Documentación de los procedimientos almacenados (PP.AA)

1. ***Actualizar el estado de la inscripción de una persona en una convocatoria en base a su nombre, apellido y el nombre asociado (curso, proyecto o beneficio).***

El procedimiento almacenado "actualizarEstadoInscripcionPorNombre" actualiza el estado de una inscripción en la base de datos según el nombre y apellido de una persona, y el nombre de un curso, proyecto o beneficio. Primero, identifica a la persona mediante su nombre y apellido en la tabla "persona", luego encuentra la convocatoria asociada al curso, proyecto o beneficio dado, y finalmente actualiza el estado de la inscripción en la tabla "inscripcion" para esa persona y convocatoria específica. Esto permite a los administradores cambiar rápidamente el estado de inscripciones utilizando nombres y apellidos y el nombre de las actividades.

```
Unset
DELIMITER //
CREATE PROCEDURE actualizarEstadoInscripcionPorNombre (
        IN p_PER_Nombre VARCHAR(45),
        IN p_PER_Apellido VARCHAR(45),
        IN p_NombreAsociado VARCHAR(45),   -- Nombre del curso, proyecto, o
beneficio
        IN p_INS_Estado ENUM('A', 'P', 'R')
)
BEGIN
        DECLARE v_PER_DocumentoIdentidad INT;
        DECLARE v_INS_IdInscripcion INT;

        -- Obtener Documento de Identidad de la persona (case-insensitive)
        SELECT PER_DocumentoIdentidad INTO v_PER_DocumentoIdentidad
        FROM persona
        WHERE LOWER(PER_Nombre) = LOWER(p_PER_Nombre)
        AND LOWER(PER_Apellido) = LOWER(p_PER_Apellido)
        LIMIT 1;
```

```
        -- Obtener ID de Inscripción correspondiente y actualizar el estado en una sola operación
        UPDATE inscripcion
        JOIN (
        SELECT con.CON_IdConvocatoria
        FROM convocatoria con
        LEFT JOIN curso cur ON cur.CON_IdConvocatoria = con.CON_IdConvocatoria AND LOWER(cur.CUR_Nombre) = LOWER(p_NombreAsociado)
        LEFT JOIN proyecto_pgp pro ON pro.CON_IdConvocatoria = con.CON_IdConvocatoria AND LOWER(pro.PRO_Nombre) = LOWER(p_NombreAsociado)
        LEFT JOIN beneficio ben ON ben.CON_IdConvocatoria = con.CON_IdConvocatoria AND LOWER(ben.BEN_Descripcion) = LOWER(p_NombreAsociado)
        LEFT JOIN curso cur2 ON cur2.CON_IdConvocatoria = con.CON_IdConvocatoria AND LOWER(cur2.CUR_Descripcion) = LOWER(p_NombreAsociado)
        WHERE cur.CON_IdConvocatoria IS NOT NULL
        OR pro.CON_IdConvocatoria IS NOT NULL
        OR ben.CON_IdConvocatoria IS NOT NULL
        OR cur2.CON_IdConvocatoria IS NOT NULL
        LIMIT 1
        ) AS sub ON inscripcion.CON_IdConvocatoria = sub.CON_IdConvocatoria
        SET inscripcion.INS_Estado = p_INS_Estado
        WHERE inscripcion.PER_DocumentoIdentidad = v_PER_DocumentoIdentidad;
END //
DELIMITER ;
```

Como ejemplo, se procederá a actualizar la inscripción número 39. Este registro pertenece a la persona con identificación CC: 1039, Laura Torres, y está relacionado con la Convocatoria: 2031, titulada "Defensa personal".

#	INS_IdInscripcion	PER_DocumentoIdentidad	CON_IdConvocatoria	INS_Estado	INS_Fecha
36	36	1036	2028	R	2024-02-12
37	37	1037	2029	A	2024-02-25
38	38	1038	2030	R	2024-02-11
39	39	1039	2031	P	2024-02-02
*	NULL	NULL	NULL	NULL	NULL

El objetivo es que el estado de la inscripción pase de “Pendiente” a “Rechazada”. Para ello se llama al procedimiento creado con anterioridad.

```
Unset
CALL actualizarEstadoInscripcionPorNombre('LAura', 'TOrreS', 'DefensA PerSonal', 'R');

SELECT * FROM inscripcion;
```

#	INS_IdInscripcio	PER_DocumentoIdentida	CON_IdConvocatori	INS_Estado	INS_Fecha
37	37	1037	2029	A	2024-02-25
38	38	1038	2030	R	2024-02-11
39	39	1039	2031	R	2024-02-02
*	NULL	NULL	NULL	NULL	NULL

inscripcion 1

Action Output

	#	Time	Action	Message
✓	1	00:06:23	CALL actualizarEstadoInscripcionPorNombre('LAura', 'TO…	1 row(s) affected
✓	2	00:06:32	SELECT * FROM inscripcion	39 row(s) returned

Nota: *Los permisos de ejecución de este procedimiento almacenado están únicamente habilitados para los jefes y los empleados.*

2. *Actualizar la fecha de un evento en base al nombre del evento (Empleado).*

El siguiente procedimiento tiene como función cambiar la fecha de un evento. Este procedimiento tiene por valores de entrada la nueva fecha del evento y el nombre del evento. Este es el código utilizado para crearlo:

```
Unset
DELIMITER $$
CREATE PROCEDURE NewDateEvent(IN NewDate DATE, IN EventName VARCHAR(45))
        BEGIN
-- Id local para buscar el id del horario del evento --
  DECLARE IdHorary INT DEFAULT 0;
-- busqueda del id del horario del evento por su nombre --
     SET IdHorary = (SELECT HOR_IdHorario FROM evento WHERE EVE_Nombre =
EventName );
-- actualización de la fecha del evento --
  UPDATE horario SET HOR_Fecha = NewDate WHERE IdHorary= HOR_IdHorario;
  END$$
DELIMITER ;
```

Realizaremos una prueba, primero miraremos la fecha del evento Conferencia de tecnología y luego la actualizaremos con el procedimiento. Para ello utilizamos el siguiente código:

```
Unset
select EVE_Nombre, HOR_Fecha from evento NATURAL JOIN horario WHERE
EVE_Nombre = 'Conferencia de Tecnología' ;
```

Lo anterior da como resultado:

	EVE_Nombre	HOR_Fecha
▶	Conferencia de Tecnología	2024-05-14

Ahora llamaremos al procedimiento y asignaremos una nueva fecha. Con el siguiente código:

```
Unset
CALL NewDateEvent('2024-05-17', 'Conferencia de Tecnología');
```

Al verificar de nuevo la fecha del evento, nos da el siguiente resultado:

	EVE_Nombre	HOR_Fecha
▶	Conferencia de Tecnología	2024-05-17

Los permisos de ejecución de este procedimiento lo tienen los jefes de cada área.

3. *Actualizar la fecha y las horas de inicio y finalización de un evento o proyecto (Jefe).*

El siguiente procedimiento tiene como función cambiar el horario de un proyecto o evento, ingresando el nombre del evento o proyecto y las nuevas especificaciones del horario. Se utilizó el siguiente código para crearlo:

```
Unset
DELIMITER $$
CREATE PROCEDURE UpdateHorary( IN NameChange VARCHAR(45), IN NewDate DATE, IN NewHourI TIME, IN NewHourF TIME)
        BEGIN
-- Conocer el id del horario --
  DECLARE IdHorary INT DEFAULT 0;
-- conocer el id del horario a partir de la consulta --
  SET IdHorary = (SELECT HOR_IdHorario
    FROM (SELECT * FROM (SELECT EVE_Nombre AS Nombre, HOR_IdHorario FROM evento) AS events_
                UNION
       SELECT * FROM (SELECT PRO_Nombre AS Nombre, HOR_IdHorario FROM proyecto_pgp) AS projects_ ) AS unionEventsAndProjects WHERE NameChange = Nombre );
-- actualización --
       UPDATE horario SET HOR_Fecha = NewDate, HOR_HoraInicio= NewHourI, HOR_HoraFinal = NewHourF WHERE HOR_IdHorario = IdHorary;
  END$$
```

```
DELIMITER ;
```

Ahora veremos el horario de un proyecto y luego lo actualizaremos con el procedimiento, veremos el horario de uqbar:

```
Unset
SELECT PRO_Nombre,HOR_Fecha, HOR_HoraInicio, HOR_HoraFinal FROM proyecto_pgp NATURAL JOIN horario WHERE PRO_Nombre = 'Uqbar';
```

	PRO_Nombre	HOR_Fecha	HOR_HoraInicio	HOR_HoraFinal
▶	Uqbar	2024-04-24	16:00:00	18:00:00

Ahora lo actualizaremos con el procedimiento y luego verificaremos está información de nuevo. Código de actualización:

```
Unset
CALL UpdateHorary('Uqbar','2024-04-27','08:00:00','11:00:00');
```

Al consultar la información anterior, tenemos que:

	PRO_Nombre	HOR_Fecha	HOR_HoraInicio	HOR_HoraFinal
▶	Uqbar	2024-04-27	08:00:00	11:00:00

Los permisos de ejecución de este procedimiento lo tienen los empleados y jefes de las áreas.

