

Proyecto Final SQL

Academia Baile Mambolaye

**Alumno:** Ernesto Vega

**Profesor:** Miguel Rodas

**Tutor:** Juan Rabazzi

Contenido

|  |  |  |
| --- | --- | --- |
| Pto. |  | Pág. |
| 1.- | Introducción | 3 |
| 2.- | Objetivo | 3 |
| 3.- | Modelo de negocio | 3 |
| 4.- | Situación problemática | 3 |
| 5.- | Diagrama E-R | 4 |
| 6.- | Descripción de tablas | 5 |
| 7.- | Scripts SQL objetos | 7 |
| 8.- | Script SQL inserción | 7 |
| 9.- | Informes generados | 8 |
| 10.- | Herramientas y tecnologías usadas | 13 |
| 11.- | Respaldo de información (BackUp) | 14 |
| 16.- | Scripts DCL & TCL | 16 |

**1.- Introducción**

Mediante el presente documento contiene toda la información recopilada, scripts generados y todos los requerimientos que han sido solicitados para presentar el Proyecto Final SQL Comisión 31270 de CoderHouse.

Los scripts SQL que se detallan en este proyecto se encuentra en el directorio **Scripts** en el presente [repositorio](https://github.com/ernieve/SQL_Coder), al inicio de cada descripción se indica el nombre del respectivo archivo con su extensión y dirección URL, esto con el fin de facilitar su ubicación.

**2.- Objetivo**

Crear una Bases de Datos relacional para llevar un mejor registro de alumnos, actividades, clases impartidas y demás personas involucradas en la Academia Mambolaye.

**3.- Modelo de Negocio**

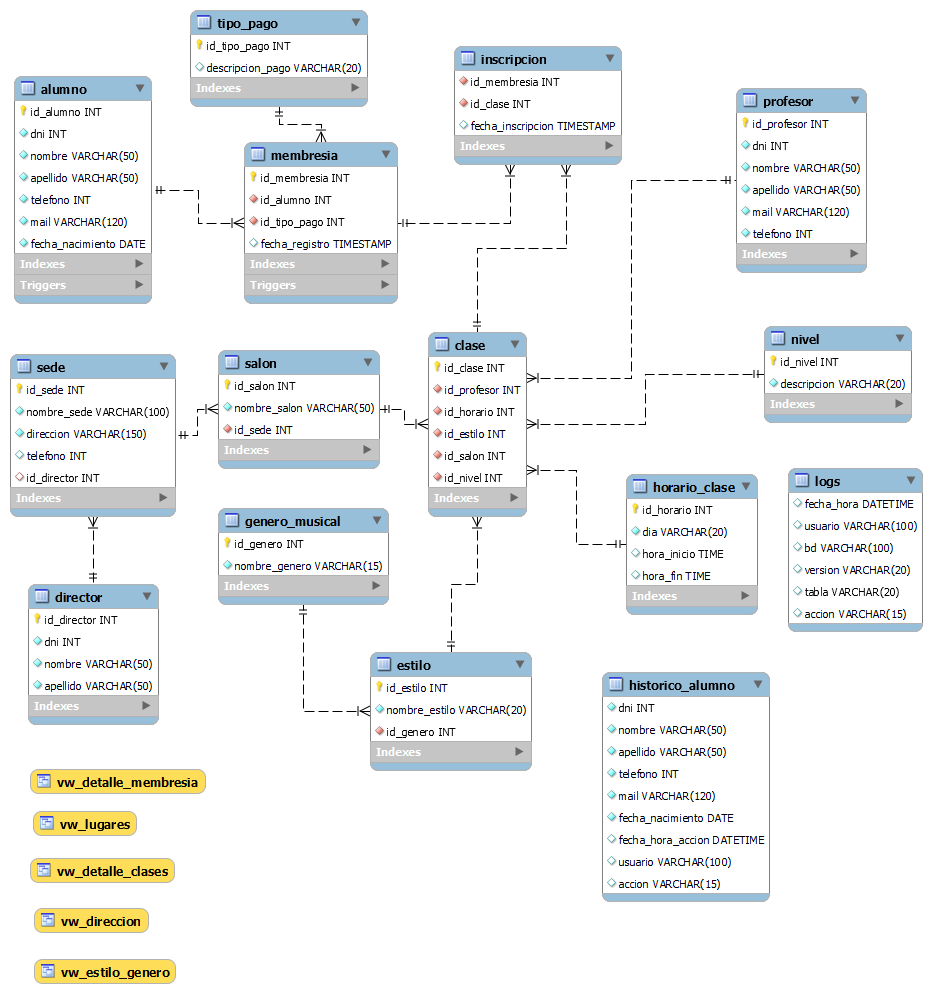
La Academia Mambolaye es una escuela de baile que se dedica a enseñar diversos estilos de baile como Mambo (Salsa on2), Mambo Lady Style, Bachata Estilo Pacheco y próximamente se irán agregando más estilos.

