PROGRAMA DE EMPREGO

FORMAWEB IV

DATA:	**Marzo 2022**
CURSO:	PROGRAMA DE EMPREGO FORMAWEB IV
MÓDULO	MF0966_3. Consulta e manipulación de información contida en xestores de datos
UNIDADE FORMATIVA	UF2214: Implementación e uso dunha base de datos
NºEXP:	36/00004/2021

Linguaxes de consulta

As linguaxes de consulta ou as linguaxes **de consulta de datos** (**DQL**) son linguaxes informáticas utilizadas para consultar bases de datos e sistemas de información .

Tipos

En liñas xerais, as linguaxes de consulta pódense clasificar como linguaxes de consulta de bases de datos ou linguaxes de consulta de recuperación de información . A diferenza é que unha linguaxe de consulta de base de datos tenta dar respostas fácticas a preguntas de feito, mentres que unha linguaxe de consulta de recuperación de información tenta atopar documentos que conteñan información relevante para unha área de investigación.

Exemplos de linguaxes de consulta

- Atomese , a linguaxe de consulta de gráficos para a base de datos de gráficos OpenCog , AtomSpace .
- Attempto Controlled English é unha linguaxe de consulta que tamén é unha linguaxe natural controlada.
- AQL é unha linguaxe de consulta para o sistema de base de datos nativo multimodelo de ArangoDB .
- .QL é unha linguaxe de consulta propietaria orientada a obxectos para consultar bases de datos relacionais ; sucesor de Datalog;
- Linguaxe de consulta contextual (CQL) unha linguaxe formal para representar consultas en sistemas de recuperación de información como índices web ou catálogos bibliográficos.
- CQLF (CODYASYL Query Language, Flat) é unha linguaxe de consulta para bases de datos de tipo CODASYL ;
- A linguaxe de consulta orientada ao concepto (COQL) úsase no modelo orientado ao concepto (COM). Baséase nun constructo, un concepto novidoso de modelado de datos e utiliza operacións como a proxección e a deproxección para análise multidimensional, operacións analíticas e inferencia;
- Cypher é unha linguaxe de consulta para a base de datos de gráficos Neo4j ;
- DMX é unha linguaxe de consulta para modelos de minería de datos ;
- Datalog é unha linguaxe de consulta para bases de datos dedutivas ;
- Discovery Query Language é unha linguaxe de consulta para acceder a Watson Discovery Services na nube de IBM ; [dúas]
- F-logic é unha linguaxe declarativa orientada a obxectos para bases de datos dedutivas e representación do coñecemento .
- FQL permítelle usar unha interface de estilo SQL para consultar os datos expostos pola API de Graph . Ofrece funcións avanzadas que non están dispoñibles na API de Graph . [3]
- Gellish English é un idioma que se pode usar para consultas de bases de datos de Gellish English, para diálogos (solicitudes e respostas), así como para modelado de información e modelado de coñecemento ; [4]
- Gremlin é unha linguaxe de atravesamento de gráficos da Apache Software Foundation para sistemas gráficos OLTP e OLAP.
- GraphQL é unha linguaxe de consulta de datos desenvolvida por Facebook como alternativa ás arquitecturas de servizos web REST e ad-hoc .
- HTSQL é unha linguaxe de consulta que traduce consultas HTTP a SQL ;
- ISBL é unha linguaxe de consulta para PRTV , un primeiro sistema de xestión de bases de datos relacionais;
- Jaql é unha linguaxe funcional de procesamento de datos e consultas máis utilizada para procesar consultas JSON;
- JSONiq é unha linguaxe de consulta declarativa deseñada para coleccións de documentos JSON;
- Kusto, ou KQL, é unha linguaxe de consulta usada en Azure Data Explorer e na ferramenta CMPivot en Microsoft System Center Configuration Manager