4. ***Actualizar el cargo y salario de un empleado por parte de los jefes.***

Con el propósito de realizar cambios sobre el salario y cargo de los empleados manejados por los jefes, se creó el siguiente procedimiento almacenado:

```
Unset
        DROP PROCEDURE IF EXISTS Info_empleado;
```

```
DELIMITER $$
CREATE PROCEDURE Info_empleado(DI INT, cargo VARCHAR(40), sueldo INT)
BEGIN
        -- Actualización de los valores en las columnas referentes al cargo y sueldo del empleado
        UPDATE empleado SET EMP_CARGO=cargo , EMP_Sueldo=sueldo WHERE PER_DocumentoIdentidad=DI;
END $$
DELIMITER;
```

Dos de los parámetros de entrada hacen referencia a los valores nuevos sobre las columnas ***EMP_Sueldo*** y ***EMP_Cargo*** de la tabla ***empleado*** para la actualización. El parámetro restante indica el Documento de identidad del empleado sobre el cual se realizará las actualizaciones. Por tanto, se indica como llave en el where clause de la actualización.

Para observar el efecto del procedimiento almacenado se realiza inicialmente la proyección de la tabla empleado:

```
Unset
        SELECT * FROM empleado;
```

PER_DocumentoIdentidad	INF_IdInfraestructura	ARE_IdArea	EMP_Cargo	EMP_Sueldo	EMP_TipoContrato
1041	28	1	Asistente de Bienestar	1500000	Término fijo
1042	4	4	Profesor de baile	2000000	Término indefinido
1043	27	3	Nutricionista	1800000	Término fijo
1044	5	5	Entrenador Deportivo	1600000	Término indefinido
1045	10	4	Profesor de música	1400000	Término fijo
1046	24	2	Asistente Social	1700000	Término indefinido
1047	11	5	Fisioterapeuta	1900000	Término fijo
1048	1	4	Recreacionista	1600000	Término indefinido
1049	22	2	Psicopedagogo	1800000	Término fijo
1050	27	3	Asesor de Salud Mental	2000000	Término indefinido
1051	35	4	Coordinador de Activid...	1800000	Término fijo
1052	11	5	Educador Físico	1600000	Término indefinido
1053	34	3	Monitor de Nutrición	1400000	Término fijo
1054	11	5	Entrenador Personal	1700000	Término indefinido
1055	28	1	Asistente de Bienestar...	1900000	Término fijo
1056	22	2	Terapeuta Ocupacional	1800000	Término indefinido
1057	1	3	Entrenador de Meditac...	1600000	Término fijo
1058	22	2	Psicoterapeuta	2000000	Término indefinido
1059	26	5	Coordinador de Activid...	1700000	Término fijo
NULL	NULL	NULL	NULL	NULL	NULL

Luego, se invoca el procedimiento dando los parámetros de entrada, tal que los cambios de sueldo y cargo están sobre el empleado con Documento de identidad 1041:

```
Unset
CALL Info_empleado (1041,'Jefe Bienestar',1800000);
```

Finalmente, se proyecta nuevamente la tabla para ver las actualizaciones sobre la tupla con el valor 1041 en la columna PER_DocumentoIdentidad.

```
Unset
SELECT * FROM empleado;
```

PER_DocumentoIdentidad	INF_IdInfraestructura	ARE_IdArea	EMP_Cargo	EMP_Sueldo	EMP_TipoContrato
1041	28	1	Jefe Bienestar	1800000	Término fijo
1042	4	4	Profesor de baile	2000000	Término indefinido
1043	27	3	Nutricionista	1800000	Término fijo
1044	5	5	Entrenador Deportivo	1600000	Término indefinido
1045	10	4	Profesor de música	1400000	Término fijo
1046	24	2	Asistente Social	1700000	Término indefinido
1047	11	5	Fisioterapeuta	1900000	Término fijo
1048	1	4	Recreacionista	1600000	Término indefinido
1049	22	2	Psicopedagogo	1800000	Término fijo
1050	27	3	Asesor de Salud Mental	2000000	Término indefinido
1051	35	4	Coordinador de Activid...	1800000	Término fijo
1052	11	5	Educador Físico	1600000	Término indefinido
1053	34	3	Monitor de Nutrición	1400000	Término fijo
1054	11	5	Entrenador Personal	1700000	Término indefinido
1055	28	1	Asistente de Bienestar...	1900000	Término fijo
1056	22	2	Terapeuta Ocupacional	1800000	Término indefinido
1057	1	3	Entrenador de Meditac...	1600000	Término fijo
1058	22	2	Psicoterapeuta	2000000	Término indefinido
1059	26	5	Coordinador de Activid...	1700000	Término fijo
NULL	NULL	NULL	NULL	NULL	NULL

Este procedimiento almacenado tiene los permisos concedidos para los jefes de los empleados.

5. *Crear una inscripción a un curso a partir de los datos de una persona.*

El siguiente procedimiento tiene como objetivo crear una inscripción (a un curso) de una persona que tiene su información en la base de datos, ingresando el nombre del curso y su número de identidad. El código para crear el procedimiento fue el siguiente:

```
Unset
DELIMITER $$
CREATE PROCEDURE NewRegistrationCourse(IN NameCourse VARCHAR(45) , IN IdentityDocument INT )
        BEGIN
-- variable para almacenar el id de la convocatoria del curso --
  DECLARE  CourseCallId INT DEFAULT 0;
-- conocer el id de la convocatoria del curso a partir del nombre --
   SET CourseCallId = (SELECT CON_IdConvocatoria FROM curso WHERE CUR_Nombre = NameCourse );
-- inserción --
          INSERT INTO inscripcion (PER_DocumentoIdentidad,CON_IdConvocatoria, INS_Estado,INS_Fecha)
  VALUES (IdentityDocument, CourseCallId, 'P',current_date());
  END $$
DELIMITER ;
```

Ahora revisaremos las inscripciones a un curso de una persona y luego le añadiremos una para verificar el correcto funcionamiento del procedimiento. A continuación, se muestra el código para consultar las inscripciones a un curso de una persona:

```
Unset
SELECT INS_IdInscripción, PER_DocumentoIdentidad,PER_Nombre,PER_Apellido,CUR_Nombre, CON_IdConvocatoria  FROM persona NATURAL JOIN inscripcion NATURAL JOIN convocatoria NATURAL JOIN curso
 WHERE PER_Nombre = 'Carlos' AND PER_Apellido = 'Ramirez';
```

Lo cual nos da como resulta lo siguiente:

	INS_IdInscripción	PER_DocumentoIdentidad	PER_Nombre	PER_Apellido	CUR_Nombre	CON_IdConvocatoria
▶	123	1003	Carlos	Ramirez	Técnica vocal	2003

Ahora realizaremos una nueva inscripción de la persona a otro curso y verificaremos de nuevo la información. A continuación, se muestra el código de llamada al procedimiento:

```
Unset
CALL NewRegistrationCourse ('Taekwondo',1003);
```

Verificando las inscripciones a los cursos de la persona, queda que:

	INS_IdInscripción	PER_DocumentoIdentidad	PER_Nombre	PER_Apellido	CUR_Nombre	CON_IdConvocatoria
▶	123	1003	Carlos	Ramirez	Técnica vocal	2003
	160	1003	Carlos	Ramirez	Taekwondo	2007

Los permisos de ejecución de este procedimiento son todos los usuarios de la base de datos.

