

Estimación de indicadores de calidad del suelo mediante modelos de regresión de machine learning a partir de análisis de imágenes multiespectrales de suelos en Cundinamarca

Maria Alejandra Enriquez Serrano¹

¹Maestría en Analítica de Datos
Universidad Central
Curso de Bases de Datos
Bogotá, Colombia
¹menriquezs@ucentral.edu.co

November 17, 2023

Contents

1	Introducción	3
2	Características del proyecto de investigación que hace uso de Bases de Datos	4
2.1	Titulo del proyecto de investigación	4
2.2	Objetivo general	4
2.2.1	Objetivos específicos	4
2.3	Alcance	4
2.4	Pregunta de investigación	4
2.5	Hipótesis	5
3	Reflexiones sobre el origen de datos e información	6
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 términos de la calidad y la consolidación?	7
3.4	¿Que espera de la utilización de un sistema de Bases de Datos para su proyecto?	8

4 Diseño del Modelo de Datos del SMBD (Sistema Manejador de Bases de Datos)	9
4.1 Características del SMBD (Sistema Manejador de Bases de Datos) para el proyecto	9
4.2 Diagrama modelo de datos	10
4.3 Imágenes de la Base de Datos	11
4.4 Código SQL - lenguaje de definición de datos (DDL)	12
4.5 Código SQL - Manipulación de datos (DML)	15
4.6 Código SQL + Resultados: Vistas	15
4.7 Código SQL + Resultados: Triggers	18
4.8 Código SQL + Resultados: Funciones	19
4.9 Código SQL + Resultados: procedimientos almacenados	23
5 Bases de Datos No-SQL	27
5.1 Diagrama Bases de Datos No-SQL	27
5.1.1 Meta-modelo conceptual	27
5.1.2 Meta - Modelo Lógico	28
5.1.3 Meta - Modelo Físico	29
5.2 SMBD utilizado para la Base de Datos No-SQL	30
6 Bibliografía	38
References	38

1 Introducción

El crecimiento acelerado de la población a nivel global ha impulsado un aumento considerable en la demanda de alimentos. Este fenómeno plantea un desafío significativo para la agricultura, que se ve obligada a adoptar prácticas intensivas para satisfacer estas necesidades, resultando en impactos negativos tanto para el medio ambiente como para la calidad del suelo. Ante esta situación, la comunidad investigadora busca alternativas que promuevan la sostenibilidad ambiental, optimicen la salud de los suelos y maximicen la productividad agrícola (Wang et al., 2017; Geisseler & Scow, 2014) y (FAO, fund for agricultural development, Unicef, food programme, & health organization, 2019).

En este contexto, surge la necesidad de brindar a los agricultores métodos eficaces y de bajo coste para evaluar la calidad del suelo. Nuestra propuesta aborda esta problemática mediante la integración de tecnologías avanzadas y enfoques tradicionales (Jung et al., 2021). En el caso específico de Colombia, es común que los agricultores evalúen la calidad del suelo antes de cada ciclo de cultivo. Sin embargo, el método tradicional empleado es lento, destructivo, implica un análisis de laboratorio costoso y produce resultados tardíos, lo que impacta negativamente en la rentabilidad (Thompson & Puntel, 2020) y (Jang et al., 2020).

En varios estudios anteriores (Paul, Coops, Johnson, Krzic, & Smukler, 2019), (A. A., Mukhtar, Rasib, Suhandri, & MOHD BUKARI, 2020), (Guan et al., 2019) y (Khanal, Fulton, Klopfenstein, Douridas, & Shearer, 2018), se ha llevado a cabo la caracterización de suelos utilizando imágenes multiespectrales. Los resultados obtenidos en estos estudios han demostrado la notable eficacia de los modelos basados en aprendizaje automático para la estimación de diversas propiedades del suelo, tales como la materia orgánica, el pH, los niveles de nitrógeno, potasio y calcio, entre otras variables de interés relacionadas con la calidad del suelo.

En consecuencia, en el presente trabajo se propone un enfoque que emplea modelos de machine learning para determinar los indicadores de calidad del suelo. Este enfoque se basa en el análisis de imágenes multiespectrales de suelos sin cobertura vegetal, representando así una contribución significativa al campo de la evaluación de la calidad del suelo.

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

2.1 Título del proyecto de investigación

Estimación de indicadores de calidad del suelo mediante modelos de regresión de machine learning a partir de análisis de imágenes multiespectrales de suelos en Cundinamarca.

2.2 Objetivo general

Determinar indicadores de calidad del suelo mediante modelos de regresión de machine learning a partir de análisis de imágenes multiespectrales de suelos en Cundinamarca.

2.2.1 Objetivos específicos

- Aplicar la metodología de caracterización de sistemas agrícolas utilizando un vehículo aéreo no tripulado (UAV) y métodos de muestreo de suelos.
- Estructurar una base de datos multidimensional con imágenes multiespectrales y datos de los resultados de indicadores de calidad del suelo.
- Evaluar el impacto de las técnicas de preprocesamiento en la base de datos multidimensional de suelos agrícolas mediante modelos de machine learning de regresión.
- Evaluar el rendimiento de los algoritmos de machine learning en la estimación de indicadores de calidad del suelo.

2.3 Alcance

El alcance del proyecto estará en diseñar y desarrollar una base de datos robusta y eficiente para almacenar y gestionar datos relacionados con imágenes multiespectrales de suelos y los resultados de laboratorio de los indicadores de calidad del suelo en la región de Cundinamarca, con el propósito de respaldar la investigación basada en machine learning. Esto es fundamental para garantizar la calidad y la disponibilidad de los datos que respaldarán tus análisis y predicciones.

2.4 Pregunta de investigación

¿Cómo determinar indicadores de calidad del suelo por medio de imágenes multiespectrales y modelos de machine learning?

2.5 Hipótesis

Hipótesis nula (H0): No existe una relación significativa entre las características de las imágenes multiespectrales del suelo y los resultados de laboratorio de los indicadores de calidad del suelo en Cundinamarca.

Hipótesis alternativa (H1): Existe una relación significativa entre las características de las imágenes multiespectrales del suelo y los resultados de laboratorio de los indicadores de calidad del suelo en Cundinamarca.

3 Reflexiones sobre el origen de datos e información

3.1 ¿Cuál es el origen de los datos e información?

El origen de los datos e información es propiedad del proyecto de regalías denominado “Desarrollo participativo de una plataforma tecnológica de teledetección para la gestión sostenible de suelos en agroecosistemas del departamento de Cundinamarca”. Estos datos fueron construidos en el marco de este proyecto, que involucra el trabajo conjunto de la gobernación de Cundinamarca y la Universidad Central. Los datos provienen de análisis de laboratorio de muestras de suelo, pruebas in situ y valores de reflectancia obtenidos de imágenes multiespectrales recopilados en las zonas de cultivo del departamento de Cundinamarca.

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

El uso de información y datos en Colombia está sujeto a varias consideraciones legales y éticas que deben ser tenidas en cuenta para garantizar el cumplimiento de las normativas y el respeto a los derechos de las personas y las instituciones involucradas. Algunas de las consideraciones legales y éticas más relevantes incluyen:

- **Consentimiento y autorización:** Para la recolección y uso de datos, especialmente aquellos relacionados con muestras de suelo, pruebas in situ y datos foliares, es importante obtener el consentimiento adecuado de las personas o entidades propietarias de los terrenos o cultivos. Esto garantiza que se cumplan las normativas de privacidad y propiedad.
- **Protección de datos personales:** Colombia cuenta con una ley de protección de datos personales (Ley 1581 de 2012) (El Congreso de la República de Colombia, 2012), que establece los principios y obligaciones para el tratamiento de información personal. Si la base de datos contiene información personal de individuos, se deben cumplir las disposiciones de esta ley, incluyendo la obtención de consentimiento, la seguridad de los datos y la notificación a las autoridades en caso de violación de datos.
- **Propiedad intelectual:** Los datos recopilados a través de investigaciones y proyectos como el mencionado deben respetar los derechos de propiedad intelectual de los investigadores y las instituciones involucradas. Cualquier uso posterior de los datos debe cumplir con los acuerdos de licencia y propiedad intelectual.
- **Uso responsable:** Los datos deben utilizarse de manera responsable y ética, respetando la privacidad y la confidencialidad de la información cuando sea necesario. Además, se deben utilizar con fines legítimos, evitando cualquier uso fraudulento o perjudicial.

- **Publicación y divulgación:** Si se planea compartir o publicar los resultados de la base de datos, se deben seguir las prácticas académicas y científicas éticas, incluyendo la atribución adecuada a los autores y la transparencia en la presentación de resultados.

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?

Los retos relacionados con la calidad y consolidación de la información y los datos que se utilizaran en el proyecto al momento de determinación de indicadores de calidad del suelo mediante modelos de regresión de machine learning a partir de análisis de imágenes multiespectrales de suelos en Cundinamarca pueden ser diversos. Aquí hay algunos aspectos clave a considerar:

- Calidad de los Datos:
 - Calibración de Instrumentos: Asegurar que los instrumentos utilizados para la toma de medidas y análisis de laboratorio estén correctamente calibrados y sean precisos para evitar errores sistemáticos en los datos.
 - Muestreo Representativo: Garantizar que las muestras de suelo representen adecuadamente las condiciones reales del terreno y los diferentes tipos de suelo en Cundinamarca. Un muestreo no representativo puede conducir a resultados sesgados.
 - Control de Calidad de Laboratorio: Verificar que los procedimientos de laboratorio sean consistentes y cumplan con estándares de calidad para obtener mediciones precisas y confiables.
- Consistencia de los Datos:
 - Integración de Fuentes: Los datos provienen de diversas fuentes, como análisis de laboratorio, pruebas in situ y valores de reflectancia de imágenes multiespectrales. Asegurar que estos datos se integren de manera coherente y consistente en tu base de datos.
 - Normalización y Estandarización: Es posible que los datos tengan diferentes unidades de medida o escalas. Debes normalizar y estandarizar los datos para que sean comparables y utilicen la misma escala en tus modelos de machine learning.
 - Limpieza de Datos: Identificar y abordar valores atípicos (outliers), datos faltantes o inconsistencias en los datos. La limpieza de datos es esencial para evitar que estos problemas afecten la precisión de tus modelos.
- Complejidad de los Datos:

- Datos Multiespectrales: Las imágenes multiespectrales pueden contener una gran cantidad de información. Se debe desarrollar métodos adecuados para procesar y extraer características relevantes de estas imágenes.
- Datos Espaciales: Considera la información espacial en los modelos, ya que los suelos pueden variar significativamente de un lugar a otro. La georreferenciación de los datos es importante.
- Limpieza de Datos: Identificar y abordar valores atípicos (outliers), datos faltantes o inconsistencias en los datos. La limpieza de datos es esencial para evitar que estos problemas afecten la precisión de tus modelos.
- Actualización y Mantenimiento: Los datos pueden cambiar con el tiempo debido a factores como cambios en el uso del suelo, prácticas agrícolas y condiciones climáticas. Se debe planificar cómo manejar la actualización de los datos a lo largo del tiempo para mantener la relevancia de tus modelos.
- Privacidad y Seguridad: Si los datos contienen información sensible o personal, se debe garantizar que se cumplan las regulaciones de privacidad y que los datos se almacenen y se utilicen de manera segura.

