GUIA DE AUTOESTUDIO 4

PL/SQL Básico

ALUMNOS:

Ignacio Andrés Castillo Rendon

Anderson Fabian Garcia Nieto

Laboratorio-Modelos de bases de datos
2024-2

DOCENTE:
MARIA IRMA DIAZ ROZO

Bogotá D.C-Colombia

INVESTIGACIÓN

A. ACCIONES REFERENCIALES

¿Para qué sirven las acciones referenciales?

Las acciones referenciales son las acciones que toma la base de datos cuando un registro que es referenciado en otra tabla, es borrado o actualizado.

Estas impiden que los usuarios de bases de datos borren por error las asociaciones entre columnas vinculadas.

¿Qué acciones soporta ORACLE? ¿Qué permite hacer cada una de ellas?

B. PL/SQL

¿Qué es PL/SQL?

PL/SQL es un lenguaje de programación creado por Oracle como una extensión de SQL, diseñado para manejar tareas más complejas en bases de datos relacionales. Aunque ambos están enfocados en la gestión de datos, presentan importantes diferencias.

SQL, que es un lenguaje de consulta estructurado, se utiliza principalmente para realizar operaciones sobre los datos almacenados, como consultarlos, actualizarlos, insertarlos o eliminarlos. Es un lenguaje declarativo, lo que significa que se especifica qué se quiere hacer, pero no cómo.

Por otro lado, PL/SQL permite la programación por procedimientos, lo que significa que, además de ejecutar consultas, se pueden realizar operaciones más avanzadas, como manejar bucles, condicionales, excepciones y ejecutar bloques completos de código. A diferencia de SQL, que ejecuta una consulta a la vez, PL/SQL permite ejecutar múltiples sentencias agrupadas en un bloque, proporcionando una mayor flexibilidad y capacidad para gestionar procesos más complejos en las bases de datos.

¿Qué motores lo soportan?

PL/SQL es soportado principalmente por Oracle Database, ya que fue desarrollado como una extensión de SQL específicamente para este sistema. Además, también es compatible con productos derivados como Oracle TimesTen In-Memory Database y Oracle Database Lite, que permiten el uso de PL/SQL en entornos más ligeros o distribuidos. Oracle TimesTen, por ejemplo, soporta muchas de las características de PL/SQL, aunque con algunas limitaciones en comparación con la base de datos principal de Oracle

Información Tomada de:

February2024. (2024, February 21). TimesTen PL/SQL Support: Reference summary. Oracle Help Center. https://docs.oracle.com/en/database/other-databases/timesten/22.1/plsql-developer/timesten-pl-sql-support-reference-

summary1.html#GUID-716087C7-55A2-401B-A105-3C6BFA9CD6BB

C. DATOS E INSTRUCCIONES EN PL/SQL

¿Cuáles son los tipos de datos que ofrece?

Tipo de datos PL/SQL	Db2® Tipo de datos SQL	Descripción
BINARY_INTEGER	INTEGER	Datos numéricos enteros
BLOB	BLOB (4096)	Datos binarios
BLOB (*n*)	BLOB (*n*) *n* = 1 a 2 147 483 647	Datos binarios de objetos grandes
BOOLEAN	BOOLEAN	Booleano lógico (true o false)
CAR	CHAR (*n*) *n* = 63 si las unidades de serie del entorno se establecen en CODEUNITS32 *n* = 255 de lo contrario	Datos de serie de caracteres de longitud fija de longitud *n*
CHAR (*n*)	CHAR (*n*) *n* = 1 a 255	Datos de serie de caracteres de longitud fija de longitud *n*
CHAR (*n* CHAR)	CHAR (*n* CODEUNITS32) *n* = 1 a 63	Datos de serie de caracteres de longitud fija de longitud *n* UTF-32 unidades de código[1]
CHAR VARYING (*n*)	VARCHAR (*n*)	Datos de serie de caracteres de longitud variable de longitud máxima *n*
CHAR VARYING (*n* CHAR)	VARCHAR (*n* CODEUNITS32) *n* = 1 a 8 168	Datos de serie de caracteres de longitud variable de longitud máxima n UTF-32 unidades de código[1]
CHARACTER	CHARACTER (*n*) *n* = 63 si las unidades de serie del entorno se establecen en CODEUNITS32 *n* = 255 de lo contrario	Datos de serie de caracteres de longitud fija de longitud *n*
CARÁCTER (*n*)	CHARACTER (*n*) *n* = 1 a 255	Datos de serie de caracteres de longitud fija de longitud *n*
CARÁCTER (*n* CHAR)	CHARACTER (*n* CODEUNITS32) *n* = 1 a 63	Datos de serie de caracteres de longitud fija de longitud *n* UTF-32 unidades de código[1]
CARÁCTER VARIABLE (*n*)	VARCHAR (*n*) *n* = 1 a 32 672	Datos de serie de caracteres de longitud variable de longitud máxima *n*
CARÁCTER VARIABLE (*n* CHAR)	VARCHAR (*n* CODEUNITS32) *n* = 1 a 8 168	Datos de serie de caracteres de longitud variable de longitud máxima *n* UTF-32 unidades de código[1]
CLOB	CLOB (1M)	Datos de objeto grande de tipo carácter
CLOB (*n*)	CLOB (*n*) *n* = 1 a 2 147 483 647	Datos de objeto grande de caracteres de longitud *n*
CLOB (*n* CHAR)	CLOB (*n* CODEUNITS32) *n* = 1 a 536 870 911	Datos de serie de objeto grande de caracteres de longitud *n* UTF-32 unidades de código[1]
DATE	FECHA [2]	Datos de fecha y hora (expresados en el segundo)
DEC	DEC (9, 2)	Datos numéricos decimales
DEC (*p*)	DEC (*p*) *p* = 1 a 31	Datos numéricos decimales de precisión *p*