6. Crear una inscripción a un proyecto a partir de los datos reales de las personas.

Ahora daremos el código con la misma funcionalidad, solo que uno para realizar una inscripción a un proyecto y otro para beneficios. Nueva inscripción proyecto:

```
Unset
DELIMITER $$
CREATE PROCEDURE NewRegistrationProject (IN NameProject VARCHAR(45) , IN IdentityDocument INT )
        BEGIN
  DECLARE ProjectCallId INT DEFAULT 0;
    SET ProjectCallId = (SELECT CON_IdConvocatoria FROM proyecto_pgp WHERE PRO_Nombre = NameProject );
         INSERT INTO inscripcion (PER_DocumentoIdentidad,CON_IdConvocatoria, INS_Estado,INS_Fecha)
  VALUES (IdentityDocument, ProjectCallId, 'P',current_date());
  END $$
DELIMITER ;
```

7. ***Crear una inscripción a un beneficio a partir de los datos reales de las personas.***

Nueva inscripción beneficio:

```
Unset
DELIMITER $$
CREATE PROCEDURE NewRegistrationBenefit (IN NameBenefit VARCHAR(45) , IN
IdentityDocument INT )
        BEGIN
   DECLARE  BenefitCallId INT DEFAULT 0;
     SET  BenefitCallId  =  (SELECT  CON_IdConvocatoria FROM beneficio WHERE
BEN_Nombre = NameBenefit );
        INSERT  INTO  inscripcion  (PER_DocumentoIdentidad,CON_IdConvocatoria,
INS_Estado,INS_Fecha)
   VALUES (IdentityDocument, BenefitCallId, 'P',current_date());
   END $$
DELIMITER ;
```

8. ***Agregar requisitos a una convocatoria.***

El procedimiento almacenado “AgregarRequisitosAConvocatoria” es utilizado para actualizar los requisitos asociados a una convocatoria en la base de datos. Sus entradas son el ID de la convocatoria que se quiere modificar y un string de números separados por comas, los cuales representan los requisitos que se desean para la convocatoria seleccionada. En primer lugar, se eliminan los requisitos que ya no están presentes en la lista proporcionada como entrada. Luego, se insertan los nuevos requisitos y se mantienen los existentes. Si un requisito ya está asociado a esa convocatoria, su asociación será actualizada.

```
Unset
DELIMITER //
CREATE PROCEDURE AgregarRequisitosAConvocatoria(
        IN p_ConvocatoriaId INT,
```

```
    IN p_Requisitos VARCHAR(255) -- Suponiendo que los IDs de requisitos se pasen como una lista separada por comas
)
BEGIN
    -- Eliminar los requisitos que ya no están en la lista de entrada
    DELETE FROM convocatoria_requisito
    WHERE CON_IdConvocatoria = p_ConvocatoriaId
    AND REQ_IdRequisito NOT IN (SELECT REQ_IdRequisito FROM requisito WHERE FIND_IN_SET(REQ_IdRequisito, p_Requisitos));

    -- Insertar nuevos requisitos y mantener los existentes
    INSERT INTO convocatoria_requisito(CON_IdConvocatoria, REQ_IdRequisito)
    SELECT p_ConvocatoriaId, REQ_IdRequisito FROM requisito WHERE FIND_IN_SET(REQ_IdRequisito, p_Requisitos)
    ON DUPLICATE KEY UPDATE CON_IdConvocatoria = p_ConvocatoriaId;

END //
DELIMITER ;
```

A continuación, se presenta un ejemplo de cómo este procedimiento puede ser utilizado:

Supongamos que se desea agregar requisitos con IDs 1,3 y 5 a una convocatoria con el ID 2001, cuyos requisitos actuales son:

#	CON_IdConvocatori	REQ_IdRequisit
1	2001	1
2	2001	2
3	2001	3

El procedimiento sería ejecutado de la siguiente manera:

```
Unset
CALL AgregarRequisitosAConvocatoria(2001, '1,3,5');

SELECT * FROM convocatoria_requisito;
```

Tras la correcta ejecución del procedimiento almacenado, se procede a revisar los requisitos asociados a la convocatoria con ID 2001 para corroborar que los registros se actualizaron correctamente.

#	CON_IdConvocatori	REQ_IdRequisit
1	2001	1
2	2001	3
3	2001	5
4	2002	1
5	2002	2

Nota: *Los permisos de ejecución de este procedimiento almacenado están únicamente habilitados para los jefes.*

9. Registrar la asistencia de los empleados a las reuniones (Jefes).

Este procedimiento almacenado inserta nuevas tuplas que registran la asistencia de los empleados a las reuniones previstas:

```
Unset
        DROP PROCEDURE IF EXISTS insrt_asistenciaReuniones;
        DELIMITER $$
        CREATE PROCEDURE insrt_asistenciaReuniones(IDReunion INT, IDEmpleado
INT)
        BEGIN
                -- Inserción de los parámetros de entrada sobre la tabla asistencia
```

```
        INSERT INTO asistencia VALUES (IDReunion,IDEmpleado);
END $$
DELIMITER ;
```

Los parámetros de entrada indican el ID de la reunión acordada y el Documento de Identidad del empleado asistente. Por tanto, los parámetros ***IDReunion*** y ***IDEmpleado*** son de tipo de dato numérico **INT**.

Se ingresa proyecta inicialmente la tabla con el fin de visualizar la función del procedimiento almacenado y observa la diferencia al momento de llamar el procedimiento:

```
Unset
        SELECT * FROM asistencia;
```

REU_IdReunion	PER_DocumentoIdentidad
1	1041
10	1041
4	1042
10	1042
4	1043
4	1045
4	1046
10	1046
1	1048
1	1051
10	1051
10	1052
1	1055
1	1059
10	1059
NULL	NULL

Se realiza el llamado al procedimiento creado ingresando por parámetros los valores para ser insertados:

```
Unset
CALL insrt_asistenciaReuniones (8,1046);
```

Nuevamente se proyecta la tabla y se visualiza la inserción de los valores ingresados por parámetros al procedimiento almacenado:

```
Unset
SELECT * FROM asistencia;
```

REU_IdReunion	PER_DocumentoIdentidad
1	1041
10	1041
4	1042
10	1042
4	1043
4	1045
4	1046
8	1046
10	1046
1	1048
1	1051
10	1051
10	1052
1	1055
1	1059
10	1059
NULL	NULL

113 08:31:28 SELECT * FROM asistencia LIMIT 0, 2000 16 row(s) returned 0.000 sec / 0.000 sec

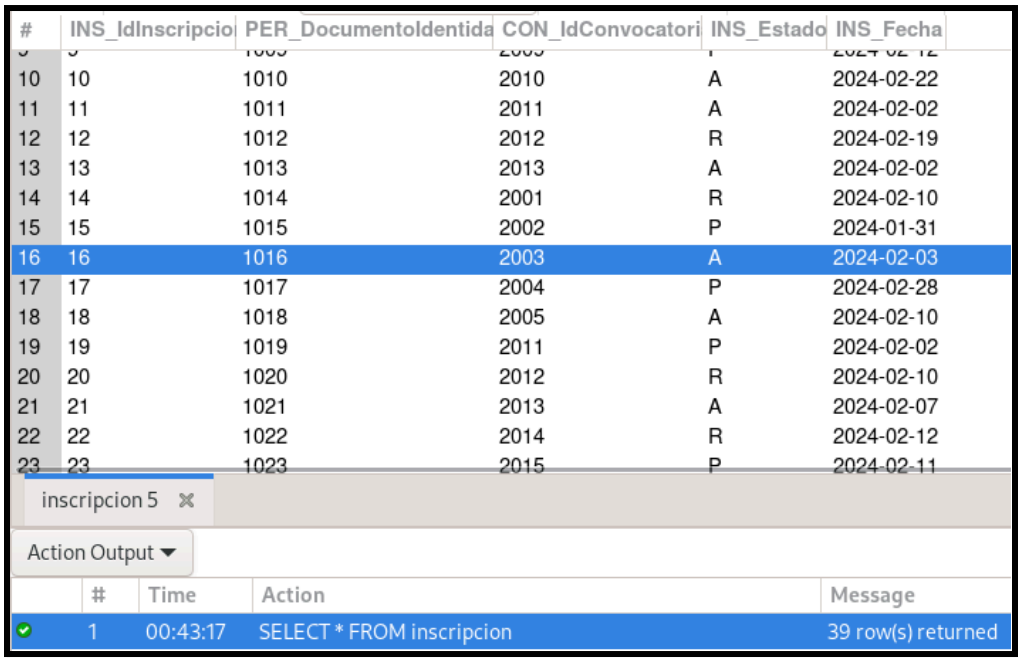
Inicialmente la proyección retornaba 15 filas, luego de realizar el llamado del procedimiento almacenado devuelve 16 filas, tal que la nueva fila hace referencia al efecto de inserción del procedimiento con los valores dados por parámetro. *Este procedimiento almacenado tiene los permisos concedidos para los jefes de los empleados.*