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

La implementación de un sistema de bases de datos en mi proyecto conlleva una serie de ventajas y expectativas clave. Esto abarca desde la eficiente gestión y almacenamiento de una amplia variedad de datos, como resultados de análisis de laboratorio, mediciones en el terreno, valores de reflectancia de imágenes multiespectrales y sus metadatos correspondientes, hasta la integración de información de múltiples fuentes en un único repositorio, facilitando así la cohesión y accesibilidad de los datos. Asimismo, se espera que el sistema permita un acceso rápido a la información mediante consultas ágiles, garantizando la ejecución de análisis exploratorios, la selección de datos para modelos de machine learning y la respuesta a interrogantes específicas del proyecto. La seguridad de los datos es una prioridad, por lo que se anticipa que el sistema proporcione sólidas medidas de protección, especialmente para datos sensibles. La escalabilidad es esencial para adaptarse al crecimiento continuo de los datos a lo largo del proyecto. Además, se confía en que el sistema facilite análisis espaciales, fomente la colaboración entre miembros del equipo y partes interesadas, registre cambios a lo largo del tiempo, y sea compatible con las necesidades de los modelos de machine learning. En última instancia, se espera que el sistema contribuya a mejorar la eficiencia de la gestión de datos, facilite el análisis, promueva la seguridad de la información y estimule la colaboración en el proyecto, manteniendo su adaptabilidad a medida que evolucione.

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

Para el proyecto se ha elegido utilizar Oracle, una destacada empresa que ofrece una amplia variedad de productos y servicios relacionados con bases de datos y tecnología. Entre sus productos más reconocidos se encuentra el sistema de gestión de bases de datos Oracle Database, que comúnmente se abrevia como Oracle DB o simplemente Oracle. Oracle Database muestra una posición destacada como uno de los sistemas de gestión de bases de datos más utilizados en el ámbito empresarial a nivel mundial. Su aplicación abarca la administración y almacenamiento de datos en aplicaciones empresariales críticas. (Lakshman, 2000) A continuación, se detallan algunas de las características clave de Oracle Database que resultan relevantes para el proyecto:

- **Modelado de Datos:** Un SMBD permite definir la estructura de la base de datos mediante esquemas, tablas, relaciones y restricciones. Con Oracle Developer, los desarrolladores pueden diseñar modelos de datos visuales y definir la estructura de las tablas utilizando Oracle SQL Developer Data Modeler, simplificando así la gestión de datos. (Beresniewicz et al., 2011)
- **Lenguaje de Consulta:** Oracle Developer admite SQL (Structured Query Language) para interactuar con la base de datos. Los desarrolladores pueden escribir consultas SQL y utilizar comandos DML (Data Manipulation Language) para insertar, actualizar, eliminar y consultar datos de manera eficiente. (Beresniewicz et al., 2011)
- **Procedimientos Almacenados:** Oracle Database admite procedimientos almacenados escritos en PL/SQL (Procedural Language/Structured Query Language). Con Oracle Developer, los desarrolladores pueden crear funciones y procedimientos almacenados para encapsular la lógica empresarial en la base de datos, lo que mejora la seguridad y el rendimiento.(Beresniewicz et al., 2011)
- **Transacciones:** Los SMBD garantizan la integridad de los datos mediante el soporte de transacciones. Oracle Developer permite a los desarrolladores gestionar transacciones de manera efectiva, asegurando que las operaciones se realicen de manera atómica y consistente.(Lakshman, 2000)
- **Seguridad:** La seguridad de los datos es crítica en cualquier aplicación. Oracle Developer y Oracle Database ofrecen un sólido conjunto de herramientas y funciones de seguridad que permiten controlar el acceso a los datos y garantizar la confidencialidad e integridad de la información. (Narayanan, 2016)

- **Optimización de Consultas:** Oracle Developer incluye herramientas de optimización de consultas que ayudan a los desarrolladores a mejorar el rendimiento de las consultas SQL. Esto es esencial para aplicaciones que manejan grandes volúmenes de datos. (Beresniewicz et al., 2011)
- **Escalabilidad:** Oracle Database es conocido por su capacidad para escalar vertical y horizontalmente. Esto significa que puede manejar tanto cargas de trabajo pequeñas como empresariales a gran escala, lo que lo convierte en una elección sólida para aplicaciones en crecimiento.(Lakshman, 2000)
- **Recuperación y Copias de Seguridad:** Oracle Developer facilita la gestión de copias de seguridad y la recuperación de datos en caso de fallos. Los SGBD, incluyendo Oracle Database, implementan estrategias de recuperación para proteger los datos contra pérdidas.(Narayanan, 2016)
- **Integración:** Oracle Developer se integra bien con otras tecnologías Oracle, lo que facilita la construcción de soluciones empresariales completas que incluyen aplicaciones web, móviles y empresariales. (Beresniewicz et al., 2011)

4.2 Diagrama modelo de datos

Se diseño una base de datos para la gestión de datos sobre la sostenibilidad de suelos en agroecosistemas que registra información sobre, los municipios de interés, los agricultores, tipos de cultivos, ortomosaicos de las zonas agrícolas, división de los cultivos, Fechas de muestreo y resultados de laboratorio de muestras de suelo.

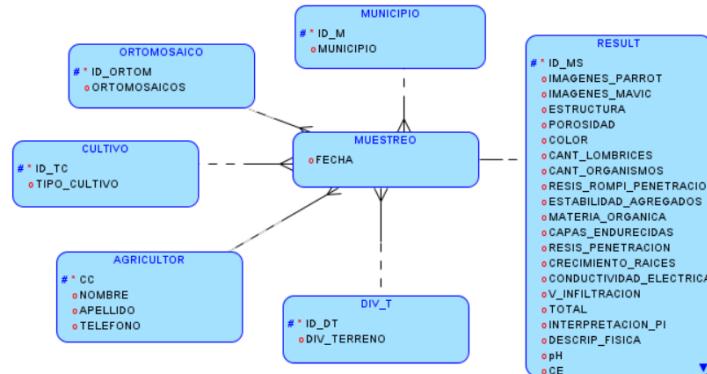


Figure 1: Diagrama ER (Entidad-Relación)

Entidades principales:

- Municipios
- Agricultor

- Cultivo
- Ortomosaico
- División del cultivo
- Eventos de muestreo
- Resultados de laboratorio

En la figura 1 se puede observar cada entidad con sus respectivos atributos y sus respectivas *llaves primarias*, también se puede ver el diagrama Entidad-Relación.

Relaciones:

Un municipio puede tener varios muestreos, Un tipo de cultivo puede tener muchos muestreos, Un agricultor tiene muchos muestreos, Cada ortomosaico puede tener múltiples muestreos, cada resultado tiene un muestreo y una división de terreno puede tener muchos muestreos

4.3 Imágenes de la Base de Datos

En la figura 2 se presentan los archivos xlsx que corresponden a cada entidad (Municipios, agricultor, tipo de cultivo, Ortomosaico, división del cultivo, eventos de muestreo, resultados de laboratorio)

AGRICULTOR	1/10/2023 9:05 p. m.	Hoja de cálculo d...	10 KB
CULTIVO	1/10/2023 9:07 p. m.	Hoja de cálculo d...	10 KB
DIV_T	1/10/2023 9:08 p. m.	Hoja de cálculo d...	9 KB
MUESTREO	1/10/2023 9:13 p. m.	Hoja de cálculo d...	14 KB
MUNICIPIO	1/10/2023 9:14 p. m.	Hoja de cálculo d...	16 KB
ORTOMOSAICO	1/10/2023 9:16 p. m.	Hoja de cálculo d...	18 KB
RESULT	1/10/2023 9:21 p. m.	Hoja de cálculo d...	54 KB

Figure 2: Bases de datos utilizada para el proyecto

En la figura 3 se visualiza el contenido de la tabla RESULT (resultados de laboratorio de las muestra de suelo).