Mambolaye se fundó en Venezuela y actualmente cuenta con un solo espacio de enseñanza en CABA ubicado en Almagro, con miras de expansión. El nombre de la academia está compuesto por dos palabras **Mambo** que es el estilo de baile basado en la Salsa y **Laye** que significa calle, la fusión de estas dos se defino como *el estilo de baile con un toque propio de la persona que solo se obtiene fuera de salones de baile*.

**4.-** **Situación Problemática**

Actualmente la academia no cuenta con un sistema de registros para los actores ni actividades que se realizan en ella, se maneja de manera manual y la automatización es casi nula, por ende, con el presente proyecto aplicando un modelo de Bases de Datos Relacional se busca resolver esta problemática y así poder llevar un mejor registro de todo lo que se realiza en la institución.

**5.- Diagrama E-R**



**6.- Descripción de tablas**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **alumno (TD)** | | | | | | |
| **Campo** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_alumno | int | autoinc | x | x |  | Datos de los alumnos de la academia |
| dni | int |  | x |  |  |
| nombre\_alumno | varchar | 50 | x |  |  |
| apellido\_alumno | varchar | 50 | x |  |  |
| telefono | int |  | x |  |  |
| mail | varchar | 120 | x |  |  |
| fecha\_nacimiento | date | 10 | x |  |  |
|  |  |  |  |  |  |  |
| **tipo\_pago (TD)** | | | | | | |
| **Campo** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_tipo\_pago | int | autoinc | x | x |  | Informacion de los metodos de pago |
| descripcion\_pago | varchar | 20 |  |  |  |
|  |  |  |  |  |  |  |
| **Membresia (TT)** | | | | | | |
| **Campo** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_membresia | int | autoinc | x | x |  | Datos de la membresia de cada alumno |
| id\_alumno | int |  | x |  | x |
| id\_tipo\_pago | int |  | x |  | x |
| fecha\_registro | Timestamp |  | x |  | x |
|  |  |  |  |  |  |  |
| **profesor (TD)** | | | | | | |
| **Campo** | **Tipo** | **Longitud** | **NOT NUL** | **PK** | **FK** | **Descripcion** |
| id\_profesor | int | autoinc | x | x |  | Contiene informacion de los profesores de baile |
| dni | int |  | x |  |  |
| nombre | varchar | 50 | x |  |  |
| apellido | varchar | 50 | x |  |  |
| mail | varchar | 120 | x |  |  |
| telefono | int |  | x |  |  |
|  |  |  |  |  |  |  |
| **genero\_musical (TD)** | | | | | | |
| **Campo** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_genero | int | autoinc | x | x |  | Descripcion del genero musical |
| nombre\_genero | varchar | 15 | x |  |  |
|  |  |  |  |  |  |  |
| **estilo (TD)** | | | | | | |
| **Campo** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_estilo | int | autoinc | x | x |  | Estilos de baile relacionados con los generos musicales |
| nombre\_estilo | varchar | 20 | x |  |  |
| id\_genero | int |  | x |  | x |
|  |  |  |  |  |  |  |
| **horario\_clase (TD)** | | | | | | |
| **Campos** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_horario | int | autoinc | x | x |  | Tabla de horarios de clases |
| dia | varchar | 20 | x |  |  |
| hora\_inicio | Time |  |  |  |  |
| hora\_fin | time |  |  |  |  |
|  |  |  |  |  |  |  |
| **director(TD)** | | | | | | |
| **Campos** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_director | int | autoinc | x | x |  | Datos de los directores de las sedes |
| dni | int |  | x |  |  |
| nombre | varchar | 50 | x |  |  |
| apellido | varchar | 50 | x |  |  |
|  |  |  |  |  |  |  |
| **sede (TD)** | | | | | | |
| **Campos** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_sede | int | autoinc | x | x |  | Contiene informacion de la(s) sede(s) de la academia |
| nombre\_sede | varchar | 100 | x |  |  |
| direccion | varchar | 150 | x |  |  |
| telefono | int |  |  |  |  |
| id\_director | int |  |  |  | x |
|  |  |  |  |  |  |  |
| **nivel (TD)** | | | | | | |
| **Campo** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_genero | int | autoinc | x | x |  | Niveles de baile |
| descripcion | varchar | 20 | x |  |  |
|  |  |  |  |  |  |  |
| **salon (TD)** | | | | | | |
| **Campos** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_salon | int | autoinc | x | x |  | Contiene datos de salon de practica |
| nombre\_salon | varchar | 50 | x |  |  |
| id\_sede | int |  | x |  | x |
|  |  |  |  |  |  |  |
| **clase (TD)** | | | | | | |
| **Campo** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_clase | int | autoinc | x | x |  | Contiene informacion sobre las clases que brinda la academia |
| id\_profesor | int |  | x |  | x |
| id\_horario | int |  | x |  | x |
| id\_estilo | int |  | x |  | x |
| id\_salon | int |  | x |  | x |
| id\_nivel | int |  | x |  | x |
|  |  |  |  |  |  |  |
| **inscripcion (TH)** | | | | | | |
| **Campo** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| id\_registro | int | autoinc | x | x |  | Contiene datos de la inscripcion de cada alumno |
| fecha\_registro | datetime | 16 | x |  |  |
| tipo\_pago | varchar | 20 | x |  |  |
| id\_clase | int |  | x |  | x |
| id\_alumno | int |  | x |  | x |
|  |  |  |  |  |  |  |
| **logs (TT)** | | | | | | |
| **Campo** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| fecha\_hora | datetime | autoinc | x | x |  | Contiene informacion sobre las clases que brinda la academia |
| usuario | varchar | 100 | x |  | x |
| bd | varchar | 100 | x |  | x |
| version | varchar | 20 | x |  | x |
| tabla | varchar | 20 | x |  | x |
| accion | varchar | 15 | x |  | x |
|  |  |  |  |  |  |  |
| **historico\_alumnos (TT)** | | | | | | |
| **Campo** | **Tipo** | **Longitud** | **NOT NULL** | **PK** | **FK** | **Descripcion** |
| dni | int |  | x |  |  | Contiene informacion sobre las clases que brinda la academia |
| nombre | varchar | 50 | x |  |  |
| apellido | varchar | 50 | x |  |  |
| telefono | int |  | x |  |  |
| mail | varchar | 120 | x |  |  |
| fecha\_nacimiento | date |  | x |  |  |
| fecha\_hora\_accion | datetime |  | x |  |  |
| usuario | varchar | 100 | x |  |  |
| accion | varchar | 15 | x |  |  |