10. Eliminar una inscripción específica.

El procedimiento almacenado EliminarInscripcion es utilizado para eliminar una inscripción de la base de datos, identificada por su ID de inscripción. El procedimiento ejecuta una instrucción DELETE en la tabla inscripcion donde el ID de inscripción coincide con el valor proporcionado como parámetro.

```
Unset
DELIMITER //
CREATE PROCEDURE EliminarInscripcion(
        IN p_InsId INT
)
BEGIN
        DELETE FROM inscripcion WHERE INS_IdInscripcion = p_InsId;
END //
DELIMITER ;
```

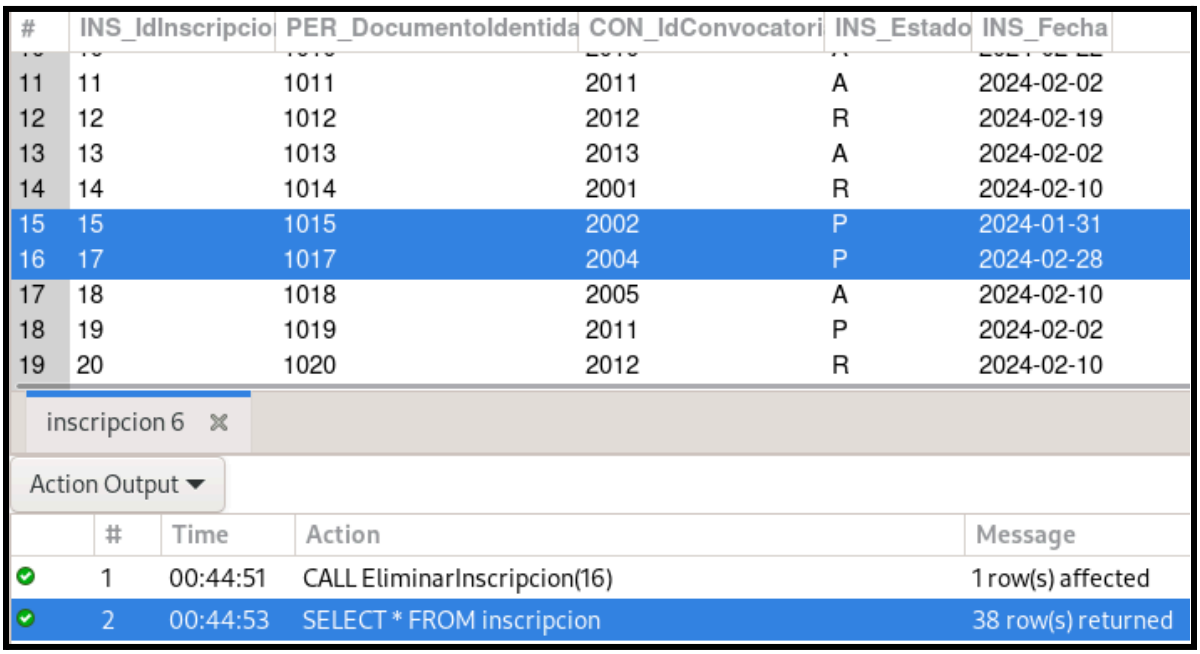
A continuación, se muestra cómo se utilizaría este procedimiento: supongamos que se desea eliminar la inscripción con el ID 16.



#	INS_IdInscripcio	PER_DocumentoIdentida	CON_IdConvocatori	INS_Estado	INS_Fecha
10	10	1010	2010	A	2024-02-22
11	11	1011	2011	A	2024-02-02
12	12	1012	2012	R	2024-02-19
13	13	1013	2013	A	2024-02-02
14	14	1014	2001	R	2024-02-10
15	15	1015	2002	P	2024-01-31
16	16	1016	2003	A	2024-02-03
17	17	1017	2004	P	2024-02-28
18	18	1018	2005	A	2024-02-10
19	19	1019	2011	P	2024-02-02
20	20	1020	2012	R	2024-02-10
21	21	1021	2013	A	2024-02-07
22	22	1022	2014	R	2024-02-12
23	23	1023	2015	P	2024-02-11

Para esto se ejecutaría el procedimiento de la siguiente manera:

```
Unset
SELECT * FROM inscripcion;
```



#	INS_IdInscripcio	PER_DocumentoIdentida	CON_IdConvocatori	INS_Estado	INS_Fecha
11	11	1011	2011	A	2024-02-02
12	12	1012	2012	R	2024-02-19
13	13	1013	2013	A	2024-02-02
14	14	1014	2001	R	2024-02-10
15	15	1015	2002	P	2024-01-31
16	17	1017	2004	P	2024-02-28
17	18	1018	2005	A	2024-02-10
18	19	1019	2011	P	2024-02-02
19	20	1020	2012	R	2024-02-10

inscripcion 6

Action Output

	#	Time	Action	Message
✓	1	00:44:51	CALL EliminarInscripcion(16)	1 row(s) affected
✓	2	00:44:53	SELECT * FROM inscripcion	38 row(s) returned

Se puede evidenciar que el registro cuyo ID de inscripción era el 16 ya no se encuentra presente dentro de la tabla después de llamar al procedimiento almacenado, por lo que el procedimiento cumplió con éxito su tarea.

Nota: *Los permisos de ejecución de este procedimiento almacenado están únicamente habilitados para los jefes y los empleados.*

11. ***Eliminar un empleado de la base de datos, lo que incluye eliminar sus registros de asistencia y su información laboral.***

El procedimiento almacenado “EliminarEmpleado” se encarga de eliminar a un empleado de la base de datos, identificado por su número de documento de identidad. Primero, elimina los registros de asistencia asociados al empleado en la tabla asistencia. Luego, elimina al empleado de la tabla “empleado”.

```
Unset
DELIMITER //
CREATE PROCEDURE EliminarEmpleado(
      IN p_DocumentoIdentidad INT
)
BEGIN
      -- Eliminar registros de asistencia del empleado
      DELETE FROM asistencia WHERE PER_DocumentoIdentidad =
p_DocumentoIdentidad;

      -- Eliminar registros de cursos a dictar
      DELETE FROM empleado_curso WHERE PER_DocumentoIdentidad =
p_DocumentoIdentidad;

      -- Eliminar al empleado
      DELETE FROM empleado WHERE PER_DocumentoIdentidad =
p_DocumentoIdentidad;
END //
DELIMITER ;
```

Supongamos que se desea eliminar al empleado con número de documento de identidad 1042, el cual está asociado a "Andrea Perez" y que actualmente se encuentra impartiendo el curso cuyo ID es 1 (Salsa y merengue).

