 127 4566	10/08/99	PU_CLERK	2500.0	0.0	114	30
120	Matthew	Weiss	MWEISS	650 123 1234	18/07/96	ST_MAN	8000.0	0.0	100	50
121	Adam	Frpp	AFRIPP	650 123 2234	10/04/97	ST_MAN	8200.0	0.0	100	50
122	Payam	Kauffman	PKAUFFIN	650 123 3234	01/05/95	ST_MAN	7900.0	0.0	100	50
123	Shanta	Vollman	SVOLLMAN	650 123 4234	10/10/97	ST_MAN	6500.0	0.0	100	50
124	Kevin	Mourgos	KMOURGOS	650 123 5234	16/11/99	ST_MAN	5800.0	0.0	100	50
125	Julia	Nayer	JNAYER	650 124 1214	16/07/97	ST_CLERK	3200.0	0.0	120	50
126	Irene	Mikkilineni	IMIKILI	650 124 1224	28/09/98	ST_CLERK	2700.0	0.0	120	50
127	James	Landry	JLANDRY	650 124 1334	14/01/99	ST_CLERK	2400.0	0.0	120	50
128	Steven	Markle	SMARKLE	650 124 1434	08/03/00	ST_CLERK	2200.0	0.0	120	50
129	Laura	Bissot	LBISOT	650 124 5234	20/08/97	ST_CLERK	3300.0	0.0	121	50
130	Mozhe	Atkinson	MATKINSO	650 124 6234	30/10/97	ST_CLERK	2800.0	0.0	121	50
131	James	Marlow	JAMRLOW	650 124 7234	16/02/97	ST_CLERK	2500.0	0.0	121	50
132	TJ	Olson	TJOLSON	650 124 8234	10/04/99	ST_CLERK	2100.0	0.0	121	50
133	Jason	Malin	JMALIN	650 127 1934	14/06/96	ST_CLERK	3300.0	0.0	122	50

Ilustración 18: Resultado de Prueba

Pantalla con rangos negativos



The screenshot shows the File Descriptor application interface. On the left, there's a sidebar with the File Descriptor logo and a section titled "Ingrese el rango de la consulta:" (Enter the range of the query:). Below this, there are input fields for "BETWEEN" with values "-5" and "10000". A section titled "Selecciona los campos para hacer la selección:" (Select the fields to make the selection:) lists various fields with checkboxes. The "Hacer consulta" (Make query) button is at the bottom of this section.