DEC (*p*, *s*)	DEC (*p*, *s*) *p* = 1 a 31; *s* = 1 a 31	Datos numéricos decimales de precisión *p* y escala *s*
DECIMAL	DECIMAL (9, 2)	Datos numéricos decimales
DECIMAL (*p*)	DECIMAL (*p*) *p* = 1 a 31	Datos numéricos decimales de precisión *p*
DECIMAL (*p*, *s*)	DECIMAL (*p*, *s*) *p* = 1 a 31; *s* = 1 a 31	Datos numéricos decimales de precisión *p* y escala *s*
DOUBLE	DOUBLE	Número de coma flotante de precisión doble
DOUBLE PRECISION	DOUBLE PRECISION	Número de coma flotante de precisión doble
FLOAT	FLOAT	Datos numéricos flotantes
FLOAT (*n*) *n* = 1 a 24	REAL	Datos numéricos reales
FLOAT (*n*) *n* = 25 a 53	DOUBLE	Datos numéricos dobles
INT	INT	Datos numéricos enteros de cuatro bytes con signo
INTEGER	INTEGER	Datos numéricos enteros de cuatro bytes con signo
LONG	CLOB (32760)	Datos de objeto grande de tipo carácter
LONG RAW	BLOB (32760)	Datos binarios de objetos grandes
LONG VARCHAR	CLOB (32760)	Datos de objeto grande de tipo carácter
NATURAL	INTEGER	Datos numéricos enteros de cuatro bytes con signo
NCHAR	NCHAR (*n*) [3] *n* = 63 si el parámetro de configuración NCHAR_MAPPING se establece en GRAPHIC_CU32 o CHAR_CU32 *n* = 127 de lo contrario	Datos de serie de caracteres nacionales de longitud fija de longitud *n*
NCHAR (*n*) *n* = 1 a 2000	NCHAR (*n*) [3]	Datos de serie de caracteres nacionales de longitud fija de longitud *n*
NCLOB [4]	NCLOB (1M) [3]	Datos de objeto grande de caracteres nacionales
NCLOB (*n*)	NCLOB (*n*) [3]	Datos de objeto grande de caracteres nacionales de longitud máxima *n*
NVARCHAR2	NVARCHAR [3]	Datos de serie de caracteres nacionales de longitud variable
NVARCHAR2 (*n*)	NVARCHAR (*n*) [3]	Datos de serie de caracteres nacionales de longitud variable de longitud máxima *n*
NUMBER	NÚMERO [5]	Datos numéricos exactos

NÚMERO (*p*)	NUMBER (*p*) [5]	Datos numéricos exactos de precisión máxima *p*
NUMBER (*p*, *s*)	NUMBER (*p*, *s*) [5] *p* = 1 a 31; *s* = 1 a 31	Datos numéricos exactos de precisión máxima *p* y escala *s*
NUMÉRICO	NUMÉRICO (9.2)	Datos numéricos exactos
NUMÉRICO (*p*)	NUMERIC (*p*) *p* = 1 a 31	Datos numéricos exactos de precisión máxima *p*
NUMERIC (*p*, s)	NUMERIC (*p*, *s*) *p* = 1 a 31; *s* = 0 a 31	Datos numéricos exactos de precisión máxima *p* y escala *s*
PLS_INTEGER	INTEGER	Datos numéricos enteros
RAW	VARBINARY(32672)	Datos de serie binaria de longitud variable
RAW (*n*)	VARBINARY (*n*) *n* = 1 a 32 672	Datos de serie binaria de longitud variable
SMALLINT	SMALLINT	Datos enteros de dos bytes con signo
TIMESTAMP (0)	TIMESTAMP (0)	Datos de fecha con información de indicación de fecha y hora
TIMESTAMP (*p*)	TIMESTAMP (*p*)	Datos de fecha y hora con segundos fraccionarios opcionales y precisión *p*
VARCHAR	VARCHAR (4096)	Datos de serie de caracteres de longitud variable con una longitud máxima de 4096
VARCHAR (*n*)	VARCHAR (*n*)	Datos de serie de caracteres de longitud variable con una longitud máxima de *n*
VARCHAR (*n* CHAR)	VARCHAR (*n* CODEUNITS32) *n* = 1 a 8 168	Datos de serie de caracteres de longitud variable de longitud máxima *n* UTF-32 unidades de código[1]
VARCHAR2 (*n*)	VARCHAR2 (*n*) [6]	Datos de serie de caracteres de longitud variable con una longitud máxima de *n*
VARCHAR2 (*n* CHAR)	VARCHAR2 (*n* CODEUNITS32) *n* = 1 a 8 168 [6]	Datos de serie de caracteres de longitud variable de longitud máxima *n* UTF-32 unidades de código[1]

¿Cuál es la forma de definir constantes y variables?
(Dentro de la creación de las tablas)
-SINTAXIS CONSTANTES
Nombre_variable [CONSTANT] TIPO [NOT NULL] [:= inicialización];
-SINTAXIS VARIABLES
Nombre_variable tipo_de_dato [:= valor_inicial];

¿Cómo se define una variable con un tipo tomado de la base de datos?
Para definir una variable en PL/SQL cuyo tipo de dato se toma directamente de una columna de la base de datos, se puede utilizar la cláusula %TYPE. Esto permite que la variable herede automáticamente el tipo de dato de la columna correspondiente, lo cual es útil si el tipo de la columna cambia en el futuro, ya que la variable se ajustará sin necesidad de modificar el código.
SINTAXIS
nombre_variable nombre_tabla.nombre_columna%TYPE;

¿Cuál es la forma de los diferentes tipos de asignación? (Son tres)
1) Asignación directa

SINTAXIS

variable := valor;

2) Asignación mediante consulta

SINTAXIS
SELECT columna_o_expresion INTO variable
FROM tabla
WHERE condición;

3) Asignación con valores por defecto

SINTAXIS
nombre_variable tipo_de_dato := valor_inicial;

D. CURSORES

¿Qué es un cursor implícito? ¿Para qué sirve?

RTA: Los cursores implícitos se utilizan para realizar consultas SELECT que devuelven un único registro. Deben tenerse en cuenta los siguientes puntos cuando se utilizan cursores implícitos:

- Con cada cursor implícito debe existir la palabra clave INTO.
- Las variables que reciben los datos devueltos por el cursor tienen que contener el mismo tipo de dato que las columnas de la tabla.
- Los cursores implícitos solo pueden devolver una única fila. En caso de que se devuelva más de una fila (o ninguna fila) se producirá una excepción.

¿Qué es un cursor explícito? ¿Para qué sirve?