**7.- Scripts SQL Objetos**

*7.1.- Crear Schema y tablas* [*Scripts/academia\_mambolaye.sql*](https://github.com/ernieve/SQL_Coder/blob/master/Scripts/academia_mambolaye.sql)

*7.2.- Vistas de la BD* [*Scripts/ScripVisVega.sql*](https://github.com/ernieve/SQL_Coder/blob/master/Scripts/ScripVisVega.sql)

*7.3.- Funciones* [*Scripts/FormatoVega.sql*](https://github.com/ernieve/SQL_Coder/blob/master/Scripts/FormatoVega.sql)

*7.4.- Stored Procedures* [*Scripts/* *StoredVega.sql*](https://github.com/ernieve/SQL_Coder/blob/master/Scripts/StoredVega.sql)

*7.5.- Triggers* [*Scripts/* *TriggersVega.sql*](https://github.com/ernieve/SQL_Coder/blob/master/Scripts/TriggersVega.sql)

**8.- Script SQL inserción**

Los registros de la BD fueron cargados mediante script ([Scripts/ScriptInsVega.sql](https://github.com/ernieve/SQL_Coder/blob/master/Scripts/ScriptInsVega.sql)), no se dispuso de algún datasets, motivado que el propósito de este proyecto es hacerlo muy cercano a la realidad actual y planes a futuro de la academia.

