



SISTEMAS DE BASES DE DATOS DISTRIBUIDOS

1. DATOS PERSONALES

ESTUDIANTE: Kevin Damian Palate Moreta

CARRERA: Tecnologías de la Información

PARALELO: "A"

ASIGNATURA: Sistemas de Bases de Datos Distribuidas **NIVEL:** 05

Semana 01

29/Aug/2025

BD Centralizadas

Es un sistema donde todos los datos se almacenan y gestionan en un solo lugar, es decir en un solo servidor (sistema central).

Ventajas: Es usado debido a su simplicidad, fácil implementación ya que no requiere muchos recursos y menor complejidad en las operaciones.

Desventajas: Punto único, es decir si falla deja de estar disponible. Limitación en rendimiento ya que puede saturar el servidor y Difícil escalabilidad.

Semana 02

01/Sep/2025

Bases de Datos Distribuidas

Una Base de datos Distribuida (DDB) es un conjunto de bases de datos que están divididas en varios sitios, pero conectados lógicamente, es decir que están relacionadas entre sí para cumplir una función y que se accede a ella a través de un Sistema de Administración de Bases de Datos DDBMS.

Para que una BD sea distribuida debe tener

- Redes de comunicación: Para conectar con varios nodos.
- Sistemas de gestión de Bases de Datos (SGBD): Para administrar los datos.
- Sistemas operativos

RED(NUBE-SGBD-CLIENTE)

Consumo de Recursos: Las bases de datos distribuidas necesitan un gran computador por ende tiene alto costo de hardware.

Transparencia: Es decir que cuando hay un cambio en un nodo se refleja en los demás haciendo que parezca una BD centralizada.

DDB (Distributed Data Base) es una base de datos tradicional dividida en diferentes partes físicas dispersas que se acceden de manera lógica.

DBMS (Data Base Management System o SGBD) permite crear, gestionar, almacenar y recuperar información. Ej. MySQL, ORACLE.

DDBMS (Distributed Database Management System) gestiona los datos ubicados en distintos sitios, gracias a la fragmentación, replicación y distribución.

Componentes del DDBMS

Estaciones de trabajo → Interfaces de usuario / Clientes

Componentes de Software-Hardware → El lenguaje SQL del SGBD - Los servidores

Medio de Comunicación → La red que permite la transferencia de datos

Procesador de Transacciones → El SGBD que se encarga de gestionar las operaciones distribuidas

Procesador de datos → **El motor de base de datos** (encargado de la ejecución de consultas, recuperación, actualización y manejo físico de los datos)

Características

1. Recibe solicitudes.
 2. Valida, analiza y descompone la solicitud (Hecha por el procesador de datos).
 3. Descompone la solicitud en varias operaciones.
 4. Busca, localiza lee y valida los datos.
 5. Garantiza la consistencia, seguridad y la integridad. (Hecho en)
 6. Valida los datos de conformidad con las condiciones (Hecho en las consultas en el SGBD).
 7. Presenta los datos seleccionados
 8. Todas estas actividades son transparentes para el usuario.
-

Clasificación de Bases de Datos

Se clasifican en base a la distribución de procesos y datos.

Los datos en una base de datos **centralizada** están en un servidor, un solo sitio de procesamiento(SPSD)

Los datos en una BD **distribuida** están en en múltiples servidores, múltiples sitios de procesamiento (MPSD).

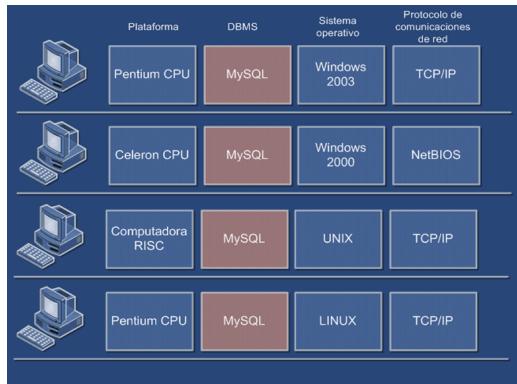
Homogénea:

- Todas las bases de datos que conforman el sistema distribuido usan el **mismo SGBD** y el mismo modelo de datos.
- Los usuarios no perciben diferencias entre los nodos.