- As expresións de consulta LINQ son unha forma de consultar varias fontes de datos en linguaxe .NET
- LDAP é un protocolo de aplicación para consultar e modificar servizos de directorio que se executan sobre TCP/IP;
- LogiQL é unha variante de Datalog e é a linguaxe de consulta do sistema LogicBlox.
- MQL é unha linguaxe de consulta cheminformática para unha busca de subestruturas que permite ademais de propiedades nominais tamén propiedades numéricas;
- MDX é unha linguaxe de consulta para bases de datos OLAP;
- N1QL é a linguaxe de consulta de Couchbase para buscar datos nos servidores de Couchbase;
- OQL é a linguaxe de consulta de obxectos;
- OCL (Linguaxe de restrición de obxectos). A pesar do seu nome, OCL é tamén unha linguaxe de consulta de obxectos e un estándar OMG;
- OPath, deseñado para o seu uso en consultas da *tenda* WinFS ;
- OttoQL, deseñado para consultar táboas, XML e bases de datos;
- Poliqarp Query Language é unha linguaxe de consulta especial deseñada para analizar textos anotados. Usado no buscador Poliqarp;
- PQL é unha linguaxe de programación de propósito especial para xestionar modelos de procesos baseados na información sobre escenarios que describen estes modelos;
- PTQL baseado en consultas relacionais sobre trazos de programas, o que permite aos programadores escribir consultas expresivas e declarativas sobre o comportamento do programa.
- QUEL é unha linguaxe de acceso a bases de datos relacionais, similar en moitos aspectos a SQL;
- RDQL é unha linguaxe de consulta RDF;
- ReQL é unha linguaxe de consulta usada en RethinkDB ;
- SMARTS é o estándar quimioinformático para a busca de subestruturas;
- SPARQL é unha linguaxe de consulta para gráficos RDF;
- SPL é unha linguaxe de consulta xerada pola máquina para big data, baseada en Unix Piping e SQL.
- SCL é a linguaxe de control de software para consultar e manipular obxectos de Endevor
- SQL é unha linguaxe de consulta moi coñecida e unha linguaxe de manipulación de datos para bases de datos relacionais;
- SuprTool é unha linguaxe de consulta propietaria para SuprTool, un programa de acceso a bases de datos usado para acceder a datos en bases de datos *Image/SQL* (anteriormente TurboIMAGE) e Oracle;
- TMQL Topic Map Query Language é unha linguaxe de consulta para Topic Maps;
- TQL é unha linguaxe utilizada para consultar a topoloxía dos produtos HP.
- O tutorial D é unha linguaxe de consulta para sistemas de xestión de bases de datos verdadeiramente relacionais (TRDBMS);
- U-SQL é unha linguaxe de procesamento de datos inventada en Microsoft
- XQuery é unha linguaxe de consulta para fontes de datos XML;
- XPath é unha linguaxe declarativa para navegar por documentos XML;
- XSPARQL é unha linguaxe de consulta integrada que combina XQuery con SPARQL para consultar fontes de datos XML e RDF ao mesmo tempo;
- YQL é unha linguaxe de consulta tipo SQL creada por Yahoo!
- Linguaxes de consulta de buscadores, p. ex. Por exemplo, como usa Google ou Bing

...

Linguaxe de consulta SQL

SQL (*Structured Query Language* ou linguaxe estruturado de consulta) está pensado para procesar bases de datos relacionais. Moitas construcións e operadores SQL están asociados con intervencións de álxebra relacional. Desde este punto de vista, os métodos de álxebra relacional son básicos para unha comprensión máis profunda do uso da linguaxe SQL.
.

Linguaxe SQL

Os diferentes sistemas de xestión de bases de datos (DBMS) usan dous tipos de linguaxe SQL:

SQL interactivo: úsase para realizar accións directamente na base de datos en modo en liña. Como regra xeral, algún programa de servidor SQL acompaña SQL interactivo. Os máis populares actualmente son Oracle SQL-Server, MS-SQL Server,
Inter Base e algúns outros.

SQL incorporado: consiste en comandos SQL incrustados directamente en programas escritos noutra linguaxe de programación. Por exemplo, sistemas de programación como Delphi, C++Builder, Visual Basic, Visual C++ teñen SQL incorporado. En varios DBMS, por exemplo en Microsoft -Access,
tamén hai oportunidades de usar comandos SQL incorporados.