	ID	MATERIAL	CANTIDAD	UNIDAD	PRECIO	DETALLE	CONDICION	DETALLE	PRECIO	DETALLE	CONDICION	DETALLE	PRECIO	DETALLE	CONDICION	DETALLE	PRECIO	
2	1	1	3	4	6	4		2	4	3	4	0	44	Moderada	4	0	44 Moderada	
2	2	CUserlymer CUserlymer	3	3	3	6	0	6	6	2	2	6	4	0	42	Moderada	4	
2	3	CUserlymer CUserlymer	3	2	6	4	4	3	2	2	2	6	4	0	40	Moderada	4	
2	4	CUserlymer CUserlymer	0	3	2	3	6	2	3	4	1	0	4	0	28	Mala Calida	4	
2	5	CUserlymer CUserlymer	3	6	4	4,5	6	3	6	4	1	2	4,5	4	0	48	Buena Calida	4
2	6	CUserlymer CUserlymer	3	4,5	6	3	4	3	3	1	2	6	4	0	40	Moderada	4	
2	7	CUserlymer CUserlymer	3	3	2	3	6	2	3,5	5	1	2	3	0	4	0	34,5 Moderada	4
2	8	CUserlymer CUserlymer	3	4,5	3	1,5	4,5	4	1,5	4	2	3	3	4	0	38	Moderada	4
2	9	CUserlymer CUserlymer	1	3	4	3	3	4	3,5	4	1	2	3	4	0	36,5	Moderada	4
2	10	CUserlymer CUserlymer	4,5	6	4	6	6	2	4,5	3	2	3	4,5	4	0	36,5	Moderada	4
2	11	CUserlymer CUserlymer	3	3	2	3	0	2	3	4	1	2	3	4	0	30	Moderada	4
2	12	CUserlymer CUserlymer	6	3	2	0	3	2	4,5	4	2	2	3	4	0	35,5	Moderada	4
2	13	CUserlymer CUserlymer	1	3	2	3	3	2	4,5	4	1	0	6	4	0	34	Moderada	4
2	14	CUserlymer CUserlymer	0	6	4	6	6	4	3	2	1	2	6	4	0	44	Moderada	4
2	15	CUserlymer CUserlymer	6	3	2	1,5	6	4	3	2	0	2	6	4	0	39,5	Moderada	4
2	16	CUserlymer CUserlymer	8	3	2	3	3	2	0	1	2	4	1,5	4	0	28,5	Mala Calida	4
2	17	CUserlymer CUserlymer	6	1,5	0	3	3	2	4,5	2	1	4	3	4	0	37	Moderada	4

Figure 3: Contenido de la tabla RESULT

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

Se crea las tablas de AGRICULTOR (Datos personales de los agricultores), CULTIVO (Tipos de cultivo), DIV_T (división de cultivo), como se muestra en la figura 4

```

CREATE TABLE agricultor (
    cc      INTEGER NOT NULL,
    nombre  VARCHAR2(50),
    apellido VARCHAR2(50),
    telefono INTEGER
);

ALTER TABLE agricultor ADD CONSTRAINT agricultor_pk PRIMARY KEY ( cc );

CREATE TABLE cultivo (
    id_tc      INTEGER NOT NULL,
    tipo_cultivo VARCHAR2(50)
);

ALTER TABLE cultivo ADD CONSTRAINT cultivo_pk PRIMARY KEY ( id_tc );

CREATE TABLE div_t (
    id_dt      INTEGER NOT NULL,
    div_terreno VARCHAR2(50)
);

ALTER TABLE div_t ADD CONSTRAINT div_t_pk PRIMARY KEY ( id_dt );

```

Figure 4: Creación de las tablas: Agricultor, Cultivo, Div_t

También se crea las tablas MUESTREO (Fechas de muestreo), MUNICIPIO (municipios donde se han realizado los muestreos) y ORTOMOSAICO (rutas donde se encuentran las imágenes), como se muestra en la figura 5.

```

CREATE TABLE muestreo (
    fecha DATE,
    municipio_id_m INTEGER NOT NULL,
    ortomosaico_id_ortom INTEGER NOT NULL,
    cultivo_id_tc INTEGER NOT NULL,
    agricultor_cc INTEGER NOT NULL,
    div_t_id_dt INTEGER NOT NULL,
    result_id_ms INTEGER NOT NULL
);

CREATE TABLE municipio (
    id_m INTEGER NOT NULL,
    municipio VARCHAR2(50)
);

ALTER TABLE municipio ADD CONSTRAINT municipio_pk PRIMARY KEY ( id_m );

CREATE TABLE ortomosaico (
    id_ortom INTEGER NOT NULL,
    ortomosaicos VARCHAR2(350)
);

ALTER TABLE ortomosaico ADD CONSTRAINT ortomosaico_pk PRIMARY KEY ( id_ortom );

```

Figure 5: Creación de las tablas: Muestreo, Municipio y Ortomosaico

y por ultimo se crea la tabla RESULTADOS (Resultados de laboratorio), como se muestra en la figura 6.

```

CREATE TABLE result (
    id_ms INTEGER NOT NULL,
    imagenes_parrot VARCHAR2(350),
    imagenes_mavic VARCHAR2(350),
    estructura FLOAT,
    porosidad FLOAT,
    color FLOAT,
    cant_lombrices FLOAT,
    cant_organismos FLOAT,
    resis_rompi_penetracion FLOAT,
    estabilidad_agregados FLOAT,
    materia_organica FLOAT,
    capas_endurecidas FLOAT,
    resis_penetracion FLOAT,

```

Figure 6: Creación de la tabla Resultados

En la figura 7 se puede observar como se crean las relaciones entre cada tabla.

```

ALTER TABLE result ADD CONSTRAINT result_pk PRIMARY KEY ( id_ms );

ALTER TABLE muestreo
    ADD CONSTRAINT muestreo_agricultor_fk FOREIGN KEY ( agricultor_cc )
        REFERENCES agricultor ( cc );

ALTER TABLE muestreo
    ADD CONSTRAINT muestreo_cultivo_fk FOREIGN KEY ( cultivo_id_tc )
        REFERENCES cultivo ( id_tc );

ALTER TABLE muestreo
    ADD CONSTRAINT muestreo_div_t_fk FOREIGN KEY ( div_t_id_dt )
        REFERENCES div_t ( id_dt );

ALTER TABLE muestreo
    ADD CONSTRAINT muestreo_municipio_fk FOREIGN KEY ( municipio_id_m )
        REFERENCES municipio ( id_m );

ALTER TABLE muestreo
    ADD CONSTRAINT muestreo_ortomosaico_fk FOREIGN KEY ( ortomosaico_id_ortom )
        REFERENCES ortomosaico ( id_ortom );

ALTER TABLE muestreo
    ADD CONSTRAINT muestreo_result_fk FOREIGN KEY ( result_id_ms )
        REFERENCES result ( id_ms );

```

Figure 7: Creación de las tablas: Agricultor, Cultivo, Div_t

A continuación, en la figura 8, se muestra que las tablas se han creado con éxito.



Figure 8: Resultado obtenido al momento de crear las tablas

4.5 Código SQL - Manipulación de datos (DML)

En la figura 9 se puede observar como se agregan nuevos registros a la tabla MUESTREO

```
--Fila 1
INSERT INTO MUESTREO (FECHA, MUNICIPIO_ID_M, ORTOMOSAICO_ID_ORTOM, CULTIVO_ID_TC, AGRICULTOR_CC, DIV_T_ID_DT, RESULT_ID_MS) VALUES (to_date('08/01/2023', 'MM/DD/YYYY'),1,1,1,69622656,1,1);
--Fila 2
INSERT INTO MUESTREO (FECHA, MUNICIPIO_ID_M, ORTOMOSAICO_ID_ORTOM, CULTIVO_ID_TC, AGRICULTOR_CC, DIV_T_ID_DT, RESULT_ID_MS) VALUES (to_date('08/01/2023', 'MM/DD/YYYY'),1,1,1,69622656,2,2);
--Fila 3
INSERT INTO MUESTREO (FECHA, MUNICIPIO_ID_M, ORTOMOSAICO_ID_ORTOM, CULTIVO_ID_TC, AGRICULTOR_CC, DIV_T_ID_DT, RESULT_ID_MS) VALUES (to_date('04/24/2023', 'MM/DD/YYYY'),1,2,2,69622656,3,3);
--Fila 4
INSERT INTO MUESTREO (FECHA, MUNICIPIO_ID_M, ORTOMOSAICO_ID_ORTOM, CULTIVO_ID_TC, AGRICULTOR_CC, DIV_T_ID_DT, RESULT_ID_MS) VALUES (to_date('04/24/2023', 'MM/DD/YYYY'),1,2,2,69622656,4,4);
--Fila 5
INSERT INTO MUESTREO (FECHA, MUNICIPIO_ID_M, ORTOMOSAICO_ID_ORTOM, CULTIVO_ID_TC, AGRICULTOR_CC, DIV_T_ID_DT, RESULT_ID_MS) VALUES (to_date('06/06/2023', 'MM/DD/YYYY'),1,3,3,69622656,1,5);
--Fila 6
INSERT INTO MUESTREO (FECHA, MUNICIPIO_ID_M, ORTOMOSAICO_ID_ORTOM, CULTIVO_ID_TC, AGRICULTOR_CC, DIV_T_ID_DT, RESULT_ID_MS) VALUES (to_date('06/06/2023', 'MM/DD/YYYY'),1,3,3,69622656,2,6);
--Fila 7
INSERT INTO MUESTREO (FECHA, MUNICIPIO_ID_M, ORTOMOSAICO_ID_ORTOM, CULTIVO_ID_TC, AGRICULTOR_CC, DIV_T_ID_DT, RESULT_ID_MS) VALUES (to_date('03/23/2023', 'MM/DD/YYYY'),1,4,4,43073097,3,7);
--Fila 8
INSERT INTO MUESTREO (FECHA, MUNICIPIO_ID_M, ORTOMOSAICO_ID_ORTOM, CULTIVO_ID_TC, AGRICULTOR_CC, DIV_T_ID_DT, RESULT_ID_MS) VALUES (to_date('03/23/2023', 'MM/DD/YYYY'),1,4,4,43073097,4,8);
--Fila 9
```

Salida de Script x Resultado de la Consulta x

SQL | Toda la Consulta Recuperada: 50 filas en 0,003 segundos

FECHA	MUNICIPIO_ID_M	ORTOMOSAICO_ID_ORTOM	CULTIVO_ID_TC	AGRICULTOR_CC	DIV_T_ID_DT	RESULT_ID_MS
1 01/08/23	1	1	1	69622656	1	1
2 01/08/23	1	1	1	69622656	2	2
3 24/04/23	1	2	2	69622656	3	3
4 24/04/23	1	2	2	69622656	4	4
5 06/06/23	1	3	3	69622656	1	5

Figure 9: Agregar nuevos registros con “INSERT” a la tabla MUESTREO

Para recuperar todos los registros y columnas de la tabla AGRICULTOR se utilizo el código SQL “SELECT * FROM Agricultor”, como se ve en la figura 10

SELECT * FROM AGRICULTOR;

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 21 en 0,003 seg

CC	NOMBRE	APELLIDO	TELEFONO
1 69622656	Guillermo	Mahecha	3052811572
2 43073097	Armando	Gaona	3125154497
3 71324643	Manuel	Lopez	3119828825
4 66017300	Orlando	Beltran	3221458880
5 62654864	Maria	Mahecha	3223090312
6 77203826	Dario	Urrego	3125217112

Figure 10: Recuperar todos los registros y columnas de la tabla AGRICULTOR

4.6 Código SQL + Resultados: Vistas

VISTA 1

Se crear una vista que muestre cuales agricultores tienen cultivos de piña, se utiliza una consulta SQL que se une a las tablas agricultor, cultivo y muestreo utilizando las claves primarias y luego filtre por el tipo de cultivo de piña, como se puede observar en la figura 11

```

CREATE VIEW VistaAgricultoresConPina AS
SELECT a.cc, a.nombre, a.apellido
FROM agricultor a
INNER JOIN muestreo m ON a.cc = m.agricultor_cc
INNER JOIN cultivo c ON m.cultivo_id_tc = c.id_tc
WHERE c.tipo_cultivo = 'Piña';

```

Figure 11: Vista que visualiza qué agricultores están cultivando piña

A continuación, en la figura 12, se presenta el resultado de la vista.