RTA: Los cursores explícitos se emplean para realizar consultas SELECT que pueden devolver cero filas o más de una fila. Para trabajar con un cursor explícito necesitamos realizar las siguientes tareas:

- Declarar el cursor.
- Abrir el cursor con la instrucción OPEN.
- Leer los datos del cursor con la instrucción FETCH.
- Cerrar el cursor y liberar los recursos con la instrucción CLOSE

¿Cuáles son las excepciones propias de estos cursores?

Excepciones asociadas a los cursores implícitos

Excepción	Explicación
NO_DATA_FOUND	Se produce cuando una sentencia SELECT intenta recuperar datos pero ninguna fila satisface sus condiciones. Es decir, cuando "no hay datos".
TOO_MANY_ROWS	Dado que cada cursor implícito sólo es capaz de recuperar una fila , esta excepción detecta la existencia de más de una fila.

Excepciones predefinidas

Excepción	Descripción
ACCESS_INTO_NULL	El programa intentó asignar valores a los atributos de un objeto no inicializado -6530.
COLLECTION_IS_NULL	El programa intentó asignar valores a una tabla anidada aún no inicializada -6531.
CURSOR_ALREADY_OPEN	El programa intentó abrir un cursor que ya se encontraba abierto. Recuerde que un cursor de ciclo FOR automáticamente lo abre y ello no se debe especificar con la sentencia OPEN -6511.
DUP_VAL_ON_INDEX	El programa intentó almacenar valores duplicados en una columna que se mantiene con restricción de integridad de un índice único (unique index) -1.
INVALID_CURSOR	El programa intentó efectuar una operación no válida sobre un cursor -1001.
INVALID_NUMBER	En una sentencia SQL, la conversión de una cadena de caracteres hacia un número falla cuando esa cadena no representa un número válido -1722.
LOGIN_DENIED	El programa intentó conectarse a Oracle con un nombre de usuario o password inválido -1017.
NO_DATA_FOUND	Una sentencia SELECT INTO no devolvió valores o el programa referenció un elemento no inicializado en una tabla indexada 100.
NOT_LOGGED_ON	El programa efectuó una llamada a Oracle sin estar conectado -1012.
PROGRAM_ERROR PL/SQL	Tiene un problema interno -6501.
ROWTYPE_MISMATCH	Los elementos de una asignación (el valor a asignar y la variable que lo contendrá) tienen tipos incompatibles. También se presenta este error cuando un parámetro pasado a un subprograma no es del tipo esperado -6504.
SELF_IS_NULL	El parámetro SELF (el primero que es pasado a un método MEMBER) es nulo -30625.
STORAGE_ERROR	La memoria se terminó o está corrupta -6500.
SUBSCRIPT_BEYOND_COUNT	El programa está tratando de referenciar un elemento de una colección indexada que se encuentra en una posición más grande que el número real de elementos de la colección -6533.
SUBSCRIPT_OUTSIDE_LIMIT	El programa está referenciando un elemento de una tabla utilizando un número fuera del rango permitido (por ejemplo, el elemento "-1") -6532.
SYS_INVALID_ROWID	La conversión de una cadena de caracteres hacia un tipo ROWID falló porque la cadena no representa un número -1410.
TIMEOUT_ON_RESOURCE	Se excedió el tiempo máximo de espera por un recurso en Oracle -51.
TOO_MANY_ROWS	Una sentencia SELECT INTO devuelve más de una fila -1422.
VALUE_ERROR	Ocurrió un error aritmético, de conversión o truncamiento. Por ejemplo, sucede cuando se intenta introducir un valor muy grande dentro de una variable más pequeña -6502.
ZERO_DIVIDE	El programa intentó efectuar una división por cero -1476.

Fuente:

- https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/107#:~:text=Los%20cursores%20implícitos%20se%20utilizan,existir%20la%20palabra%20clave%20INTO.

E. MODULARIDAD

¿Cuál es la estructura general de un bloque PL/SQL?

RTA:

Los bloques PL/SQL presentan una estructura específica compuesta de tres partes bien diferenciadas:

- **La sección declarativa, donde se declaran todas las constantes y variables que se van a utilizar en la ejecución del bloque.**
- **La sección de ejecución, que incluye las instrucciones a ejecutar en el bloque PL/SQL.**
- **La sección de excepciones, en donde se definen los manejadores de errores que soportará el bloque PL/SQL.**

FUENTE:
https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/107#:~:text=Los%20bloques%20PL%2FSQL%20presentan,en%20el%20bloque%20PL%2FSQL.

```
[declare | is | as ]    /*Parte declarativa*/
begin    /*Parte de ejecucion*/
[ exception ]    /*Parte de excepciones*/
end;
```

¿Para qué sirven las diferentes estructuras modulares? (bloque anónimo, procedimiento, función y disparador)

RTA:

Bloque anónimo: La sentencia de bloque anónimo PL/SQL es una sentencia ejecutable que puede contener sentencias de control PL/SQL y sentencias SQL. Se puede utilizar para implementar la lógica de procedimiento en un lenguaje de scripts. En contextos PL/SQL, esta sentencia la puede compilar y ejecutar el servidor de datos Db2
Fuente: https://www.ibm.com/docs/es/db2/11.5?topic=plsql-anonymous-block-statement

Procedimiento: El servidor de datos Db2® da soporte a la compilación y ejecución de procedimientos PL/SQL. Los procedimientos PL/SQL son objetos de base de datos que contienen lógica de procedimiento PL/SQL y sentencias SQL que se pueden invocar en contextos en los que las referencias de procedimiento o sentencia CALL son válidas.
Fuente:
https://www.ibm.com/docs/es/db2/11.5?topic=support-procedures-plsql

Funciones: El servidor de datos Db2® da soporte a la compilación y ejecución de funciones PL/SQL escalares e interconectadas. Las funciones PL/SQL escalares se pueden invocar en contextos en los que las expresiones son válidas. Cuando se evalúa, una función PL/SQL escalar devuelve un valor que se sustituye dentro de la expresión en la que se incorpora la función. Las funciones PL/SQL interconectadas se pueden invocar en la cláusula FROM de las sentencias SELECT y calcular una tabla de una fila a la vez.
Fuente:
https://www.ibm.com/docs/es/db2/11.5?topic=support-functions-plsql

Disparador: La principal función de los trigger es contribuir a mejorar la gestión de la base de datos. Gracias a ellos muchas operaciones se pueden realizar de forma automática, sin necesidad de intervención humana, lo que permite ahorrar mucho tiempo.