#	PER_DocumentoIdentida	INF_IdInfraestructur	ARE_IdArea	EMP_Cargo	EMP_Sueldo	EMP_TipoContrat
1	1041	28	1	Asistente de Bienestar	1500000	Término fijo
2	1042	4	4	Profesor de baile	2000000	Término indefinido
3	1043	27	3	Nutricionista	1800000	Término fijo
4	1044	5	5	Entrenador Deportivo	1600000	Término indefinido

#	PER_DocumentoIdentida	CUR_IdCurso
1	1042	1
2	1044	7
3	1044	9
4	1045	2
5	1052	8
6	1054	10
*	NULL	NULL

Se ejecutaría el procedimiento de la siguiente manera:

```
Unset
CALL EliminarEmpleado(1042);


SELECT * FROM empleado;

SELECT * FROM empleado_curso;
```

De esta forma, se eliminarían todos los registros de asistencia relacionados con el empleado, los cursos a los que este está asociado y luego se eliminaría al empleado de la base de datos.

#	PER_DocumentoIdentida	INF_IdInfraestructur	ARE_IdArea	EMP_Cargo	EMP_Sueldo	EMP_TipoContrat
1	1041	28	1	Asistente de Bienestar	1500000	Término fijo
2	1043	27	3	Nutricionista	1800000	Término fijo
3	1044	5	5	Entrenador Deportivo	1600000	Término indefinido
4	1045	10	4	Profesor de música	1400000	Término fijo
5	1046	24	2	Asistente Social	1700000	Término indefinido
6	1047	11	5	Fisioterapeuta	1900000	Término fijo

empleado 13

Action Output

#	Time	Action	Message
13	01:04:21	SELECT * FROM empleado_curso	5 row(s) returned
14	01:04:35	SELECT * FROM empleado	18 row(s) returned

#	PER_DocumentoIdentida	CUR_IdCurso
1	1044	7
2	1044	9
3	1045	2
4	1052	8
5	1054	10
*	NULL	NULL

Nota: *Los permisos de ejecución de este procedimiento almacenado están únicamente habilitados para los jefes.*

12. Eliminar un curso de la base de datos según su nombre.

El siguiente procedimiento tiene como función eliminar un curso, tiene como parámetro de entrada el nombre del curso. El código utilizado para el crear el procedimiento es el siguiente:

```
Unset
DELIMITER $$
CREATE PROCEDURE DeleteCourse(IN NameCourse VARCHAR(45))
        BEGIN
-- Eliminar el curso --
  DELETE FROM curso WHERE CUR_Nombre = NameCourse;
  END $$
DELIMITER ;
```

Ahora para verificar el funcionamiento vamos a insertar un curso, verificarlo en la tabla de cursos y luego borrarlo con el procedimiento. El código para insertar y luego verificar que está en la tabla curso es el siguiente:

```
Unset
INSERT INTO convocatoria (CON_IdConvocatoria , CON_FechaInicio , CON_FechaFin,
CON_Tipo) VALUES (2032,'2024-01-27','2024-02-20','C');
```

```
INSERT INTO curso(ARE_IdArea, CON_IdConvocatoria ,CUR_Cupos, CUR_Nombre,CUR_Descripcion) VALUES (5,2032,20,'Natación','Curso basico de natación para la comunidad educativa');
SELECT * FROM curso WHERE CUR_Nombre = 'Natación';
```

Al buscar el curso en la tabla de curso, nos da como resultado el siguiente:

CUR_IdCurso	ARE_IdArea	CON_IdConvocatoria	CUR_Cupos	CUR_Nombre	CUR_Descripcion
12	5	2032	20	Natación	Curso basico de natación para la comunidad ed...
NULL	NULL	NULL	NULL	NULL	NULL

Luego de esto vamos a utilizar el procedimiento para eliminarlo.El código para llamar el procedimiento es el siguiente:

```
Unset
CALL DeleteCourse('Natación');
```

Al buscar del nuevo el curso en la tabla curso, da como resultado:

CUR_IdCurso	ARE_IdArea	CON_IdConvocatoria	CUR_Cupos	CUR_Nombre	CUR_Descripcion
NULL	NULL	NULL	NULL	NULL	NULL

Este procedimiento tiene permisos de ejecución para los jefes de cada área.

13. ***Eliminar un beneficio asociado a una convocatoria específica, identificados por sus respectivos IDs.***

El procedimiento a continuación cumple con el borrado de un beneficio indicado, tal que recibe por parámetros de entrada el ID del beneficio, así como el ID de la convocatoria específica de la cual se quiere quitar el beneficio. Al ser posible la existencia de un mismo beneficio en distintas convocatorias, se tiene en cuenta la convocatoria específica para la cual ese beneficio indicado no se ofrecerá:

El procedimiento almacenado está dado por:

```
Unset
DROP PROCEDURE IF EXISTS del_beneficio;

DELIMITER $$

CREATE PROCEDURE del_beneficio(id_benef INT, id_convo INT)

BEGIN

-- Borrado indicado las llaves de la condición como los parámteros de
entrada

DELETE FROM beneficio WHERE BEN_IdBeneficio=id_benef AND
CON_IdConvocatoria=id_convo;

END $$

DELIMITER ;
```

Por medio de los parámetros de entrada (IN) ***id_benf*** y ***id_convo*** se indica el ID del beneficio y de la convocatoria sobre la cual este beneficio no existirá.

La sentencia de borrado está dada por la expresión:

```
Unset
DELETE FROM beneficio WHERE BEN_IdBeneficio=id_benef AND

CON_IdConvocatoria=id_convo;
```

Para probar el funcionamiento del procedimiento almacenado, se realiza la inserción de una nueva tupla sobre la tabla beneficio:

```
Unset
INSERT INTO beneficio VALUES (11,1,2020,50,'Nuevo

beneficio','2024-03-01','2024-06-30','A');
```

Se realiza la proyección para ver la nueva tupla:

```
Unset
SELECT * FROM beneficio;
```

BEN_IdBeneficio	ARE_IdArea	CON_IdConvocatoria	BEN_Cupos	BEN_Descripcion	BEN_Inicio	BEN_Finalizacion	BEN_TipoBeneficio
1	1	2011	100	Descuento en matrícula	2024-02-28	2024-06-02	M
2	1	2012	101	Descuento en transporte	2024-02-28	2024-06-02	T
3	2	2013	102	Descuento en residencia universitaria	2024-02-28	2024-06-02	M
4	2	2014	103	Subsidio de alimentos	2024-02-28	2024-06-02	A
5	1	2015	104	Beca para deportistas	2024-02-28	2024-06-02	M
6	1	2016	105	Beca por investigación	2024-02-28	2024-06-02	M
7	1	2017	106	Subsidio para prácticas profesionales	2024-02-28	2024-06-02	M
8	1	2018	107	Beca para estudiantes de bajos recursos	2024-02-28	2024-06-02	M
9	1	2019	108	Beca para estudios de posgrado	2024-02-28	2024-06-02	M
10	2	2020	109	Subsidio para material tecnológico	2024-02-28	2024-06-02	M
11	1	2020	50	Nuevo beneficio	2024-03-01	2024-06-30	A
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Luego, se realiza el llamado del procedimiento almacenado con la sentencia:

```
Unset
CALL del_beneficio (11,2020);
```

Finalmente se realiza nuevamente la proyección de la tabla para ver el efecto de borrado sobre la tupla indicada:

```
Unset
SELECT * FROM beneficio;
```

BEN_IdBeneficio	ARE_IdArea	CON_IdConvocatoria	BEN_Cupos	BEN_Descripcion	BEN_Inicio	BEN_Finalizacion	BEN_TipoBeneficio
1	1	2011	100	Descuento en matrícula	2024-02-28	2024-06-02	M
2	1	2012	101	Descuento en transporte	2024-02-28	2024-06-02	T
3	2	2013	102	Descuento en residencia universitaria	2024-02-28	2024-06-02	M
4	2	2014	103	Subsidio de alimentos	2024-02-28	2024-06-02	A
5	1	2015	104	Beca para deportistas	2024-02-28	2024-06-02	M
6	1	2016	105	Beca por investigación	2024-02-28	2024-06-02	M
7	1	2017	106	Subsidio para prácticas profesionales	2024-02-28	2024-06-02	M
8	1	2018	107	Beca para estudiantes de bajos recursos	2024-02-28	2024-06-02	M
9	1	2019	108	Beca para estudios de posgrado	2024-02-28	2024-06-02	M
10	2	2020	109	Subsidio para material tecnológico	2024-02-28	2024-06-02	M
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Además de ADMIN de la base de datos, los jefes tienen permisos para acceder a este procedimiento almacenado.

