# UCENFOTEC

# SEDE CENTRAL

## 

### PROGRAMACIÓN DE BASES DE DATOS

### PROYECTO FINAL

### AJEDREZ EN PL/SQL

### INTEGRANTES

### BRANDON RUTCH MURILLO

### PROFESOR

### LUIS MONGE

### III CUATRIMESTRE 2019

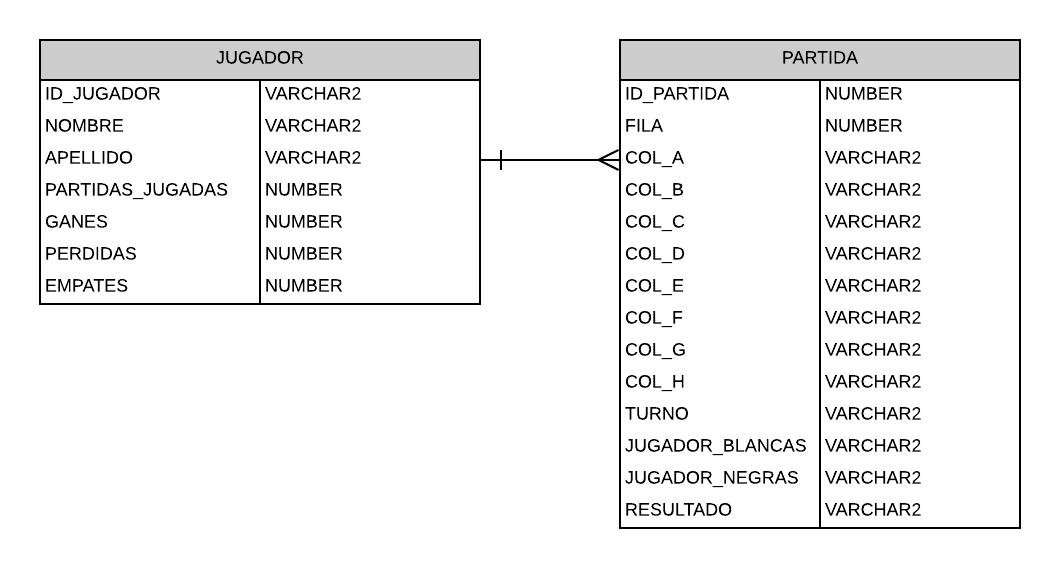
**OBJETIVOS**

1. Diseñar el modelo entidad relación de una base de datos para almacenar las partidas de ajedrez y los jugadores.
2. Implementar una solución en base de datos para el desarrollo del juego de ajedrez.
3. Realizar movimiento únicamente cuando sean válidos en el tablero de ajedrez.
4. Cumplir con los movimientos específicos para cada ficha del ajedrez.
5. Crear un método efectivo para que dos personas puedan jugar n cantidad de partidas de ajedrez por medio de programas de PL/SQL.
6. Utilizar los conocimientos adquiridos dentro de la asignatura para poder realizar el proyecto.

**REGLAS**

* Los peones pueden moverse únicamente hacia el frente un espacio excepto en el primer movimiento que puede moverse dos espacios, se come en diagonal de un espacio.
* Las torres se mueven únicamente en horizontal y vertical y no pueden pasar sobre fichas de ningún tipo, se come en el mismo movimiento y se posiciona sobre la ficha que come.
* Los alfiles se mueven en diagonal, no pasan sobre otras fichas y comen en estas mismas direcciones, posicionándose sobre estos lugares.
* Los caballos se mueven en forma de L tres espacios hacia derecha, izquierda, arriba y abajo y un espacio en la línea perpendicular. Este puede pasar sobre otras fichas pero su destino debe estar vacío o contener una ficha del contrario.
* El Rey se puede mover un espacio en cualquier dirección dentro del tablero siempre y cuando no haya una ficha en su destino.
* La reina se puede mover cualquier cantidad de espacios bajo las mismas condiciones de la torre y el alfil.
* El jugador debe ingresar coordenadas de filas y columnas dentro del tablero, como es que en las filas el número vaya de 1 a 8 y las columnas de A a H.
* El jugador que esté en turno puede mover únicamente las fichas de su color.
* El juego termina cuando el jugador del equipo contrario ha realizado el movimiento de Jaque Mate, es decir, se ha comido el rey.

**DIAGRAMA ENTIDAD-RELACIÓN DE LAS TABLAS DEL AJEDREZ**



1. Tabla JUGADOR: esta tabla corresponde a la entidad jugador y almacena atributos como el ID\_JUGADOR (llave primaria, única, varchar2(10)), NOMBRE(varchar2(20)), APELLIDO(varchar2(20)), PARTIDAS\_JUGADAS(number), GANES(number), PERDIDAS(number) Y EMPATES(number).
2. Tabla PARTIDA: esta tabla corresponde a la entidad PARTIDA y almacena atributos como el ID\_PARTIDA(llave primaria, única, number), FILA(number), COL\_A(varchar2(2)), COL\_B(varchar2(2)), COL\_C(varchar2(2)), COL\_D(varchar2(2)), COL\_E(varchar2(2)), COL\_F(varchar2(2)), COL\_G(varchar2(2)), COL\_H(varchar2(2)), TURNO(varchar2(20)), JUGADOR\_BLANCAS(varchar2(10), llave foránea procedente de tabla JUGADOR), JUGADOR\_NEGRAS(varchar2(10), llave foránea procedente de tabla JUGADOR).

**MANUAL DE USO DE PL/SQL AJEDREZ Y DESCRIPCIÓN DE PROGRAMAS, BLOQUES, FUNCIONES Y PROCEDIMIENTOS.**

**PROGRAMA**

* PROGRAMA\_CREAR\_TABLAS\_AJEDREZ.SQL

Contiene los drops, la creación de tablas y de constraints.

**BLOQUES**

* AGREGAR\_JUGADORES.SQL

Este bloque pide los datos de id, nombre y apellido para insertar un nuevo jugador a la base de datos. Los 3 datos deben ser de tipo varchar2, siendo el id de tamaño 10 y el nombre y apellido de tamaño 20.

* CONSULTAR\_TABLERO.SQL

Este bloque solicita el id de la partida para imprimir el tablero correspondiente, sus fichas, coordenadas, el color de turno y los jugadores de esa partida. El id de la partida debe ser de una partida existente y debe ser un número entero positivo.

* CONSULTAR\_ESTADISTICAS\_JUGADORES.SQL

Este bloque solicita el id del jugadores y luego muestra la información de partidas jugadas, empates, pérdidas y ganes del jugador respectivo. El id del jugador debe corresponder a un id existente y ser de tamaño 10.

* CREAR\_PARTIDA.SQL

Este bloque inserta una nueva partida y para ello solicita el id de los dos jugadores que jugarán dicha partida. Los id deben pertenecer a jugadores existentes, no deben ser el mismo id y debe ser de tamaño 10.

* MOVER\_PIEZA.SQL

Este bloque solicita el id de partida, la fila de origen, la columna de origen, la fila de destino y la columna de destino, todo ello para validar si el movimiento a registrar es válido y con ello actualiza las posiciones de las fichas en el tablero. El valor del id de partida debe ser un número entero positivo, las filas deben ser números entero positivos entre 1 y 8 y las columnas deben ser un carácter entre a y h.

* SOLICITAR\_EMPATE.SQL

Este bloque solicita el id de partida, los id de los dos jugadores a modo de acuerdo para finalizar la partida y actualiza la información en la base de datos. El id partida debe ser un número entero positivo y los de los jugadores un varchar2 de tamaño 10 máximo;

**PROGRAMA FUNCIONES\_AJEDREZ.SQL**

* FUNCION\_ESTADISTICAS(VARCHAR2)

Recibe el id del jugador del cuál se quiere consultar su información de partidas y retorna un dato varchar2 con toda la información concatenada.

* VALIDAR\_PARTIDA(NUMBER)

Recibe el id de la partida y retorna un boolean según exista o no en las partidas registradas.

* VALIDAR\_PARTIDA\_PENDIENTE(NUMBER)

Recibe el id de la partida y retorna un boolean verdadero en caso de que la partida esté pendiente y un false en caso de que esté finalizada.

* VALIDAR\_JUGADOR(VARCHAR2)

Recibe el id del jugador y retorna un boolean según exista o no en el registro de jugadores.

* VALIDAR\_ULTIMA\_PARTIDA

Retorna un dato tipo number con el id de la última partida registrada.

* INICIAR\_TABLERO(VARCHAR2, VARCHAR2)

Recibe 2 id pertenecientes a jugadores registrados, llama a validar\_ultima\_partida y con base a ello inserta el valor del id de la nueva partida, las filas, columnas, fichas, jugadores, turno en juego y resultado en la tabla de partidas. Al final retorna el valor de la partida para que los jugadores sepan el id de la partida creada.

* VALIDAR\_JUGADOR\_PARTIDA(NUMBER, VARCHAR2)

Recibe el id de la partida y el id del jugador para validar que ese jugador pertenece a la partida. Retorna boolean según sea el caso.