The main area displays a table titled "Tabla Completa Selección Proyección". The table has columns: EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID, and DEPARTMENT_ID. The table contains 20 rows of employee data. An error message box is overlaid on the table, stating "Error El rango inicial no puede ser menor a cero" (Error The initial range cannot be less than zero) with an "Aceptar" (Accept) button.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
101	Neena	Kochhar	NKOCHHAR	515 123 4568	21/09/89	AD_VP	17000.0	0.0	100	90
102	Lex	De Haan	LDEHAAN	515 123 4569	13/01/93	AD_VP	17000.0	0.0	100	90
103	Alexander	Hunold	AHUNOLD	590 423 4567	03/01/90	IT_PROG	9000.0	0.0	102	60
104	Bruce	Ernst	BERNST	590 423 4568	21/05/91	IT_PROG	6000.0	0.0	103	60
105	David	Austin	DAUSTIN	590 423 4569	25/06/97	IT_PROG	4800.0	0.0	103	60
106	Valli	Pataballa	VPATABAL	590 423 4560	05/02/98	IT_PROG	4800.0	0.0	103	60
107	Diana	Lorentz	DLORENTZ	590 423 5567	07/02/99	IT_PROG	4200.0	0.0	103	60
108	Nancy	Greenberg	NGREENBE	515 124 4569	17/08/94	FL_MGR	12000.0	0.0	101	100
109	Daniel	Faviet	DFAVIET	515 124 4169	16/08/94	FL_ACCOUNT	9000.0	0.0	108	100
110	John	Chen	JCHEN	515 124 4269	28/09/97	FL_ACCOUNT	8200.0	0.0	108	100
111	Ismael	Sciarra	ISCIARRA	515 124 4369	30/09/97	FL_ACCOUNT	7700.0	0.0	108	100
112	Jose Manuel	Uman	JMUR	Error		FL_ACCOUNT	7800.0	0.0	108	100
113	Luis	Popp	LPOPP	Error		FL_ACCOUNT	6900.0	0.0	108	100
114	Den	Raphaely	DRAP	Error		PU_MAN	11000.0	0.0	100	30
115	Alexander	Khoo	AKHO	Error		PU_CLERK	3100.0	0.0	114	30
116	Shell	Baida	SBAID	Error		PU_CLERK	2900.0	0.0	114	30
117	Sigal	Tobias	STOBIAS	515 127 4564	24/07/97	PU_CLERK	2800.0	0.0	114	30
118	Guy	Himuro	GHIMURO	515 127 4565	15/11/98	PU_CLERK	2600.0	0.0	114	30
119	Karen	Coimnare	KCOLMENA	515 127 4566	10/08/99	PU_CLERK	2500.0	0.0	114	30
120	Matthew	Weiss	MWEISS	650 123 1234	18/07/96	ST_MAN	8000.0	0.0	100	50
121	Adam	Frpp	AFRIPP	650 123 2234	10/04/97	ST_MAN	8200.0	0.0	100	50
122	Payam	Kauffman	PKAUFFIN	650 123 3234	01/05/95	ST_MAN	7900.0	0.0	100	50
123	Shanta	Vollman	SVOLLMAN	650 123 4234	10/10/97	ST_MAN	6500.0	0.0	100	50
124	Kevin	Mourgos	KMOURGOS	650 123 5234	16/11/99	ST_MAN	5800.0	0.0	100	50
125	Julia	Nayer	JNAYER	650 124 1214	16/07/97	ST_CLERK	3200.0	0.0	120	50
126	Irene	Mikkilineni	IMIKILI	650 124 1224	28/09/98	ST_CLERK	2700.0	0.0	120	50
127	James	Landry	JLANDRY	650 124 1334	14/01/99	ST_CLERK	2400.0	0.0	120	50
128	Steven	Markle	SMARKLE	650 124 1434	08/03/00	ST_CLERK	2200.0	0.0	120	50
129	Laura	Bissot	LBISOT	650 124 5234	20/08/97	ST_CLERK	3300.0	0.0	121	50
130	Mozhe	Atkinson	MATKINSO	650 124 6234	30/10/97	ST_CLERK	2800.0	0.0	121	50
131	James	Marlow	JAMRLOW	650 124 7234	16/02/97	ST_CLERK	2500.0	0.0	121	50
132	TJ	Olson	TJOLSON	650 124 8234	10/04/99	ST_CLERK	2100.0	0.0	121	50
133	Jason	Malin	JMALIN	650 127 1934	14/06/96	ST_CLERK	3300.0	0.0	122	50

Ilustración 19: Resultado de Prueba

Pruebas de Caja Blanca.

Método de Between:

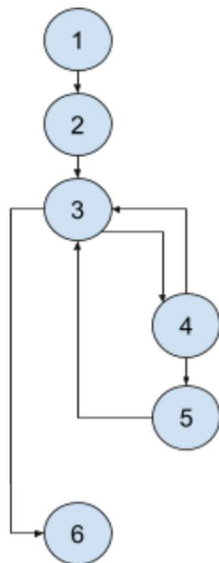
```
@Override
public Table between(Double lowestValue, Double highestValue, Table table) {
    Table tableResult = new Table();
    tableResult.setTableName(table.getTableName());

    for (Tuple t : table.getTuplesList()) {
        TupleEmployee e = (TupleEmployee) t;

        if (e.getSalary() >= lowestValue && e.getSalary() <= highestValue) {
            tableResult.addTuple(e);
        }
    }

    return tableResult;
}
```

Ilustración 20: Código método between



Complejidad celomática

$$C(G) = 3$$

Ilustración 21: Grado de flujo asociado

Casos de Prueba:

No.	Camino independiente	Datos	Resultado
1	1-2-3-6	En el caso de que el archivo esté vacío, la tabla se imprimirá sin contenido	0 resultados
1	1-2-3-4-3-6	El archivo tiene datos, los datos evaluados en el "if" no cumplen la condición, por lo tanto, tenemos 0 resultados	0 resultados
1	1-2-3-4-5-3-6	el archivo tiene contenido, entre los datos evaluados hay quienes cumplen la condición del "if" y se muestran los resultados	tabla con resultados