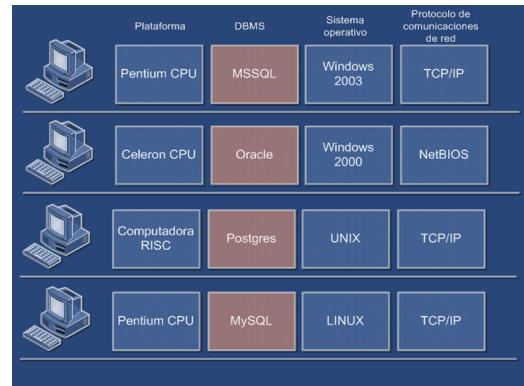
Heterogénea:

- El sistema distribuido está compuesto por **diferentes SGBD**, sistemas operativos o modelos de datos.
- Pueden existir nodos con MySQL, Oracle, SQL Server, PostgreSQL,

- La comunicación y coordinación más sencillas, por la uniformidad:
 - Lenguaje de consulta (ej. SQL).
 - Estructuras de datos.
 - Protocolos de acceso.



- etc.
- Requiere mecanismos de **traducción e integración** (middleware o gateways) para que los usuarios accedan de forma unificada.



Interacción con la BD mediante Postman, Swagger

Ventajas de BDD

- Datos localizados cerca del sitio de mayor demanda
- Acceso mas rápido a los datos
- Procesamiento mas rápido de datos
- Facilita el crecimiento (escalable)
- Comunicación mejorada
- Costos de operación distribuida
- Interface de usuario fácil de usar
- Tolerante a fallas, ya que hay menos peligro si falla un solo punto

Problemas de BDD

- Rendimiento afectado por la carga de trabajo.
- La complejidad de los componentes que requiere el sistema distribuido: ordenadores, red, almacenamiento, transacción, replicaciones, etc.
- Mayor complejidad, altos gastos de construcción y mantenimiento
- Mayor complejidad en el diseño e implementación del sistema.

- Independencia del procesador por la fragmentación



Mayor Consumo de Recursos

Niveles de transparencia de una BDD

La **transparencia**: significa que el usuario trabaja con la BD de manera sencilla, sin preocuparse por los detalles técnicos de cómo está implementada. Es decir, el sistema oculta la complejidad interna (dónde están los datos, cómo se fragmentan o replican) y le muestra al usuario solo lo necesario para que use la base de datos como si fuera una sola.

1. Trasparencia de heterogeneidad
 2. Trasparencia de desempeño
 3. Trasparencia de falla
 4. Trasparencia de replicación
 5. Trasparencia de transacción
 6. Trasparencia de distribución (transparencia de fragmentación, ubicación y ubicación local)
 7. Independencia de datos
 8. Datos
-

Transparencia de Distribución

Permite manejar una base de datos dispersas físicamente como si fuera centralizada. Con 3 niveles de transparencia de distribución:

- Transparencia de fragmentación
- Transparencia de ubicación
- Transparencia de ubicación local



Fragmentación Horizontal ⇒ divide por filas.
(Mediante una condición en común atributo-columna)

Fragmentación Vertical ⇒ divide por columnas. (Mediante separación de columnas junto a la PK)