The screenshot shows the Oracle SQL Developer interface. On the left, the object browser displays various database objects under the 'Vistas' category, with 'VISTAAGRICULTORESCONPIÑA' highlighted by a red rectangle. To the right, the main pane shows the SQL code for creating the view:

```

CREATE VIEW VistaAgricultoresConPina AS
SELECT a.cc, a.nombre, a.apellido
FROM agricultor a
INNER JOIN muestreo m ON a.cc = m.agricultor_cc
INNER JOIN cultivo c ON m.cultivo_id_tc = c.id_tc
WHERE c.tipo_cultivo = 'Piña';

```

Below this, another SQL statement is shown:

```

SELECT * FROM VistaAgricultoresConPina;

```

The results of the query are displayed in a table titled 'Resultado de la Consulta' (Query Result). The table has columns CC, NOMBRE, and APELLIDO. The data is as follows:

CC	NOMBRE	APELLIDO
1	Dario	Urrego
2	Dario	Urrego
3	Dario	Urrego
4	Dario	Urrego
5	Pedro	Gutierrez
6	Pedro	Gutierrez
7	Pedro	Gutierrez
8	Pedro	Gutierrez

Figure 12: Resultados de la Vista que visualiza qué agricultores están cultivando piña

VISTA 2

Ahora creamos una segunda vista denominada VistaAgricultoresConCoculo mediante una consulta que combina las tablas Agricultor, Muestreo y Cultivo, utilizando las claves primarias y las relaciones pertinentes. Luego, aplicamos un filtro a los resultados utilizando la condición = Coculo para mostrar únicamente los agricultores que cultivan Coculo. Se puede observar el resultado en la figura 13.

```

CREATE VIEW VistaAgricultoresConCoculo AS
SELECT a.cc, a.nombre, a.apellido
FROM agricultor a
INNER JOIN muestreo m ON a.cc = m.agricultor_cc
INNER JOIN cultivo c ON m.cultivo_id_tc = c.id_tc
WHERE c.tipo_cultivo = 'Coculo';

```

Figure 13: Vista que visualiza qué agricultores están cultivando Coculo

El resultado de la segunda vista se puede observar en la figura 14

The screenshot shows the Object Explorer on the left with a tree view of database objects. A red box highlights the 'VISTAAGRICULTORESCONCOCULO' node under 'Tables'. To the right, the 'Script Editor' window displays the SQL code for creating the view:

```

CREATE VIEW VistaAgricultoresConCoculo AS
SELECT a.cc, a.nombre, a.apellido
FROM agricultor a
INNER JOIN muestreo m ON a.cc = m.agricultor_cc
INNER JOIN cultivo c ON m.cultivo_id_tc = c.id_tc
WHERE c.tipo_cultivo = 'Coculo';

SELECT * FROM VistaAgricultoresConCoculo;

```

Below the code, the 'Resultado de la Consulta' tab shows the output:

	CC	NOMBRE	APELLIDO
1	73515682	Nelson	Mancera
2	73515682	Nelson	Mancera
3	62383365	Yamid	Silva
4	62383365	Yamid	Silva
5	59797382	Maria	Munar
6	59797382	Maria	Munar
7	35744648	Gabriela	Correa

Figure 14: Resultados de la Vista que visualiza qué agricultores están cultivando Cocolo

VISTA 3

Se crea una tercera vista como se puede observar el resultado en la figura 15, llamada “VistaAgricultoresConBosque”. Para ello, empleamos una consulta que combina las tablas Agricultor, Muestreo y Cultivo mediante las claves primarias y relaciones apropiadas. Luego, seleccionamos las columnas que deseamos visualizar en la vista, que incluyen el código del agricultor (cc_agricultor), nombre del agricultor (nombre_agricultor), apellido del agricultor (apellido_agricultor), fecha del cultivo (fecha_cultivo) y tipo de cultivo (tipo_cultivo). Finalmente, aplicamos un filtro a los resultados utilizando la condición = Bosque para mostrar únicamente a los agricultores que cultivan bosque.

```

CREATE VIEW VistaAgricultoresConBosque AS
SELECT a.cc AS cc_agricultor, a.nombre AS nombre_agricultor, a.apellido AS apellido_agricultor,
       m.fecha AS fecha_cultivo, c.tipo_cultivo AS tipo_cultivo
FROM agricultor a
INNER JOIN muestreo m ON a.cc = m.agricultor_cc
INNER JOIN cultivo c ON m.cultivo_id_tc = c.id_tc
WHERE c.tipo_cultivo = 'Bosque';

```

Figure 15: Vista que visualiza qué agricultores que tienen Bosque

El resultado de la segunda vista se puede observar en la figura 16

The screenshot shows the Object Explorer on the left with several database objects listed, including 'VIEW_WORLOAD', 'PRODUCT_PRIVS', 'SCHEDULER_JOB_ARGS', 'SCHEDULER_PROGRAM_ARGS', 'VISTAAGRICULTORESCONBOSQUE' (which is highlighted with a red box), 'VISTAAGRICULTORESCONCOLOCU', 'VISTAAGRICULTORESCONPINA', 'Procedimientos', 'Funciones', 'Operadores', 'Colas', 'Tablas de Colas', 'Disparadores', 'REGISTRAR_FECHA_MUESTREO', 'Tipos', 'Secuencias', 'Vistas Materializadas', 'Logs de Vistas Materializadas', 'Síntomas', 'Síntomas Públicos', 'Enlaces de Base de Datos', and 'Enlaces de Base de Datos Pública'. To the right, there is a 'Salida de Script' window with the following SQL code:

```

CREATE VIEW VistaAgricultoresConBosque AS
SELECT a.cc AS cc_agricultor, a.nombre AS nombre_agricultor, a.apellido AS apellido_agricultor,
       c.fecha_cultivo AS fecha_cultivo, c.tipo_cultivo AS tipo_cultivo
FROM agricultor
INNER JOIN muestreo a ON a.cc = m.agricultor_cc
INNER JOIN cultivo c ON m.cultivo_id_tc = c.id_tc
WHERE c.tipo_cultivo = 'Bosque';

SELECT * FROM VistaAgricultoresConBosque;

```

Below the code is a 'Resultado de la Consulta' grid showing the results of the query. The columns are CC_AGRICULTOR, NOMBRE_AGRICULTOR, APELLIDO_AGRICULTOR, FECHA_CULTIVO, and TIPO_CULTIVO. The data is as follows:

	CC_AGRICULTOR	NOMBRE_AGRICULTOR	APELLIDO_AGRICULTOR	FECHA_CULTIVO	TIPO_CULTIVO
1	69622656	Guillermo	Mahecha	06/06/23	Bosque
2	69622656	Guillermo	Mahecha	06/06/23	Bosque
3	47001020	Pedro	Gutierrez	18/02/23	Bosque
4	47001020	Pedro	Gutierrez	18/02/23	Bosque
5	73515682	Nelson	Mancera	20/02/23	Bosque
6	73515682	Nelson	Mancera	20/02/23	Bosque
7	62383345	Yamid	Silva	16/07/23	Bosque
8	62383345	Yamid	Silva	16/07/23	Bosque
9	59511808	Manuel	Rodriguez	10/08/23	Bosque

Figure 16: Resultados de la Vista que visualiza qué agricultores que tienen Bosque

4.7 Código SQL + Resultados: Triggers

Se ha implementado un trigger denominado registrar_fecha_muestreo. Los triggers son objetos de la base de datos que se activan de forma automática en respuesta a eventos específicos, como inserciones (INSERT), actualizaciones (UPDATE) o eliminaciones (DELETE) en una tabla determinada. En este contexto, el trigger se ha configurado para activarse antes de ejecutar una inserción en la tabla MUESTREO. Este trigger garantiza que cada vez que se realice una inserción en la tabla MUESTREO, el campo fecha en la nueva fila se establecerá automáticamente en la fecha y hora actual del sistema antes de que se complete la operación de inserción. Este enfoque se utiliza para registrar automáticamente la fecha y hora en que se realizó el muestreo, eliminando la necesidad de que el usuario proporcione esta información manualmente. Como se puede observar en la figura 16

The screenshot shows a script editor window with the following SQL code:

```

CREATE OR REPLACE TRIGGER registrar_fecha_muestreo
BEFORE INSERT ON muestreo
FOR EACH ROW
BEGIN
:NEW.fecha := SYSTIMESTAMP;
END;

```

Figure 17: Creación de trigger

BEFORE INSERT ON muestreo: Esta línea especifica que el trigger se activará antes de que se realice una operación de inserción en la tabla muestreo.

FOR EACH ROW: Esto indica que el trigger se ejecutará una vez por cada fila que se inserte en la tabla muestreo.

SYSTIMESTAMP: Esta línea asigna la fecha actual y la hora del sistema al campo fecha de la fila que se va a insertar en la tabla muestreo.