Na linguaxe SQL, hai moitos comandos que se relacionan con determinadas accións realizadas. Polo tanto, SQL pódese dividir en subconxuntos específicos:

DQL - linguaxe de consulta, os comandos están deseñados para extraer datos das táboas.
DML - linguaxe de manipulación de datos.
TPL - é unha linguaxe de procesamento de transaccións. Os comandos permítenche combinar equipos de linguaxe HTML en grupos de transaccións. Se non se pode executar un dos comandos, todos os comandos anteriores na mesma transacción son abortados: prodúcese unha "reversión da transacción".
DDL - é unha linguaxe de definición de datos.
Tamén inclúe instrucións de integridade dos datos. Por exemplo, comandos para crear táboas e organizar relacións entre elas.
CCL - Linguaxe de control do cursor. Permite seleccionar unha fila do conxunto resultante de consultas para procesar.
DCL - é unha linguaxe de xestión de datos.
Contén instrucións polas que se realiza a asignación de dereitos de acceso á base de datos, a múltiples táboas ou vistas.

Funcións da linguaxe SQL

SQL (Structured Query Language): é unha linguaxe de consulta estruturada. A linguaxe está orientada a traballar con bases de datos relacionais (tabulares). A linguaxe é sinxela e de feito, consiste en comandos (interpretados), a través dos cales se pode traballar con grandes conxuntos de datos (bases de datos), borrando, engadindo, cambiando información neles e realizando unha cómoda procura.

Para traballar con **código SQL**, necesita un sistema de xestión de bases de datos (DBMS), que proporciona funcionalidades para traballar con bases de datos.

O **sistema de xestión de bases de datos (DBMS)** é unha combinación de linguaxe e ferramentas de software deseñadas para crear, manter e compartir unha base de datos por moitos usuarios.

Compoñentes SQL

A linguaxe SQL consta dos seguintes compoñentes:

- Linguaxe de manipulación de datos (DML)
- Linguaxe de definición de datos (DDL)
- Linguaxe de control de datos (DCL)

1. A linguaxe de manipulación de datos (DML) consta de catro comandos principais:
    - Selección de información da base de datos - SELECT
    - Inserir información nunha táboa de base de datos - INSERT
    - Actualizar (cambiar) a información nas táboas da base de datos - ALTER
    - Eliminar información dunha base de datos - DELETE
    - A linguaxe de definición de datos (DDL) úsase para crear e modificar a estrutura da base de datos e os seus compoñentes: táboas, índices, vistas (táboas virtuais), así como disparadores e procedementos de almacenamento.
      Estes son só algúns dos comandos básicos da linguaxe:

Creación de bases de datos - CREATE DATABASE
Crear táboa - CREATE TABLE
Cambio de táboa (estrutura) - ALTER TABLE
Drop table - DROP TABLE

3. Data Control Language (DCL) utilízase para controlar os dereitos de acceso aos datos e executar procedementos nun ambiente multiusuario.

Análise da linguaxe de consulta para BD relacionais

Durante os anos sesenta e setenta Edgar Frank Codd traballou no que hoxe levou ao modelo relacional. Foi en 1970 cando publicou unha das súas obras máis importantes «Un modelo de datos relacionales para grandes bancos de datos compartidos» e ese foi o xerme das bases de datos actuais.

Xunto co modelo relacional, propón unha sublinguaxe para acceder a ditos datos denominada SEQUEL (Structured English Query Language) que foi o predecesor de SQL (Structured Query Language).

Actualmente utilízase a segunda versión chamada SQL2 ou SQL92, que é un estándar revisado e ampliado do primeiro SQL (SQL1 ou SQL86). A maioría dos actuais sistemas de xestión de bases de datos funcionan con esta versión, que permite unha gran variedade de operacións, aínda que foi revisada posteriormente, introducindo novas funcionalidades ou pequenas modificacións ás existentes.

SQL ten dous subcompoñentes moi importantes: a linguaxe de definición de datos ou LDD e a linguaxe de manipulación de datos interactiva ou LMD:

- O LDD ou Data Definition Language ten todo tipo de comandos para crear as estruturas e esquemas de relación, así como a súa eliminación e modificación. Co LDD, entre outras operacións, podemos crear ou modificar táboas que son os contedores básicos onde se almacenan os datos nunha base de datos relacional, podemos crear relacións entre elas, establecer os seus atributos, etc., así como borralas. Tamén co LDD poderase definir restricións de integridade que deben cumprir os datos almacenados nas devanditas táboas.

