

# Proyecto de creación de base de datos como repositorio para que los modelos de machine learning puedan determinar la calidad vino

Luis Alberto Mosquera<sup>1</sup>, Jeisson Javier Garcia<sup>2</sup>, Johan Steven Peñaloza<sup>3</sup>

<sup>1-2</sup>Dpto. de Ciencias exactas e Ingenieria  
Universidad Central  
Maestría en Analítica de Datos  
Curso de Bases de Datos  
Bogotá, Colombia  
{<sup>1</sup>lmosquerad, <sup>2</sup>jgarciaa15, <sup>3</sup>jpenalozal}@ucentra.edu.co  
May 25, 2023

## Contents

<b>1</b>	<b>Introducción</b>	<b>3</b>
<b>2</b>	<b>Características del proyecto de investigación que hace uso de Bases de Datos</b>	<b>4</b>
2.1	Titulo del proyecto de investigación . . . . .	4
2.2	Objetivo general . . . . .	4
2.2.1	Objetivos especificos . . . . .	4
2.3	Alcance . . . . .	4
2.4	Pregunta de investigación . . . . .	4
2.5	Hipotesis . . . . .	5
<b>3</b>	<b>Reflexiones sobre el origen de datos e información</b>	<b>6</b>
3.1	¿Cual es el origen de los datos e información . . . . .	6
3.2	¿Cuales son las consideraciones legales o eticas del uso de la información? . . . . .	6
3.3	¿Cuales son los retos de la información y los datos que utilizara en la base de datos en terminos de la calidad y la consolidación? . . . . .	7
3.4	¿Que espera de la utilización de un sistema de Bases de Datos para su proyecto? . . . . .	7

<b>4</b>	<b>Diseño del Modelo de Datos del SMBD (Sistema Manejador de Bases de Datos)</b>	<b>8</b>
4.1	Características del SMBD (Sistema Manejador de Bases de Datos) para el proyecto . . . . .	8
4.2	Diagrama modelo de datos . . . . .	9
4.3	Imágenes de la Base de Datos . . . . .	9
4.4	Código SQL - lenguaje de definición de datos (DDL) . . . . .	10
4.5	Código SQL - Manipulación de datos (DML) . . . . .	14
4.6	Código SQL + Resultados: Vistas . . . . .	15
4.7	Código SQL + Resultados: Triggers . . . . .	15
4.8	Código SQL + Resultados: Funciones . . . . .	16
4.9	Código SQL + Resultados: procedimientos . . . . .	16
<b>5</b>	<b>Bases de Datos No-SQL</b>	<b>17</b>
5.1	Diagrama Bases de Datos No-SQL . . . . .	17
5.2	SMBD utilizado para la Base de Datos No-SQL . . . . .	18
<b>6</b>	<b>Aplicación de ETL (Extract, Transform, Load)</b>	<b>19</b>
6.1	Ejemplo de aplicación de ETL . . . . .	19
6.2	Ejemplo de lago de datos . . . . .	22
<b>7</b>	<b>Lecciones aprendidas</b>	<b>24</b>
<b>8</b>	<b>Bibliografía</b>	<b>25</b>

# 1 Introducción

El vino se ha empezado a posicionar como una bebida apetecida, debido a su contenido de antioxidantes como el resveratrol. Algunos expertos han indicado que tomar con moderación se considera saludable porque ayuda a prevenir enfermedades de las arterias coronarias, la afección que provoca los ataques cardíacos.

Para la evaluación sensorial del vino, se tienen unos parámetros como lo son el color, el aroma y el sabor.

El color del vino durante su proceso de observación es compleja, laboriosa, costosa y sujeta a error, debido a la subjetividad del juicio. Las condiciones de observación varían de acuerdo con la luminosidad del lugar en donde se realice la cata; si la luminosidad es baja, el color se tornará oscuro, con una mala tonalidad; por ello, es importante que el tono sea neutro.

El aroma son las percepciones olfativas que generan los vinos y el olor adquirido por los vinos envejecidos durante su proceso de fermentación.

El sabor del vino tiene tres etapas, la primera etapa es llevar el vino a la boca aparece el sabor dulce, el segundo sabor el cual es salado se percibe paulatinamente, la tercera etapa la cual es amargo y resulta al mantener por cierto tiempo la sensación ácida aumenta durante el tiempo en el que se mantenga el vino en la boca.

En los análisis de los vinos es frecuente utilizar escalas conocidas por los jueces y, con ello, presentar de forma muy sencilla la descripción sensorial del producto analizado. Estas representaciones permiten, además, comparar de forma visual las diferencias sensoriales entre vinos. Algunas veces los datos obtenidos durante la evaluación pueden ser analizados mediante análisis de varianza (ANOVA), interpretando las percepciones de los jueces frente a los vinos.

## **2 Características del proyecto de investigación que hace uso de Bases de Datos**

### **2.1 Título del proyecto de investigación**

Creación de bases de datos como repositorio para que los modelos de machine learning puedan determinar la calidad vino.

### **2.2 Objetivo general**

Construir un modelo relacional de base de datos y su implementación el cual pueda servir como insumo para predecir la calidad, dentro de un intervalo de confianza, del vino tinto y vino blanco tomando como base sus estándares de calidad.