Verificamos como se muestra en la figura 18 el ultimo registro de la tabla

MUESTREO (Fila 126)

	FECHA	MUNICIPIO_ID_M	ORTOMOSAICO_ID_ORTOM	CULTIVO_ID_TC	AGRICULTOR_CC	DIV_T_ID_DT	RESULT_ID_MS
122	11/07/23	3	61	16	50838356	2	122
123	30/07/23	3	62	16	50838356	3	123
124	30/07/23	3	62	16	50838356	4	124
125	22/03/23	3	63	16	50838356	1	125
126	22/03/23	3	63	16	50838356	2	126

Figure 18: Creación de trigger

Ahora vamos a agregar un nuevo registro y se selecciona la tabla para ver el cambio

	FECHA	MUNICIPIO_ID_M	ORTOMOSAICO_ID_ORTOM	CULTIVO_ID_TC	AGRICULTOR_CC	DIV_T_ID_DT	RESULT_ID_MS
124	30/07/23	3	62	16	50838356	4	124
125	22/03/23	3	63	16	50838356	1	125
126	22/03/23	3	63	16	50838356	2	126
127	04/10/23	1	3	3	12345	4	5

Figure 19: Creación de trigger

En la figura 19, se puede comprobar que el trigger está funcionando correctamente. A pesar de que al momento de insertar un nuevo registro la fecha sea 2052-04-12, en la tabla se observa que el trigger ha registrado automáticamente la fecha del día en que se ejecutó.

4.8 Código SQL + Resultados: Funciones

FUNCIÓN 1

Se crea una función como se puede observar en la figura 20 que pueda calcular el promedio de Materia orgánica (mo) de la tabla result para un agricultor específico (ac_c) y un tipo de cultivo específico (ctc). La función también busca el nombre del agricultor y el tipo de cultivo correspondientes en las tablas agricultor y cultivo, respectivamente. Declara tres variables locales: promedio_mo (para almacenar el promedio), mensaje (para almacenar un mensaje de resultado) y dos variables para el nombre del agricultor y el tipo de cultivo (agricultor_nombre y cultivo_tipo). La función realiza tres consultas SQL para recuperar información de la base de datos:

La primera consulta busca el nombre del agricultor (agricultor_nombre) que corresponde al número de identificación del agricultor (ac_c). La segunda consulta busca el tipo de cultivo (cultivo_tipo) que corresponde al tipo de cultivo (ctc). La tercera consulta calcula el promedio de la columna mo en la tabla result para registros que cumplen con ciertas condiciones de filtro. Estas condiciones se basan en los valores de ac_c y ctc.

Luego, la función verifica si el valor de promedio_mo es nulo (NULL). Si es nulo, significa que no se encontraron registros en la tabla RESULT para el agricultor y el tipo de cultivo dados. En este caso, la función establece el mensaje de resultado para indicar que el cultivo no pertenece al agricultor. Si el valor de promedio_mo no es nulo, significa que se encontraron registros en la tabla RESULT y se calculó un promedio. La función establece el mensaje de resultado para mostrar el promedio de materia orgánica (MO) para el agricultor y el tipo de cultivo dados. Finalmente, la función devuelve el mensaje resultante.

```

CREATE OR REPLACE FUNCTION calcular_PromedioMo(ac_c IN INTEGER, ctc IN INTEGER)
RETURN VARCHAR2
IS
    promedio_mo NUMBER;
    mensaje VARCHAR2(200);
    agricultor_nombre VARCHAR2(50);
    cultivo_tipo VARCHAR2(50);
BEGIN
    SELECT a.nombre INTO agricultor_nombre
    FROM agricultor a
    WHERE a.ac_c = ac_c;
    SELECT c.tipo_cultivo INTO cultivo_tipo
    FROM cultivo c
    WHERE c.id_ctc = ctc;
    SELECT AVG(mo) INTO promedio_mo
    FROM result r
    INNER JOIN muestra m ON r.id_ms = m.result_id_ms
    WHERE m.agricultor_ac_c = ac_c
    AND m.cultivo_id_ctc = ctc;
    IF promedio_mo IS NULL THEN
        mensaje := 'El cultivo ' || cultivo_tipo || ' no pertenece al agricultor ' || agricultor_nombre || '.';
    ELSE
        mensaje := 'El promedio de MO para el agricultor ' || agricultor_nombre || ' del cultivo ' || cultivo_tipo || ' es ' || TO_CHAR(promedio_mo);
    END IF;
    RETURN mensaje;
END calcular_PromedioMo;

```

Figure 20: Creación de una función para calcular el promedio de Materia orgánica

A continuación, se presentan algunas pruebas a la función

El agricultor Guillermo tiene tres tipos de cultivos (Caucho, Pastizal y Bosque). Al ingresar la cédula de ciudadanía de Guillermo (CC= 69622656) y el tipo de cultivo 'Pastizal' (ID=2) en la función, podemos concluir que la función se ejecuta con éxito, como se evidencia en la figura 21.

```

SELECT calcular_PromedioMo(69622656, 2) AS promedio_Materia_onica FROM dual;

```

lida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,161 segundos

PROMEDIO_MATERIA_ONICA

1 El promedio de MO para el agricultor Guillermo del cultivo Pastizal es 13,6

Figure 21: Resultados de la función para calcular el promedio de Materia orgánica

Se lleva a cabo otra prueba, en la cual se proporciona el número de cédula del agricultor Guillermo (CC=69622656) y se especifica el tipo de cultivo “Helecho” (ID=15) como entrada para la función. En este escenario, la función configura el mensaje de resultado para indicar que dicho cultivo no está asociado al agricultor. Esto se puede comprobar en la figura 22.

```

SELECT calcular_PromedioMo(69622656, 15) AS promedio_Materia_onica FROM dual;

```

lida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,007 segundos

PROMEDIO_MATERIA_ONICA

1 El cultivo "Helecho" no pertenece al agricultor "Guillermo".

Figure 22: Resultados de la función para calcular el promedio de Materia orgánica

FUNCIÓN 2

Se creo una segunda función como se puede observar en la figura 23. La finalidad de esta función es determinar y contar los distintos tipos de cultivo asociados a un agricultor específico, proporcionando además un mensaje informativo sobre los tipos de cultivo que posee dicho agricultor. En su implementación, la función declara tres variables locales: cult_tipos (para almacenar una lista de tipos de cultivo), cantidad (para almacenar la cantidad de tipos de cultivo) y agri_nombre (para almacenar el nombre del agricultor).

La función comienza realizando una consulta SQL para buscar el nombre del agricultor (almacenado en agri_nombre) correspondiente a la cédula del agricultor (indicada como cedula_agri) en la tabla agricultor. Posteriormente, la función lleva a cabo otra consulta SQL con el propósito de contar la cantidad de tipos de cultivo distintos asociados al agricultor. Para ello, se utiliza una subconsulta que busca los identificadores de los tipos de cultivo (cultivo_id_tc) en la tabla MUESTREO que corresponden al agricultor especificado en cedula_agri. El resultado de esta cuenta se almacena en la variable cantidad. La función verifica si la cantidad de tipos de cultivo es igual a cero. En caso afirmativo,

esto indica que no se encontraron tipos de cultivo asociados al agricultor. En consecuencia, la función devuelve un mensaje indicando que el agricultor no posee tipos de cultivo registrados. Si la cantidad de tipos de cultivo es mayor que cero, la función procede a realizar otra consulta SQL para obtener una lista de los tipos de cultivo asociados al agricultor. Esto se logra mediante el uso de la función LISTAGG. La lista de tipos de cultivo se almacena en la variable cult_tipos.

Finalmente, la función construye un mensaje informativo que incluye el nombre del agricultor, la cantidad de tipos de cultivo y la lista de tipos de cultivo. Dicho mensaje se genera concatenando estas variables y se devuelve como resultado.

```

CREATE OR REPLACE FUNCTION contarTiposCultivo(cedula_agri IN INTEGER)
RETURN VARCHAR2
IS
    cult_tipos VARCHAR2(4000);
    cantidad NUMBER;
    agri_nombre VARCHAR2(50);
BEGIN
    SELECT a.nombre INTO agri_nombre
    FROM agricultor a
    WHERE a.cc = cedula_agri;
    SELECT COUNT(DISTINCT c.tipo_cultivo) INTO cantidad
    FROM cultivo c
    WHERE c.id_tc IN (
        SELECT DISTINCT m.cultivo_id_tc
        FROM muestreo m
        WHERE m.agricultor_cc = cedula_agri );
    IF cantidad = 0 THEN
        RETURN 'El agricultor no tiene tipos de cultivo registrados.';
    ELSE
        SELECT LISTAGG(c.tipo_cultivo, ', ') WITHIN GROUP (ORDER BY c.tipo_cultivo)
        INTO cult_tipos
        FROM cultivo c
        WHERE c.id_tc IN (
            SELECT DISTINCT m.cultivo_id_tc
            FROM muestreo m
            WHERE m.agricultor_cc = cedula_agri);
        RETURN 'El agricultor ' || agri_nombre || ' tiene ' || TO_CHAR(cantidad) || ' tipo(s) de cultivo: ' || cult_tipos;
    END IF;
END;
/

```

Figure 23: Función para determinar y contar los distintos tipos de cultivo asociados a un agricultor

La agricultora María Munar tiene en su posesión tres tipos de cultivo: Aguacate, Cocolo y Rusco. Al ejecutar la función, se puede afirmar que está funcionando correctamente, tal como se evidencia en la figura 24

```

SELECT contarTiposCultivo(59797382) AS Agricultor_cultivos FROM dual;

```

Salida de Script | Resultado de la Consulta | SQL | Todas las Filas Recuperadas: 1 en 0,035 segundos

AGRICULTOR_CULTIVOS

1 El agricultor Maria tiene 3 tipo(s) de cultivo: Aguacate, Cocolo, Rusco

Figure 24: Resultados de la función para determinar y contar los distintos tipos de cultivo asociados a un agricultor

El agricultor Yamid Silva cuenta con 3 tipos de cultivo (Bosque, Eucalipta y Rusco) Al ejecutar la función, se puede afirmar que está funcionando correctamente, tal como se evidencia en la figura 24

```

SELECT contarTiposCultivo(62383365) AS Agricultor_cultivos FROM dual;

```

Salida de Script | Resultado de la Consulta | SQL | Todas las Filas Recuperadas: 1 en 0,008 segundos

AGRICULTOR_CULTIVOS

1 El agricultor Yamid tiene 3 tipo(s) de cultivo: Bosque, Cocolo, Eucalipta

Figure 25: Resultados de la función para determinar y contar los distintos tipos de cultivo asociados a un agricultor

4.9 Código SQL + Resultados: procedimientos almacenados

PROCEDIMIENTO 1

Se ha creado un procedimiento denominado *insertar_agricultor*, el cual recibe cuatro parámetros: p_cc, p_nombre, p_apellido y p_telefono. Estos parámetros corresponden a las columnas de la tabla "agricultor". A continuación, se describe el funcionamiento del procedimiento: Se lleva a cabo una inserción en la tabla "agricultor" utilizando los valores proporcionados en los parámetros mencionados. Después de la inserción, se realiza un COMMIT para confirmar la transacción y guardar los cambios en la base de datos. En caso de que se produzca algún error durante la ejecución del procedimiento, se captura la excepción correspondiente. Si se detecta un error, se muestra un mensaje de error al usuario para informar sobre la situación. Además, se realiza un ROLLBACK para deshacer cualquier cambio que se haya realizado en la base de datos en caso de error.