Outra característica deste idioma é que pode establecer dereitos de acceso tanto ás relacións como ás vistas, establecendo así o nivel de seguridade necesario.
- Co DML ou Data Management Language pódense consultar estas estruturas e recuperar os datos segundo os criterios establecidos. Esta linguaxe baséase na álxebra relacional e no cálculo relacional. O seu poder e robustez reside sobre todo na base matemática destas dúas últimas linguas.
- Tamén existe o LCD ou Data Control Language que permite ao administrador do sistema ou propietario dun obxecto conceder ou eliminar privilexios sobre el.

Neste capítulo veremos algunhas das características da linguaxe SQL.

Enunciados de definición de datos.

A linguaxe de definición de datos nunha base de datos como o seu nome indica define ou modifica a estrutura dunha base de datos. Ten catro operacións básicas (CREATE, ALTER, DROP e TRUNCATE) e serven para crear, modificar, eliminar ou definir os obxectos dunha base de datos.

Crear (`CREATE`).

Co comando `create` pódense crear novos obxectos nas bases de datos, como táboas, vistas, unha nova base de datos ou procedementos almacenados, entre outros.

Un exemplo básico de creación dunha táboa sería o seguinte:

```
CREATE TABLE raquetas
( raqueta_id number(10) not null,
raqueta_modelo varchar2(50) not null,
raqueta_marca varchar2(50)
);
```

Na declaración anterior podes ver como se crea a táboa de raquetas mediante a cláusula "crear táboa". Entre as parénteses especifícanse os campos dos que constará a táboa. O primeiro será racket_id que é un campo numérico con 10 posicións e non pode ser nulo. O segundo campo almacena o nome da raqueta que, como o anterior, non pode ser nula pero contén un varchar2 ou cadea de caracteres alfanuméricos. O terceiro campo almacena a marca e pode conter datos baleiros (nesta táboa pode haber raquetas sen marca asociada).

> Como podes ver, o comando SQL sempre remata cun punto e coma.

Modificación (`ALTER`).

A través da cláusula ALTER, SQL permite modificar obxectos da base de datos. Por exemplo, ALTER TABLE modificará a estrutura da táboa.

O seguinte exemplo mostra como modificar a estrutura da táboa creada previamente engadindo un novo campo racket_description.

```
ALTER TABLE raquetas
ADD raqueta_descripcion varchar2(150);
```

Tamén é posible mediante a cláusula ALTER modificar un campo xa existente. O seguinte comando cambia a lonxitude do campo racket_description e configúrao para evitar que se introduzan datos baleiros.

```
ALTER TABLE raquetas
MODIFY raqueta_descripcion varchar2(200) not null;
```

Ademais, mediante a mesma cláusula é posible eliminar columnas da táboa. No seguinte exemplo eliminamos o campo racket_description da táboa.

```
ALTER TABLE raquetas
DROP COLUMN raqueta_descripcion;
```

Eliminación (`DROP`)

O comando DROP axudaranos a eliminar ambos índices, táboas ou bases de datos. Vexamos un exemplo diso:

```
DROP TABLE raquetas;
```

Imaxinemos que queremos eliminar a base de datos da tenda. Bastaría con executar o seguinte comando:

```
DROP DATABASE database_name;
```

Para soltar un índice, o comando pode variar dun xestor de bases de datos a outro. Vexamos algúns exemplos:

Os seguintes comandos eliminarán o índice de indraquetas da táboa de raquetas.

Sintaxe do índice DROP en MS Access:

```
DROP INDEX indraquetas ON raquetas;
```

Sintaxe do índice DROP en MS SQL Server:

```
DROP INDEX raquetas.indraquetas;
```

Sintaxe do índice DROP en DB2/Oracle:

```
DROP INDEX indraquetas;
```

Sintaxe do índice DROP en MySQL:

```
ALTER TABLE raquetas DROP INDEX indraquetas;
```

Eliminado ou truncado (`TRUNCATE`).