**9.- Informes generados**

**9.1.- Vistas**

**Vista 1: vw\_direccion**

**Objetivo**: Poder observar las sedes de la academia y sus respectivos directores en una sola consulta.

**Descripción:** Se realiza una consulta a la tabla *sede* (*nombre\_sede*), luego se realiza un join de la concatenación con la tabla *director* (concatenación campos *nombre* y *apellido*), la asociación de los campos se realiza mediante el *id\_director* entre ambas tablas

|  |  |
| --- | --- |
| **Ejemplo:** |  |

**Vista 2: vw\_ estilo\_genero**

**Objetivo:** Debido que los estilos de baile pueden variar para un género musical, mediante esta vista se puede obtener el género y sus distintos tipos de estilo.

**Descripción:** Se realiza consulta a la tabla *estilo* (*nombre\_estilo*), posteriormente se ejecuta un join hacia la tabla *genero\_musical* (*nombre\_genero*), la asociación entre las dos tablas se realiza mediante el campo *id\_genero.*

|  |  |
| --- | --- |
| **Ejemplo:** |  |

**Vista 3: vw\_detalle\_clases**

**Objetivo:** La tabla clase está conformada por distintos ID que pertenecen a otras tablas, esto a la vista humana es poco entendible, por ende, esta vista asocia los distintos ID con su respectiva tabla trayendo la descripción de los mismos y así la información sea más acorde a la vista del usuario.

**Descripción:** Esta vista se basa en la consulta a las tablas *clase* (*id\_clase)*, *estilo* (*nombre\_estilo*), *profesor* (concatenación *nombre* y *apellido*), horario\_clase (concatenación *dia, hora\_inicio, hora\_fin*), *nivel* (*descripción*) y *salón* (*nombre\_salon*) las asociaciones para mostrar los valores correspondientes se realizan mediante varios join’s de los id’s que conforman la tabla clase y que son llaves foráneas de las otras tablas antes mencionadas.

|  |  |
| --- | --- |
| **Ejemplo:** |  |

**Vista 4: vw\_lugares**

**Objetivo:** Observar cuales son los salones que se encuentran en cada sede en una sola consulta.

**Descripción:** Realiza consulta a la tabla *sede* (*nombre\_sede, dirección, teléfono*) y a la tabla *salón* (*nombre\_salon*) asociando los distintos valores mediante el *id\_sede*.

|  |  |
| --- | --- |
| **Ejemplo:** |  |

**Vista 5: vw\_detalle\_membresia**

**Objetivo:** La tabla membresía está conformada por ID’s que hacen referencia a las otras tablas, esto a la vista del usuario es inentendible o de difícil compresión, por ello esta vista buscar observar toda la información de los alumnos que han obtenido una membresía y con cual método de pago fue registrado ordenada de manera ascendente.

**Descripción:** La presente vista realiza una consulta general a la tabla alumno y a *tipo\_pago* (*descripcion\_pago*), y luego realiza un join con la tabla *membresia* asociando los distintos ID’s que conforman dicha tabla.

|  |  |
| --- | --- |
| **Ejemplo:** |  |

**9.2.- Funciones**

**Función 1: cant\_alumnos\_x\_clase**

**Objetivo:** Contabilizar cuantos alumnos inscritos en una clase especifica indicando el id de la clase.

**Descripción:** La función recibe por parámetro un valor entero (INT) el cual debe corresponder a una clase y la función realiza un SELECT COUNT de la tabla *inscripción*.

|  |  |
| --- | --- |
| **Ejemplo:** |  |

**Función 2: ult\_alumno\_x\_clase**

**Objetivo:** Retornar el nombre del último alumno que se registró a una clase en especifica.

**Descripción:** Se ingresa un entero que corresponde al ID de la clase a consultar y la función retornar una cadena de texto (VARCHAR) con el nombre y apellido del estudiante.

|  |  |
| --- | --- |
| **Ejemplo:** |  |

**9.3.- Stored Procedures**

**SP 1: sp\_orden\_x\_campo**

**Objetivo:** Mostrar datos ordenados de una tabla indicada por el usuario, esto permitirá al usuario que trabaje directamente en la BD ahorrar tiempo y código de sentencia.

**Descripción:** Para hacer uso del SP solo deberá ingresar el nombre de la tabla, la columna y si es en orden ascendente o descendente.

|  |  |
| --- | --- |
| **Ejemplo:** |  |

**SP 2: sp\_delete\_alumno**

**Objetivo:** Eliminar alumno que ya no se encuentre en la academia, el SP ayuda a evitar la posibilidad de eliminar registros por equivocación y omisión de la cláusula WHERE. Cuenta con una validación de longitud por si el usuario no ingresa por completo la secuencia de números que conforman un DNI.