#### **2.2.1 Objetivos específicos**

- Diseñar el modelo de entidad relación para mostrar la relación y la información de los datos.
- Crear los objetos (tablas, vistas, procedimientos almacenados, funciones, trigger) necesarios en la base de datos.
- Realizar el cargue de los registros en el motor de base de datos MSQl.

### **2.3 Alcance**

Para determinar la evaluación sensorial del vino existen varias formas. Un panel de expertos, conformado por personas de gran experiencia, que en muchos casos son enólogos reconocidos; un panel de jueces entrenados, integrados por personas capacitadas para actuar como evaluadores capaces de percibir las sensaciones por su conocimiento y algún tipo de experiencia en la evaluación sensorial, y el panel de jueces evaluadores, consumidores de vino elegidos al azar. Durante este proceso se califican través de una escala para que posteriormente sean analizados. En muchos casos los resultados de las pruebas hacen que sean subjetivos y muy costosos, perdiendo el foco principal que es dar una calificación objetiva sobre la calidad el vino. Para esto se utilizará un modelo relacional de base datos el cual represente la información recopilada que realizan los expertos de la evaluación sensorial.

### **2.4 Pregunta de investigación**

¿Cómo el modelo relacional de base de datos nos permite recopilar la información necesaria predecir la calidad del vino aplicando métodos estadísticos?

## **2.5 Hipotesis**

La implementación de un modelo relacional de base de datos, nos puede ayudar en recopilar y almenar la información suministrada para conseguir una mejor imparcialidad y ahorro de dinero en el proceso de la evaluación de la calidad del vino, esto debido a que en el planteamiento encontramos que determinar la calidad del vino se determina por diferentes personas que puedes agregar un sesgo por una inclinación hacía una marca.

## 3 Reflexiones sobre el origen de datos e información

### 3.1 ¿Cual es el origen de los datos e información

Los datos provienen de la base de datos de vinos portugueses "Vinho Verde". Fueron recopilados por un equipo de investigadores de la Universidade do Minho en Portugal, liderados por Paulo Cortez y António Cerdeira. Estos datos fueron hechos públicos por primera vez en un artículo de investigación titulado "Modeling wine preferences by data mining from physicochemical properties" (en español: "Modelado de las preferencias de vinos mediante minería de datos de propiedades fisicoquímicas"), publicado en la revista científica Decision Support Systems en 2009.

Los datos contienen información sobre varios atributos fisicoquímicos (como pH, acidez, contenido de alcohol, etc.) y sensoriales (como la calidad del vino) de vinos tintos y blancos de la región de Vinho Verde en Portugal. Estos atributos se utilizaron para predecir la calidad del vino en una escala del 0 al 10, que es la variable objetivo de este conjunto de datos. Los datos fueron recopilados en 2004 y 2005.

Se tiene en consideración, se toma como base los datos que nosotros tomamos como ejemplo para el presente proyecto para el desarrollo y para complementar el modelo se adicional otras tablas, con el fin de crear un modelo de entidad relación. A continuación, algunos antecedentes de modelos predictivos de calidad del vino:

- Análisis sensorial
- Análisis quumico.
- Modelos multivariantes.
- Fuente de datos "kaggle(<https://www.kaggle.com/datasets/hufe09/winequality>)".

### 3.2 ¿Cuales son las consideraciones legales o eticas del uso de la información?

El uso de cualquier conjunto de datos debe ser cuidadosamente evaluado para asegurar el cumplimiento de las leyes y regulaciones aplicables y para respetar los derechos de privacidad y confidencialidad de las personas involucradas. Teniendo en cuenta lo siguiente:

**1. Anonimización de los datos:** La anonimización de los datos es un aspecto clave para garantizar la privacidad de los individuos involucrados. En el caso de los datos del conjunto de Calidad del Vino, se desconoce si se han tomado medidas adecuadas para anonimizar los datos y prevenir la identificación de individuos específicos.

**2. Uso legítimo de los datos:** El uso de los datos del conjunto debe ser consistente con el propósito original para el que fueron recopilados. En este caso, el propósito original fue investigar cómo los atributos fisicoquímicos y sensoriales de los vinos afectan a su calidad. Cualquier uso de los datos para fines diferentes a este debe ser cuidadosamente evaluado para asegurar que sea ético y legal.

### **3.3 ¿Cuales son los retos de la información y los datos que utilizara en la base de datos en terminos de la calidad y la consolidación?**

Realizar una revisión de los datos contra el tipo de dato para así poder realizar los ajustes que sean necesarios. Mantener. Velar por respetar los derechos de privacidad y confidencialidad de las personas involucradas y así mismo como el uso legítimo de los datos.

### **3.4 ¿Que espera de la utilización de un sistema de Bases de Datos para su proyecto?**

Afianzar los conocimientos adquiridos durante la clase sobre base de datos relaciones y no relacionales, como lo son la creación, modificación y borrado de objetos, modelo entidad relación, consultas hacia la base de datos para seleccionar, actualizar, insertar y borrar registros.

Entender y colocar en práctica el concepto de ETL, bodega de datos.