Imaxinemos que só queremos borrar os datos da táboa de raquetas. Bastaría con truncar a táboa para conseguilo:

```
TRUNCATE TABLE raquetas;
```

Sentencias de manipulación de datos.

As instrucións de manipulación de datos forman parte da linguaxe de manipulación de datos (DML). A linguaxe de manipulación de datos permite realizar consultas, actualizar información, inserila e eliminala. As instrucións utilizadas son SELECT, INSERT, UPDATE e DELETE. Vexamos cada un deles con máis detalle:

> **Optimizar consultas.**
>
> - Unha consulta, dependendo de como se escriba, pode levar ao xestor de menos dun segundo a máis de 15 minutos (ou incluso máis).
> - Moitas veces os administradores de datos avalían as consultas e optimízanas para que se executen máis rápido no xestor e non conxestionen o sistema. Dependendo do xestor de base de datos, a optimización pode ser diferente.
> - Ás veces os administradores teñen que crear índices para acelerar as consultas.
> - A creación de índices non se pode facer dun xeito tolo xa que melloran as consultas senón que penalizan as inserción s e actualizacións, polo que hai que facelo con criterio.

Selección de datos (`SELECT`).

A instrución SELECT básica en SQL contén as cláusulas SELECT e FROM. Vexamos un exemplo diso:

```
SELECT nome, apelido FROM alumno;
```

Esta declaración selecciona o nome e apelidos de todos os estudantes existentes.

Podemos filtrar os estudantes cuxo nome é "EMMA". Para iso teremos que engadir a cláusula WHERE.

```
SELECT nome, apelido FROM alumno WHERE nome = 'EMMA';
```

A cláusula WHERE, como podemos ver, úsase para filtrar o resultado dunha instrución SELECT. Moitas veces temos que usalo porque non sempre queremos recuperar todos os datos dunha táboa.

Cláusula `DISTINCT`.

Imaxinemos que queremos extraer os distintos apelidos da táboa do alumnado. Necesitamos executar a seguinte instrución:

> SELECT apelido DISTINTO DO estudante;

DISTINCT eliminará as filas duplicadas que atope no resultado.

Cláusula `ORDER BY`.

Polo xeral, gustaríanos ter ordenados os resultados da consulta, para iso utilizaremos a instrución ORDER BY. Un exemplo de ORDER BY sería o seguinte:

> SELECT apellidos,nombre FROM alumno ORDER BY apellidos ASC, nombre;

Ou ben:

> SELECT apellidos,nombre FROM alumno ORDER BY 1, 2;

As cláusulas ASC e DESC axudaranos a ordenar o resultado ascendente (ASC) ou descendente (DESC).

Inserción de datos (`INSERT`).

A cláusula INSERT permítelle inserir datos nunha táboa. Xeralmente úsase a oración normal ou abreviada. A declaración normal tería o seguinte formato:

> INSERT INTO ‘nombreTabla’ (‘columna1’,[‘columna2,... ‘]);
>
> VALUES (‘valor1’, [‘valor2,...’])

Un exemplo de INSERT sería o seguinte:

> INSERT INTO alumnos (nombre, apellido)
>
> VALUES (‘Robert’, ‘Sinnock’);

A instrución abreviada é similar pero omítense os nomes das columnas porque se supón que se coñecen:

```
INSERT INTO nombreTabla VALUES (‘valor1’, [‘valor2,...’]);
```

O seguinte exemplo asume que "nome" e "apelido" son as únicas columnas da táboa "alumnos":

```
INSERT INTO alumnos

VALUES (‘Juan Carlos’, ‘Moreno’);
```

Nese caso, e para evitar erros, hai que poñer todas as columnas da táboa e na mesma orde na que foron creadas.

Actualización de datos (`UPDATE`).

Coa cláusula UPDATE podemos modificar os valores dunha serie de rexistros dunha táboa.