Ilustración 22: Casos de prueba between

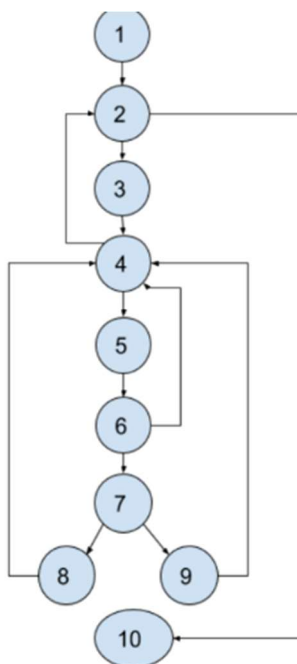
Método filterRepeatedTuples:

```
@Override
public List<List<String>> filterRepeatedTuples( List<List<String>> table) {

    List<List<String>> tableAux = new ArrayList(table);

    for (List lista : tableAux) {
        List listaStr = (List) lista;
        int repeatCont=0;
        for (List t2 : table) {
            List e2 = (List) t2;
            if ( listaStr.equals(e2)) {
                if(repeatCont>0){
                    tableAux.remove(t2);
                }else{
                    repeatCont=1;
                }
            }
        }
    }
    return tableAux;
}
```

Ilustración 23: Código método filterRepeatedTuples



Complejidad celomática:
 $C(G) = 5$

Ilustración 24: Grado de flujo asociado

Casos de Prueba

No.	Camino Independiente	Datos	Resultado
1	1-2-10	En el caso de que después de la selección no se guarden elementos que cumplan los rangos.	0 resultados
2	1-2-3-4-2-10	En el caso de que los datos en tabla se perdieran entre el paso 2 y 4, debido a que son los mismos utilizados para el paso 2	0 resultados
3	1-2-3-4-5-6-4-2-10	En caso de que el primer if verifique que un dato de los resultados de la proyección no es igual a los demás	Tabla de proyección con datos filtrados
4	1-2-3-4-5-6-7-8-4-2-10	En caso de que el segundo if encuentre que algunos resultados después de hacer la proyección son iguales, pero no se tiene ninguno dentro de la tabla final de filtrado, se toma este y se enciende la bandera.	Tabla de proyección con datos filtrados
5	1-2-3-4-5-6-7-9-4-2-10	En caso de que el segundo if encuentre que algunos resultados después de hacer la proyección son iguales y ya se ha guardado uno de este tipo	Tabla de proyección con datos filtrados

Ilustración 25: Casos de prueba método filterRepeatedTuples

Método Projection:

```
@Override
public List<List<String>> projection(List<String> attrs, Table employeesTable) {
    List<List<String>> salida = new ArrayList();
    if(attrs == null) return salida;

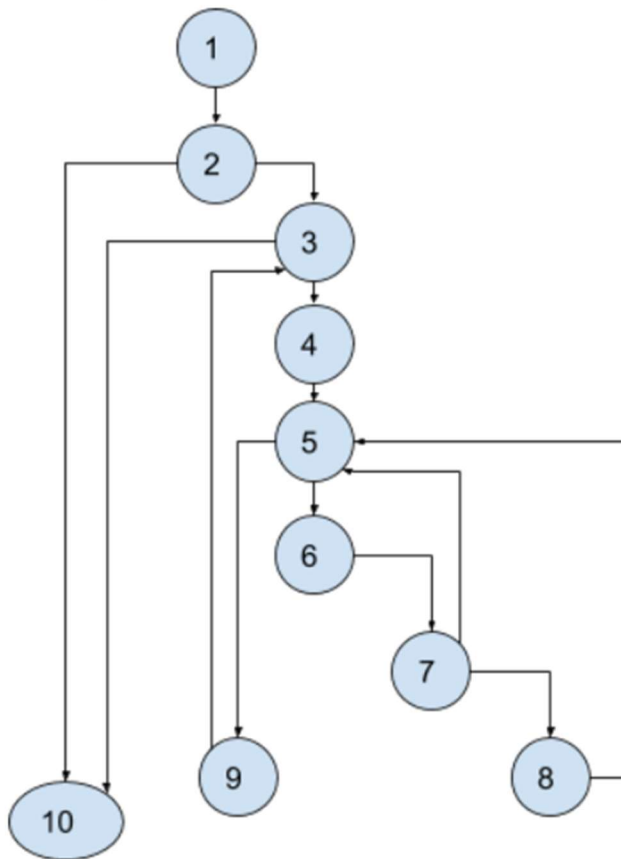
    for (Tuple t : employeesTable.getTuplesList()) {
        TupleEmployee employee = (TupleEmployee) t;
        List<String> tuplaSalida = new ArrayList();

        for (String attr : attrs) {
            String value = employee.getAttrFromString(attr);
            if (value != null) {
                tuplaSalida.add(value);
            }
        }

        salida.add(tuplaSalida);
    }

    return salida;
}
```