**Descripción:** Se debe ingresa el numero de DNI a eliminar de la tabla alumno.

|  |  |
| --- | --- |
| **Ejemplo:** |  |

**9.4.- Triggers**

**General:** Se idearon y crearon varios disparadores para el registro de algunas sentencias DML que el usuario pudiese realizar en las tablas membresías y alumnos.

**Nomenclatura de los triggers:**

|  |  |
| --- | --- |
| TipoTrigger | BEF = before (antes de ejecutar la acción) |
| AFT = after (luego de ejecutar la acción) |
| AccionDML | DEL = Eliminar registro |
| INS = Insertar registro |
| UPD = Actualización registro |

**Sintaxis de uso:**

CREATE TRIGGER `<TipoTrigger>\_<AccionDML>\_<NombreTabla>`

… Bloque de acción del trigger…

**TR Logs:**

**Objetivo:** Generar un registro de acciones DML que se puedan realizar sobre las tablas membresía y alumnos.

**Descripción:** Por cada acción Insert, Update y/o Delete que realice el usuario se realizara un registro en la tabla logs con la siguiente información: Fecha de la acción, nombre de usuario, nombre BD, versión de la BD, nombre tabla (membresía o alumno) y acción (INSERT/UPDATE/DELETE).

|  |  |
| --- | --- |
| Ejemplo: |  |

**TR Histórico**

**Objetivo:** Generar una copia de los registros que se insertan o actualizan en la tabla alumnos generando de este modo un respaldo de los alumnos agregados y actualizaciones que se puedan ejecutar en la tabla.

**Descripción:** Por cada acción INSERT y UPDATE que realiza el usuario sobre la tabla alumno se generara un registro en la tabla historico\_alumno, con todos los campos hayan sido modificados o no, seguido de fecha\_hora\_accion, nombre de usuario y breve descripción de la acción que genero ese registro

|  |  |
| --- | --- |
| Ejemplo: |  |

**10.- Herramientas y tecnologías usadas**

**Draw.io:** Se realizo en primera instancia para la creación y moldeado de las tablas, posibles campos que contendrían cada tabla y bosquejo de Diagrama ER.

**Microsoft Excel:** Se utilizo para el armado de los detalles correspondientes a cada tabla y así poder asegurar el tipo de datos que puede contener cada campo

**MySQL Workbench:** SGBD usado para la creación de los scripts SQL de todos los objetos de la BD de este proyecto.

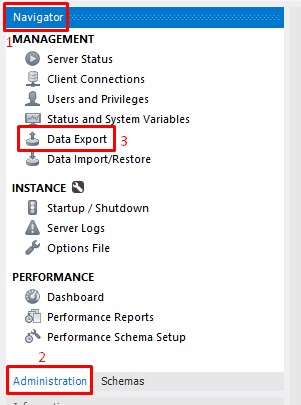
**Git:** Herramienta de control de versioneslocal, de este modo pude tener un control de todos los cambios que se pudiese realizar a lo largo del proyecto y poder comparar en caso de algún error.

**GitHub:** Plataforma basada en sistema de GIT que permite la publicación de repositorios públicos, fue usada para la entrega de los desafíos de toda la comisión.