14. Eliminar un proyecto de PGP especificado por su nombre y el ID de la convocatoria asociada.

Al igual que el procedimiento anterior este procedimiento almacenado realiza un borrado, no obstante sobre la tabla proyecto pgp. Acepta como parámetros de entrada el nombre del proyecto pgp y el id de la convocatoria, donde no se va a ofrecer ese proyecto pgp.

La creación del procedimiento almacenado está dada por:

```
Unset
DROP PROCEDURE IF EXISTS del_proyectopgp;
DELIMITER %%
CREATE PROCEDURE del_proyectopgp(nom_proy VARCHAR(40), id_convo INT)
BEGIN
        -- Declaración de la variable
        DECLARE id_proy INT;
  -- Asignación del valor a la variable
     SELECT PRO_IdProyectoPGP INTO id_proy FROM proyecto_pgp WHERE
PRO_Nombre=nom_proy AND CON_IdConvocatoria=id_convo;
  -- Borrado con la llave indicada
  DELETE FROM proyecto_pgp WHERE PRO_IDProyectoPGP=id_proy;
END %%
DELIMITER ;
```

Aquí nom_proy y id_convo son los parámetros de entrada. Además, se implementa una variable de procedimiento la cual se declara con ***DECLARE id_proy IN***T, es decir es de tipo de dato entero. Los parámetros de entrada indican la condición para la proyección del ID del proyecto pgp, tal que este se almacena en la variable ***id_proy*** previamente definida.

Así mismo, se realiza el borrado sobre la tabla proyecto_pgp donde se indica en el where clause el id del proyecto pgp como el valor asignado sobre la variable ***id_proy*** a partir de la proyección.

Para probar el funcionamiento del procedimiento almacenado, se realiza la inserción de una nueva tupla sobre la tabla proyecto pgp:

```
Unset
INSERT INTO proyecto_pgp VALUES (11,2,21,2026,'Nuevo proyecto pgp','New project','Sem');
```

Y se realiza la proyección de la tabla por verificación:

```
Unset
SELECT * FROM proyecto_pgp;
```

PRO_IdProyectoPGP	ARE_IdArea	HOR_IdHorario	CON_IdConvocatoria	PRO_Descripción	PRO_Nombre	PRO_TipoProyecto
2	2	22	2022	El grupo TLÖN busca plantear soluciones de inte…	Tlön	Sem
3	2	23	2023	Convocatoria Nacional de Apoyo a la Difusión d…	JCUN 2023	Ini
4	2	24	2024	Las líneas de investigación del grupo se clasifica…	BIOT	Gru
5	2	25	2025	El grupo de investigación se enfoca en el estudi…	UNSECURELAB	Sem
6	2	26	2026	Este grupo se centra en la aplicación de tecnolo…	CyberTech	Ini
7	2	27	2027	QuantumSafe es un grupo de investigación que …	QuantumSafe	Gru
8	2	28	2028	GreenIT es una iniciativa que busca investigar y…	GreenIT	Sem
9	2	29	2029	EduTech es un grupo de investigación que se e…	EduTech	Ini
10	2	30	2030	BioInformatics es un grupo de investigación inte…	BioInformatics	Gru
11	2	21	2026	Nuevo proyecto pgp	New project	Sem
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Se hace el llamado del procedimiento ingresando por parámetros el nombre del proyecto pgp como *'New project'* y el ID de la convocatoria *2026*, con el fin de borrar la tupla insertada previamente.

```
Unset
CALL del_proyectopgp ('New project',2026);
```

Realizando la proyección total de la tabla afectada , se observa la no existencia de la tupla indicada por parámetros en el procedimiento almacenado:

```
Unset
SELECT* FROM proyecto_pgp;
```

PRO_IdProyectoPGP	ARE_IdArea	HOR_IdHorario	CON_IdConvocatoria	PRO_Descripción	PRO_Nombre	PRO_TipoProyecto
1	2	21	2021	El único grupo estudiantil de la Universidad Naci...	Uqbar	Gru
2	2	22	2022	El grupo TLÖN busca plantear soluciones de inte...	Tlön	Sem
3	2	23	2023	Convocatoria Nacional de Apoyo a la Difusión d...	JCUN 2023	Ini
4	2	24	2024	Las líneas de investigación del grupo se clasifica...	BIOT	Gru
5	2	25	2025	El grupo de investigación se enfoca en el estudi...	UNSECURELAB	Sem
6	2	26	2026	Este grupo se centra en la aplicación de tecnolo...	CyberTech	Ini
7	2	27	2027	QuantumSafe es un grupo de investigación que ...	QuantumSafe	Gru
8	2	28	2028	GreenIT es una iniciativa que busca investigar y...	GreenIT	Sem
9	2	29	2029	EduTech es un grupo de investigación que se e...	EduTech	Ini
10	2	30	2030	BioInformatics es un grupo de investigación inte...	BioInformatics	Gru
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Además de ADMIN de la base de datos, los jefes tiene permisos para acceder a este procedimiento almacenado.

15. Obtener el número de inscritos en cada curso para el año seleccionado.

Este procedimiento almacenado “ObtenerInscripcionesPorCursoAnio” toma un año como parámetro de entrada y devuelve una lista de cursos junto con el número de personas inscritas en cada curso para el año especificado. Filtra las inscripciones aprobadas y agrupa los resultados por nombre del curso, contando cuántas personas están inscritas en cada curso dentro del año dado.

```
Unset
DELIMITER //
CREATE PROCEDURE ObtenerInscripcionesPorCursoAnio(IN p_anio INT)
BEGIN
	SELECT
	CUR_Nombre AS 'Nombre curso',
	COUNT(PER_DocumentoIdentidad) AS 'Personas inscritas'
	FROM
	(SELECT * FROM inscripcion WHERE INS_Estado = 'A') AS aprobados
	NATURAL JOIN convocatoria NATURAL JOIN curso
	WHERE
	YEAR(CON_FechaInicio) = p_anio
	GROUP BY
	CUR_Nombre;
```

```
END //
DELIMITER ;
```

Se puede evidenciar el funcionamiento de este procedimiento almacenado con el siguiente ejemplo. Suponga que se desea consultar los inscritos a los cursos ofrecidos en el año "2024". Llamamos al procedimiento almacenado de la siguiente manera:

```
Unset
CALL ObtenerInscripcionesPorCursoAnio(2024);
```

	Nombre curso	Personas inscritas
▶	Salsa y Merengue	1
	Técnica vocal	1
	Danza Contemporánea	1
	Taekwondo	1
	Tenis de campo	1

Nota: *Todos los usuarios con acceso a la base de datos puede ejecutar este procedimiento almacenado.*

16. Obtener los estudiantes de un profesor especificado por su nombre y apellido.

El siguiente procedimiento tiene como valores de entrada el nombre y apellido de un trabajador asignado a un curso, y da como resultado los estudiantes que hacen parte de ese curso al que el profesor esté asignado, el código para realizar este procedimiento va de la siguiente manera:

```
Unset
DELIMITER $$
CREATE PROCEDURE StudentsTeacher(NameTeacher VARCHAR(45),
LastNameTeacher VARCHAR(45))
    BEGIN
```

```
SELECT    PER_Nombre,    PER_Apellido,    PER_CorreoElectronico,
PER_Telefono
FROM
    persona
        NATURAL JOIN
       (SELECT  *  FROM  inscripcion  WHERE  INS_Estado  =  'A')  AS
aprobados
        NATURAL JOIN
    convocatoria
        NATURAL JOIN
    curso
WHERE
    (SELECT CUR_IdCurso
        FROM
            persona
                NATURAL JOIN
            empleado
                NATURAL JOIN
            empleado_curso
                NATURAL JOIN
            curso
        WHERE
            PER_Nombre = NameTeacher
                        AND  PER_Apellido  =  LastNameTeacher)  =
CUR_IdCurso;
     END $$
DELIMITER ;
```

Ahora realizaremos una prueba, le pasaremos al procedimiento los nombres de un profesor para ver su retorno. El código utilizado para llamar el procedimiento fue el siguiente:

```
Unset
CALL StudentsTeacher('Andrea','Pérez');
```