Ilustración 26: Código método projection



Complejidad ciclomática:
 $C(G) = 3$

Ilustración 27: Grado de flujo asociado

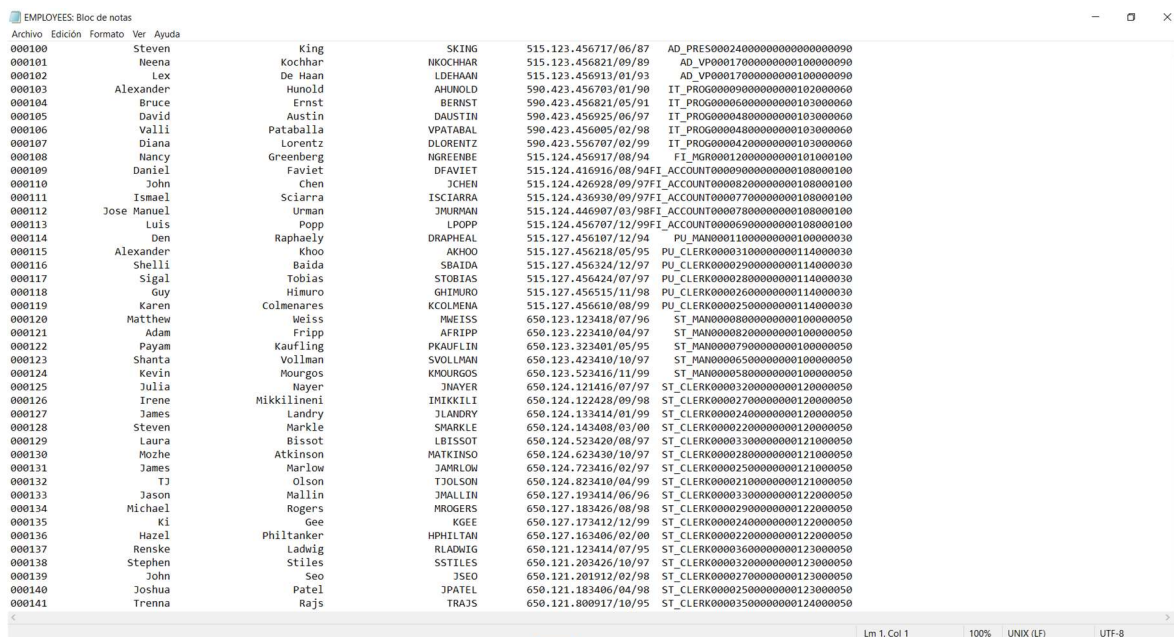
Casos de Prueba:

Tabla 4: Casos de prueba projection

No.	Camino Independiente	Datos	Resultado
1	1-2-10	si al evaluar el "if" cumple que es null, retorna la lista (salida)	lista(salida)
2	1-2-3-10	evalúa el "if" si es diferente de null, inicializa una variable donde va tomando cada valor de la lista de tuplas	lista(salida)
3	1-2-3-4-5-9-3-10	crea la lista salida, evalúa si los atributos son igual a null,	lista(salida)
4	1-2-3-4-5-6-7-5-9-3-10	verifica que los atributos no sean "null", agrega a "t" elementos de Lista de Tuplas, vuelve a "t" un tipo "employee", crea "tuplaSalida" toma los atributos y evalúa que sean diferentes de "null"	lista(salida) sin atributos
5	1-2-3-4-5--6-7-8-5-9-3-10	verifica que haya atributos, toma de TuplesList las tuplas y las pone en "t", lee los atributos y los evalúa, si son diferentes de null se vuelve una "tuplaSalida" y se añaden a la lista (salida)	lista(salida) con tuplas que cumplen las condiciones