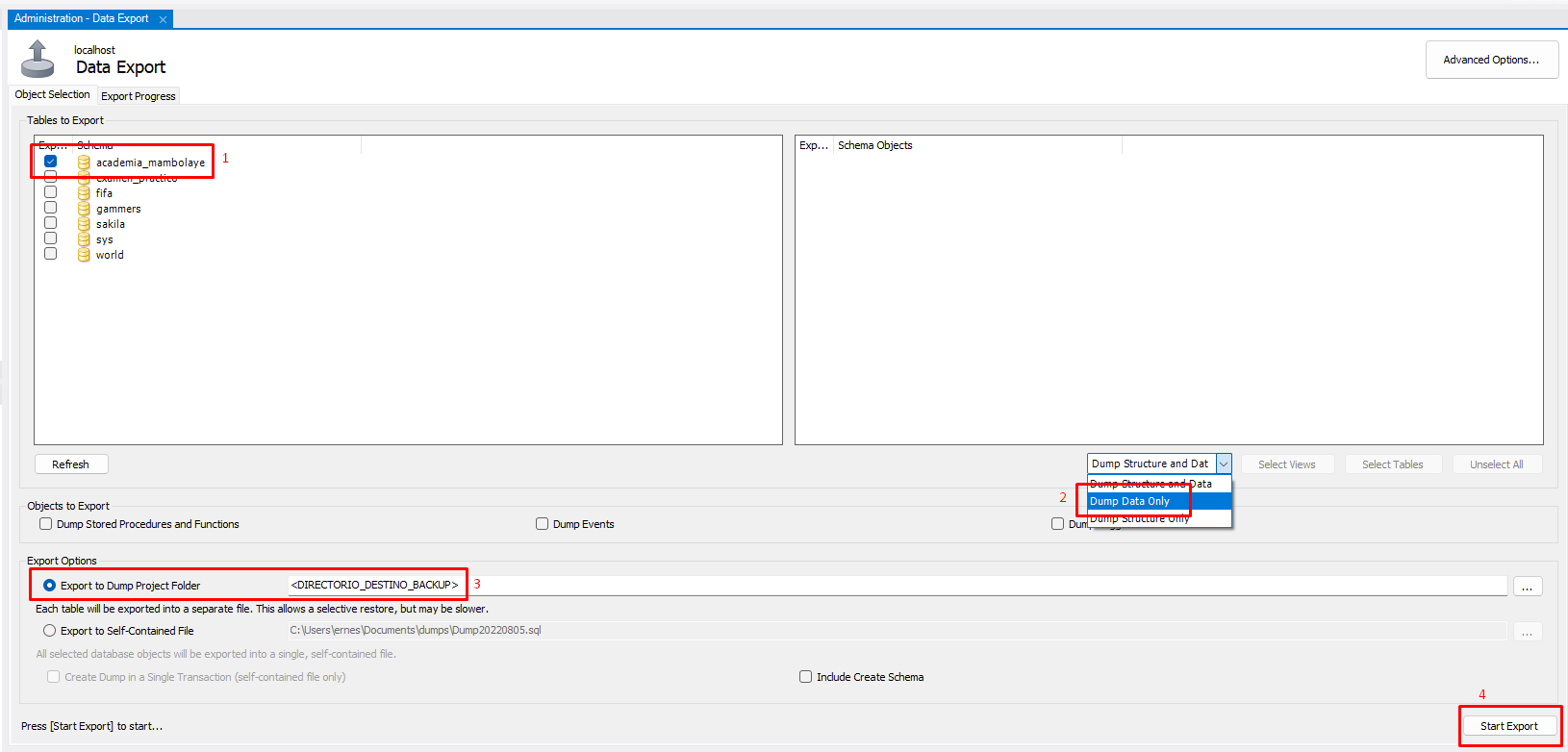
**11.- Respaldo de información (BackUp):**

Se realiza un paso a paso del procedimiento a seguir para la creación del SQL de respaldo correspondiente a la BD del proyecto:

Paso 1.- Navigator > Administration > Data Export



Paso 2.- Seleccionar Schema > Dump Data Only > Ingresar Directorio Destino > Start Export



Este proceso generará un archivo .sql con el nombre que se le haya indicado el cual contendrá toda la información de la BD seleccionado en el paso 2.

Para este proyecto se generó el archivo [BackupVega.sql](https://github.com/ernieve/SQL_Coder/blob/master/Scripts/BackupVega.sql) que contiene el respaldo de la información del proyecto.

Para la recuperación o importación de la información solo se debe seleccionar la opción **Data Import/Restore** del paso 1 y seleccionar el archivo que contiene la información que se desea recuperar.

**12.- Scripts DCL & TCL**

**- DCL Data Control Language (**[**Scripts/SentenciasVega.sql**](https://github.com/ernieve/SQL_Coder/blob/master/Scripts/SentenciasVega.sql)**):** Script para la creación de usuarios user\_1 con todos los permisos sobre la BD y tablas, y user\_2 con solo permiso de lectura, inserción y actualización de registros.

**- TCL Transaction Control Language (**[**Scripts/TCLVega.sql**](https://github.com/ernieve/SQL_Coder/blob/master/Scripts/TCLVega.sql)**):** Script del desafío 20 de la cursada donde se ejecutan acciones de borrado con y sin Rollback, y además se realiza la implementación de SAVEPOINT con registros de prueba