Otra de sus funciones es aumentar la seguridad e integridad de la información. Esto lo consiguen gracias a la programación de restricciones o requerimientos de verificación que permiten minimizar los errores y sincronizar la información.Por otra parte, entre sus principales ventajas es que todas estas funciones se pueden realizar desde la propia base de datos, es decir, no es necesario recurrir a lenguajes externos de programación.

Fuente:

https://ayudaleyprotecciondatos.es/bases-de-datos/trigger/

PRACTICANDO

```
----TABLAS
CREATE TABLE Band (
   band_no INT DEFAULT num_sequence.NEXTVAL,
   band_name VARCHAR2(20) NOT NULL,
   band_home INT NOT NULL,
   band_type VARCHAR2(10) NULL,
   b_date DATE NULL,
   band_contact INT NOT NULL
);

CREATE TABLE Performer (
   perf_no INT NOT NULL,
   perf_is INT NOT NULL,
   instrument VARCHAR2(50) NULL,
   perf_type VARCHAR2(50) NOT NULL
);

CREATE TABLE Composer (
   comp_no INT NOT NULL,
   comp_is INT NOT NULL,
   comp_type VARCHAR2(50) NOT NULL
);


CREATE TABLE Musician (
   m_no INT NOT NULL,
   m_name VARCHAR2(100) NOT NULL,
   born DATE NOT NULL,
   died DATE NULL,
```

```
    born_in INT NOT NULL,
    living_in INT NOT NULL
);

CREATE TABLE Place (
    place_no INT NOT NULL,
    place_town VARCHAR2(100) NOT NULL,
    place_country VARCHAR2(100) NOT NULL
);

CREATE TABLE Plays_in (
    player INT NOT NULL,
    band_id INT NOT NULL
);

CREATE TABLE Performance (
    pfmnce_no INT NOT NULL,
    gave INT NOT NULL,
    performed INT NOT NULL,
    conducted_by INT NOT NULL,
    performed_in INT NOT NULL
);

--XTABLAS
ALTER TABLE Band DROP PRIMARY KEY CASCADE;
ALTER TABLE Performer DROP PRIMARY KEY CASCADE;
ALTER TABLE Composer DROP PRIMARY KEY CASCADE;
ALTER TABLE Musician DROP PRIMARY KEY CASCADE;
ALTER TABLE Place DROP PRIMARY KEY CASCADE;
ALTER TABLE Plays_in DROP PRIMARY KEY CASCADE;
ALTER TABLE Performance DROP PRIMARY KEY CASCADE;

DROP TABLE Band;
DROP TABLE Performer;
DROP TABLE Composer;
DROP TABLE Musician;
DROP TABLE Place;
DROP TABLE Plays_in;
DROP TABLE Performance;

----Atributos
--Primary Keys
ALTER TABLE Band
ADD CONSTRAINT PK_BAND PRIMARY KEY (band_no);

ALTER TABLE Performer
ADD CONSTRAINT PK_PERFORMER PRIMARY KEY (perf_no);
```

```
ALTER TABLE Composer
ADD CONSTRAINT PK_COMPOSER PRIMARY KEY (comp_no);

ALTER TABLE Musician
ADD CONSTRAINT PK_MUSICIAN PRIMARY KEY (m_no);

ALTER TABLE PLACE
ADD CONSTRAINT PK_PLACE PRIMARY KEY (place_no);

ALTER TABLE PLAYS_IN
ADD CONSTRAINT PK1_PLAY PRIMARY KEY (player, band_id);

ALTER TABLE PERFORMANCE
ADD CONSTRAINT PK_PERFORMANCE PRIMARY KEY (pfmnce_no);

--Unique keys

ALTER TABLE Band
ADD CONSTRAINT UK_BAND UNIQUE (band_name);

--Foreign keys

ALTER TABLE PLAYS_IN
ADD CONSTRAINT FK_PERFORMER_PLAY FOREIGN KEY (player) references
performer(perf_no)
ADD CONSTRAINT FK_BAND_PLAY FOREIGN KEY (band_id) references band(band_no);


ALTER TABLE MUSICIAN
ADD CONSTRAINT FK1_MUSICIAN_PLACE FOREIGN KEY (born_in) references
place(place_no)
ADD CONSTRAINT FK2_MUSICIAN_PLACE FOREIGN KEY (living_in) references
place(place_no);

ALTER TABLE PERFORMANCE
ADD CONSTRAINT FK_PERFORMANCE_PLACE FOREIGN KEY (performed_in)
references place(place_no);

ALTER TABLE BAND
ADD CONSTRAINT FK_BAND_PLACE FOREIGN KEY (band_home) references
place(place_no)
ADD CONSTRAINT FK_BAND_MUSICIAN FOREIGN KEY (band_contact) references
musician(m_no);

ALTER TABLE PERFORMANCE
ADD CONSTRAINT FK_PERFORMANCE_BAND FOREIGN KEY (gave) references
band(band_no)
```

```
ADD CONSTRAINT FK_PERFORMANCE_MUSICIAN FOREIGN KEY (conducted_by)
references musician(m_no);

ALTER TABLE COMPOSER
ADD CONSTRAINT FK_COMPOSER_MUSICIAN FOREIGN KEY (comp_is) references
musician(m_no);

ALTER TABLE PERFORMER
ADD CONSTRAINT FK_PERFORMER_MUSICIAN FOREIGN KEY (perf_is) references
musician(m_no);


----TUPLAS

--TConsecutive Entero(9) Positivo consecutivo
CREATE SEQUENCE num_sequence
   START WITH 1
   INCREMENT BY 1;

--TName Cadena(20) Minimo una palabra
ALTER TABLE Band ADD CONSTRAINT TName
CHECK (LENGTH(band_name)>1);

--TType
ALTER TABLE Band ADD CONSTRAINT TType
CHECK (band_type IN ('rock','classical','blues','pop','soul', 'jazz'));

ALTER TABLE BAND
DROP CONSTRAINT TType;

----ATRIBUTOSOK
INSERT INTO place(place_no,place_town,place_country) VALUES (1,'chia','colombia');
INSERT INTO musician(m_no,m_name,born,died) VALUES (1,'Luis
Carlos',TO_DATE('2000-08-30','YYYY-MM-DD'),null);
INSERT INTO band(band_name,band_home,band_type,b_date,band_contact) VALUES
('LOCOS',1,'rock',TO_DATE('2023-08-30','YYYY-MM-DD'),1);
INSERT INTO band(band_name,band_home,band_type,b_date,band_contact) VALUES
('TAPS',1,'rock',TO_DATE('2015-08-30','YYYY-MM-DD'),1);

----ATRIBUTOSNOOK
--Aca no deja agregar los datos, porque en la tabla de places no existe nada, y desde este
insert se esta referenciado informacion del otro lado
INSERT INTO band(band_name,band_home,band_type,b_date,band_contact) VALUES
('LOCOS',1,'rock',TO_DATE('2023-08-30','YYYY-MM-DD'),1);

--Aca no deja ingresar los datos porqye m_no es una clave principal y estas no pueden ser
vacias.
```

```
INSERT INTO musician(m_no,m_name,born,died) VALUES (null,'Luis
Carlos',TO_DATE('2000-08-30','YYYY-MM-DD'),null);

--Aca no deja ingresar los datos porque band_name tiene la caracteristica que es UNIQUE.
INSERT INTO band(band_name,band_home,band_type,b_date,band_contact) VALUES
('LOCOS',1,'rock',TO_DATE('2023-08-30','YYYY-MM-DD'),1);
INSERT INTO band(band_name,band_home,band_type,b_date,band_contact) VALUES
('LOCOS',1,'rock',TO_DATE('2023-08-30','YYYY-MM-DD'),1);

----TUPLASOK