## 4 Diseño del Modelo de Datos del SMBD (Sistema Manejador de Bases de Datos)

### 4.1 Características del SMBD (Sistema Manejador de Bases de Datos) para el proyecto

MySQL es un sistema de administración de bases de datos relacionales. Es un software de código abierto desarrollado por Oracle. Se considera como la base de datos de código abierto más utilizada en el mundo.

Las principales características son:

- 1. Código abierto:** MySQL utiliza la Licencia Pública General de GNU, por lo que se puede descargar, utilizar y modificar a voluntad. Esto facilita su uso tanto académico como profesional.
- 2. Uso multiplataforma:** Una de sus características principales y de mayor ventaja es que puede instalarse en entornos con sistemas operativos diversos como Windows, Mac y la mayoría de distribuciones Linux, así como en ambientes Unix.
- 3. Escalabilidad:** Tiene soporte para 40-50 millones de registros, 150.000-200.000 tablas y 5000 millones de filas.
- 4. Tipos de datos:** Soporta una amplia gama de tipos de datos, lo que permite tener una gran versatilidad en cuanto a las situaciones, industrias o casos de uso donde puede implementarse una base de datos MySQL. Puede emplearse para la industria financiera, al manejar datos con mucha precisión; por otro lado, también puede utilizarse en ámbitos de geolocalización por sus datos de tipo espacial. De igual forma puede competir, en ciertas situaciones, con las bases de datos no relacionales con su tipo.
- 5. Conjuntos de caracteres:** Es compatible con un gran listado de conjuntos de caracteres e idiomas, lo que le permite adaptarse a cualquier parte del mundo. Sin duda alguna, es un aspecto que le ha ayudado a posicionarse en los sistemas de internet a lo largo y ancho del planeta.
- 6. Clientes gráficos:** Si bien MySQL utiliza su propio lenguaje para administrar los datos almacenados, existen diversas herramientas o clientes gráficos que nos permiten interactuar con las bases de datos, ayudando a que dicha interacción sea más sencilla y, por lo tanto, más rápida.
- 7. Soporte para lenguajes de programación:** Las características y ventajas de MySQL son muchas, pero sin duda todas ellas son mejor explotadas cuando están integradas dentro de un sistema de información. Para ello existe un amplio abanico de API nativas, librerías, paquetes, etc. que permiten integrar

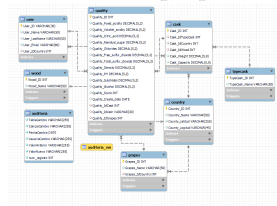


una base de datos MySQL en un sistema desarrollado en cualquier lenguaje de programación.

**8. Documentación actualizada:** Al ser muy popular y utilizado, permite que exista una documentación oficial muy amplia, además de una comunidad enorme siempre dispuesta a ayudar, colaborar y aportar al conocimiento compartido

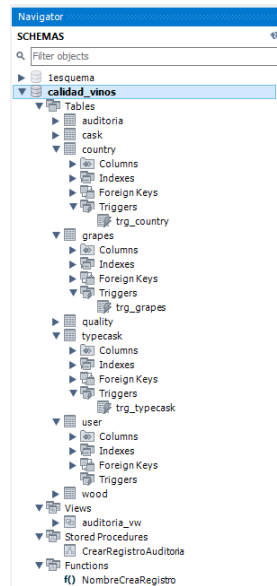
## 4.2 Diagrama modelo de datos

Figure 1: Modelo de datos propuesto para el trabajo



## 4.3 Imágenes de la Base de Datos

Figure 2: Base de datos donde se muestran los objetos creados



#### 4.4 Código SQL - lenguaje de definición de datos (DDL)

```
CREATE SCHEMA calidad_vinos;  
USE calidad_vinos;
```

```
CREATE TABLE Auditoria (  
    TablaCambio VARCHAR(255) NOT NULL,  
    CampoCambio VARCHAR(255) NOT NULL,  
    FechaCambio DATE NOT NULL,  
    UsuarioCambio VARCHAR(255) NOT NULL,  
    ValorAnterior VARCHAR(10) NOT NULL,  
    ValorNuevo VARCHAR(10) NOT NULL,  
    num_registro INT NOT NULL  
);
```

```
CREATE TABLE country (  
    Country_ID INT NOT NULL AUTO INCREMENT,  
    Country_Name varchar(50) DEFAULT NULL,  
    Contry_Longitud varchar(255) DEFAULT NULL,  
    Contry_Latitud varchar(255) DEFAULT NULL,  
    PRIMARY KEY (Country_ID)  
);
```

```
CREATE TABLE typecask (  
    TypeCask_ID INT NOT NULL AUTO INCREMENT,  
    TypeCask_Name varchar(50) DEFAULT NULL,  
    PRIMARY KEY (TypeCask_ID)  
);
```

```
CREATE TABLE wood (  
    Wood_ID INT NOT NULL AUTO INCREMENT,  
    Wood_Name varchar(50) DEFAULT NULL,  
    PRIMARY KEY (Wood_ID)  
);
```

```
CREATE TABLE grapes (  
    Grapes_ID INT NOT NULL AUTO INCREMENT,  
    Grapes_Name varchar(50) DEFAULT NULL,  
    Grapes_IdCountry int DEFAULT NULL,  
    PRIMARY KEY (Grapes_ID),  
    KEY Grapes_IdCountry (Grapes_IdCountry),  
    CONSTRAINT grapes_country_fk FOREIGN KEY  
    (grapes_IdCountry) REFERENCES country (Country_ID)  
);
```

```
CREATE TABLE user (  