* VALIDAR\_COLOR\_FICHA\_ORIGEN(NUMBER, NUMBER, VARCHAR2)

Recibe el id de la partida, la fila de origen y la columna de origen, valida contra la información de la partida si la ficha que está en las coordenadas de origen pertenece al color del turno. Retorna boolean según sea el caso.

* VALIDAR\_POSICION\_DESTINO(NUMBER, NUMBER, VARCHAR2)

Recibe el id de la partida, la fila de destino y la columna de destino, valida contra la información de la partida si la ficha o campo que está en las coordenadas de destino es contrario al color del turno o si es un campo vacío. Retorna boolean según sea el caso.

* VALIDAR\_MOVIMIENTO\_O\_D(NUMBER, NUMBER, VARCHAR2, NUMBER, VARCHAR2)

Recibe el ide la partida, las coordenadas de la ficha de origen y las coordenadas del campo de destino. Tramita el movimiento según sea peón, reina, alfil, torre, rey, caballo y retorna un boolean según el movimiento sea permitido o no.

* VALIDAR\_MUERTE\_REY(NUMBER, NUMBER, VARCHAR2)

Recibe id de partida y coordenadas de campo de destino, valida si la ficha que está en esa posición es el rey. Retorna un boolean verdadero en caso de ser el rey y un boolean falso en caso contrario.

* DEVOLVER\_FICHA\_CAMPO(NUMBER, NUMBER, VARCHAR2)

Recibe el id de la partida, la fila y la columna del campo, devuelve un varchar2 con el valor que tenga ese campo para esa partida.

* DEVOLVER\_COLOR\_TURNO(PART)

Recibe el id de la partida y devuelve un varchar2 con el color de turno de esa partida.

* ACTUALIZAR\_MOVIMIENTO\_JAQUE\_MATE(NUMBER, NUMBER, VARCHAR2, NUMBER, VARCHAR2)

Recibe id de partida, coordenadas de campo de origen y coordenadas de campo de destino. Actualiza la tabla de partida y la tabla de jugador con los datos correspondientes al jaque mate. Devuelve un dato tipo varchar2 con la información de la jugada.

* ACTUALIZAR\_MOVIMIENTO(NUMBER, NUMBER, VARCHAR2, NUMBER, VARCHAR2)

Recibe id de partida, coordenadas de campo de origen y coordenadas de campo de destino. Actualiza la tabla de partida con los datos correspondientes. Devuelve un dato tipo varchar2 con la información de la jugada.

* DEVOLVER\_CHAR\_COLUMNA(NUMBER)

Recibe el número de columna y retorna un varchar2 con la letra correspondiente a ese número.

* DEVOLVER\_NUMERO\_COLUMNA(VARCHAR2)

Recibe la letra de la columna y retorna un number con el número de columna correspondiente a esa letra.

**PROGRAMA PROCEDIMIENTOS\_AJEDREZ.SQL**

* PROC\_CONSULTAR\_TABLERO(VARCHAR2)

Recibe el id de la partida y consulta la información de las filas y columnas respectivas. Imprime en pantalla el tablero con dicha información.

* PROC\_AGREGAR\_NUEVO\_JUGADOR(VARCHAR2, VARCHAR2, VARCHAR2)

Recibe el id, el nombre y el apellido del jugador a agregar. Con esa información se ejecuta el insert del nuevo jugador en la tabla jugador. Imprime el resultado del insert.

* PROC\_CREAR\_NUEVA\_PARTIDA(VARCHAR2, VARCHAR2)

Este procedimiento recibe el id de los dos jugadores que van a jugar. Llama a función iniciar\_tablero y actualiza la información de la nueva partida en tabla jugador. Imprime el resultado de la creación de la nueva partida.

* PROC\_MOVER\_FICHA(NUMBER, NUMBER, VARCHAR2, NUMBER, VARCHAR2)

Recibe el id de la partida, la fila y la columna de origen, la fila y la columna de destino. Llama a las funciones de validar\_partida, validar\_partida\_pendiente, validar\_color\_ficha\_origen, validar\_posicion\_destino, validar\_movimiento\_o\_d, validar\_muerte\_rey, actualizar\_movimiento\_jaque\_mate y actualizar\_movimiento. Imprime el resultado del movimiento según sea válido o no.

* PEDIR\_EMPATE(NUMBER, VARCHAR2, VARCHAR2)

Recibe el id de la partida, el id de los jugadores de esa partida y actualiza en la tabla partida y en la tabla jugadore la información correspondiente al empate. Imprime el resultado del trámite.

**¿CÓMO JUGAR?**

* **Agregar jugador**

1. Escribir en línea de comandos la instrucción @agregar\_jugadores.sql.
2. Escribir el id del jugador (10 caracteres máximo). No debe ser igual al de un jugador existente.
3. Escribir el nombre del jugador (20 caracteres máximo).
4. Escribir el apellido del jugador (20 caracteres máximo).
5. Listo el jugador se ha registrado.

* **Nueva partida**

1. Escribir en línea de comandos la instrucción @crear\_partida.sql, enter.
2. Ingresar 2 id correspondientes a jugadores previamente registrados. Si es un nuevo jugador debe agregarlo primero (Ver agregar jugador).
3. Listo la partida nueva se ha creado.

* **Continuar partida**

1. Para ello sólo debe de usar @mover\_pieza.sql(Ver mover ficha). Para tener una guía de cómo están las fichas y de quién es el turno puedo usar @consultar\_tablero.sql(Ver consultar partida).

* **Consultar partida**

1. Escribir en línea de comandos la instrucción @consultar\_tablero.sql

* **Consultar información de jugador**

1. Escribir en línea de comandos la instrucción @consultar\_estadisticas\_jugadores.sql
2. Ingresar el id del jugador del cual quiere obtener información, el id debe ser un conjunto de máximo 10 dígitos y debe pertenecer al id de un jugador previamente registrado.
3. Se imprimirá en pantalla la información del jugador como su nombre, partidas jugadas, ganadas, perdidas y empates.

* **Mover ficha**

1. Escribir en línea de comandos la instrucción @mover\_pieza.sql
2. Ingresar el id de la partida donde quiere tramitar el movimiento, debe ser un id correspondiente a una partida existente y que se encuentre sin finalizar.
3. Ingresar la fila de origen (número entero positivo entre 1 y 8).
4. Ingresar columna de origen (un caracter entre a y h);
5. Ingresar la fila de destino (número entero positivo entre 1 y 8).
6. Ingresar columna de destino (un caracter entre a y h);
7. La ficha elegida debe corresponder a la ficha del color del turno en transcurso.
8. El movimiento debe ser uno válido para la ficha elegida.
9. Listo el movimiento se actualizará en la partida

* **Pedir empate**

1. Escribir en línea de comandos la instrucción @solicitar\_empate.sql
2. Ingresar el id de la partida donde quiere tramitar el empate, debe ser un id correspondiente a una partida existente.
3. Ingrese el id de cada uno de los jugadores que están participando en dicha partida.
4. Listo el empate se ha efectuado.

Anexo 1. Creación de tablas

/\*\*

Universidad Cenfotec

Proyecto de Programación de Bases de Datos

Profesor:

-Luis Monge

Elaborado por:

- Brandon Rutch Murillo

\*/

/\*\* Seccion drop tablas \*/

DROP TABLE JUGADOR CASCADE CONSTRAINTS;

DROP TABLE PARTIDA CASCADE CONSTRAINTS;

/\*\* Seccion drop secuencias \*/

DROP SEQUENCE SEQ\_PARTIDA;

/\*\* TABLAS \*/

CREATE TABLE JUGADOR(

ID\_JUGADOR VARCHAR2(10) NOT NULL ,

NOMBRE VARCHAR2(20)NOT NULL,

APELLIDO VARCHAR2(20) NOT NULL,

PARTIDAS\_JUGADAS NUMBER NOT NULL,

GANES NUMBER NOT NULL ,

PERDIDAS NUMBER NOT NULL,

EMPATES NUMBER NOT NULL,

CONSTRAINT PK\_JUGADOR PRIMARY KEY (ID\_JUGADOR)

);

CREATE TABLE PARTIDA(

ID\_PARTIDA NUMBER NOT NULL ,

FILA NUMBER NOT NULL ,

COL\_A VARCHAR2(2),

COL\_B VARCHAR2(2),

COL\_C VARCHAR2(2),

COL\_D VARCHAR2(2),

COL\_E VARCHAR2(2),

COL\_F VARCHAR2(2),

COL\_G VARCHAR2(2),

COL\_H VARCHAR2(2),

TURNO VARCHAR2(20),

JUGADOR\_BLANCAS VARCHAR2(10),

JUGADOR\_NEGRAS VARCHAR2(10),

RESULTADO VARCHAR2(20),

CONSTRAINT PK\_PARTIDA PRIMARY KEY (ID\_PARTIDA, FILA)

);

/\*\* SEQUENCIAS \*/

CREATE SEQUENCE SEQ\_PARTIDA

INCREMENT BY 1

START WITH 1

MAXVALUE 1000;