--TTYPE
INSERT INTO band(band_name,band_home,band_type,b_date,band_contact) VALUES
('LOCOS',1,'rock',TO_DATE('2023-08-30','YYYY-MM-DD'),1);

--TCONSECUTIVE
INSERT INTO band(band_name,band_home,band_type,b_date,band_contact) VALUES
('LOCOS',1,'rock',TO_DATE('2023-08-30','YYYY-MM-DD'),1);

--TNAME
INSERT INTO band(band_name,band_home,band_type,b_date,band_contact) VALUES
('LOCOS',1,'rock',TO_DATE('2023-08-30','YYYY-MM-DD'),1);


----TUPLASNOOK
--Aca no deja agregar nada porque los generos musicales son diferentes a los
preestablecidos en la base de datos
INSERT INTO place(place_no,place_town,place_country) VALUES (1,'chia','colombia');
INSERT INTO musician(m_no,m_name,born,died) VALUES (1,'Luis
Carlos',TO_DATE('2000-08-30','YYYY-MM-DD'),null);
INSERT INTO band(band_name,band_home,band_type,b_date,band_contact) VALUES
('LOCOS',1,'vallenato',TO_DATE('2023-08-30','YYYY-MM-DD'),1);

--Aca no deja agregar nada porque el nombre de la banda es menor a una palabra
INSERT INTO band(band_name,band_home,band_type,b_date,band_contact) VALUES ('
',1,'vallenato',TO_DATE('2023-08-30','YYYY-MM-DD'),1);



----Acciones referenciales
--Plays_in
ALTER TABLE PLAYS_IN
DROP CONSTRAINT FK_PERFORMER_PLAY;

ALTER TABLE PLAYS_IN
ADD CONSTRAINT FK_PERFORMER_PLAY FOREIGN KEY (player)
REFERENCES performer(perf_no)
ON DELETE CASCADE;
```

```
ALTER TABLE PLAYS_IN
DROP CONSTRAINT FK_BAND_PLAY;

ALTER TABLE PLAYS_IN
ADD CONSTRAINT FK_BAND_PLAY FOREIGN KEY (band_id)
REFERENCES band(band_no)
ON DELETE CASCADE;

--Musician
ALTER TABLE MUSICIAN
DROP CONSTRAINT FK1_MUSICIAN_PLACE;

ALTER TABLE MUSICIAN
ADD CONSTRAINT FK1_MUSICIAN_PLACE FOREIGN KEY (born_in)
REFERENCES place(place_no)
ON DELETE SET NULL;

ALTER TABLE MUSICIAN
DROP CONSTRAINT FK2_MUSICIAN_PLACE;

ALTER TABLE MUSICIAN
ADD CONSTRAINT FK2_MUSICIAN_PLACE FOREIGN KEY (living_in)
REFERENCES place(place_no)
ON DELETE SET NULL;

--Band
ALTER TABLE BAND
DROP CONSTRAINT FK_BAND_PLACE;

ALTER TABLE BAND
ADD CONSTRAINT FK_BAND_PLACE FOREIGN KEY (band_home)
REFERENCES place(place_no)
ON DELETE SET NULL;

ALTER TABLE BAND
DROP CONSTRAINT FK_BAND_MUSICIAN;

ALTER TABLE BAND
ADD CONSTRAINT FK_BAND_MUSICIAN FOREIGN KEY (band_contact)
REFERENCES musician(m_no)
ON DELETE CASCADE;

--Composer
ALTER TABLE COMPOSER
DROP CONSTRAINT FK_COMPOSER_MUSICIAN;

ALTER TABLE COMPOSER
ADD CONSTRAINT FK_COMPOSER_MUSICIAN FOREIGN KEY (comp_is)
```

```
REFERENCES musician(m_no)
ON DELETE SET NULL;

--Performer
ALTER TABLE PERFORMER
DROP CONSTRAINT FK_PERFORMER_MUSICIAN;

ALTER TABLE PERFORMER
ADD CONSTRAINT FK_PERFORMER_MUSICIAN FOREIGN KEY (perf_is)
REFERENCES musician(m_no)
ON DELETE SET NULL;

--Performance
ALTER TABLE PERFORMANCE
DROP CONSTRAINT FK_PERFORMANCE_BAND;

ALTER TABLE PERFORMANCE
ADD CONSTRAINT FK_PERFORMANCE_BAND FOREIGN KEY (gave)
REFERENCES band(band_no)
ON DELETE CASCADE;


ALTER TABLE PERFORMANCE
DROP CONSTRAINT FK_PERFORMANCE_PLACE;

ALTER TABLE PERFORMANCE
ADD CONSTRAINT FK_PERFORMANCE_PLACE FOREIGN KEY (performed_in)
REFERENCES place(place_no)
ON DELETE SET NULL;

ALTER TABLE PERFORMANCE
DROP CONSTRAINT FK_PERFORMANCE_MUSICIAN;

ALTER TABLE PERFORMANCE
ADD CONSTRAINT FK_PERFORMANCE_MUSICIAN FOREIGN KEY(conducted_by)
REFERENCES musician(m_no)
ON DELETE SET NULL;


INSERT INTO PLACE(place_no, place_town, place_country) VALUES (1, 'Manchester',
'England');
INSERT INTO PLACE(place_no, place_town, place_country) VALUES (2, 'Edinburgh',
'Scotland');
INSERT INTO PLACE(place_no, place_town, place_country) VALUES (3, 'Salzburg',
'Austria');
INSERT INTO PLACE(place_no, place_town, place_country) VALUES (4, 'New York',
'USA');
```

```
INSERT INTO PLACE(place_no, place_town, place_country) VALUES (5, 'Birmingham', 'England');
INSERT INTO PLACE(place_no, place_town, place_country) VALUES (6, 'Glasglow', 'Scotland');
INSERT INTO PLACE(place_no, place_town, place_country) VALUES (7, 'London', 'England');
INSERT INTO PLACE(place_no, place_town, place_country) VALUES (8, 'Chicago', 'USA');
INSERT INTO PLACE(place_no, place_town, place_country) VALUES (9, 'Amsterdam', 'Netherlands');

INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (1, 'Fred Bloggs', '02/03/1948', NULL, 1, 2);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (2, 'John Smith', '03/03/1950', NULL, 3, 4);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (3, 'Helen Smyth', '08/08/48', NULL, 4, 5);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (4, 'Harriet Smithson', '09/05/1909', '20/09/80', 5 ,6);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (5, 'James First', '10/06/1965', NULL, 7 ,7);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (6, 'Theo Mengel', '12/08/1948', NULL, 7 ,1);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (7, 'Sue Little', '21/02/1945', NULL, 8 ,9);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (8, 'Harry Forte', '28/02/1951', NULL, 1 ,8);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (9, 'Phil Hot', '30/06/1942', NULL, 2 ,7);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (10, 'Jeff Dawn', '12/12/1945', NULL, 3 ,6);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (11, 'Rose Spring', '25/05/1948', NULL, 4 ,5);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (12, 'Davis Heavan', '03/10/1975', NULL, 5 ,4);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (13, 'Lovely Time', '28/12/1948', NULL, 6 ,3);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (14, 'Alan Fluff', '15/01/1935', '15/05/1997', 7 ,2);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (15, 'Tony Smythe', '02/04/1932', NULL, 8 ,1);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (16, 'James Quick', '08/08/1924', NULL, 9 ,2);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (17, 'Freda Miles', '04/07/1920', NULL, 9 ,3);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (18, 'Elsie James', '06/05/1947', NULL, 8 ,5);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (19, 'Andy Jones', '08/10/1958', NULL, 7 ,6);