Prueba de Caja Gris

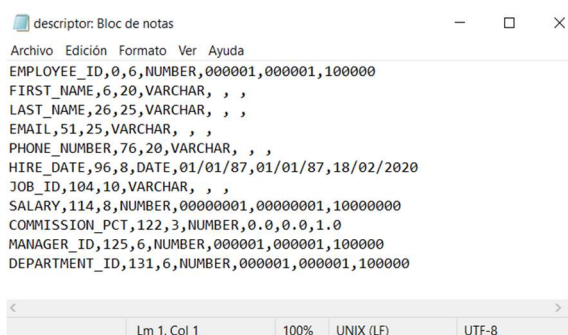
Los datos en principio se encuentran contenidos en un archivo de texto, ubicado en la carpeta data del proyecto, en él se puede observar la manera en que serán tomados los datos uno a uno para la creación de la estructura.



EMPLOYEE_ID	NAME	JOB	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	HIRE_DATE
000100	Steven	King	515.123.456717/06/87		AD_PRES000240000000000000000000		
000101	Neena	Kochhar	515.123.456821/09/89		AD_VP000170000000000000000000		
000102	Lex	De Haan	515.123.456913/01/93		AD_VP000170000000000000000000		
000103	Alexander	Hunold	590.423.456703/01/90		IT_PROG000090000000000000000000		
000104	Bruce	Ernst	590.423.456821/05/91		IT_PROG000060000000000000000000		
000105	David	Austin	590.423.456925/06/97		IT_PROG000048000000000000000000		
000106	Valli	Pataballa	590.423.456905/02/98		IT_PROG000048000000000000000000		
000107	Diana	Lorentz	590.423.456707/02/99		IT_PROG000042000000000000000000		
000108	Nancy	Greenberg	515.124.456917/08/94		FI_MGR000120000000000000000000		
000109	Daniel	Faviet	515.124.416916/08/94		FI_ACCOUNT000090000000000000000000		
000110	John	Chen	515.124.426928/09/97		FI_ACCOUNT000082000000000000000000		
000111	Ismael	Sciarra	515.124.436930/09/97		FI_ACCOUNT000077000000000000000000		
000112	Jose Manuel	Urman	515.124.446907/03/98		FI_ACCOUNT000078000000000000000000		
000113	Luis	Popp	515.124.456707/12/99		FI_ACCOUNT000069000000000000000000		
000114	Den	Raphaely	515.127.456107/12/94		PU_MAN000110000000000000000000		
000115	Alexander	Khoo	515.127.456218/05/95		PU_CLERK000031000000000000000000		
000116	Shelli	Baida	515.127.456324/12/97		PU_CLERK000029000000000000000000		
000117	Sigal	Tobias	515.127.456424/07/97		PU_CLERK000028000000000000000000		
000118	Guy	Himuro	515.127.456515/11/98		PU_CLERK000026000000000000000000		
000119	Karen	Colmenares	515.127.456610/08/99		PU_CLERK000025000000000000000000		
000120	Matthew	Weiss	650.123.123418/07/96		ST_MAN000080000000000000000000		
000121	Adam	Fripp	650.123.223410/04/97		ST_MAN000082000000000000000000		
000122	Payam	Kaufling	650.123.323401/05/95		ST_MAN000079000000000000000000		
000123	Shanta	Vollman	650.123.423410/10/97		ST_MAN000065000000000000000000		
000124	Kevin	Mourgos	650.123.523416/11/99		ST_MAN000058000000000000000000		
000125	Julia	Hayer	650.124.121416/07/97		ST_CLERK000032000000000000000000		
000126	Irene	Mikkilineni	650.124.122428/09/98		ST_CLERK000027000000000000000000		
000127	James	Landry	650.124.133414/01/99		ST_CLERK000024000000000000000000		
000128	Steven	Markle	650.124.143408/03/00		ST_CLERK000022000000000000000000		
000129	Laura	Bissot	650.124.523420/08/97		ST_CLERK000033000000000000000000		
000130	Mozhe	Atkinson	650.124.623430/10/97		ST_CLERK000028000000000000000000		
000131	James	Marlow	650.124.723416/02/97		ST_CLERK000025000000000000000000		
000132	TJ	TJOLSON	650.124.823410/04/99		ST_CLERK000021000000000000000000		
000133	Jason	Mallin	650.127.193414/06/96		ST_CLERK000033000000000000000000		
000134	Michael	Rogers	650.127.183426/08/98		ST_CLERK000029000000000000000000		
000135	Ki	Gee	650.127.173412/12/99		ST_CLERK000024000000000000000000		
000136	Hazel	Philtanker	650.127.163406/02/00		ST_CLERK000022000000000000000000		
000137	Renske	Ladwig	650.121.123414/07/95		ST_CLERK000036000000000000000000		
000138	Stephen	Stiles	650.121.203426/10/97		ST_CLERK000032000000000000000000		
000139	John	Seo	650.121.201912/02/98		ST_CLERK000027000000000000000000		
000140	Joshua	Patel	650.121.183406/04/98		ST_CLERK000025000000000000000000		
000141	Trenna	Rajs	650.121.800917/10/95		ST_CLERK000035000000000000000000		