La figura 26 contiene el código utilizado para crear este procedimiento

```
CREATE OR REPLACE PROCEDURE insertar_agricultor(
    p_cc INTEGER,
    p_nombre VARCHAR2,
    p_apellido VARCHAR2,
    p_telefono INTEGER
)
IS
BEGIN
    INSERT INTO agricultor(cc, nombre, apellido, telefono)
    VALUES (p_cc, p_nombre, p_apellido, p_telefono);

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Registro de agricultor insertado correctamente.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error al insertar el registro de agricultor: ' || SQLERRM);
        ROLLBACK;
END insertar_agricultor;
```

Figure 26: Procedimiento para agregan nuevos registros a la tabla AGRICULTOR

y la figura 27 proporciona el resultado de este procedimiento.

The screenshot shows the Oracle SQL Developer interface. At the top, there is a script editor window containing the following PL/SQL code:

```
BEGIN
    insertar_agricultor(12345, 'Juan', 'Pérez', 5551234567);
END;
```

Below the script editor is a results window titled "Resultado de la Consulta". It displays the output of the "SELECT * FROM AGRICULTOR;" query. The results are presented in a table with columns CC, NOMBRE, APELLIDO, and TELEFONO. The last row, which corresponds to the inserted record (CC=12345, NOMBRE='Juan', APELLIDO='Pérez', TELEFONO=5551234567), is highlighted with a red border.

CC	NOMBRE	APELLIDO	TELEFONO
6	Dario	Urrego	3125217112
7	Pedro	Gutierrez	3017090312
8	Nelson	Mancera	3163532065
9	Miguel	Castro	3213322520
10	Yamid	Silva	3122222520
11	Maria	Munar	3211260726
12	Manuel	Rodriguez	3118800137
13	Gloria	Esperanza	3118549345
14	Gabriela	Correa	3151617451
15	Julio	Rozo	3120824107
16	Miguel	Rozo	312198472
17	Guilberto	Gutierrez	3120179606
18	Hector	Sarmiento	3118755489
19	Marlen	Rodriguez	3120972619
20	Rosa	Maldonado	3119401166
21	Fabio	Rodriguez	3122595310
22	Juan	Pérez	5551234567

Figure 27: Resultados del procedimiento para agregan nuevos registros a la tabla AGRICULTOR

PROCEDIMIENTO 2

Se crea un segundo procedimiento almacenado llamado “calcular_resultado_muestreo” como se muestra en la figura 28 . Este procedimiento recibe un parámetro

de entrada llamado p_result_id_ms, de tipo INTEGER, y tiene como finalidad calcular un valor total mediante la suma de varios campos de la tabla RESULT. Luego, actualiza un campo llamado “Prueba_Procedimiento” en la misma tabla con el valor calculado. A continuación, se detalla el funcionamiento de este procedimiento:

Se declara una variable local denominada v_total, de tipo FLOAT, con el propósito de almacenar el valor total calculado. Se ejecuta una consulta SQL para calcular el valor total. Esta consulta suma los valores de los campos “estructura”, “porosidad”, “color” y “cant_lombrices” en la tabla RESULT. El resultado de esta suma se almacena en la variable v_total. Se realiza una consulta SQL de actualización (UPDATE) en la tabla RESULT para establecer el valor del campo “Prueba_Procedimiento” con el valor calculado (v_total). La actualización se aplica al registro que coincide con el ID especificado en el parámetro p_result_id_ms. Se lleva a cabo un COMMIT para confirmar los cambios en la base de datos. Esto asegura que los cambios realizados se guarden de manera permanente.

Se implementa una estructura de manejo de excepciones (EXCEPTION) para abordar posibles errores: Si la consulta SQL no encuentra ningún registro que coincida con el ID especificado (p_result_id_ms), se captura la excepción NO_DATA_FOUND y se muestra un mensaje de error que indica que no se encontró un registro con el ID especificado. Si se produce cualquier otro error durante la ejecución, se captura la excepción OTHERS y se muestra un mensaje de error que incluye información sobre el error utilizando la función SQLERRM. Además, se realiza un ROLLBACK para deshacer cualquier cambio pendiente en caso de error.

```

CREATE OR REPLACE PROCEDURE calcular_resultado_muestreo (
    p_result_id_ms INTEGER
) AS
    v_total FLOAT;
BEGIN
    -- Calcular el total (suma de varios campos) según tus necesidades
    SELECT (estructura + porosidad + color + cant_lombrices) INTO v_total
    FROM result
    WHERE id_ms = p_result_id_ms;
    -- Actualizar el campo 'total' en la tabla 'result' con el valor calculado
    UPDATE result
    SET Prueba_Procedimiento = v_total
    WHERE id_ms = p_result_id_ms;
    -- Otros cálculos o actualizaciones que deseas realizar aquí
    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No se encontró un registro con el ID especificado.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error al calcular y actualizar el resultado: ' || SQLERRM);
        ROLLBACK;
END calcular_resultado_muestreo;

```

Figure 28: Procedimiento para calcular suma de algunas variables

Para probar este procedimiento primero se crea una nueva Columna en la tabla RESULT. Se selecciona la tabla RESULT para ver la ultima columna como

se ve en la figura 29

	N_AMONICAL	N_INTRICO	SATURACION_MG	SATURACION_SODIO	SATURACION_ALUMINIO	SATURACION_K	SATURACION_CA	RELACION_CA_MG	RELACION_CA_K	RELACION_MG_K	RELACION_CAMG_K
1	308	24,1	332	18,6	0,612	52,1	7,63	73,1	3,93	9,57	2,43
2	428	23,8	406	18,7	0,666	39,1	7,07	73,6	3,94	10,4	2,64
3	14,6	22,1	63,5	19,1	0,461	43,5	3,36	77	4,03	22,9	5,69

Figure 29: Visualizar tabla RESULT

Al seleccionarla nos damos cuenta que la ultima columna de RESULT es “RELACION_CAMG_K”, como se puede ver en la figura 29 , ahora agregamos la nueva columna llamada “Prueba_Procedimiento”, como se puede ver en la figura 30.

```
ALTER TABLE result
ADD Prueba_Procedimiento FLOAT;
```

Figure 30: Crear nueva columna en la tabla RESULT

Volvemos a visualizar la tabla RESULT en la figura 31.

	SATURACION_MG	SATURACION_SODIO	SATURACION_ALUMINIO	SATURACION_K	SATURACION_CA	RELACION_CA_MG	RELACION_CA_K	RELACION_MG_K	RELACION_CAMG_K	PRUEBA_PROCEDIMIENTO
1	332	18,6	0,612	52,1	7,63	73,1	3,93	9,57	2,43	(null)
2	406	18,7	0,666	39,1	7,07	73,6	3,94	10,4	2,64	(null)
3	63,5	19,1	0,461	43,5	3,36	77	4,03	22,9	5,69	(null)

Figure 31: Visualizar tabla RESULT

Luego ejecutamos el procedimiento y nos volvemos a visualizar la tabla RESULT como se ve en la figura 32.

	SATURACION_MG	SATURACION_SODIO	SATURACION_ALUMINIO	SATURACION_K	SATURACION_CA	RELACION_CA_MG	RELACION_CA_K	RELACION_MG_K	RELACION_CAMG_K	PRUEBA_PROCEDIMIENTO
1	332	18,6	0,612	52,1	7,63	73,1	3,93	9,57	2,43	16
2	406	18,7	0,666	39,1	7,07	73,6	3,94	10,4	2,64	(null)
3	63,5	19,1	0,461	43,5	3,36	77	4,03	22,9	5,69	(null)
4	78	18,9	0,38	20,2	2,96	77,7	4,12	26,2	6,36	(null)

Figure 32: Resultado del procedimiento para calcular suma de algunas variables

Verificamos que la suma de (Estructura + Porosidad + Color + Cant_lombrices) corresponda al resultado del procedimiento ($3 + 3 + 4 + 6 = 16$), Ver figura 33.

	ESTRUCTURA	POROSIDAD	COLOR	CANT_LOMBRICES
1	3	3	4	6
2	3	3	3	3
3	6	3	2	6
4	0	3	2	3
5	3	6	4	4,5

Figure 33: Resultado del procedimiento para calcular suma de algunas variables

5 Bases de Datos No-SQL

5.1 Diagrama Bases de Datos No-SQL

A continuación se presentan las entidades y sus relaciones:

Entidades: Ortomosaico, Agricultor, División de Terreno, Tipo de Cultivo, Muestreo, Municipio, Resultado.

Relaciones:

- **Muestreo a Municipio**

Muchos a Uno: Un muestreo pertenece a un municipio, pero un municipio puede tener muchos muestreos.