/\*\* FOREIGN KEYS \*/

ALTER TABLE PARTIDA ADD CONSTRAINT FK\_JUGADOR1 FOREIGN KEY (JUGADOR\_BLANCAS) REFERENCES JUGADOR(ID\_JUGADOR);

ALTER TABLE PARTIDA ADD CONSTRAINT FK\_JUGADOR2 FOREIGN KEY (JUGADOR\_NEGRAS) REFERENCES JUGADOR(ID\_JUGADOR);

Anexo 2. Funciones

CREATE OR REPLACE FUNCTION FUNCION\_ESTADISTICAS(IDENT VARCHAR2)

RETURN VARCHAR2

IS

RESULTADO VARCHAR2(200);

CURSOR CU\_JUGADORES IS

SELECT ID\_JUGADOR, NOMBRE, APELLIDO, PARTIDAS\_JUGADAS, GANES, PERDIDAS, EMPATES FROM JUGADOR;

BEGIN

FOR JUG IN CU\_JUGADORES

LOOP

IF JUG.ID\_JUGADOR = IDENT THEN

RESULTADO := 'EL JUGADOR '||JUG.NOMBRE||' '||JUG.APELLIDO||' HA JUGADO UN TOTAL DE: '||TO\_CHAR(JUG.PARTIDAS\_JUGADAS)||' . DE LAS CUALES HA GANADO '||TO\_CHAR(JUG.GANES)||', HA PERDIDO '||TO\_CHAR(JUG.PERDIDAS)||' Y HA EMPATADO '||TO\_CHAR(JUG.EMPATES)||'.';

RETURN RESULTADO;

ELSE

RESULTADO := 'EL ID INGRESADO NO CORRESPONDE A UN JUGADOR REGISTRADO.';

END IF;

EXIT WHEN CU\_JUGADORES%NOTFOUND;

END LOOP;

RETURN RESULTADO;

END;

/

CREATE OR REPLACE FUNCTION VALIDAR\_PARTIDA(IDE\_PARTIDA NUMBER)

RETURN BOOLEAN

IS

EXISTE BOOLEAN := FALSE;

CURSOR CU\_PARTIDAS IS

SELECT ID\_PARTIDA FROM PARTIDA;

BEGIN

FOR PART IN CU\_PARTIDAS

LOOP

IF PART.ID\_PARTIDA = IDE\_PARTIDA THEN

EXISTE := TRUE;

RETURN EXISTE;

END IF;

END LOOP;

RETURN EXISTE;

END;

/

CREATE OR REPLACE FUNCTION VALIDAR\_PARTIDA\_PENDIENTE(IDE\_PARTIDA NUMBER)

RETURN BOOLEAN

IS

EXISTE BOOLEAN := FALSE;

ESTADO VARCHAR2(20);

BEGIN

SELECT DISTINCT RESULTADO INTO ESTADO FROM PARTIDA WHERE ID\_PARTIDA = IDE\_PARTIDA;

IF ESTADO = 'PENDIENTE' THEN

EXISTE := TRUE;

RETURN EXISTE;

END IF;

RETURN EXISTE;

END;

/

CREATE OR REPLACE FUNCTION VALIDAR\_JUGADOR(VAR\_ID VARCHAR2)

RETURN BOOLEAN

IS

EXISTE BOOLEAN := FALSE;

CURSOR CU\_JUGADORES IS

SELECT ID\_JUGADOR FROM JUGADOR;

BEGIN

FOR JUG IN CU\_JUGADORES

LOOP

IF JUG.ID\_JUGADOR = VAR\_ID THEN

EXISTE := TRUE;

RETURN EXISTE;

END IF;

END LOOP;

RETURN EXISTE;

END;

/

CREATE OR REPLACE FUNCTION DEVOLVER\_FICHA\_CAMPO(PART NUMBER, FIL NUMBER, COL VARCHAR2)

RETURN VARCHAR2

IS

LINEA\_PART PARTIDA%ROWTYPE;

CAMPO VARCHAR2(10);

RESULTADO BOOLEAN;

BEGIN

SELECT \* INTO LINEA\_PART FROM PARTIDA WHERE ID\_PARTIDA = PART AND fila = FIL;

RESULTADO := FALSE;

CASE UPPER(COL)

WHEN 'A' THEN

CAMPO := LINEA\_PART.COL\_A;

WHEN 'B' THEN

CAMPO := LINEA\_PART.COL\_B;

WHEN 'C' THEN

CAMPO := LINEA\_PART.COL\_C;

WHEN 'D' THEN

CAMPO := LINEA\_PART.COL\_D;

WHEN 'E' THEN

CAMPO := LINEA\_PART.COL\_E;

WHEN 'F' THEN

CAMPO := LINEA\_PART.COL\_F;

WHEN 'G' THEN

CAMPO := LINEA\_PART.COL\_G;

WHEN 'H' THEN

CAMPO := LINEA\_PART.COL\_H;

END case ;

RETURN CAMPO;

END;

/

CREATE OR REPLACE FUNCTION DEVOLVER\_CHAR\_COLUMNA (NUM\_COL NUMBER) RETURN VARCHAR2 IS

RESULTADO VARCHAR2(2);

BEGIN

CASE NUM\_COL

WHEN 1 THEN RESULTADO := 'A';

WHEN 2 THEN RESULTADO := 'B';

WHEN 3 THEN RESULTADO := 'C';

WHEN 4 THEN RESULTADO := 'D';

WHEN 5 THEN RESULTADO := 'E';

WHEN 6 THEN RESULTADO := 'F';

WHEN 7 THEN RESULTADO := 'G';

WHEN 8 THEN RESULTADO := 'H';

END CASE;

RETURN RESULTADO;

end;

/

CREATE OR REPLACE FUNCTION DEVOLVER\_NUMERO\_COLUMNA (CHAR\_COL VARCHAR2) RETURN NUMBER IS

RESULTADO NUMBER;

BEGIN

CASE upper(CHAR\_COL)

WHEN 'A' THEN RESULTADO := 1;

WHEN 'B' THEN RESULTADO:= 2;

WHEN 'C' THEN RESULTADO := 3;

WHEN 'D' THEN RESULTADO := 4;

WHEN 'E' THEN RESULTADO := 5;

WHEN 'F' THEN RESULTADO := 6;

WHEN 'G' THEN RESULTADO := 7;

WHEN 'H' THEN RESULTADO := 8;

END CASE;

RETURN RESULTADO;

end;

/

CREATE OR REPLACE FUNCTION DEVOLVER\_COLOR\_TURNO(PART NUMBER)

RETURN VARCHAR2

IS

COLOR\_TURNO VARCHAR2(20);

BEGIN

SELECT DISTINCT TURNO INTO COLOR\_TURNO FROM PARTIDA WHERE ID\_PARTIDA = PART;

RETURN COLOR\_TURNO;

END;

/

CREATE OR REPLACE FUNCTION VALOR\_ULTIMA\_PARTIDA

RETURN NUMBER

IS

NUM NUMBER;

BEGIN

SELECT MAX(ID\_PARTIDA) INTO NUM FROM PARTIDA;

IF NUM IS NULL THEN

RETURN 0;

ELSE

RETURN NUM;

END IF;

END;

/

CREATE OR REPLACE FUNCTION INCIAR\_TABLERO(JUG\_B VARCHAR2, JUG\_N VARCHAR2)

RETURN NUMBER

IS

NUM\_PARTIDA NUMBER;

BEGIN

NUM\_PARTIDA := VALOR\_ULTIMA\_PARTIDA;

NUM\_PARTIDA := NUM\_PARTIDA + 1;

INSERT INTO PARTIDA VALUES (NUM\_PARTIDA, 8, 'TN', 'CN', 'AN', 'KN', 'RN', 'AN', 'CN', 'TN','BLANCAS',JUG\_B,JUG\_N,'PENDIENTE');

INSERT INTO PARTIDA VALUES (NUM\_PARTIDA, 7, 'PN', 'PN', 'PN', 'PN', 'PN', 'PN', 'PN', 'PN','BLANCAS',JUG\_B,JUG\_N,'PENDIENTE');

INSERT INTO PARTIDA VALUES (NUM\_PARTIDA, 6, ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ','BLANCAS',JUG\_B,JUG\_N,'PENDIENTE');

INSERT INTO PARTIDA VALUES (NUM\_PARTIDA, 5, ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ','BLANCAS',JUG\_B,JUG\_N,'PENDIENTE');

INSERT INTO PARTIDA VALUES (NUM\_PARTIDA, 4, ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ','BLANCAS',JUG\_B,JUG\_N,'PENDIENTE');

INSERT INTO PARTIDA VALUES (NUM\_PARTIDA, 3, ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ','BLANCAS',JUG\_B,JUG\_N,'PENDIENTE');

INSERT INTO PARTIDA VALUES (NUM\_PARTIDA, 2, 'PB', 'PB', 'PB', 'PB', 'PB', 'PB', 'PB', 'PB','BLANCAS',JUG\_B,JUG\_N,'PENDIENTE');