```

```
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (20,
'Louise Simpson', '10/01/1948', '11/02/1998', 6 ,6);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (21,
'James Steeple', '10/01/1947', NULL, 5 ,6);
INSERT INTO MUSICIAN(m_no, m_name, born, died, born_in, living_in) VALUES (22,
'Steven Chaytors', '11/03/1956', NULL, 6 ,7);


INSERT INTO BAND(band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES(1, 'ROP', 5, 'classical', '30/01/2001', 11);
INSERT INTO BAND(band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES(2, 'AASO', 6, 'classical', NULL, 10);
INSERT INTO BAND(band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES(3, 'The J Bs', 8, 'jazz', NULL, 12);
INSERT INTO BAND(band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES(4, 'BBSO', 9, 'classical', NULL, 21);
INSERT INTO BAND(band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES(5, 'The left Overs', 2, 'jazz', NULL, 8);
INSERT INTO BAND(band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES(6, 'Somebody Love this', 1, 'jazz', NULL, 6);
INSERT INTO BAND(band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES(7, 'Oh well', 4, 'classical', NULL, 3);
INSERT INTO BAND(band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES(8, 'Swinging strings', 4, 'classical', NULL, 7);
INSERT INTO BAND(band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES(9, 'The Rest', 9, 'jazz', NULL, 16);

INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(1,1,'jazz');
INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(2,3,'classical');
INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(3,5,'jazz');
INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(4,7,'classical');
INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(5,9,'jazz');
INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(6,11,'rock');
INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(7,13,'classical');
INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(8,15,'jazz');
INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(9,17,'classical');
INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(10,19,'jazz');
INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(11,10,'rock');
INSERT INTO COMPOSER(comp_no, comp_is, comp_type) VALUES(12,8,'jazz');


INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(1,2,'violin','classical');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(2,4,'viola','classical');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(3,6,'banjo','jazz');
```

```
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(4,8,'violin','classical');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(5,12,'guitar','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(6,14,'violin','classical');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(7,16,'trumpet','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(8,18,'viola','classical');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(9,20,'bass','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(10,2,'flute','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(11,20,'cornet','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(12,6,'violin','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(13,8,'drums','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(14,10,'violin','classical');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(15,12,'cello','classical');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(16,14,'viola','classical');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(17,16,'flute','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(18,18,'guitar','not known');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(19,20,'trombone','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(20,3,'horn','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(21,5,'violin','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(22,7,'cello','classical');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(23,2,'bass','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(24,4,'violin','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(25,6,'drums','classical');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(26,8,'clarinet','jazz');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(27,10,'bass','jazz');
```

```
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(28,12,'viola','classical');
INSERT INTO PERFORMER(perf_no, perf_is, instrument, perf_type)
VALUES(29,18,'cello','classical');

INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(1,1,1,21,1);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(2,1,3,21,1);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(3,1,5,21,1);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(4,1,2,1,2);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(5,2,4,21,2);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(6,2,6,21,2);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(7,4,19,9,3);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(8,4,20,10,3);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(9,5,12,10,4);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(10,5,13,11,4);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(11,3,5,13,5);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(12,3,6,13,5);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(13,3,7,13,5);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(14,6,20,14,6);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(15,8,12,15,7);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(16,9,16,21,8);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(17,9,17,21,8);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(18,9,18,21,8);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(19,9,19,21,8);
INSERT INTO PERFORMANCE(pfmnce_no, gave, performed, conducted_by,
performed_in) VALUES(20,4,12,10,3);

INSERT INTO PLAYS_IN(player, band_id) VALUES(1,1);
INSERT INTO PLAYS_IN(player, band_id) VALUES(1,7);
```

```
INSERT INTO PLAYS_IN(player, band_id) VALUES(3,1);
INSERT INTO PLAYS_IN(player, band_id) VALUES(4,1);
INSERT INTO PLAYS_IN(player, band_id) VALUES(4,7);
INSERT INTO PLAYS_IN(player, band_id) VALUES(5,1);
INSERT INTO PLAYS_IN(player, band_id) VALUES(6,1);
INSERT INTO PLAYS_IN(player, band_id) VALUES(6,7);
INSERT INTO PLAYS_IN(player, band_id) VALUES(7,1);
INSERT INTO PLAYS_IN(player, band_id) VALUES(8,1);
INSERT INTO PLAYS_IN(player, band_id) VALUES(8,7);
INSERT INTO PLAYS_IN(player, band_id) VALUES(10,2);
INSERT INTO PLAYS_IN(player, band_id) VALUES(12,2);
INSERT INTO PLAYS_IN(player, band_id) VALUES(13,2);
INSERT INTO PLAYS_IN(player, band_id) VALUES(14,2);
INSERT INTO PLAYS_IN(player, band_id) VALUES(14,8);
INSERT INTO PLAYS_IN(player, band_id) VALUES(15,2);
INSERT INTO PLAYS_IN(player, band_id) VALUES(15,8);
INSERT INTO PLAYS_IN(player, band_id) VALUES(17,2);
INSERT INTO PLAYS_IN(player, band_id) VALUES(18,2);
INSERT INTO PLAYS_IN(player, band_id) VALUES(19,3);
INSERT INTO PLAYS_IN(player, band_id) VALUES(20,3);
INSERT INTO PLAYS_IN(player, band_id) VALUES(21,4);
INSERT INTO PLAYS_IN(player, band_id) VALUES(22,4);
INSERT INTO PLAYS_IN(player, band_id) VALUES(23,4);
INSERT INTO PLAYS_IN(player, band_id) VALUES(25,5);
INSERT INTO PLAYS_IN(player, band_id) VALUES(26,6);
INSERT INTO PLAYS_IN(player, band_id) VALUES(27,6);
INSERT INTO PLAYS_IN(player, band_id) VALUES(28,7);
INSERT INTO PLAYS_IN(player, band_id) VALUES(28,8);
INSERT INTO PLAYS_IN(player, band_id) VALUES(29,7);

TRUNCATE TABLE PLAYS_IN;
TRUNCATE TABLE PERFORMANCE;
TRUNCATE TABLE COMPOSER;
TRUNCATE TABLE PERFORMER;
TRUNCATE TABLE BAND;
TRUNCATE TABLE MUSICIAN;
TRUNCATE TABLE PLACE;

---ON DELETE CASCADE con Plays_in y performer
DELETE FROM Performer WHERE perf_no = 28;
DELETE FROM Performer WHERE perf_no = 25;

SELECT * FROM PLAYS_IN;
SELECT * FROM PERFORMER;

---ON DELETE CASCADE con Band y Performance
DELETE FROM Band WHERE band_no = 5;
DELETE FROM Band WHERE band_no = 1;
```

```
SELECT * FROM BAND;
SELECT * FROM PERFORMANCE;

---ON DELETE SET NULL con Musician y Place
DELETE FROM Place WHERE place_no = 4;
DELETE FROM Place WHERE place_no = 1;

SELECT * FROM MUSICIAN;
SELECT * FROM PLACE;

---ON MUSICIAN SET NULL con Performer y Musician
DELETE FROM Musician WHERE m_no = 21;
DELETE FROM Musician WHERE m_no = 7;


---Disparadores
---El número se genera automáticamente: es un consecutivo
---El nombre de la banda debe tener mínimo una palabra
---Si no se indica el tipo de banda, se asume que es 'rock'
---La fecha debe ser menor a la del dia de hoy, si se indica.
CREATE OR REPLACE TRIGGER TR_BAND_BI
BEFORE INSERT ON Band
FOR EACH ROW
BEGIN
   -- Generar el número automáticamente (esto se maneja por la secuencia)
   IF :NEW.band_no IS NULL THEN
      SELECT num_sequence.NEXTVAL INTO :NEW.band_no FROM dual;  -- Utiliza la
secuencia para el band_no
   END IF;

   -- Verificar que el nombre de la banda tenga al menos una palabra
   IF :NEW.band_name IS NULL OR LENGTH(TRIM(:NEW.band_name)) = 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'El nombre de la banda debe tener al menos
una palabra.');
   END IF;

   -- Asumir que el tipo de banda es 'rock' si no se indica
   IF :NEW.band_type IS NULL THEN
      :NEW.band_type := 'rock';
   END IF;

   -- Verificar que la fecha sea menor que la de hoy si se proporciona
   IF :NEW.b_date IS NOT NULL AND :NEW.b_date >= TRUNC(SYSDATE) THEN
      RAISE_APPLICATION_ERROR(-20002, 'La fecha debe ser anterior a la fecha
actual.');
   END IF;
END;
```

```
/


---De los datos generales, únicamente se pude modificar la fecha, si no se había dado.
CREATE OR REPLACE TRIGGER TR_BAND_BU
BEFORE UPDATE ON Band
FOR EACH ROW
BEGIN
   -- Permitir la actualización de la fecha solo si no se había dado
   IF :OLD.b_date IS NOT NULL AND :NEW.b_date IS NOT NULL THEN
      RAISE_APPLICATION_ERROR(-20003, 'Solo se puede modificar la fecha si no se
había dado previamente.');
   END IF;
END;
/


---Es posible adicionar y eliminar músicos, no modificarlos.
CREATE OR REPLACE TRIGGER TR_PLAYS_IN_BU
BEFORE UPDATE ON Plays_in
FOR EACH ROW
BEGIN
   RAISE_APPLICATION_ERROR(-20004, 'No se puede modificar el registro de un músico
en Plays_in. Solo se permiten inserciones y eliminaciones.');
END;
/


---Máximo pueden existir 10 músicos en una banda
CREATE OR REPLACE TRIGGER TR_PLAYS_IN_BI
BEFORE INSERT ON Plays_in
FOR EACH ROW
DECLARE
   v_count INT;
BEGIN
   -- Contar el número de músicos en la banda
   SELECT COUNT(*)
   INTO v_count
   FROM Plays_in
   WHERE band_id = :NEW.band_id;

   IF v_count >= 10 THEN
      RAISE_APPLICATION_ERROR(-20005, 'No se pueden agregar más de 10 músicos en
una banda.');
   END IF;
END;
```