- **Muestreo a Ortomosaico**

Muchos a Uno: Un muestreo pertenece a un ortomosaico, pero un ortomosaico puede tener muchos muestreos.

- **Muestreo a Cultivo**

Muchos a Uno: Un muestreo pertenece a un tipo de cultivo, pero un tipo de cultivo puede tener muchos muestreos.

- **Muestreo a Agricultor**

Muchos a Uno: Un muestreo pertenece a un agricultor, pero un agricultor puede tener muchos muestreos.

- **Muestreo a División de Terreno**

Muchos a Uno: Un muestreo pertenece a una división de terreno, pero una división de terreno puede tener muchos muestreos.

- **Muestreo a Resultado** Uno a Uno: Cada muestreo se relaciona con un resultado y viceversa.

5.1.1 Meta-modelo conceptual

El meta-modelo conceptual describe las entidades principales y sus relaciones de una manera abstracta y fácilmente comprensible. Aquí se representan las entidades y las relaciones sin preocuparse demasiado por detalles técnicos.

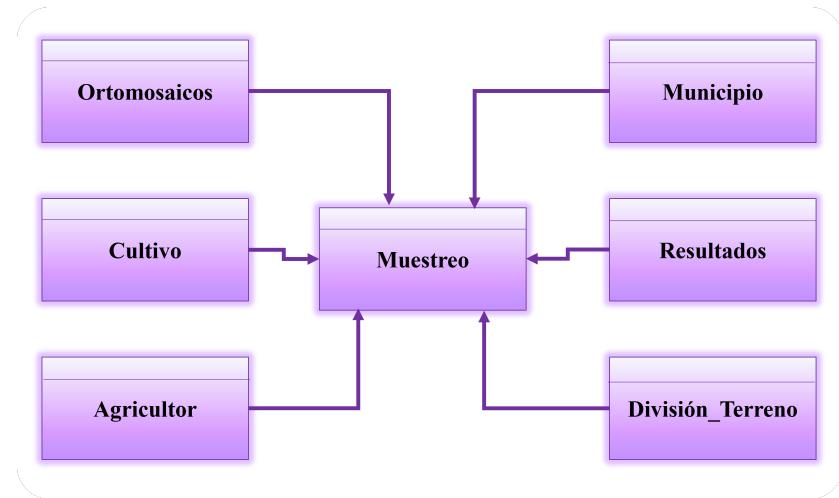


Figure 34: modelo conceptual

5.1.2 Meta - Modelo Lógico



Figure 35: Modelo Lógico

5.1.3 Meta - Modelo Físico

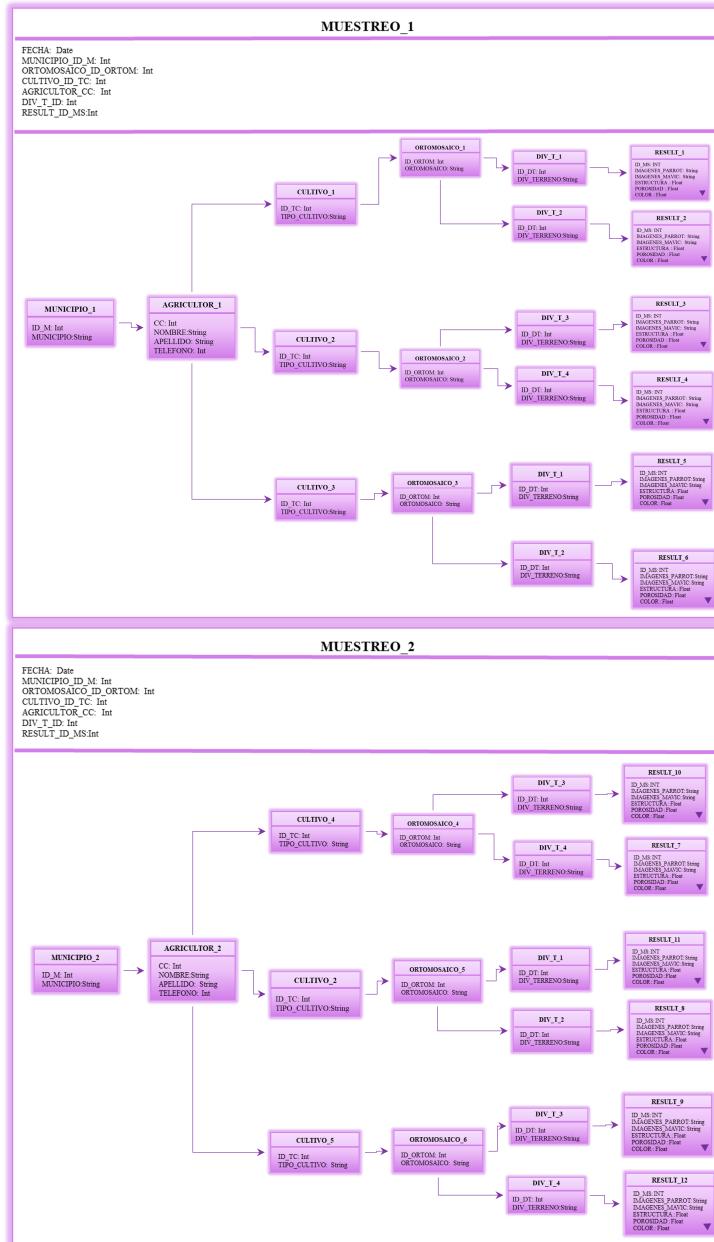


Figure 36: Modelo Físico

5.2 SMBD utilizado para la Base de Datos No-SQL

Para el proyecto que tiene como objetivo determinar indicadores de calidad del suelo mediante modelos de regresión de machine learning a partir de análisis de imágenes multiespectrales de suelos en Cundinamarca, se ha optado por utilizar MongoDB como sistema de gestión de bases de datos. La elección de MongoDB se fundamenta en su Modelo de Datos Flexible, el cual utiliza un formato de datos basado en documentos JSON llamado BSON. Esta característica posibilita almacenar datos de manera versátil, permitiendo que los documentos puedan contener campos de diferentes tipos sin estar limitados a una estructura fija. Además, se ha considerado el Soporte para Consultas Ricas de MongoDB, ya que este sistema no solo facilita consultas complejas mediante su lenguaje de consulta con operadores avanzados y funciones de agregación, sino que también cuenta con soporte para índices, agilizando el proceso de recuperación de datos, aspectos cruciales para el análisis detallado de imágenes multiespectrales y la ejecución eficiente de modelos de regresión de machine learning en este contexto específico.

Se creó una base de datos llamada "DB_multispectral" en la cual se crearon siete colecciones (AGRICULTOR, CULTIVO, DIV_T, MUESTREO, MUNICIPIO, ORTOMOSAICO y RESULT) como se puede observar en la siguiente figura



Figure 37: Creación de Base de datos y colecciones

En la colección de AGRICULTOR, cada documento representa a un agricultor con detalles como cédula de ciudadanía, nombre, apellido y teléfono.

DB_Multidimensional.AGRICULTOR

STORAGE SIZE: 20KB LOGICAL DATA SIZE: 2.15KB TOTAL DOCUMENTS: 21 INDEXES: 0

Find Indexes Schema Anti-Patterns 0 Aggregation

Filter Type a query: { field: 'value' }

```
_id: ObjectId('6556b29dd48ed0b21af87e10')
CC: "69622656"
NOMBRE: "Guillermo"
APELLIDO: "Mahecha"
TELEFONO: "3052811572"
```

```
_id: ObjectId('6556b29dd48ed0b21af87e11')
CC: "43073097"
NOMBRE: "Armando"
APELLIDO: "Gaona"
TELEFONO: "3125154497"
```

```
_id: ObjectId('6556b29dd48ed0b21af87e12')
CC: "71324643"
NOMBRE: "Manuel"
APELLIDO: "Lopez"
TELEFONO: "3119828825"
```

Figure 38: Colección AGRICULTO

En la colección de CULTIVO, cada documento representa a un tipo de cultivo.

DB_Multidimensional.CULTIVO

STORAGE SIZE: 20KB LOGICAL DATA SIZE: 971B TOTAL DOCUMENTS: 16 INDEXES TOTAL S

Find Indexes Schema Anti-Patterns 0 Aggregation

Filter Type a query: { field: 'value' }

QUERY RESULTS: 1-16 OF 16

```
_id: ObjectId('6556b2dad48ed0b21af87e25')
TIPO_CULTIVO: "Caucho"
```

```
_id: ObjectId('6556b2dad48ed0b21af87e26')
TIPO_CULTIVO: "Pastizal"
```

```
_id: ObjectId('6556b2dad48ed0b21af87e27')
TIPO_CULTIVO: "Bosque"
```

```
_id: ObjectId('6556b2dad48ed0b21af87e28')
TIPO_CULTIVO: "Yuca"
```

```
_id: ObjectId('6556b2dad48ed0b21af87e29')
TIPO_CULTIVO: "Citricos"
```

Figure 39: Colección CULTIVO

En la colección de DIV_T, cada documento representa una division de terreno, en este caso existen cuatro posibles divisiones (Sur, norte, este y oeste).

DB_Multidimensional.DIV_T

STORAGE SIZE: 20KB LOGICAL DATA SIZE: 249B TOTAL DOCUMENTS

Find

Indexes

Schema Anti-Patterns 0

Filter 

Type a query: { field: 'value' }

QUERY RESULTS: 1-4 OF 4

```
_id: ObjectId('6556b30ed48ed0b21af87e35')
DIV_TERRENO: "Zona Sur"
```

```
_id: ObjectId('6556b30ed48ed0b21af87e36')
DIV_TERRENO: "Zona Norte"
```

```
_id: ObjectId('6556b30ed48ed0b21af87e37')
DIV_TERRENO: "Zona Este"
```

```
_id: ObjectId('6556b30ed48ed0b21af87e38')
DIV_TERRENO: "Zona Oeste"
```

Figure 40: Colección DIV_T

En la colección de MUNICIPIO, cada documento representa a un municipio, en este caso existen tres: Sesquile, Paratebueno y Cachipay. Los cuales son los municipios de donde se ha recolectado la información.