```

```

User_Id varchar(30) NOT NULL,
User_Name varchar(50) DEFAULT NULL,
User_LastName varchar(50) DEFAULT NULL,
User_Email varchar(80) DEFAULT NULL,
User_IdCountry int DEFAULT NULL,
PRIMARY KEY (User_Id),
KEY User_IdCountry (User_IdCountry),
CONSTRAINT user_country_fk FOREIGN KEY (User_IdCountry)
REFERENCES country (Country_Id)
);

```

```

CREATE TABLE cask (
Cask_Id int NOT NULL AUTO_INCREMENT,
Cask_IdTypeCask int DEFAULT NULL,
Cask_IdCountry int DEFAULT NULL,
Cask_IdWood int DEFAULT NULL,
Cask_Weight decimal(5,0) DEFAULT NULL,
Cask_Capacity decimal(5,0) DEFAULT NULL,
PRIMARY KEY (Cask_Id),
KEY Cask_IdCountry (Cask_IdCountry),
KEY Cask_IdTypeCask (Cask_IdTypeCask),
KEY Cask_IdWood (Cask_IdWood),
CONSTRAINT cask_country_fk FOREIGN KEY (Cask_IdCountry)
REFERENCES country (Country_Id),
CONSTRAINT cask_type_cask_fk FOREIGN KEY
(Cask_IdTypeCask) REFERENCES type_cask (TypeCask_Id),
CONSTRAINT cask_wood_fk FOREIGN KEY (Cask_IdWood)
REFERENCES wood (Wood_Id)
);

```

```

CREATE TABLE quality (
Quality_Id int NOT NULL AUTO_INCREMENT,
Quality_Fixed_acidity decimal(5,2) DEFAULT NULL,
Quality_Volatile_acidity decimal(5,2) DEFAULT NULL,
Quality_Citric_acid decimal(5,2) DEFAULT NULL,
Quality_Residual_sugar decimal(5,2) DEFAULT NULL,
Quality_Chlorides decimal(5,2) DEFAULT NULL,
Quality_Free_sulfur_dioxide decimal(5,2) DEFAULT NULL,
Quality_Total_sulfur_dioxide decimal(5,2) DEFAULT NULL,
Quality_Density decimal(5,2) DEFAULT NULL,
Quality_Ph decimal(5,2) DEFAULT NULL,
Quality_Sulphates decimal(5,2) DEFAULT NULL,
Quality_Alcohol decimal(5,2) DEFAULT NULL,
Quality_Score int DEFAULT NULL,
Quality_Create_date date DEFAULT NULL,
Quality_IdCask int DEFAULT NULL,

```

```

Quality_IDUser varchar(30) DEFAULT NULL,
Quality_IDGrapes int DEFAULT NULL,
PRIMARY KEY (Quality_ID),
KEY Quality_IDCask (Quality_IDCask),
KEY Quality_IDUser (Quality_IDUser),
KEY Quality_IDGrapes (Quality_IDGrapes),
CONSTRAINT quality_cask_fk FOREIGN KEY (Quality_IDCask)
REFERENCES cask (Cask_ID),
CONSTRAINT quality_user_fk FOREIGN KEY (Quality_IDUser)
REFERENCES user (User_ID),
CONSTRAINT quality_grapes_fk FOREIGN KEY
(Quality_IDGrapes) REFERENCES grapes (Grapes_ID)
);

```

```

CREATE PROCEDURE CrearRegistroAuditoria(IN nombre_tabla VARCHAR(255),
IN campo_tabla VARCHAR(255),
IN usuario_cambio VARCHAR(255),
IN valor_anterior VARCHAR(255),
IN valor_nuevo VARCHAR(255),
IN num_registro INT)
BEGIN
INSERT INTO Auditoria (TablaCambio,
CampoCambio,
FechaCambio,
UsuarioCambio,
ValorAnterior,
ValorNuevo,
num_registro)
VALUES (nombre_tabla,
campo_tabla,
SYSDATE(),
usuario_cambio,
valor_anterior,
valor_nuevo,
num_registro);
END

```

```

CREATE TRIGGER trg_country
BEFORE UPDATE ON country
FOR EACH ROW
BEGIN
IF NEW.CountryName <> OLD.CountryName THEN
CALL CrearRegistroAuditoria('Country',
'Name', user(), OLD.CountryName,
NEW.CountryName, OLD.Country_ID);
ENDIF;