INSERT INTO PARTIDA VALUES (NUM\_PARTIDA, 1, 'TB', 'CB', 'AB', 'RB', 'KB', 'AB', 'CB', 'TB','BLANCAS',JUG\_B,JUG\_N,'PENDIENTE');

COMMIT;

RETURN NUM\_PARTIDA;

END;

/

create or replace function validar\_jugador\_partida(part number, jug varchar2)

return boolean

is

AUX\_PART NUMBER;

resultado boolean;

SUMATORIA number;

begin

AUX\_PART := part;

resultado := false;

select count(jugador\_blancas) into sumatoria from partida where id\_partida = aux\_part;

if SUMATORIA is not null then

resultado := true;

return resultado;

else

select count(jugador\_negras) into sumatoria from partida where id\_partida = aux\_part;

if sumatoria is not null then

resultado := true;

return resultado;

end if;

end if;

return resultado;

end;

/

CREATE OR REPLACE FUNCTION VALIDAR\_COLOR\_FICHA\_ORIGEN(PART NUMBER, FIL\_O NUMBER, COL\_O VARCHAR2)

RETURN BOOLEAN

IS

RESULTADO BOOLEAN;

LINEA\_PART PARTIDA%ROWTYPE;

COLOR\_TURNO VARCHAR2(20);

FICHA VARCHAR2(2);

BEGIN

COLOR\_TURNO := DEVOLVER\_COLOR\_TURNO(PART);

SELECT \* INTO LINEA\_PART FROM PARTIDA WHERE ID\_PARTIDA = PART AND fila = FIL\_O;

RESULTADO := FALSE;

FICHA := DEVOLVER\_FICHA\_CAMPO(PART, FIL\_O, COL\_O);

CASE COLOR\_TURNO

WHEN 'BLANCAS' THEN

if ficha = 'PB' or ficha = 'CB' or ficha = 'AB' or ficha = 'RB' or ficha = 'TB' or ficha = 'KB' then

RESULTADO := TRUE;

RETURN RESULTADO;

end if;

WHEN 'NEGRAS' THEN

if ficha = 'PN' or ficha = 'CN' or ficha = 'AN' or ficha = 'RN' or ficha = 'TN' or ficha = 'KN' then

RESULTADO := TRUE;

RETURN RESULTADO;

end if;

END CASE;

RETURN RESULTADO;

END;

/

CREATE OR REPLACE FUNCTION VALIDAR\_POSICION\_DESTINO(PART NUMBER, FIL\_D NUMBER, COL\_D VARCHAR2)

RETURN BOOLEAN

IS

RESULTADO BOOLEAN;

LINEA\_PART PARTIDA%ROWTYPE;

COLOR\_TURNO VARCHAR2(20);

CAMPO\_DESTINO VARCHAR2(2);

BEGIN

COLOR\_TURNO := DEVOLVER\_COLOR\_TURNO(PART);

RESULTADO := FALSE;

SELECT \* INTO LINEA\_PART FROM PARTIDA WHERE ID\_PARTIDA = PART AND fila = FIL\_D;

CAMPO\_DESTINO := DEVOLVER\_FICHA\_CAMPO(PART, FIL\_D, COL\_D);

CASE COLOR\_TURNO

WHEN 'NEGRAS' THEN

if CAMPO\_DESTINO = 'PB' or CAMPO\_DESTINO= 'CB' or CAMPO\_DESTINO= 'AB' or CAMPO\_DESTINO= 'RB' or CAMPO\_DESTINO= 'TB' or CAMPO\_DESTINO= 'KB' OR CAMPO\_DESTINO = ' ' then

RESULTADO := TRUE;

RETURN RESULTADO;

end if;

WHEN 'BLANCAS' THEN

if CAMPO\_DESTINO = 'PN' or CAMPO\_DESTINO = 'CN' or CAMPO\_DESTINO = 'AN' or CAMPO\_DESTINO = 'RN' or CAMPO\_DESTINO = 'TN' or CAMPO\_DESTINO = 'KN' OR CAMPO\_DESTINO = ' ' then

RESULTADO := TRUE;

RETURN RESULTADO;

end if;

END CASE;

RETURN RESULTADO;

END;

/

CREATE OR REPLACE FUNCTION VALIDAR\_MUERTE\_REY(PART NUMBER, FIL\_D VARCHAR2, COL\_D VARCHAR2)

RETURN BOOLEAN

IS

RESULTADO BOOLEAN;

FIGURA VARCHAR2(2);

BEGIN

RESULTADO := FALSE;

FIGURA := DEVOLVER\_FICHA\_CAMPO(PART, FIL\_D, COL\_D);

IF FIGURA = 'KN' OR FIGURA = 'KB' THEN

RESULTADO := TRUE;

RETURN RESULTADO;

END IF;

RETURN RESULTADO;

END;

/

CREATE OR REPLACE FUNCTION ACTUALIZAR\_MOVIMIENTO(PART NUMBER, FIL\_O NUMBER, COL\_O VARCHAR2, FIL\_D NUMBER, COL\_D VARCHAR2)

RETURN VARCHAR2

IS

RESULTADO VARCHAR2(1000);

COLOR\_TURNO VARCHAR2(10);

NUEVO\_COLOR\_TURNO VARCHAR2(10);

FICHA\_ORIGEN VARCHAR2(2);

BEGIN

COLOR\_TURNO := DEVOLVER\_COLOR\_TURNO(PART);

FICHA\_ORIGEN := DEVOLVER\_FICHA\_CAMPO(PART, FIL\_O, COL\_O);

CASE COLOR\_TURNO

WHEN 'BLANCAS' THEN

NUEVO\_COLOR\_TURNO := 'NEGRAS';

WHEN 'NEGRAS' THEN

NUEVO\_COLOR\_TURNO := 'BLANCAS';

END CASE;

UPDATE PARTIDA SET TURNO = NUEVO\_COLOR\_TURNO WHERE ID\_PARTIDA = PART;

CASE UPPER(COL\_O)

WHEN 'A' THEN

UPDATE PARTIDA SET COL\_A = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'B' THEN

UPDATE PARTIDA SET COL\_B = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'C' THEN

UPDATE PARTIDA SET COL\_C = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'D' THEN

UPDATE PARTIDA SET COL\_D = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'E' THEN

UPDATE PARTIDA SET COL\_E = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'F' THEN

UPDATE PARTIDA SET COL\_F = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'G' THEN

UPDATE PARTIDA SET COL\_G = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'H' THEN

UPDATE PARTIDA SET COL\_H = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

END CASE;

CASE UPPER(COL\_D)

WHEN 'A' THEN

UPDATE PARTIDA SET COL\_A = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'B' THEN

UPDATE PARTIDA SET COL\_B = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'C' THEN

UPDATE PARTIDA SET COL\_C = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'D' THEN

UPDATE PARTIDA SET COL\_D = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'E' THEN

UPDATE PARTIDA SET COL\_E = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'F' THEN

UPDATE PARTIDA SET COL\_F = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'G' THEN

UPDATE PARTIDA SET COL\_G = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'H' THEN

UPDATE PARTIDA SET COL\_H = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

END CASE;

RESULTADO := 'EL JUGADOR DE LAS FICHAS '||COLOR\_TURNO||' HA MOVIDO LA FICHA '||FICHA\_ORIGEN||' DE '||TO\_CHAR(FIL\_O)||COL\_O||' A '||TO\_CHAR(FIL\_D)||COL\_D||'.'||' EL SIGUIENTE MOVIMIENTO ES DEL JUGADOR DE LAS FICHAS '||NUEVO\_COLOR\_TURNO||'.';

COMMIT;

RETURN RESULTADO;

END;

/

CREATE OR REPLACE FUNCTION ACTUALIZAR\_MOVIMIENTO\_JAQUE\_MATE(PART NUMBER, FIL\_O NUMBER, COL\_O VARCHAR2, FIL\_D NUMBER, COL\_D VARCHAR2)

RETURN VARCHAR2

IS

RESULTADO VARCHAR2(2000);

COLOR\_TURNO VARCHAR2(10);

COLOR\_ADVERSARIO VARCHAR2(10);

FICHA\_ORIGEN VARCHAR2(2);

JUG\_BLANCAS VARCHAR2(10);

NOMBRE\_JUG\_BLANCAS VARCHAR2(20);

JUG\_NEGRAS VARCHAR2(10);

NOMBRE\_JUG\_NEGRAS VARCHAR2(20);

GANADOR VARCHAR2(20);

PERDEDOR VARCHAR(20);

BEGIN

COLOR\_TURNO := DEVOLVER\_COLOR\_TURNO(PART);

FICHA\_ORIGEN := DEVOLVER\_FICHA\_CAMPO(PART, FIL\_O, COL\_O);

SELECT DISTINCT PARTIDA.JUGADOR\_BLANCAS, JUGADOR.NOMBRE INTO JUG\_BLANCAS, NOMBRE\_JUG\_BLANCAS FROM PARTIDA, JUGADOR WHERE ID\_PARTIDA = PART AND PARTIDA.JUGADOR\_BLANCAS = JUGADOR.ID\_JUGADOR;