Un exemplo básico da estrutura deste comando sería o seguinte.

```
UPDATE nombreTabla SET campo1 = ‘valor1’ WHERE campo2 = ‘valor2’;
```

Un exemplo de uso sería o seguinte:

```
UPDATE alumnos SET nombre = ‘Sancho’ WHERE apellidos = ‘Torres Maestre’;
```

A declaración anterior actualizará o campo do nome en todos os rexistros (cambiarao a 'Sancho') cuxo campo de apelidos é igual a 'Torres Maestre'.

Eliminación de datos (`DELETE`).

Coa cláusula DELETE podemos eliminar todos ou varios dos rexistros dunha táboa.

A estrutura básica do comando DELETE sería a seguinte:

```
DELETE FROM tableName;
```

ou

```
DELETE FROM tableName WHERE column1 = 'valor1';
```

Exemplos de uso

```
DELETE FROM alumnos;
```

Neste comando anterior borraranse todos os datos da táboa. Se queremos filtrar que datos se eliminarán podemos usar o seguinte comando:

```
DELETE FROM estudantes WHERE apelidos = 'Moreno';
```

Neste caso, só se eliminarán aqueles alumnos cuxo apelido sexa 'Moreno'.

Sentenzas de concesión e revogación de privilexios.

Os comandos para conceder e denegar permisos en SQL son GRANT e REVOKE. Ambos pertencen á linguaxe de control de datos LCD ou SQL. O administrador do sistema ou o propietario da base de datos usa estes dous comandos para conceder ou eliminar privilexios nun obxecto de base de datos.

A sintaxe deste comando (GRANT) é a seguinte:

```
GRANT privilegio

ON objeto

TO usuario | PUBLIC | rol

[WITH GRANT OPTION];
```

Vexamos unha a unha o que significa cada unha destas cláusulas:

- privilexio. É o privilexio da base de datos que quere conceder, por exemplo SELECT.
- obxecto. É o obxecto ao que se lle vai conceder o privilexio. Por exemplo a mesa dos estudantes
- usuario, PÚBLICO ou rol. Son os destinatarios do privilexio, pode ser un usuario concreto, PUBLIC sería para todos ou podemos asignar o privilexio a un rol.
- CON OPCIÓN DE SUBVENCIÓN. É un argumento opcional e significa que o usuario poderá conceder privilexios sobre ese obxecto a outros usuarios.

```
GRANT SELECT ON alumnos TO emma WITH GRANT OPTION;
```

Vexamos a sintaxe do comando REVOKE:

```
REVOKE privilegio

ON objeto

FROM usuario | PUBLIC | rol;
```

Un exemplo deste comando sería o seguinte:

```
REVOKE SELECT ON alumnos FROM emma;
```

Como podes ver, no comando anterior elimínase o permiso SELECT do usuario emma na táboa de estudantes.

Funcións e procedementos almacenados.

Os procedementos e funcións almacenadas son moi utilizados en grandes bases de datos xa que permiten automatizar algunha serie de procesos na base de datos e, polo tanto, mellorar o rendemento xa que os clientes que se conectan á base de datos non teñen que enviar tanta información senón que é a base de datos a que si. o traballo.

Gran parte do traballo faise agora no servidor e non no cliente e, polo tanto, aumenta un pouco a carga do servidor. Non obstante, o uso de funcións e procedementos almacenados en xeral mellora o rendemento do sistema.

Polo xeral, cando hai moitos procedementos almacenados, o máis sensato é crear bibliotecas agrupándoos por características.

> Algúns sistemas de xestión de bases de datos usan a linguaxe Java como linguaxe de programación.

Cando é mellor usar funcións e procedementos almacenados?

- Cando varias aplicacións cliente acceden á mesma base de datos realizando as mesmas operacións.
- Cando queremos simplificar a programación de aplicacións cliente.
- Cando queremos aumentar a seguridade. Os trámites permítennos garantir que as operacións se realizan correctamente e que non existen accesos ou operacións indebidas.