El resultado de llamar el procedimiento fue el siguiente:

	PER_Nombre	PER_Apellido	PER_CorreoElectronico	PER_Telefono
▶	María	Pérez	MariaP1001@unal.edu.co	3202011122

Lo cual da un resultado correcto, *para este procedimiento tiene permisos de ejecución el usuario jefes y usuario empleado.*

17. Proyección de los eventos asignados para un año específico, indicado por parámetro de entrada.

El siguiente procedimiento almacenado supervisa la entrada de un string de 4 caracteres, tal que indique el año para el cual se quieren visualizar todos los eventos previstos:

```
Unset
        DROP PROCEDURE IF EXISTS EVENT_YEAR;

        DELIMITER %%

        CREATE PROCEDURE EVENT_YEAR (year VARCHAR(4))

        BEGIN

                IF length(year)=4 THEN

                        SELECT

                                EV.EVE_idEvento AS 'ID del Evento',

                                EV.EVE_Nombre AS 'Nombre evento',
```

```
                HOR_Fecha AS 'Fecha evento',
                INF_Nombre 'Nombre del Lugar del evento',
                INF_Locacion 'Información locación',
                ARE_Nombre 'Área líder del evento'
            FROM
                evento EV
                    JOIN
                horario HOR ON EV.HOR_idHorario = HOR.HOR_idHorario
                    JOIN
                area AR ON EV.ARE_IdArea = AR.ARE_IdArea
                    JOIN
                infraestructura INF ON INF.INF_idInfraestructura =
HOR.INF_idInfraestructura
            WHERE
                YEAR(HOR_Fecha) = year;
        END IF;
    END %%
    DELIMITER ;
```

Tal que la información proyectada por el procedimiento almacenada pertenece a distintas tablas, se hace uso de **JOINS** para relacionar las tablas y poder extraer las columnas necesarias. Así mismo, se recalca el uso de la función **YEAR()** con el fin de extraer el año de la columna ***HOR_Fecha*** con tipo de dato **DATE**, y así compararlo con el valor dado por parámetro al procedimiento almacenado.

Para verificar el funcionamiento del procedimiento almacenado, se realiza la invocación del mismo:

```
Unset
CALL EVENT_YEAR ('2024');
```

Obteniendo el siguiente resultado:

ID del Evento	Nombre evento	Fecha evento	Nombre del Lugar del evento	Información locación	Área líder del evento
1	Conferencia de Tecnología	2024-05-14	Polideportivo	Cerca a la Biblioteca Central	Área de Acompañamiento Integral
2	Taller de Robótica	2024-02-08	Cancha de Fútbol	Detrás del edificio CyT y cerca del Estadio	Área de Acompañamiento Integral
4	Feria de Empleo	2024-03-30	Concha acústica	En frente del Estadio	Área de Acompañamiento Integral
10	Competencia de Programación	2024-02-09	Áreas Verdes(La Playita)	Detrás del Edificio insignia de Ingeniería Julio Ga...	Área de Acompañamiento Integral
11	Clase Magistral de Matemáticas	2024-02-29	Cancha de Fútbol	Detrás del edificio CyT y cerca del Estadio	Área de Acompañamiento Integral
12	Encuentro de Egresados	2024-03-29	Cancha de Tenis	Cerca del Estadio	Área de Acompañamiento Integral
13	Workshop de Innovación	2024-05-04	Concha acústica	En frente del Estadio	Área de Acompañamiento Integral
14	Mesa Redonda de Filosofía	2024-03-18	Estadio Alfonso López	Por la ruta hacia la hemeroteca	Área de Acompañamiento Integral
16	Presentación de Proyectos de Ingeniería	2024-04-16	Hemeroteca	Por la salida de la Carrera 45	Área de Acompañamiento Integral

73 19:43:57 CALL EVENT_YEAR ('2024') 20 row(s) returned 0.047 sec / 0.000 sec

18. Actualizar los datos secundarios de las personas de la universidad por las personas de la comunidad universitaria (NO directivos).

El siguiente procedimiento almacenado permite la actualización de datos personales generales como el teléfono y correo electrónico, a través del ingreso de estos valores por parámetro. Así mismo, entre los parámetros se indica el número de identificación de la persona sobre la cual se realizarán los cambios. Se debe tener en cuenta el establecimiento de una condición, que supervisa por búsqueda de caracteres o strings la coincidencia de los valores ingresados como correo electrónico como un correo institucional. Por tanto, su dominio debe ser '@unal.edu.co', como se indica con ***LIKE '%@unal.edu.co'.***

```
Unset
DROP PROCEDURE IF EXISTS UPDATE_GeneralData;

DELIMITER %%
```

```
CREATE PROCEDURE UPDATE_GeneralData (DI INT, Correo_E VARCHAR(40),Telefono BIGINT)
BEGIN
    DECLARE correo VARCHAR(40);
  SET correo = Correo_E;
    IF correo LIKE '%@unal.edu.co' THEN
        UPDATE persona SET PER_CorreoElectronico=Correo_E, PER_Telefono=Telefono WHERE PER_DocumentoIdentidad=DI;
    END IF;
END %%
DELIMITER ;
```

Para constatar la efectividad de la condición establecida para el procedimiento almacenado, y el funcionamiento del mismo, se realizan dos invocaciones del PA, tal que en una de ellas se ingresa sobre el valor de correo electrónico un dominio incorrecto.

Inicialmente, al realizar la proyección de la tabla persona se obtiene:

```
Unset
SELECT * FROM persona;
```

PER_DocumentoIdentidad	PER_TipoDocumento	PER_Nombre	PER_Apellido	PER_CorreoElectronico	PER_Edad	PER_Telefono
1001	CE	María	Pérez	Maria1001@unal.edu.co	19	310310310
1002	CE	María	Pérez	JuanG1002@unal.edu.co	19	3102022233
1003	CE	Carlos	Ramirez	CarlosR1003@unal.edu.co	20	3002033344
1004	DNI	Laura	Martínez	LauraM1004@unal.edu.co	22	3152044455
1005	CC	Diego	González	DiegoG1005@unal.edu.co	26	3012055566
1006	TI	Ana	López	AnaL1006@unal.edu.co	17	3122066677
1007	CE	Luis	Patiño	LuisP1007@unal.edu.co	23	3132077788
1008	DNI	María	Sánchez	MariaS1008@unal.edu.co	32	3142088899
1009	CC	José	Hernández	JoseH1009@unal.edu.co	23	3152099900
1010	TI	Andrea	Díaz	AndreaD1010@unal.edu.co	16	3162000011

Se llama el procedimiento almacenado ingresando por parámetros un correo electrónico con un dominio incorrecto a @unal.edu.co.

```
Unset
CALL UPDATE_GeneralData (1001,'MARIA@gmail.com',310310310);
```

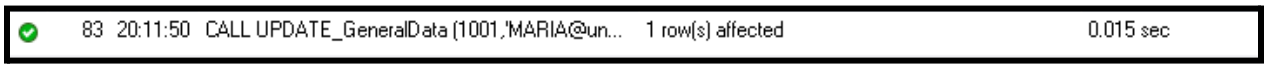
Se evidencia que ninguna fila es afectada. Por tanto, falla el procedimiento almacenado. Proyectando la tabla persona se observa que no ocurre actualización alguna:

PER_DocumentoIdentidad	PER_TipoDocumento	PER_Nombre	PER_Apellido	PER_CorreoElectronico	PER_Edad	PER_Telefono
1001	CE	María	Pérez	Maria1001@unal.edu.co	19	310310310
1002	CE	María	Pérez	JuanG1002@unal.edu.co	19	3102022233
1003	CE	Carlos	Ramirez	CarlosR1003@unal.edu.co	20	3002033344
1004	DNI	Laura	Martínez	LauraM1004@unal.edu.co	22	3152044455
1005	CC	Diego	González	DiegoG1005@unal.edu.co	26	3012055566
1006	TI	Ana	López	AnaL1006@unal.edu.co	17	3122066677
1007	CE	Luis	Patiño	LuisP1007@unal.edu.co	23	3132077788
1008	DNI	María	Sánchez	MariaS1008@unal.edu.co	32	3142088899
1009	CC	José	Hernández	JoseH1009@unal.edu.co	23	3152099900
1010	TI	Andrea	Díaz	AndreaD1010@unal.edu.co	16	3162000011