```

*END*

```
CREATE TRIGGER trg_typecask
BEFOREUPDATEONtypecask
FOREACHROW
BEGIN
IFNEW.TypeCask_Name <> OLD.TypeCask_NameTHEN
CALLCrearRegistroAuditoria('TypeCask',' Name',user(),OLD.TypeCask_Name,
NEW.TypeCask_Name,OLD.TypeCask_ID);
ENDIF;
END
```

```
CREATE TRIGGER trg_grapes
BEFOREUPDATEONgrapes
FOREACHROW
BEGIN
IFNEW.Grapes_Name <> OLD.Grapes_NameTHEN
CALLCrearRegistroAuditoria('TypeCask',' Name',user(),OLD.Grapes_Name,
NEW.Grapes_Name,OLD.Grapes_ID);
ENDIF;
END
```

```
CREATE TRIGGER trg_wood
BEFOREUPDATEONwood
FOREACHROW
BEGIN
IFNEW.Wood_Name <> OLD.Wood_NameTHEN
CALLCrearRegistroAuditoria('TypeCask',' Name',user(),OLD.Wood_Name,
NEW.Wood_Name,OLD.Wood_ID);
ENDIF;
END
```

```
DELIMITER
CREATE TRIGGER trg_quality
BEFOREUPDATEONquality
FOREACHROW
BEGIN
IFNEW.Quality_Fixed_acidity <> OLD.Quality_Fixed_acidityTHEN
CALLCrearRegistroAuditoria('quality',' Quality_Fixed_acidity',user()),
CONVERT(NEW.Quality_Fixed_acidity,CHAR),
CONVERT(OLD.Quality_Fixed_acidity,CHAR),OLD.Quality_ID);
ENDIF;
```

```
IF NEW.Quality_Volatile_acidity <> OLD.Quality_Fixed_acidityTHEN
CALLCrearRegistroAuditoria('quality',' Quality_Volatile_acidity',user(),
CONVERT(NEW.Quality_Volatile_acidity,CHAR),
```

```

CONVERT(OLD.QualityVolatileacidity, CHAR), OLD.QualityID);
ENDIF;

```

```

IF NEW.QualityCitricacid <> OLD.QualityCitricacid THEN
CALLCrearRegistroAuditoria('quality', 'QualityCitricacid', user()),
CONVERT(NEW.QualityCitricacid, CHAR),
CONVERT(OLD.QualityCitricacid, CHAR), OLD.QualityID);
ENDIF;

```

```

IF NEW.QualityResidualsugar <> OLD.QualityResidualsugar THEN
CALLCrearRegistroAuditoria('quality', 'QualityCitricacid', user()),
CONVERT(NEW.QualityResidualsugar, CHAR),
CONVERT(OLD.QualityResidualsugar, CHAR), OLD.QualityID);
ENDIF;
END

```

```

CREATE FUNCTION NombreCreaRegistro(registro INT,
nombreTabla VARCHAR(255))
RETURNS VARCHAR(255)
DETERMINISTIC
BEGIN
DECLARE nombreCrea VARCHAR(255);
IF nombreTabla = 'quality' THEN
SELECT Qualitycreatedate
INTOnombreCrea
FROMquality
WHEREQualityID = registro;
ELSE
SETnombreCrea = CONCAT('Sinnombre, perosecreequeeselusuario', user());
ENDIF;

RETURN nombreCrea;
END

```

```

CREATE VIEW auditoriawAS
SELECTTablaCambio, CampoCambio, FechaCambio, UsuarioCambio,
ValorAnterior, ValorNuevo, numrregistro,
NombreCreaRegistro(numrregistro, TablaCambio)CreadorRegistro
FROMauditoria;

```

#### 4.5 Código SQL - Manipulación de datos (DML)

```

INSERT INTO country VALUES (1,'España', '-4.0000000', '40.0000000'), (2,'Portugal', '-8.0000000', '39.5000000')
INSERT INTO typecask VALUES (1,'Barril Rehabilitado'),
(2,'Barril Americano'),

```

```

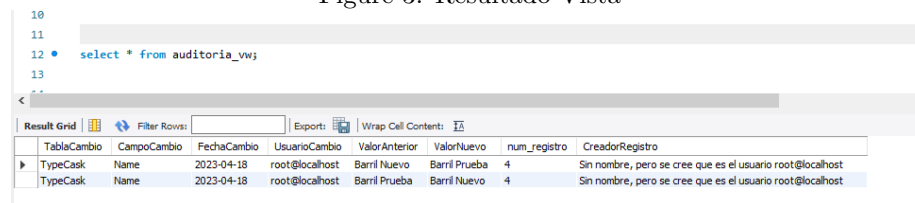
(3,'Barril Francés'),
(4,'Barril Nuevo');
INSERT INTO wood VALUES (1,'Roble francés'),
(2,'Roble americano'),
(3,'Cerezo'),
(4,'Acacia');
INSERT INTO grapes VALUES (1,'Suavinhon blanc',2),
(2,'Merlot',1),
(3,'pinot noir',4);
INSERT INTO user VALUES ('1234545','Johan','Peñaloza','jpéñalozal@ucentral.edu.co',3),
('13412342','luis','Mosquera','lmosquerad@ucentral.edu.co',3),
('154345345','Javier','Garcia','jgarciaa15@ucentral.edu.co',3);
INSERT INTO cask VALUES (1,1,3,1,90,200),
(2,2,4,2,120,300);
INSERT INTO quality VALUES (3,7.1,1.7,0.6,8,0,23,40,2,6,6,3,1,'2023-04-11',1,'1234545',1);
INSERT INTO quality VALUES (4,8.2,1.8,0.9,9,0,24,32,2,8,6,4,9,'2023-04-11',1,'154345345',2);
INSERT INTO quality VALUES (5,9.3,2.6,0.3,6,0,29,60,2,5,6,8,3,'2023-04-11',2,'154345345',3);
INSERT INTO quality VALUES (6,1.4,3.5,0.4,7,0,26,50,2,3,6,7,6,'2023-04-11',1,'1234545',1);
INSERT INTO quality VALUES (7,2.5,1.8,0.7,9,0,25,38,2,2,6,9,9,'2023-04-11',2,'1234545',2);