Vexamos un exemplo dunha función en MySQL:

```
mysql> delimiter //

mysql> CREATE FUNCTION hola (s CHAR(25)) RETURNS CHAR(50)
-> RETURN CONCAT('Hola ',s);
-> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;

mysql> SELECT hola('Emma') as saludo;
+————————+
|   saludo    |
+————————+
| Hola Emma  |
+————————+
1 row in set (0.00 sec)
```

Comentemos liña por liña cada un dos comandos:

- límite //. Neste comando dicímoslle a MySQL que use // como delimitador en lugar do punto e coma xa que imos usar o punto e coma dentro da nosa función e non queremos que o tome como o final da función.
- CREATE FUNCTION hello (s CHAR(25)) RETURNS CHAR(50) RETURN CONCAT('Hello',s);. Nestas liñas creamos a función. Como podes ver, a función hello admite un parámetro s que é de tipo CHAR de lonxitude 25 e devolve un CHAR de lonxitude 50. A función usa CONCAT para concatenar a palabra Hello ao parámetro co que se chama e devolve todo isto usando a declaración RETURN. .
- delimitar;. Restablecemos o delimitador a un punto e coma de novo.

- SELECCIONE ola('Emma') como saúdo;. Finalmente facemos unha chamada á función coa instrución SELECT pasando o parámetro 'Emma'.

Vexamos agora un exemplo dun procedemento almacenado en MySQL:

```
mysql> delimiter //

mysql> CREATE PROCEDURE elemental (OUT par INT)
-> BEGIN
->  SELECT COUNT(*) INTO par FROM alumnos;
-> END
-> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;

mysql> CALL elemental(@cuantos);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @cuantos;
+———————+
| @cuantos  |
+———————+
|    1000    |
+———————+
1 row in set (0.00 sec)
```

Como podes ver, o procedemento elemental chamado comeza coas cláusulas CREATE PROCEDURE e remata coa instrución END. O procedemento devolve un parámetro (especificado con OUT) que chamamos par que é de tipo numérico. No corpo do procedemento realízase un reconto (COUNT(*)) das filas que ten a táboa do alumno e o resultado gárdase por parellas.

Finalmente, para executar o procedemento denominado elemental, empregamos a instrución CALL e posteriormente mostramos con SELECT o valor que devolve esta.

Disparadores.

Un disparador ou disparador é un procedemento que se executa cando se produce unha determinada condición nas operacións nunha base de datos. Os disparadores adoitan configurarse para activarse cando se produce unha inserción (INSERT), borrado (DELETE) ou actualización (UPDATE), pero algunhas bases de datos poden executar disparadores ao crear, eliminar ou modificar usuarios, táboas, etc.

A sintaxe MySQL dun disparador é a seguinte:

```
CREATE
[DEFINER = { usuario | CURRENT_USER }]
TRIGGER trigger_nombre
trigger_momento trigger_evento
ON tabla_nombre FOR EACH ROW
trigger_cuerpo
```

Onde a cláusula trigger_moment pode ter os seguintes valores:

- BEFORE
- AFTER

Onde a cláusula trigger_event pode ter os seguintes valores:

- INSERT
- UPDATE
- DELETE

Vexamos un exemplo sinxelo de disparadores feitos en MySQL:

Imos crear a táboa de débedas. A táboa terá dous campos, o nome do cliente e a débeda.

```
CREATE TABLE deudas (cliente INT, debe DECIMAL(10,2));
```

A continuación imos crear o disparador chamado sum_debt que acumula na variable suma o total das débedas:

```
CREATE TRIGGER suma_deuda BEFORE INSERT ON deudas

FOR EACH ROW SET @suma = @suma + NEW.debe;
```

A continuación inicializamos a variable suma a cero e inserimos as débedas de tres clientes (cliente 1, 2 e 3).

```
mysql> SET @sum = 0;

mysql> INSERT INTO deudas VALUES(1,20.90);

mysql> INSERT INTO deudas VALUES(2,200.50);

mysql> INSERT INTO deudas VALUES(3,50.00);

mysql> SELECT @sum AS 'Deuda total';

+————————-+

| Deuda total  |

+————————-+

| 271.40     |

+————————-+
```

Finalmente, consultamos a variable suma (SELECT @sum AS 'Total Debt';) á que lle damos o alias "Total Debt".

MRZ 2022


galicia
Xacobeo 21·22
CONCELLO DE VIGO
OBRADOIROS DE EMPREGO DE GALICIA
UNIÓN EUROPEA
XUNTA DE GALICIA