SELECT DISTINCT PARTIDA.JUGADOR\_NEGRAS, JUGADOR.NOMBRE INTO JUG\_NEGRAS, NOMBRE\_JUG\_NEGRAS FROM PARTIDA, JUGADOR WHERE ID\_PARTIDA = PART AND PARTIDA.JUGADOR\_NEGRAS = JUGADOR.ID\_JUGADOR;

UPDATE PARTIDA SET RESULTADO = 'FINALIZADA' WHERE ID\_PARTIDA = PART;

CASE COLOR\_TURNO

WHEN 'BLANCAS' THEN

GANADOR := NOMBRE\_JUG\_BLANCAS;

PERDEDOR := NOMBRE\_JUG\_NEGRAS;

COLOR\_ADVERSARIO := 'NEGRAS';

UPDATE JUGADOR SET GANES = GANES+1 WHERE ID\_JUGADOR = JUG\_BLANCAS;

UPDATE JUGADOR SET PERDIDAS = PERDIDAS+1 WHERE ID\_JUGADOR = JUG\_NEGRAS;

WHEN 'NEGRAS' THEN

GANADOR := NOMBRE\_JUG\_NEGRAS;

PERDEDOR := NOMBRE\_JUG\_BLANCAS;

COLOR\_ADVERSARIO := 'BLANCAS';

UPDATE JUGADOR SET GANES = GANES+1 WHERE ID\_JUGADOR = JUG\_NEGRAS;

UPDATE JUGADOR SET PERDIDAS = PERDIDAS+1 WHERE ID\_JUGADOR = JUG\_BLANCAS;

END CASE;

CASE UPPER(COL\_O)

WHEN 'A' THEN

UPDATE PARTIDA SET COL\_A = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'B' THEN

UPDATE PARTIDA SET COL\_B = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'C' THEN

UPDATE PARTIDA SET COL\_C = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'D' THEN

UPDATE PARTIDA SET COL\_D = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'E' THEN

UPDATE PARTIDA SET COL\_E = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'F' THEN

UPDATE PARTIDA SET COL\_F = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'G' THEN

UPDATE PARTIDA SET COL\_G = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

WHEN 'H' THEN

UPDATE PARTIDA SET COL\_H = ' ' WHERE ID\_PARTIDA = PART AND FILA = FIL\_O;

END CASE;

CASE UPPER(COL\_D)

WHEN 'A' THEN

UPDATE PARTIDA SET COL\_A = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'B' THEN

UPDATE PARTIDA SET COL\_B = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'C' THEN

UPDATE PARTIDA SET COL\_C = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'D' THEN

UPDATE PARTIDA SET COL\_D = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'E' THEN

UPDATE PARTIDA SET COL\_E = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'F' THEN

UPDATE PARTIDA SET COL\_F = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'G' THEN

UPDATE PARTIDA SET COL\_G = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

WHEN 'H' THEN

UPDATE PARTIDA SET COL\_H = FICHA\_ORIGEN WHERE ID\_PARTIDA = PART AND FILA = FIL\_D;

END CASE;

RESULTADO := 'EL JUGADOR DE LAS FICHAS '||COLOR\_TURNO||' HA MOVIDO LA FICHA '||FICHA\_ORIGEN||' DE '||TO\_CHAR(FIL\_O)||COL\_O||' A '||TO\_CHAR(FIL\_D)||COL\_D||'. EN ESTE MOVIMIENTO ELIMINO AL REY DEL EQUIPO DE LAS FICHAS '||COLOR\_ADVERSARIO||'. RESULTADO: JAQUE MATE. EL GANADOR ES '||GANADOR||' Y EL PERDEDOR ES '||PERDEDOR;

COMMIT;

RETURN RESULTADO;

END;

/

CREATE OR REPLACE FUNCTION VALIDAR\_CASTILLO(PART NUMBER, FIL\_O NUMBER, COL\_O VARCHAR2, FIL\_D NUMBER, COL\_D VARCHAR2)

RETURN BOOLEAN

IS

RESULTADO BOOLEAN;

CAMPO\_AUX VARCHAR2(2);

CAMPO\_LIBRE BOOLEAN;

NUM\_COL\_O NUMBER;

NUM\_COL\_D NUMBER;

COLUMNA VARCHAR2(2);

BEGIN

RESULTADO := FALSE;

CAMPO\_LIBRE := FALSE;

IF FIL\_O = FIL\_D THEN

NUM\_COL\_O := DEVOLVER\_NUMERO\_COLUMNA(COL\_O);

NUM\_COL\_D := DEVOLVER\_NUMERO\_COLUMNA(COL\_D);

IF NUM\_COL\_O - NUM\_COL\_D = -1 OR NUM\_COL\_O - NUM\_COL\_D = 1 THEN

RESULTADO := TRUE;

RETURN RESULTADO;

END IF;

IF NUM\_COL\_D - NUM\_COL\_O > 1 THEN

FOR I IN NUM\_COL\_O+1..NUM\_COL\_D-1

LOOP

COLUMNA := DEVOLVER\_CHAR\_COLUMNA(I);

CAMPO\_AUX := DEVOLVER\_FICHA\_CAMPO(PART, FIL\_O, COLUMNA);

IF CAMPO\_AUX = ' ' THEN

CAMPO\_LIBRE := TRUE;

ELSE

RESULTADO:= FALSE;

RETURN RESULTADO;

END IF;

END LOOP;

END IF;

IF NUM\_COL\_O - NUM\_COL\_D > 1 THEN

FOR I IN NUM\_COL\_D+1..NUM\_COL\_O-1

LOOP

COLUMNA := DEVOLVER\_CHAR\_COLUMNA(I);

CAMPO\_AUX := DEVOLVER\_FICHA\_CAMPO(PART, FIL\_O, COLUMNA);

IF CAMPO\_AUX = ' ' THEN

CAMPO\_LIBRE := TRUE;

ELSE

RESULTADO:= FALSE;

RETURN RESULTADO;

END IF;

END LOOP;

END IF;

END IF;

IF LOWER(COL\_O) = LOWER(COL\_D) THEN

IF FIL\_O - FIL\_D = -1 OR FIL\_O - FIL\_D = 1 THEN

RESULTADO := TRUE;

RETURN RESULTADO;

END IF;

IF FIL\_D - FIL\_O > 1 THEN

FOR I IN FIL\_O+1..FIL\_D-1

LOOP

CAMPO\_AUX := DEVOLVER\_FICHA\_CAMPO(PART, I, COL\_O);

IF CAMPO\_AUX = ' ' THEN

CAMPO\_LIBRE := TRUE;

ELSE

RESULTADO:= FALSE;

RETURN RESULTADO;

END IF;

END LOOP;

END IF;

IF FIL\_O - FIL\_D > 1 THEN

FOR I IN FIL\_D+1..FIL\_O-1

LOOP

CAMPO\_AUX := DEVOLVER\_FICHA\_CAMPO(PART, I, COL\_O);

IF CAMPO\_AUX = ' ' THEN

CAMPO\_LIBRE := TRUE;

ELSE

RESULTADO:= FALSE;

RETURN RESULTADO;

END IF;

END LOOP;

END IF;

END IF;

IF CAMPO\_LIBRE THEN

RESULTADO := TRUE;

END IF;

RETURN RESULTADO;

END;

/

CREATE OR REPLACE FUNCTION VALIDAR\_CABALLO(PART NUMBER, FIL\_O NUMBER, COL\_O VARCHAR2, FIL\_D NUMBER, COL\_D VARCHAR2)

RETURN BOOLEAN

IS

NUM\_COL\_O NUMBER;

NUM\_COL\_D NUMBER;

RESULTADO BOOLEAN;

BEGIN

RESULTADO := FALSE;

IF FIL\_O <> FIL\_D AND COL\_O <> COL\_D THEN

NUM\_COL\_O := DEVOLVER\_NUMERO\_COLUMNA(COL\_O);

NUM\_COL\_D := DEVOLVER\_NUMERO\_COLUMNA(COL\_D);

IF (ABS(FIL\_O-FIL\_D) = 2 AND ABS(NUM\_COL\_O - NUM\_COL\_D) = 1) OR (ABS(FIL\_O-FIL\_D) = 1 AND ABS(NUM\_COL\_O - NUM\_COL\_D) = 2) THEN

RESULTADO := TRUE;

END IF;

END IF;

RETURN RESULTADO;

END;

/

create or replace function VALIDAR\_REY (FILAO NUMBER, COLO VARCHAR2, FILAD NUMBER, COLD VARCHAR2) RETURN BOOLEAN IS

FILA\_ORGIEN NUMBER := FILAO;

FILA\_DESTINO NUMBER:= FILAD;

COLUMNA\_ORIGEN VARCHAR2(10):= UPPER(COLO);

COLUMNA\_DESTINO VARCHAR2(10):= UPPER(COLD);

BEGIN

IF (COLUMNA\_ORIGEN = COLUMNA\_DESTINO AND (FILAD = FILAO-1) OR (FILAD = FILAO+1))THEN

RETURN TRUE;

end if;