```

## 4.6 Código SQL + Resultados: Vistas

Se adjunta imagen donde se consulta la vista de auditora, la cual es una tabla que almacena info cada que se actualiza un campo de una tabla. Esta actualización es ejecutada por un trigger el cual llama a un procedimiento que realiza la inserción.

Figure 3: Resultado Vista



	TablaCambio	CampoCambio	FechaCambio	UsuarioCambio	ValorAnterior	ValorNuevo	num_registro	CreadorRegistro
TypeCask	Name		2023-04-18	root@localhost	Barril Nuevo	Barril Prueba	4	Sin nombre, pero se cree que es el usuario root@localhost
TypeCask	Name		2023-04-18	root@localhost	Barril Prueba	Barril Nuevo	4	Sin nombre, pero se cree que es el usuario root@localhost

## 4.7 Código SQL + Resultados: Triggers

Los triggers creados para este ejercicio son before update es decir que antes actualizar la tabla primero inserta en la tabla de a auditoria. La imagen corresponde a un proceso de actualización que luego se inserta en la tabla de

auditoria. Se adjunta imagen

Figure 4: Resultado triggers

```
11 • UPDATE typecask SET typecask_Name = 'Barril Nuevo'
12   WHERE typecask_ID = 4;
13
14 • select * from Auditoria;
```

TablaCambio	CampoCambio	FechaCambio	UsuarioCambio	ValorAnterior	ValorNuevo	num_registro
TypeCask	Name	2023-04-18	root@localhost	Barril Nuevo	Barril Prueba	4
TypeCask	Name	2023-04-18	root@localhost	Barril Prueba	Barril Nuevo	4

## 4.8 Código SQL + Resultados: Funciones

La funcion creada devuelve el nombre de un usuario que realiza un cambio. Si no lo encuentra entonces muestra el nombre del usuario que esta logueado en la base de datos. Se adjunta imagen

Figure 5: Resultado Funcion

```
9
10 • select NombreCreaRegistro(4,'TypeCask') from dual;
```

NombreCreaRegistro(4,'TypeCask')
Sin nombre, pero se cree que es el usuario root@localhost

## 4.9 Código SQL + Resultados: procedimientos

Este procedimiento es ejecutado desde los triggers e insertan los datos necesarios para crear la auditoria. Se adjunta imagen



Figure 6: Resultado Vista



## 5 Bases de Datos No-SQL

### 5.1 Diagrama Bases de Datos No-SQL

Para esta sección se adicionan se crea una base de datos y 3 colecciones. Las colecciones creadas son:

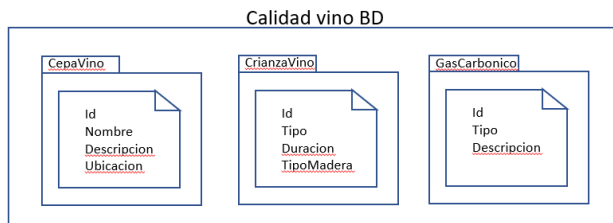
**Cepa:** La cepa de vino se refiere al tronco de la vid o dicho de manera más sencilla, al tipo de uva con la cual se fabrica el vino.

**Gas carbónico:** El gas carbónico es la cantidad de burbujas que tiene una botella. Es importante destacar que los vinos espumosos no forman parte de esta categoría, debido a la cantidad excesiva de gas carbónico.

**Crianza :** Esta categoría se determina por la crianza en barrica o botella.

A continuación se muestra un modelado orientado a documentos:

Figure 7: Modelado de base datos mongo



## 5.2 SMBD utilizado para la Base de Datos No-SQL

La base de datos NoSql a utilizar es MongoDB (del inglés humongous, "enorme"). El cual es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico. Al ser un proyecto de código abierto, sus binarios están disponibles para los sistemas operativos Windows, GNU/Linux, OS X y Solaris.

### Características de MongoDB

**Consultas ad hoc:** Con MongoDb podemos realizar todo tipo de consultas. Podemos hacer búsqueda por campos, consultas de rangos y expresiones regulares. Además, estas consultas pueden devolver un campo específico del documento, pero también puede ser una función JavaScript definida por el usuario.

**Indexación:** El concepto de índices en MongoDB es similar al empleado en bases de datos relacionales, con la diferencia de que cualquier campo documentado puede ser indexado y añadir múltiples índices secundarios.

**Replicación:** Del mismo modo, la replicación es un proceso básico en la gestión de bases de datos. MongoDB soporta el tipo de replicación primario-secundario. De este modo, mientras podemos realizar consultas con el primario, el secundario actúa como réplica de datos en solo lectura a modo copia de seguridad con la particularidad de que los nodos secundarios tienen la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.