Ilustración 28: Archivo datos de la tabla employees

La estructura fue definida en el archivo descriptor



```

EMPLOYEE_ID,0,6,NUMBER,000001,000001,100000
FIRST_NAME,6,20,VARCHAR, , ,
LAST_NAME,26,25,VARCHAR, , ,
EMAIL,51,25,VARCHAR, , ,
PHONE_NUMBER,76,20,VARCHAR, , ,
HIRE_DATE,96,8,DATE,01/01/87,01/01/87,18/02/2020
JOB_ID,104,10,VARCHAR, , ,
SALARY,114,8,NUMBER,00000001,00000001,10000000
COMMISSION_PCT,122,3,NUMBER,0.0,0.0,1.0
MANAGER_ID,125,6,NUMBER,000001,000001,100000
DEPARTMENT_ID,131,6,NUMBER,000001,000001,100000
  
```

Ilustración 29: Archivo descriptor de archivos

Dicha estructura se encuentra implementada en las clases “Tuple” y “TupleEmployee”.



Ilustración 30: Clases de tipo tupla

La clase Tuple cuenta con métodos para la comparación de tuplas, obtención de atributos a partir de su nombre y conversión de tuplas a arreglos para su posterior impresión en pantalla

```
1 package model;
2
3 import java.util.List;
4
5 public class Tuple {
6
7     public Boolean equalsTo(Tuple tuple){
8         return true;
9     }
10
11     public String getAttrFromString(String attr){
12         return null;
13     }
14
15     public List<String> toArrayString(){
16         return null;
17     }
18 }
```

Ilustración 31: Clase tuple

La clase TupleEmployee es extendida de la clase Tuple por lo que cuenta con sus métodos, además de tener los atributos solicitados en el archivo descriptor y que posteriormente serán cargados a la clase Table la cual consta de una lista de tuplas.

```
7 public class TupleEmployee extends Tuple {
8
9     private int employeeId;
10    private String firstName;
11    private String lastName;
12    private String email;
13    private String phoneNumber;
14    private String hireDate;
15    private String jobId;
16    private Double salary;
17    private Double commissionPct;
18    private int managerId;
19    private int departmentId;
20
21    public TupleEmployee(int employeeId, String firstName, String lastName, String email, String phoneNumber, String hireDate, String jobId, Double salary, Double commissionPct, int managerId, int departmentId) {
22        this.employeeId = employeeId;
23        this.firstName = firstName;
24        this.lastName = lastName;
25        this.email = email;
26        this.phoneNumber = phoneNumber;
27        this.hireDate = hireDate;
28        this.jobId = jobId;
29        this.salary = salary;
30        this.commissionPct = commissionPct;
31        this.managerId = managerId;
32        this.departmentId = departmentId;
33    }
34 }
```

Ilustración 32: Clase Tuple employee

Después la lista será cargada a la interfaz gráfica y será impresa en una tabla donde cada atributo pasará a ser una columna en la tabla.