IF (

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'A') AND (COLUMNA\_DESTINO='B') AND (FILA\_ORGIEN = FILA\_DESTINO) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'B') AND (COLUMNA\_DESTINO='A') OR (COLUMNA\_DESTINO= 'C') AND (FILA\_ORGIEN = FILA\_DESTINO) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'C') AND (COLUMNA\_DESTINO='D') OR (COLUMNA\_DESTINO= 'B') AND (FILA\_ORGIEN = FILA\_DESTINO) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'D') AND (COLUMNA\_DESTINO='C') OR (COLUMNA\_DESTINO= 'E') AND (FILA\_ORGIEN = FILA\_DESTINO) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'E') AND (COLUMNA\_DESTINO='D') OR (COLUMNA\_DESTINO= 'F') AND (FILA\_ORGIEN = FILA\_DESTINO) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'F') AND (COLUMNA\_DESTINO='E') OR (COLUMNA\_DESTINO= 'G') AND (FILA\_ORGIEN = FILA\_DESTINO) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'G') AND (COLUMNA\_DESTINO='F') OR (COLUMNA\_DESTINO= 'H') AND (FILA\_ORGIEN = FILA\_DESTINO) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'H') AND (COLUMNA\_DESTINO='G') AND (FILA\_ORGIEN = FILA\_DESTINO)

) THEN

RETURN TRUE;

end if;

IF (

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'A') AND (COLUMNA\_DESTINO='B') AND (FILA\_ORGIEN = FILA\_DESTINO) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'B') AND (COLUMNA\_DESTINO='A') OR (COLUMNA\_DESTINO= 'C') AND (FILAD = FILAO-1) OR (FILAD = FILAO+1) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'C') AND (COLUMNA\_DESTINO='D') OR (COLUMNA\_DESTINO= 'B') AND (FILAD = FILAO-1) OR (FILAD = FILAO+1) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'D') AND (COLUMNA\_DESTINO='C') OR (COLUMNA\_DESTINO= 'E') AND (FILAD = FILAO-1) OR (FILAD = FILAO+1) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'E') AND (COLUMNA\_DESTINO='D') OR (COLUMNA\_DESTINO= 'F') AND (FILAD = FILAO-1) OR (FILAD = FILAO+1) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'F') AND (COLUMNA\_DESTINO='E') OR (COLUMNA\_DESTINO= 'G') AND (FILAD = FILAO-1) OR (FILAD = FILAO+1) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'G') AND (COLUMNA\_DESTINO='F') OR (COLUMNA\_DESTINO= 'H') AND (FILAD = FILAO-1) OR (FILAD = FILAO+1) OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'H') AND (COLUMNA\_DESTINO='G') AND (FILAD = FILAO-1) OR (FILAD = FILAO+1)

) THEN

RETURN TRUE;

end if;

RETURN FALSE;

END;

/

CREATE OR REPLACE FUNCTION VALIDAR\_PEON (PART NUMBER, FILAO NUMBER, COLO VARCHAR2, FILAD NUMBER, COLD VARCHAR2) RETURN BOOLEAN IS

ORIGEN VARCHAR2(2);

DESTINO varchar2(2);

FILA\_ORGIEN NUMBER := FILAO;

FILA\_DESTINO NUMBER:= FILAD;

COLUMNA\_ORIGEN VARCHAR2(10):= UPPER(COLO);

COLUMNA\_DESTINO VARCHAR2(10):= UPPER(COLD);

BEGIN

ORIGEN := DEVOLVER\_FICHA\_CAMPO(PART, FILA\_ORGIEN, COLUMNA\_ORIGEN);

DESTINO := DEVOLVER\_FICHA\_CAMPO(PART, FILA\_DESTINO, COLUMNA\_DESTINO);

IF (

(ORIGEN = 'PB') AND (FILA\_ORGIEN = 2) AND ((FILA\_DESTINO = FILA\_ORGIEN+2) OR (FILA\_DESTINO= FILA\_ORGIEN+1)) AND (COLUMNA\_ORIGEN = COLUMNA\_DESTINO) OR

(ORIGEN = 'PN') AND (FILA\_ORGIEN = 7) AND ((FILA\_DESTINO = FILA\_ORGIEN-2) OR (FILA\_DESTINO= FILA\_ORGIEN-1)) AND (COLUMNA\_ORIGEN = COLUMNA\_DESTINO)

)THEN

RETURN TRUE;

end if;

IF (

((ORIGEN = 'PB') AND (FILA\_DESTINO= FILA\_ORGIEN+1)) AND (COLUMNA\_ORIGEN = COLUMNA\_DESTINO) OR

((ORIGEN = 'PN') AND (FILA\_DESTINO= FILA\_ORGIEN-1)) AND (COLUMNA\_ORIGEN = COLUMNA\_DESTINO)

)THEN

RETURN TRUE;

end if;

IF (

(

ORIGEN = 'PB' AND

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'A') AND (COLUMNA\_DESTINO = 'B') AND (FILAD = FILAO + 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'B') AND (COLUMNA\_DESTINO = 'A') OR (COLUMNA\_DESTINO = 'C') AND (FILAD = FILAO + 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'C') AND (COLUMNA\_DESTINO = 'D') OR (COLUMNA\_DESTINO = 'B') AND (FILAD = FILAO + 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'D') AND (COLUMNA\_DESTINO = 'C') OR (COLUMNA\_DESTINO = 'E') AND (FILAD = FILAO + 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'E') AND (COLUMNA\_DESTINO = 'D') OR (COLUMNA\_DESTINO = 'F') AND (FILAD = FILAO + 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'F') AND (COLUMNA\_DESTINO = 'E') OR (COLUMNA\_DESTINO = 'G') AND (FILAD = FILAO + 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'G') AND (COLUMNA\_DESTINO = 'F') OR (COLUMNA\_DESTINO = 'H') AND (FILAD = FILAO + 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'H') AND (COLUMNA\_DESTINO = 'G') AND (FILAD = FILAO - 1)

) OR(

ORIGEN = 'PN' AND

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'A') AND (COLUMNA\_DESTINO = 'B') AND (FILAD = FILAO - 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'B') AND (COLUMNA\_DESTINO = 'A') OR (COLUMNA\_DESTINO = 'C') AND (FILAD = FILAO - 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'C') AND (COLUMNA\_DESTINO = 'D') OR (COLUMNA\_DESTINO = 'B') AND (FILAD = FILAO - 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'D') AND (COLUMNA\_DESTINO = 'C') OR (COLUMNA\_DESTINO = 'E') AND (FILAD = FILAO - 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'E') AND (COLUMNA\_DESTINO = 'D') OR (COLUMNA\_DESTINO = 'F') AND (FILAD = FILAO - 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'F') AND (COLUMNA\_DESTINO = 'E') OR (COLUMNA\_DESTINO = 'G') AND (FILAD = FILAO - 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'G') AND (COLUMNA\_DESTINO = 'F') OR (COLUMNA\_DESTINO = 'H') AND (FILAD = FILAO - 1) AND DESTINO != ' ' OR

(COLUMNA\_ORIGEN != COLUMNA\_DESTINO) AND (COLUMNA\_ORIGEN = 'H') AND (COLUMNA\_DESTINO = 'G') AND (FILAD = FILAO - 1)

)

) THEN

RETURN TRUE;

end if;

RETURN FALSE;

end;

/

CREATE OR REPLACE FUNCTION VALIDAR\_ALFIL(PART NUMBER, FILAO NUMBER, COLO VARCHAR2, FILAD NUMBER, COLD VARCHAR2) RETURN BOOLEAN IS

FILA\_ORGIEN NUMBER := FILAO;

FILA\_DESTINO NUMBER:= FILAD;

COL\_O VARCHAR2(10):= UPPER(COLO);

COL\_D VARCHAR2(10):= UPPER(COLD);

NUM\_COL\_O NUMBER;

NUM\_COL\_D NUMBER;

AUX\_COL VARCHAR2(10);

ESTADO BOOLEAN;

Y NUMBER;

BEGIN

NUM\_COL\_O := DEVOLVER\_NUMERO\_COLUMNA(COL\_O);

NUM\_COL\_D := DEVOLVER\_NUMERO\_COLUMNA(COL\_D);

IF FILAO = FILAD OR COLO = COLD THEN

RETURN FALSE;

ELSE

IF (ABS(FILA\_ORGIEN-FILA\_DESTINO) <> ABS(NUM\_COL\_O-NUM\_COL\_D)) THEN

RETURN FALSE;

ELSE

ESTADO := TRUE;

IF NUM\_COL\_O < NUM\_COL\_D AND FILA\_ORGIEN < FILA\_DESTINO THEN

Y := FILA\_ORGIEN+1;

WHILE ESTADO AND Y <= FILA\_DESTINO-1 LOOP

FOR X IN NUM\_COL\_O+1..NUM\_COL\_D-1 LOOP

AUX\_COL:= DEVOLVER\_CHAR\_COLUMNA(X);