**Balanceo de carga:** Resulta muy interesante cómo MongoDB puede escalar la carga de trabajo. MongoDB tiene la capacidad de ejecutarse de manera simultánea en múltiples servidores, ofreciendo un balanceo de carga o servicio de replicación de datos, de modo que podemos mantener el sistema funcionando en caso de un fallo del hardware.

**Almacenamiento de archivos:** Aprovechando la capacidad de MongoDB para el balanceo de carga y la replicación de datos, Mongo puede ser utilizado también como un sistema de archivos. Esta funcionalidad, llamada GridFS e incluida en la distribución oficial, permite manipular archivos y contenido.

**Ejecución de JavaScript del lado del servidor:** MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

## 6 Aplicación de ETL (Extract, Transform, Load)

### 6.1 Ejemplo de aplicación de ETL

Para esta sección lo que se realizará sera extraer y unificar la información creada en las tablas de mysql y convertirlo en un objeto json para ser insertado en un colección de una base de datos de mongo. La colección se llamara calidad y el json tendrá la siguientes estructura:

```
{
  "cask": {
    "caskId": 1,
    "castCapacity": 200.0,
    "castWeight": 90.0,
    "country": {
      "countryId": 3,
      "countryName": "Colombia"
    },
    "typecask": {
      "typecastId": 1,
      "typecastName": "Barril Rehabilitado"
    },
    "wood": {
      "woodId": 1,
      "woodName": "Roble franc  es"
    }
  },
  "grape": {
    "country": {
      "countryId": 2,
      "countryName": "Portugal"
    },
    "grapeId": 1,
    "grapeName": "Suavinhon blanc"
  },
  "qualityAlcohol": 9.0,
  "qualityChlorides": 0.0,
  "qualityCitricAcid": 0.8,
  "qualityDensity": 1.0,
  "qualityFixedAcidity": 7.5,
  "qualityFreeSulfurDioxide": 11.3,
  "qualityID": 1,
  "qualityPH": 4.0,
  "qualityResidualSugar": 2.0,
  "qualityScore": 5,
  "qualitySulphates": 1.0,
```

```

"qualityTotalSulfurDioxide": 34.0,
"qualityVolatileAcidity": 1.8,
"user": {
  "country": {
    "countryId": 2,
    "countryName": "Portugal"
  },
  "userEmail": "lmosquerad@ucentral.edu.co",
  "userId": "13412342",
  "userLastName": "Mosquera",
  "userName": "luis"
}
}

```

Para la ejecución de este ETL, se crean unas clases en python así: **conexionbd.py**  
 : Clase que se encarga de las conexiones y todos los procesos de base de datos de de mongo y mysql.

Figure 8: Clase conexion

```

1  import pymysql
2  from pymongo.mongo_client import MongoClient
3  from pymongo.server_api import ServerApi
4  import bson.json_util
5
6  1 usage
7  > def crearconexionmysql():...
8
9  1 usage
10 > def crearconexionmongo():
11
12  16 >     try:...
13
14  21     except Exception as e:
15
16  22         print("Error al conectarse con mongo: ", e)
17
18  1 usage
19  23 > def selectquality():...
20
21  1 usage
22  41 > def selectCask(caskid):...
23
24  1 usage
25  49 > def selectuser(userid):...
26
27  1 usage
28  54 > def selectgrapes(grapesid):...
29
30  1 usage
31  > def fabricaconsulta(tipo, valor):...
32
33  2 usages
34
35  70 > class ConexionBD:...
36
37  89

```

**etl.py** : Clase que se del proceso de etl.

Figure 9: Clase ETL

```
1  import conexionbd
2  import quality
3
4  1 usage
5  class Etl:
6
7      1 usage
8      def extraer(self):
9          consult = conexionbd.ConexionBD()
10         registers = consult.crearconsulta('quality')
11         return registers, consult
12
13     1 usage
14     def tranformar(self, registers, consult):
15         listaRegistros = []
16         for registro in registers:...
17         return listaRegistros
18
19     1 usage
20     def cargar(self, lista):
21         consult = conexionbd.ConexionBD()
22         consult.cargarmongo(lista)
```

**qualiy.py** : Es la clase resultande que luego se convierte a json para ser cargado a mongo.

**main.py** : Clase principal que ejecuta la aplicacación.

Asi mismo se adjunta una imagen de la base de datos en mongo donde está la coleccion calidad la cual es donde se almacena el json anteriormente nombrado.