Diccionario	Tabla Completa	Selección	Proyección							
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21/09/89	AD_VP	17000.0	0.0	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13/01/93	AD_VP	17000.0	0.0	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03/01/90	IT_PROG	9000.0	0.0	102	80
104	Bruce	Ernst	BERNST	590.423.4568	21/05/91	IT_PROG	6000.0	0.0	103	60
105	David	Austin	DAUSTIN	590.423.4569	25/06/97	IT_PROG	4800.0	0.0	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	05/02/98	IT_PROG	4800.0	0.0	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07/02/99	IT_PROG	4200.0	0.0	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17/08/94	FI_MGR	12000.0	0.0	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16/08/94	FI_ACCOUNT	9000.0	0.0	108	100
110	John	Chen	JCHEN	515.124.4269	28/09/97	FI_ACCOUNT	8200.0	0.0	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	30/09/97	FI_ACCOUNT	7700.0	0.0	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	07/03/98	FI_ACCOUNT	7800.0	0.0	108	100
113	Luis	Popp	LPOPP	515.124.4567	07/12/99	FI_ACCOUNT	6900.0	0.0	108	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	07/12/94	PU_MAN	11000.0	0.0	100	30
115	Alexander	Khoo	AKHOO	515.127.4562	18/05/95	PU_CLERK	3100.0	0.0	114	30
116	Shelli	Baida	SBAIDA	515.127.4563	24/12/97	PU_CLERK	2900.0	0.0	114	30
117	Gigay	Tobias	STOBIAS	515.127.4564	24/07/97	PU_CLERK	2800.0	0.0	114	30
118	Guy	Himuro	GHIMURO	515.127.4565	15/11/98	PU_CLERK	2600.0	0.0	114	30
119	Karen	Colmenares	KCOLMENA	515.127.4566	10/08/99	PU_CLERK	2500.0	0.0	114	30
120	Matthew	Weiss	MWEISS	650.123.1234	18/07/96	ST_MAN	8000.0	0.0	100	50
121	Adam	Fripp	AFRIPP	650.123.2234	10/04/97	ST_MAN	8200.0	0.0	100	50
122	Payam	Kaufling	PKAUFLLN	650.123.3234	01/05/95	ST_MAN	7900.0	0.0	100	50
123	Shanta	Vollman	SVOLLMAN	650.123.4234	10/10/97	ST_MAN	6500.0	0.0	100	50
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16/11/99	ST_MAN	5800.0	0.0	100	50
125	Julia	Nayer	JNAYER	650.124.1214	16/07/97	ST_CLERK	3200.0	0.0	120	50
126	Irene	Mikkilineni	IMIKKILI	650.124.1224	28/09/98	ST_CLERK	2700.0	0.0	120	50
127	James	Landry	JLANDRY	650.124.1334	14/01/99	ST_CLERK	2400.0	0.0	120	50
128	Steven	Markle	SMARKLE	650.124.1434	08/03/00	ST_CLERK	2200.0	0.0	120	50
129	Laura	Bissot	LBISOT	650.124.5234	20/08/97	ST_CLERK	3300.0	0.0	121	50
130	Mozhe	Atkinson	MATKINSO	650.124.6234	30/10/97	ST_CLERK	2800.0	0.0	121	50
131	James	Marlow	JAMRLOW	650.124.7234	16/02/97	ST_CLERK	2500.0	0.0	121	50
132	TJ	Olson	TJOLSON	650.124.8234	10/04/99	ST_CLERK	2100.0	0.0	121	50
133	Jason	Mallin	JMALLIN	650.127.1934	14/06/96	ST_CLERK	3300.0	0.0	122	50

Ilustración 33: Vista gráfica del diccionario

Como se puede apreciar los datos se encuentran de la misma manera que en el archivo debido a que como son cargados uno a uno a cada tupla asegura que ningún dato se pierda en el proceso o se escriba en un campo erróneo.

Bibliografía

P. (2015). ¿Qué es el sistema manejador de bases de datos?. 22/02/20, de PowerData Sitio web: <https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/406549/Qu-es-el-sistema-manejador-de-bases-de-datos>

J.. (2016). Ques es un DBMS. 22/02/20, de Plataforma Sistemas Sitio web: <https://plataformasistemas.wordpress.com/bases-de-datos/ques-es-un-dbms/>

Guzman A. (1985). The file descriptor. En The file descriptor: use of a descriptive tool to retrieve general queries to file (5-7). Zacatenco, México, D.F.: instituto Politécnico Nacional.