IF DEVOLVER\_FICHA\_CAMPO(PART, Y, AUX\_COL) <> ' ' THEN

ESTADO := FALSE;

RETURN ESTADO;

EXIT WHEN ESTADO = FALSE;

end if;

Y := Y + 1 ;

end loop;

end loop;

ELSIF ((NUM\_COL\_O < NUM\_COL\_D) AND (FILA\_ORGIEN > FILA\_DESTINO)) THEN

ESTADO := TRUE;

Y := FILA\_ORGIEN-1;

WHILE ESTADO AND Y >= FILA\_DESTINO+1 LOOP

FOR X IN NUM\_COL\_O+1..NUM\_COL\_D-1 LOOP

AUX\_COL := DEVOLVER\_CHAR\_COLUMNA(X);

IF DEVOLVER\_FICHA\_CAMPO(PART, Y, AUX\_COL) <> ' ' THEN

ESTADO := FALSE;

RETURN ESTADO;

EXIT WHEN ESTADO = FALSE;

end if;

Y:= Y - 1;

end loop;

end loop;

ELSIF ((NUM\_COL\_O > NUM\_COL\_D) AND (FILA\_ORGIEN < FILA\_DESTINO)) THEN

Y := FILA\_ORGIEN+1;

WHILE ESTADO AND Y <= FILA\_DESTINO-1 LOOP

FOR X IN NUM\_COL\_D+1..NUM\_COL\_O-1 LOOP

AUX\_COL:= DEVOLVER\_CHAR\_COLUMNA(X);

IF (DEVOLVER\_FICHA\_CAMPO(PART, Y, AUX\_COL) != ' ')THEN

ESTADO := FALSE;

RETURN ESTADO;

EXIT WHEN ESTADO = FALSE;

end if;

Y:= Y + 1;

end loop;

end loop;

ELSIF ((NUM\_COL\_O > NUM\_COL\_D) AND (FILA\_ORGIEN > FILA\_DESTINO)) THEN

Y := FILA\_ORGIEN-1;

WHILE ESTADO AND Y >= FILA\_DESTINO+1 LOOP

FOR X IN NUM\_COL\_D+1..NUM\_COL\_O-1 LOOP

AUX\_COL:= DEVOLVER\_CHAR\_COLUMNA(X);

IF (DEVOLVER\_FICHA\_CAMPO(PART, Y, AUX\_COL) != ' ')THEN

ESTADO := FALSE;

RETURN ESTADO;

EXIT WHEN ESTADO = FALSE;

end if;

Y:= Y - 1;

end loop;

end loop;

ELSE

RETURN FALSE;

END IF;

end if;

end if;

RETURN ESTADO;

end;

/

CREATE OR REPLACE FUNCTION VALIDAR\_REINA(PART NUMBER, FIL\_O NUMBER, COL\_O VARCHAR2, FIL\_D NUMBER, COL\_D VARCHAR2)

RETURN BOOLEAN

IS

RESULTADO BOOLEAN;

BEGIN

RESULTADO := FALSE;

IF VALIDAR\_CASTILLO(PART, FIL\_O, COL\_O, FIL\_D, COL\_D) THEN

RESULTADO := TRUE;

RETURN RESULTADO;

END IF;

IF VALIDAR\_ALFIL(PART, FIL\_O, COL\_O, FIL\_D, COL\_D) THEN

RESULTADO := TRUE;

RETURN RESULTADO;

END IF;

RETURN RESULTADO;

END;

/

CREATE OR REPLACE FUNCTION VALIDAR\_MOVIMIENTO\_O\_D(PART NUMBER, FIL\_O NUMBER, COL\_O VARCHAR2, FIL\_D NUMBER, COL\_D VARCHAR2)

RETURN BOOLEAN

IS

RESULTADO BOOLEAN;

FICHA\_CAMPO VARCHAR2(2);

BEGIN

RESULTADO := FALSE;

FICHA\_CAMPO := DEVOLVER\_FICHA\_CAMPO(PART, FIL\_O, COL\_O);

CASE FICHA\_CAMPO

WHEN 'PB' THEN

RESULTADO := VALIDAR\_PEON(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

WHEN 'CB' THEN

RESULTADO := VALIDAR\_CABALLO(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

WHEN 'AB' THEN

RESULTADO := VALIDAR\_ALFIL(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

WHEN 'RB' THEN

RESULTADO := VALIDAR\_REINA(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

WHEN 'KB' THEN

RESULTADO := VALIDAR\_REY(FIL\_O, COL\_O, FIL\_D, COL\_D);

WHEN 'TB' THEN

RESULTADO := VALIDAR\_CASTILLO(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

WHEN 'PN' THEN

RESULTADO := VALIDAR\_PEON(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

WHEN 'CN' THEN

RESULTADO := VALIDAR\_CABALLO(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

WHEN'AN' THEN

RESULTADO := VALIDAR\_ALFIL(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

WHEN'RN' THEN

RESULTADO := VALIDAR\_REINA(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

WHEN 'KN' THEN

RESULTADO := VALIDAR\_REY( FIL\_O, COL\_O, FIL\_D, COL\_D);

WHEN 'TN' THEN

RESULTADO := VALIDAR\_CASTILLO(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

END CASE;

RETURN RESULTADO;

END;

/

Anexo 3. Procedimientos

CREATE OR REPLACE PROCEDURE PROC\_CONSULTAR\_TABLERO(ID\_PART NUMBER)

IS

NOMBREB VARCHAR2(15);

APELLIDOB VARCHAR2(15);

NOMBREN VARCHAR2(15);

APELLIDON VARCHAR2(15);

MUEVEN VARCHAR2(20);

CURSOR CU\_FIL\_COL IS

SELECT FILA, COL\_A, COL\_B, COL\_C, COL\_D, COL\_E, COL\_F, COL\_G, COL\_H FROM PARTIDA WHERE ID\_PARTIDA = ID\_PART ORDER BY FILA DESC;

BEGIN

IF VALIDAR\_PARTIDA(ID\_PART) THEN

SELECT DISTINCT JUGADOR.NOMBRE, JUGADOR.APELLIDO, PARTIDA.TURNO INTO NOMBREB, APELLIDOB, MUEVEN FROM JUGADOR, PARTIDA WHERE PARTIDA.ID\_PARTIDA = ID\_PART AND JUGADOR.ID\_JUGADOR = PARTIDA.JUGADOR\_BLANCAS;

SELECT DISTINCT JUGADOR.NOMBRE, JUGADOR.APELLIDO INTO NOMBREN, APELLIDON FROM JUGADOR, PARTIDA WHERE PARTIDA.ID\_PARTIDA = ID\_PART AND JUGADOR.ID\_JUGADOR = PARTIDA.JUGADOR\_NEGRAS;

DBMS\_OUTPUT.PUT\_LINE('TABLERO DE PARTIDA: '||TO\_CHAR(ID\_PART));

DBMS\_OUTPUT.PUT\_LINE('BLANCAS: '||NOMBREB||' '||APELLIDOB||' '||

'NEGRAS: '||NOMBREN||' '||APELLIDON||' '||'MUEVEN: '||MUEVEN);

FOR CAMPOS IN CU\_FIL\_COL

LOOP

DBMS\_OUTPUT.PUT\_LINE(' \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_');

DBMS\_OUTPUT.PUT\_LINE(TO\_CHAR(CAMPOS.FILA)||' | '||CAMPOS.COL\_A||' | '||CAMPOS.COL\_B||' | '||CAMPOS.COL\_C||' | '||CAMPOS.COL\_D||' | '||CAMPOS.COL\_E||' | '||CAMPOS.COL\_F||' | '||CAMPOS.COL\_G||' | '||CAMPOS.COL\_H||' | ');

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(' \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_');

DBMS\_OUTPUT.PUT\_LINE(' A B C D E F G H ');

ELSE

DBMS\_OUTPUT.PUT\_LINE('EL ID DE PARTIDA INGRESADO NO SE ENCUENTRA REGISTRADO.');

END IF;

END;

/

CREATE OR REPLACE PROCEDURE PROC\_AGREGAR\_NUEVO\_JUGADOR(VAR\_ID VARCHAR2, VAR\_NOMBRE VARCHAR2, VAR\_APELLIDO VARCHAR2)

IS

BEGIN

IF VALIDAR\_JUGADOR(VAR\_ID) THEN

DBMS\_OUTPUT.PUT\_LINE('EL ID INGRESADO YA EXISTE, NO SE PUEDE REGISTRAR UN ID EXISTENTE.');

ELSE

INSERT INTO JUGADOR VALUES(VAR\_ID, VAR\_NOMBRE, VAR\_APELLIDO, 0, 0, 0,0);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('JUGADOR REGISTRADO SATISFACTORIAMENTE');

END IF;

END;

/

CREATE OR REPLACE PROCEDURE PROC\_CREAR\_NUEVA\_PARTIDA(JUG\_B VARCHAR2, JUG\_N VARCHAR2)

IS

NUM\_PARTIDA NUMBER;