Figure 10: Clase quality

```
import json

9 usages
class Quality:
    1 usage
    def tojson(self):
        return json.dumps(self, default=lambda o: o.__dict__,
                           sort_keys=True, indent=4)
```

Figure 11: main

```
import etl

if __name__ == '__main__':
    try:
        etlprocessor = etl.Etl()
        # Extraer
        registers, consult = etlprocessor.extraer()
        # Transformar
        resultantly = etlprocessor.tranformar(registers, consult)
        # Cargar
        etlprocessor.cargar(resultantly)
    except Exception as e:
        print("Error al ejecutar la transaccion: ", e)
```

Figure 12: Clase quality

The screenshot shows the MongoDB Compass interface. On the left, a sidebar lists databases and collections. The 'CalidadVino' database is selected, and the 'Calidad' collection is highlighted. The main panel displays the 'CalidadVino.Calidad' collection with 7 documents. A query is entered: 'Type a query: { field: "value" }'. The query results show 7 of 7 documents. The first document is highlighted with a red box and contains the following JSON structure:

```
{
  "_id": "ObjectID('6467f8aba799ba958cac3647')",
  "cask": "Object",
  "grape": "Object",
  "qualityAlcohol": 9,
  "qualityColorIdeas": 9,
  "qualityCttrickIdeas": 8.8,
  "qualityDensity": 1,
  "qualityFresculFurDioxide": 7.5,
  "qualityFresculFurDioxideIdeas": 11.3,
  "qualityID": 1,
  "qualityPH": 4,
  "qualityResidualSugar": 2,
  "qualityScore": 5,
  "qualitySulphates": 1,
  "qualityTotalSulFurDioxide": 34,
  "qualityVolatleAcidity": 1.9,
  "user": "Object"
}
```

## 6.2 Ejemplo de lago de datos

En el siguiente ejemplo se crea un lago de datos sencillo en python. Lo se hace es crear 3 data set, realizar una transformación leve a la información y por último

crear un archivo de salida.

Figure 13: Lago de datos

```
[12] import pandas as pd
dfr1 = pd.read_csv('result.csv', sep=',', encoding='latin-1')
dfr2 = pd.read_csv('result1.csv', sep=',', encoding='latin-1')
dfr3 = pd.read_csv('result2.csv', sep=',', encoding='latin-1')
dfr1.head()
```

	Id	cask	castCapacity	castWeight	country	countryName	typecask	typecastName	wood	woodName
0	6467f8abaf99ba058cac3647	1	200	90	3	Colombia	1	Barril Rehabilitado	1	Roble francés

1 rows x 33 columns

```
[7] dff = pd.concat([dfr1, dfr2, dfr3])
dff.head()
```

	Id	cask	castCapacity	castWeight	country	countryName	typecask	typecastName	wood	woodName
0	6467f8abaf99ba058cac3647	1	200.0	90.0	3	Colombia	1	Barril Rehabilitado	1	Roble francés
1	6467f8abaf99ba058cac3648	1	200.0	90.0	3	Colombia	1	Barril Rehabilitado	1	Roble francés
2	6467f8abaf99ba058cac3649	1	200.0	90.0	3	Colombia	1	Barril Rehabilitado	1	Roble francés

3 rows x 33 columns

```
dff.to_csv('lago_de_datos.csv', index=False)
```

## 7 Lecciones aprendidas

**Planificación adecuada:** Antes de comenzar cualquier proyecto de gestión de datos, es esencial realizar una planificación adecuada. Definir los objetivos, los requisitos y los plazos del proyecto. Y que esto ayudará a evitar problemas y retrasos en etapas posteriores.

**Diseño flexible y escalable:** Siempre se debe asegurar de que el modelo entidad-relación que se plantea flexible y escalable; es decir diseñar una estructura que pueda adaptarse fácilmente a modificaciones sin afectar negativamente la funcionalidad existente.

**Validación y limpieza de datos:** Antes de cargar los datos en el sistema, se debe realiza una exhaustiva validación y limpieza de los mismos. Identificar y corregir los errores, elimina los datos duplicados y estandariza el formato de los datos en la medida de lo posible. Esto garantizará la integridad y la calidad de los datos almacenados.

**Documentación:** Se debe Documentar todo el proceso de desarrollo y configuración del sistema. Esto incluye descripciones detalladas de las tablas, relaciones, procedimientos almacenados, triggers, funciones, vistas, ETL y lago de datos. Una documentación adecuada facilitará el mantenimiento y la comprensión del sistema en el futuro.

**Pruebas:** Se debe muy riguroso con las pruebas en cada etapa del proyecto. Verificar y validar la integridad de los datos cargados. Comprobar el funcionamiento correcto de los procedimientos almacenados, triggers, funciones, vistas y demas objetos creados en la base de datos. Además, en lo posible realizar pruebas de rendimiento para garantizar que el sistema pueda manejar grandes volúmenes de datos sin problemas.



## 8 Bibliografía

Eficiencia Operativa De MongoDB Atlas. (s/f). MongoDB. Recuperado el 12 de mayo de 2023, de <https://www.mongodb.com/es/cloud/atlas/efficiency>

Guía sobre tipos de vino: características y variaciones. (2021, agosto 12). Aprende Institute. <https://aprende.com/blog/gastronomia/vinos/tipos-de-vino/>

Robledano, A. (2019, octubre 28). Qué es MongoDB. Openwebinars.net. <https://openwebinars.net/blog/que-es-mongodb/>

(S/f). Edu.co. Recuperado el 12 de mayo de 2023, de <https://repository.udistrital.edu.co/bitstream/handle/11349/2742/PovedaGalvisJuanPablo2015.pdf;jsessionid=F124FF475BAED58AC6E9F23551B689A9?sequence=1>

Robledano, A. (2019a, septiembre 24). Qué es MySQL: Características y ventajas. Openwebinars.net. <https://openwebinars.net/blog/que-es-mysql/>