Nuevamente, se llama el procedimiento almacenado, no obstante se ingresa un correo electrónico con dominio correcto por parámetro:

```
Unset
CALL UPDATE_GeneralData (1001,'MARIA@unal.edu.co',310310310);
```

Como se observa una fila fue afectada:



Realizando nuevamente la proyección de la tabla persona se observa la actualización en los datos sobre el registro con PER_DocumentoIdentidad = 1001:

PER_DocumentoIdentidad	PER_TipoDocumento	PER_Nombre	PER_Apellido	PER_CorreoElectronico	PER_Edad	PER_Telefono
1001	CE	María	Pérez	MARIA@unal.edu.co	19	310310310
1002	CE	María	Pérez	JuanG1002@unal.edu.co	19	3102022233
1003	CE	Carlos	Ramirez	CarlosR1003@unal.edu.co	20	3002033344
1004	DNI	Laura	Martínez	LauraM1004@unal.edu.co	22	3152044455
1005	CC	Diego	González	DiegoG1005@unal.edu.co	26	3012055566
1006	TI	Ana	López	AnaL1006@unal.edu.co	17	3122066677
1007	CE	Luis	Patiño	LuisP1007@unal.edu.co	23	3132077788
1008	DNI	María	Sánchez	MariaS1008@unal.edu.co	32	3142088899
1009	CC	José	Hernández	JoseH1009@unal.edu.co	23	3152099900
1010	TI	Andrea	Díaz	AndreaD1010@unal.edu.co	16	3162000011

1006

Nota: *Este procedimiento únicamente lo puede ejecutar el usuario "Persona U".*

19. Permite la actualización de los datos personales específicos de mayor importancia sobre las personas (por directivos o jefes).

El procedimiento almacenado recibe por parámetros el documento de identidad, el nombre, el apellido, el tipo de documento y la edad, los cuales por medio del procedimiento almacenado son actualizados.

```
Unset
        DROP PROCEDURE IF EXISTS UPDATE_SpecificData;

        DELIMITER %%

        CREATE PROCEDURE UPDATE_SpecificData (DI INT, Nombre VARCHAR(30), Apellido VARCHAR(30), TipoDocumento VARCHAR(3), Edad INT)

        BEGIN

                IF length(TipoDocumento)<=3 AND (TipoDocumento='CC' OR TipoDocumento='TI' OR TipoDocumento='CE' OR TipoDocumento='DNI') THEN

                UPDATE persona SET PER_Nombre=Nombre, PER_Apellido=Apellido, PER_TipoDocumento=TipoDocumento, PER_Edad=Edad WHERE PER_DocumentoIdentidad = DI;

            END IF;
```

```
END %%

DELIMITER ;
```

Para corroborar el correcto ingreso de los valores por parámetros se verifica el número de caracteres del tipo de documento con la función ***length()***. Además, se incluye la igualdad del valor del parámetro con uno de los tipos de documento de identidad definidos al momento de crear la base de datos y las tablas.

Se aplican dos casos de invocación del PA para evaluar la efectividad de las condiciones establecidas para los parámetros:

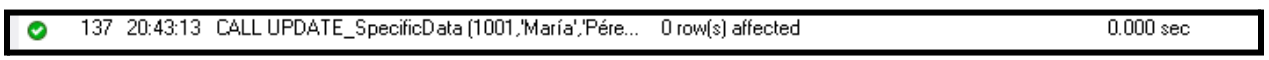
Primero se proyecta la tabla para ver el efecto de las actualizaciones:

```
Unset
SELECT * FROM persona;
```

PER_DocumentoIdentidad	PER_TipoDocumento	PER_Nombre	PER_Apellido	PER_CorreoElectronico	PER_Edad	PER_Telefono
1001	CC	María	Pérez	MariaP1001@unal.edu.co	18	3202011122
1002	TI	Juan	Gómez	JuanG1002@unal.edu.co	16	3102022233
1003	CE	Carlos	Ramirez	CarlosR1003@unal.edu.co	20	3002033344
1004	DNI	Laura	Martínez	LauraM1004@unal.edu.co	22	3152044455
1005	CC	Diego	González	DiegoG1005@unal.edu.co	26	3012055566
1006	TI	Ana	López	AnaL1006@unal.edu.co	17	3122066677
1007	CE	Luis	Patiño	LuisP1007@unal.edu.co	23	3132077788
1008	DNI	María	Sánchez	MariaS1008@unal.edu.co	32	3142088899
1009	CC	José	Hernández	JoseH1009@unal.edu.co	23	3152099900
1010	TI	Andrea	Díaz	AndreaD1010@unal.edu.co	16	3162000011

```
Unset
-- Llamado del PA con parámetro de Tipo de Documento incorrecto:

CALL UPDATE_SpecificData (1001,'María','Pérez','PA',19);
```



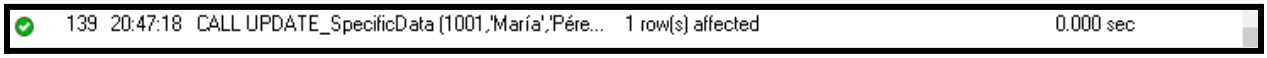
Ninguna fila es afectada, así que no ocurre cambio alguno en la tabla:

```
Unset
SELECT * FROM persona;
```

PER_DocumentoIdentidad	PER_TipoDocumento	PER_Nombre	PER_Apellido	PER_CorreoElectronico	PER_Edad	PER_Telefono
1001	CC	María	Pérez	MariaP1001@unal.edu.co	18	3202011122
1002	TI	Juan	Gómez	JuanG1002@unal.edu.co	16	3102022233
1003	CE	Carlos	Ramirez	CarlosR1003@unal.edu.co	20	3002033344
1004	DNI	Laura	Martínez	LauraM1004@unal.edu.co	22	3152044455
1005	CC	Diego	González	DiegoG1005@unal.edu.co	26	3012055566
1006	TI	Ana	López	AnaL1006@unal.edu.co	17	3122066677
1007	CE	Luis	Patiño	LuisP1007@unal.edu.co	23	3132077788
1008	DNI	María	Sánchez	MariaS1008@unal.edu.co	32	3142088899
1009	CC	José	Hernández	JoseH1009@unal.edu.co	23	3152099900
1010	TI	Andrea	Díaz	AndreaD1010@unal.edu.co	16	3162000011

```
Unset
-- Llamado del PA con parámetro de Tipo de Documento correcto:

CALL UPDATE_SpecificData (1001,'María','Pérez','DNI',19);
```



Se puede observar que la fila es afectada. Al proyectar la tabla se evidencia los cambios en el registro:

```
Unset
SELECT * FROM persona;
```

PER_DocumentoIdentidad	PER_TipoDocumento	PER_Nombre	PER_Apellido	PER_CorreoElectronico	PER_Edad	PER_Telefono
1001	DNI	María	Pérez Rodríguez	MariaP1001@unal.edu.co	20	3202011122
1002	TI	Juan	Gómez	JuanG1002@unal.edu.co	16	3102022233
1003	CE	Carlos	Ramirez	CarlosR1003@unal.edu.co	20	3002033344
1004	DNI	Laura	Martínez	LauraM1004@unal.edu.co	22	3152044455
1005	CC	Diego	González	DiegoG1005@unal.edu.co	26	3012055566
1006	TI	Ana	López	AnaL1006@unal.edu.co	17	3122066677
1007	CE	Luis	Patiño	LuisP1007@unal.edu.co	23	3132077788
1008	DNI	María	Sánchez	MariaS1008@unal.edu.co	32	3142088899
1009	CC	José	Hernández	JoseH1009@unal.edu.co	23	3152099900
1010	TI	Andrea	Díaz	AndreaD1010@unal.edu.co	16	3162000011

Nota: *Este procedimiento almacenado únicamente lo puede ejecutar aquella persona que tenga el usuario "Jefes".*