/

```
DROP TRIGGER TR_BAND_BI;
DROP TRIGGER TR_BAND_BU;
DROP TRIGGER TR_PLAYS_IN_BU;
DROP TRIGGER TR_PLAYS_IN_BI;


---Uso del disparador (que funcione)

---TR_BAND_BI
INSERT INTO Band (band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES (num_sequence.NEXTVAL, 'The Rockers', 2, 'rock', TO_DATE('2015-05-01',
'YYYY-MM-DD'), 5);

INSERT INTO Band (band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES (num_sequence.NEXTVAL, 'Blues Brothers', 3, 'blues', TO_DATE('2010-10-15',
'YYYY-MM-DD'), 8);

INSERT INTO Band (band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES (num_sequence.NEXTVAL, 'Soul Vibes', 5, 'soul', TO_DATE('2018-07-20', 'YYYY-
MM-DD'), 11);

INSERT INTO Band (band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES (num_sequence.NEXTVAL, 'Pop Stars', 7, 'pop', TO_DATE('2020-03-05', 'YYYY-
MM-DD'), 15);

INSERT INTO Band (band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES (num_sequence.NEXTVAL, 'Classical Harmony', 9, 'classical', TO_DATE('2005-11-
25', 'YYYY-MM-DD'), 22);

INSERT INTO Band (band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES (num_sequence.NEXTVAL, 'The New Popes', 4, 'classical', NULL, 12);

INSERT INTO Band (band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES (num_sequence.NEXTVAL, 'Chin Chan', 2, 'jazz', NULL, 21);

INSERT INTO Band (band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES (num_sequence.NEXTVAL, 'Amino', 6, 'rock', NULL, 5);

SELECT * FROM BAND;

---TR_BAND_BU
UPDATE Band
SET b_date = TO_DATE('2020-01-01', 'YYYY-MM-DD')
WHERE band_no = 11;
```

```
UPDATE Band
SET b_date = TO_DATE('2006-03-02', 'YYYY-MM-DD')
WHERE band_no = 12;

UPDATE Band
SET b_date = TO_DATE('2006-03-27', 'YYYY-MM-DD')
WHERE band_no = 13;


---TR_PLAYS_IN_BU
INSERT INTO PLAYS_IN(player, band_id) VALUES(3,7);
INSERT INTO PLAYS_IN(player, band_id) VALUES(4,7);
INSERT INTO PLAYS_IN(player, band_id) VALUES(19,11);
INSERT INTO PLAYS_IN(player, band_id) VALUES(12,6);
INSERT INTO PLAYS_IN(player, band_id) VALUES(8,10);


--TR_PLAYS_IN_BI
BEGIN
   FOR i IN 1..10 LOOP
      INSERT INTO Plays_in (player, band_id)
      VALUES (i, 8);
   END LOOP;
END;

SELECT * FROM PLAYS_IN

---Uso del disparador (RAISE_APPLICATION_ERROR)

---TR_BAND_BI

---Debe fallar porque band_name = NULL
INSERT INTO Band (band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES (num_sequence.NEXTVAL, NULL, 2, 'jazz', TO_DATE('2015-05-01', 'YYYY-MM-
DD'), 5);

---Como band_type = NULL, se toma automáticamente como 'rock'
INSERT INTO Band (band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES (num_sequence.NEXTVAL, 'The Messis', 2, NULL, TO_DATE('2015-05-01',
'YYYY-MM-DD'), 5);

---Debe fallar porque el año es en el 2025 y la fecha debe ser menor a la actual
INSERT INTO Band (band_no, band_name, band_home, band_type, b_date, band_contact)
VALUES (num_sequence.NEXTVAL, 'Football', 2, 'blues', TO_DATE('2025-05-01', 'YYYY-
MM-DD'), 5);

---TR_BAND_BU
```

```
---Debe fallar porque se está modificando una fecha existente; solo se puede modificar si
b_date = NULL
UPDATE Band
SET b_date = TO_DATE('2020-01-01', 'YYYY-MM-DD')
WHERE band_no = 6;

UPDATE Band
SET b_date = TO_DATE('2006-03-02', 'YYYY-MM-DD')
WHERE band_no = 7;

UPDATE Band
SET b_date = TO_DATE('2006-03-27', 'YYYY-MM-DD')
WHERE band_no = 8;

UPDATE Band
SET b_date = TO_DATE('2010-12-17', 'YYYY-MM-DD')
WHERE band_no = 9;

UPDATE Band
SET b_date = TO_DATE('2005-09-14', 'YYYY-MM-DD')
WHERE band_no = 10;

---TR_PLAYS_IN_BU

---Debe fallar porque solo se pueden agregar o eliminar músicos, no eliminarlos
UPDATE Plays_in
SET player = 3
WHERE player = 4 AND band_id = 7;

UPDATE Plays_in
SET player = 27
WHERE player = 12 AND band_id = 6;

---TR_PLAYS_IN_BI
INSERT INTO PLAYS_IN(player, band_id) VALUES (11,8);
INSERT INTO PLAYS_IN(player, band_id) VALUES (12,8);
```