BEGIN

IF JUG\_B <> JUG\_N THEN

IF (VALIDAR\_JUGADOR(JUG\_B) AND VALIDAR\_JUGADOR(JUG\_N)) THEN

NUM\_PARTIDA := INCIAR\_TABLERO(JUG\_B, JUG\_N);

UPDATE JUGADOR SET PARTIDAS\_JUGADAS=PARTIDAS\_JUGADAS+1 WHERE ID\_JUGADOR = JUG\_B;

UPDATE JUGADOR SET PARTIDAS\_JUGADAS=PARTIDAS\_JUGADAS+1 WHERE ID\_JUGADOR = JUG\_N;

DBMS\_OUTPUT.PUT\_LINE('SE HA CREADO NUEVA PARTIDA, NUMERO DE PARTIDA: '||TO\_CHAR(NUM\_PARTIDA));

COMMIT;

ELSE

DBMS\_OUTPUT.PUT\_LINE('LOS ID INGRESADOS NO PERTENECEN A JUGADORES REGISTRADOS.');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('LOS ID INGRESADOS DEBEN SER DIFERENTES ENTRE SI');

END IF;

END;

/

create or replace procedure pedir\_empate(par number, jug\_blancas varchar2, jug\_negras varchar2)

is

begin

if validar\_partida(par) then

if jug\_blancas <> jug\_negras then

if validar\_jugador\_partida(par, jug\_blancas) and validar\_jugador\_partida(par,jug\_negras) then

update partida set resultado = 'Finalizada' where id\_partida = par;

update jugador set empates = empates+1 where id\_jugador=jug\_blancas or id\_jugador = jug\_negras;

dbms\_output.put\_line('Empate registrado correctamente');

else

dbms\_output.put\_line('Los id ingresados no coinciden a los registradas en la partida '||to\_char(par)||'.');

end if;

else

dbms\_output.put\_line('Los id de los jugadores deben ser diferentes.');

end if;

else

dbms\_output.put\_line('La partida no existe en los registros.');

end if;

end;

/

CREATE OR REPLACE PROCEDURE PROC\_MOVER\_FICHA(PART NUMBER, FIL\_O NUMBER, COL\_O VARCHAR2, FIL\_D NUMBER, COL\_D VARCHAR2)

IS

RESULTADO VARCHAR2(1000);

BEGIN

IF VALIDAR\_PARTIDA(PART) THEN

IF VALIDAR\_PARTIDA\_PENDIENTE(PART) THEN

IF (FIL\_O >= 1 AND FIL\_O <=8) AND (FIL\_D >= 1 AND FIL\_D <= 8) THEN

IF lower(substr(COL\_O,1,1)) in ('a','b','c','d','e','f','g','h') AND lower(substr(COL\_D,1,1)) in ('a','b','c','d','e','f','g','h') THEN

IF FIL\_O = FIL\_D AND COL\_O = COL\_D THEN

DBMS\_OUTPUT.PUT\_LINE('EL CAMPO DE ORIGEN Y EL DE DESTINO NO PUEDE SER EL MISMO');

ELSE

IF VALIDAR\_COLOR\_FICHA\_ORIGEN(PART, FIL\_O, COL\_O) THEN

IF VALIDAR\_POSICION\_DESTINO(PART, FIL\_D, COL\_D) THEN

IF VALIDAR\_MOVIMIENTO\_O\_D(PART, FIL\_O, COL\_O, FIL\_D, COL\_D) THEN

IF VALIDAR\_MUERTE\_REY(PART, FIL\_D, COL\_D) THEN

RESULTADO := ACTUALIZAR\_MOVIMIENTO\_JAQUE\_MATE(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

DBMS\_OUTPUT.PUT\_LINE(RESULTADO);

ELSE

RESULTADO := ACTUALIZAR\_MOVIMIENTO(PART, FIL\_O, COL\_O, FIL\_D, COL\_D);

DBMS\_OUTPUT.PUT\_LINE(RESULTADO);

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('LA FICHA SELECCIONADA NO PUEDE REALIZAR ESE TIPO DE MOVIMIENTO.');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('EL CAMPO DE DESTINO SELECCIONADO NO ES POSIBLE');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('EL CAMPO SELECCIONADO NO CORRESPONDE A UNA FICHA DEL COLOR DE TURNO.');

END IF;

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('LAS LETRAS INGRESADAS EN LAS COLUMNAS SE SALEN DE LAS COORDENADAS DEL TABLERO');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('EL NUMERO INGRESADO EN LAS FILAS SE SALE DE LAS COORDENADAS DEL TABLERO');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('LA PARTIDA SE ENCUENTRA FINALIZADA. NO PUEDE REALIZAR MOVIMIENTOS EN UNA PARTIDA QUE NO SEA PENDIENTE.');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('LA PARTIDA INGRESADA NO EXISTE.');

END IF;

END;

/

Anexo 4. Bloques

DECLARE

VAR\_ID JUGADOR.ID\_JUGADOR%TYPE;

VAR\_NOMBRE JUGADOR.NOMBRE%TYPE;

VAR\_APELLIDO JUGADOR.APELLIDO%TYPE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('INGRESE EL ID DEL JUGADOR: ');

VAR\_ID := '&ID';

DBMS\_OUTPUT.PUT\_LINE('INGRESE EL NOMBRE DEL JUGADOR: ');

VAR\_NOMBRE := '&NOMBREJ';

DBMS\_OUTPUT.PUT\_LINE('INGRESE EL PRIMER APELLIDO DEL JUGADOR: ');

VAR\_APELLIDO := '&APELLIDOJ';

PROC\_AGREGAR\_NUEVO\_JUGADOR(VAR\_ID, VAR\_NOMBRE, VAR\_APELLIDO);

END;

/

DECLARE

JUGADOR\_BLANCAS VARCHAR2(10);

JUGADOR\_NEGRAS VARCHAR2(10);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('NUEVA PARTIDA');

DBMS\_OUTPUT.PUT\_LINE('INGRESE EL ID DEL JUGADOR QUE USARA FICHAS BLANCAS: ');

JUGADOR\_BLANCAS := '&IDBLANCAS';

DBMS\_OUTPUT.PUT\_LINE('INGRESE EL ID DEL JUGADOR QUE USARA FICHAS NEGRAS: ');

JUGADOR\_NEGRAS := '&IDNEGRAS';

PROC\_CREAR\_NUEVA\_PARTIDA(JUGADOR\_BLANCAS, JUGADOR\_NEGRAS);

END;

/

DECLARE

IDENT\_PARTIDA NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('PARA VER EL TABLERO DE UNA PARTIDA INGRESE EL ID: ');

IDENT\_PARTIDA := &PARTIDA;

PROC\_CONSULTAR\_TABLERO(IDENT\_PARTIDA);

END;

/

DECLARE

IDENT\_JUGADOR VARCHAR2(10);

RESULTADO VARCHAR2(500);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('PARA VER LAS ESTADISTICAS, INGRESE ID DEL JUGADOR: ');

IDENT\_JUGADOR := '&ID\_DEL\_JUGADOR';

RESULTADO := FUNCION\_ESTADISTICAS(IDENT\_JUGADOR);

DBMS\_OUTPUT.PUT\_LINE(RESULTADO);

END;

/

DECLARE

NUM\_PARTIDA NUMBER;

FILA\_ORIGEN NUMBER;

COL\_ORIGEN VARCHAR2(5);

FILA\_DESTINO NUMBER;

COL\_DESTINO VARCHAR2(5);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('INGRESAR EL NUMERO DE PARTIDA: ');

NUM\_PARTIDA := &PARTI;

DBMS\_OUTPUT.PUT\_LINE('INGRESAR EL NUMERO DE FILA DE ORIGEN: ');

FILA\_ORIGEN := &FIL\_O;

DBMS\_OUTPUT.PUT\_LINE('INGRESAR LA COLUMNA DE ORIGEN: ');

COL\_ORIGEN := '&COL\_O';

DBMS\_OUTPUT.PUT\_LINE('INGRESAR EL NUMERO DE FILA DE DESTINO: ');

FILA\_DESTINO := &FIL\_D;

DBMS\_OUTPUT.PUT\_LINE('INGRESAR LA COLUMNA DE DESTINO: ');

COL\_DESTINO := '&COL\_D';

PROC\_MOVER\_FICHA(NUM\_PARTIDA , FILA\_ORIGEN, COL\_ORIGEN, FILA\_DESTINO, COL\_DESTINO);

END;

/

declare

var\_partida number;

var\_jug\_blancas varchar2(10);

var\_jug\_negras varchar2(10);

begin

dbms\_output.put\_line('Solicitud de empate:');

dbms\_output.put\_line('Ingrese el id de la partida:');

var\_partida := &part;

dbms\_output.put\_line('Ingrese el id del jugador de las fichas blancas de dicha partida: ');

var\_jug\_blancas := '&jugadorBlancas';

dbms\_output.put\_line('Ingrese el id del jugador de las fichas negras de dicha partida: ');

var\_jug\_negras := '&jugadorNegras';

pedir\_empate(var\_partida, var\_jug\_blancas, var\_jug\_negras);

end;

/