DB_Multidimensional.MUNICIPIO

STORAGE SIZE: 20KB LOGICAL DATA SIZE: 178B TOTAL DOCUMENTS: 3

Find

Indexes

Schema Anti-Patterns 0

Agg

Filter 

Type a query: { field: 'value' }

QUERY RESULTS: 1-3 OF 3

```
_id: ObjectId('6556b4cad48ed0b21af87ef6')
MUNICIPIO: "Paratebueno"
```

```
_id: ObjectId('6556b4cad48ed0b21af87ef7')
MUNICIPIO: "Cachipay"
```

```
_id: ObjectId('6556b4cad48ed0b21af87ef8')
MUNICIPIO: "Sesquile"
```

Figure 41: Colección MUNICIPIO

En la colección de ORTOMOSAICO, cada documento representa a una ruta donde esta la imagen de un ortomosaico, cada ortomosaico es generado por un cultivo.

DB_Multidimensional.ORTOMOSAICO

STORAGE SIZE: 20KB LOGICAL DATA SIZE: 11.25KB TOTAL DOCUMENTS: 63 INDEXES TOTAL SIZE: 20KB

[Find](#) [Indexes](#) [Schema Anti-Patterns \(0\)](#) [Aggregation](#) [Search Indexes](#)

Filter Type a query: { field: 'value' }

```
_id: ObjectId('6556b463d48ed0b21af87eb7')
ORTOMOSAICOS: "C:\Users\menri\Desktop\Proyecto Bases de datos\Municipios\Paratebueno\..."
```

```
_id: ObjectId('6556b463d48ed0b21af87eb8')
ORTOMOSAICOS: "C:\Users\menri\Desktop\Proyecto Bases de datos\Municipios\Paratebueno\..."
```

```
_id: ObjectId('6556b463d48ed0b21af87eb9')
ORTOMOSAICOS: "C:\Users\menri\Desktop\Proyecto Bases de datos\Municipios\Paratebueno\..."
```

```
_id: ObjectId('6556b463d48ed0b21af87eba')
ORTOMOSAICOS: "C:\Users\menri\Desktop\Proyecto Bases de datos\Municipios\Paratebueno\..."
```

Figure 42: Colección ORTOMOSAICO

En la colección de RESULT, cada documento representa a un resultado de laboratorio de suelos.

DB_Multidimensional.RESULT

STORAGE SIZE: 64KB LOGICAL DATA SIZE: 186.14KB TOTAL DOCUMENTS: 126 INDEXES TOTAL SIZE: 20KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes

Filter Type a query: { field: 'value' }

```
_id: ObjectId('6556b588d48ed0b21af87ef9')
IMAGENES_PARROT: "C:\Users\menri\Desktop\Proyecto Bases de datos\Municipios\Paratebueno\..."
IMAGENES_MAVIC: "C:\Users\menri\Desktop\Proyecto Bases de datos\Municipios\Paratebueno\..."
ESTRUCTURA: "3.0"
POROSIDAD: "3.0"
COLOR: "4"
CANT_LOMBRICES: "6.0"
CANT_ORGANISMOS: "6.0"
RESIS_ROMPI_PENETRACION: "4"
ESTABILIDAD_AGREGADOS: "3.0"
MATERIA_ORGANICA: "2"
CAPAS_ENDURECIDAS: "2.0"
RESIS_PENETRACION: "4"
CRECIMIENTO_RAICES: "3.0"
CONDUCTIVIDAD_ELECTRICA: "4"
V_INFILTRACION: "0"
TOTAL: "44.0"
INTERPRETACION_PI: "Moderada Calidad"
DESCRIP_FISICA: "SÓLIDO PARDO OSCURO CON PRESENCIA DE MATERIAL VEGETAL"
PH: "6.02"
CE: "3.08"
SH: "33.8"
```

Figure 43: Colección RESULT

En la colección de MUESTREO, cada documento representa la realización de un muestreo asociada a: un agricultor, cultivo, ortomosaico del cultivo, división de terreno y resultados de laboratorio. Los campos terminados en (`_id`) establecen una relación con las colecciones (Agricultor, Cultivo, ortomosaico, Div_t y Result) utilizando los identificadores únicos.

DB_Multidimensional.MUESTREO

STORAGE SIZE: 24KB LOGICAL DATA SIZE: 22.69KB TOTAL DOCUMENTS: 126 INDE

[Find](#) [Indexes](#) [Schema Anti-Patterns](#) (0) [Aggregation](#)

[Filter](#) Type a query: { field: 'value' }

```

_id: ObjectId('6556b3aad48ed0b21af87e39')
FECHA: 2023-08-01T00:00:00.000+00:00
ORTOMOSAICO_id: ObjectId('6556b463d48ed0b21af87eb7')
MUNICIPIO_id: ObjectId('6556b4cad48ed0b21af87ef6')
RESULT_id: ObjectId('6556b588d48ed0b21af87ef9')
AGRICULTOR_id: ObjectId('6556b29dd48ed0b21af87e10')
DIV_T_id: ObjectId('6556b30ed48ed0b21af87e35')
CULTIVO_id: ObjectId('6556b30ed48ed0b21af87e35')

_id: ObjectId('6556b3aad48ed0b21af87e3a')
FECHA: 2023-08-01T00:00:00.000+00:00
ORTOMOSAICO_id: ObjectId('6556b463d48ed0b21af87eb7')
MUNICIPIO_id: ObjectId('6556b4cad48ed0b21af87ef6')
AGRICULTOR_id: ObjectId('6556b29dd48ed0b21af87e10')
CULTIVO_id: ObjectId('6556b30ed48ed0b21af87e35')
DIV_T_id: ObjectId('6556b30ed48ed0b21af87e36')
RESULT_id: ObjectId('6556b588d48ed0b21af87efa')

```

Figure 44: Colección MUESTREO

6 Bibliografía

References

- A. A., M., Mukhtar, N., Rasib, A., Suhandri, H., & MOHD BUKARI, S. (2020, 06). Mapping of peat soil physical properties by using drone-based multispectral vegetation imagery. *IOP Conference Series: Earth and Environmental Science*, 498, 012021. doi: 10.1088/1755-1315/498/1/012021
- Beresniewicz, J., Billington, A., Büchi, M., Caffrey, M., Crisco, R., & Cunningham, L. (2011). *Expert pl/sql practices: for oracle developers and dbas* (1. Aufl. ed.). Berkeley, CA: Apress.
- Beresniewicz, J., Billington, A., Büchi, M., Caffrey, M., Crisco, R., Cunningham, L., ... Shamsudeen, R. (2011). *Expert pl/sql practices: for oracle developers and dbas*. Berkeley, CA: Apress.
- El Congreso de la República de Colombia. (2012). *Ley 1581 de 2012*. Retrieved from https://www.funcionpublica.gov.co/eva/gestornormativo/norma_pdf.php?i=49981
- FAO, fund for agricultural development, I., Unicef, food programme, W., & health organization, W. (2019). *The state of food security and nutrition in the world*. Retrieved from <https://www.fao.org/3/ca5162en/ca5162en.pdf> (Online; accessed 15 september 2023)
- Geisseler, D., & Scow, K. M. (2014). Long-term effects of mineral fertilizers on soil microorganisms – a review. *Soil Biology and Biochemistry*, 75, 54-63. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0038071714001187> doi: <https://doi.org/10.1016/j.soilbio.2014.03.023>
- Guan, S., Fukami, K., Matsunaka, H., Okami, M., Tanaka, R., Nakano, H., ... Takahashi, K. (2019). Assessing correlation of high-resolution ndvi with fertilizer application level and yield of rice and wheat crops using small uavs. *Remote Sensing*, 11(2). Retrieved from <https://www.mdpi.com/2072-4292/11/2/112> doi: 10.3390/rs11020112
- Jang, G., Kim, J., Yu, J.-K., Kim, H.-J., Kim, Y.-H., Kim, D.-W., ... Chung, Y. S. (2020, 03). Review: Cost-effective unmanned aerial vehicle (uav) platform for field plant breeding application. *Remote Sensing*, 12, 998. doi: 10.3390/rs12060998
- Jung, J., Maeda, M., Chang, A., Bhandari, M., Ashapure, A., & Landivar-Bowles, J. (2021). The potential of remote sensing and artificial intelligence as tools to improve the resilience of agriculture production systems. *Current opinion in biotechnology*, 70, 15-22.
- Khanal, S., Fulton, J., Klopfenstein, A., Douridas, N., & Shearer, S. (2018). Integration of high resolution remotely sensed data and machine learning techniques for spatial prediction of soil properties and corn yield. *Computers and Electronics in Agriculture*, 153, 213-225. Retrieved from <https://www.sciencedirect.com/science/article/>

- pii/S0168169918300334 doi: <https://doi.org/10.1016/j.compag.2018.07.016>
- Lakshman, B. (2000). *Oracle developer forms techniques* (1st edition ed.). Indianapolis, Ind: Sams.
- Narayanan, A. (2016). *Oracle sql developer: learn database design, development, and administration using the feature-rich sql developer 4.1 interface*. Packt Publishing.
- Paul, S., Coops, N., Johnson, M., Krzic, M., & Smukler, S. (2019). Evaluating sampling efforts of standard laboratory analysis and mid-infrared spectroscopy for cost effective digital soil mapping at field scale. *Geoderma*, 356, 113925. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0016706119307475> doi: <https://doi.org/10.1016/j.geoderma.2019.113925>
- Thompson, L., & Puntel, L. (2020, 05). Transforming unmanned aerial vehicle (uav) and multispectral sensor into a practical decision support system for precision nitrogen management in corn. *Remote Sensing*, 12, 1597. doi: 10.3390/rs12101597
- Wang, C., Zheng, M., Song, W., Wen, S., Wang, B., Zhu, C., & Shen, R. (2017). Impact of 25 years of inorganic fertilization on diazotrophic abundance and community structure in an acidic soil in southern china. *Soil Biology and Biochemistry*, 113, 240-249. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0038071717304017> doi: <https://doi.org/10.1016/j.soilbio.2017.06.019>