Filas
→
Registros
Columnas
→
Campos

```
-- ACTIVIDAD
-- Crea una base de datos llamada universidad con 3 extensiones.
CONNECT SYSTEM;
CREATE USER UNIVERSIDAD IDENTIFIED BY UNIVERSIDAD;
GRANT CONNECT, RESOURCE, UNLIMITED TABLESPACE TO UNIVERSIDA
D;

CONNECT UNIVERSIDAD;
-- TABLAS PRINCIPALES
CREATE TABLE UNIVERSIDADES(
ID_UNI VARCHAR(5) PRIMARY KEY,
NOM_UNI VARCHAR(50) NOT NULL
);

-- EXTENSION
CREATE TABLE CAMPUS(
ID_CAM VARCHAR(5) PRIMARY KEY,
NOM_CAM VARCHAR(100) NOT NULL,
DIR_CAM VARCHAR(50) NOT NULL,
ID_UNI_PER VARCHAR(5) NOT NULL REFERENCES UNIVERSIDADES(ID_UN
I)
);

CREATE TABLE CARRERAS(
ID_CAR VARCHAR(5) PRIMARY KEY,
NOM_CAR VARCHAR(50) NOT NULL,
ID_CAM_PER VARCHAR(5) NOT NULL REFERENCES CAMPUS(ID_CAM));
```

```

-- UNIVERSIDADES
INSERT INTO UNIVERSIDADES VALUES('U001', 'UNIVERSIDAD TECNICA D
E AMBATO');

-- CAMPUS
INSERT INTO CAMPUS VALUES('C001', 'CAMPUS HUACHI', 'AV LOS CHAS
QUIS', 'U001');
INSERT INTO CAMPUS VALUES('C002', 'CAMPUS INGAHURCO', 'AV INGA
HURCO', 'U001');
INSERT INTO CAMPUS VALUES('C003', 'CAMPUS QUEROCHACA', 'CALLE
QUEROCHACA', 'U001');

-- CARRERAS
INSERT INTO CARRERAS VALUES('CAR01', 'TECNOLOGIAS DE LA INFORM
ACION', 'C001');
INSERT INTO CARRERAS VALUES('CAR02', 'FISIOTERAPIA', 'C002');
INSERT INTO CARRERAS VALUES('CAR03', 'AGRONOMIA', 'C003');

SELECT NOM_CAR FROM CARRERAS
WHERE ID_CAM_PER = 'C001';

SELECT NOM_CAR FROM CARRERAS
WHERE ID_CAM_PER = 'C002';

SELECT NOM_CAR FROM CARRERAS
WHERE ID_CAM_PER = 'C003';

```

CASO 1: La base de datos soporta transparencia de fragmentación.

```

ALTER TABLE CARRERAS
ADD CUP_DIS NUMBER;
UPDATE CARRERAS SET CUP_DIS = 30;
UPDATE CARRERAS SET CUP_DIS = 20 WHERE ID_CAR = 'CAR01';
ALTER TABLE CARRERAS
MODIFY CUP_DIS NUMBER NOT NULL;

```

-- Fragmentacion horizontal porque se divide la tabla de acuerdo a la cantidad de cupos.

```
SELECT NOM_CAR FROM CARRERAS  
WHERE CUP_DIS >= 25;
```

CASO 2: La base de datos soporta transparencia de ubicación.

```
SELECT *  
FROM CARRERAS  
WHERE ID_CAM_PER = 'C001' AND CUP_DIS>=20  
UNION ALL  
SELECT *  
FROM CARRERAS  
WHERE ID_CAM_PER = 'C002' AND CUP_DIS>=20  
UNION ALL  
SELECT *  
FROM CARRERAS  
WHERE ID_CAM_PER = 'C003' AND CUP_DIS>=20;
```

05/Sep/2025

▼ Ejercicio de BD Distribuidas UTA_Distribuida

```
/* ======  
=====  
UTA_Distribuida – Universidad Técnica de Ambato (transparencia dist  
ribuida)  
Sitios: HUACHI, INGAHURCO, QUEROCHACA  
Motor: SQL Server (T-SQL)  
Autor: Jose Caiza  
Concepto docente: simular sedes (schemas) y ofrecer capa GLOBAL  
(vistas)  
que ocultan localización/fragmentación/replicación.  
======  
===== */  
  
-- Limpieza opcional  
  
IF DB_ID('UTA_Distribuida') IS NOT NULL  
BEGIN
```

```
ALTER DATABASE UTA_Distribuida SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
DROP DATABASE UTA_Distribuida;
END
GO
```

```
CREATE DATABASE UTA_Distribuida;
GO
```

```
USE UTA_Distribuida;
GO
```

```
/* =====
ESQUEMAS (sitios físicos)
===== */
```

```
CREATE SCHEMA global AUTHORIZATION dbo;
GO
```

```
CREATE SCHEMA site_huachi AUTHORIZATION dbo;
GO
```

```
CREATE SCHEMA site_ingahurco AUTHORIZATION dbo;
GO
```

```
CREATE SCHEMA site_querochaca AUTHORIZATION dbo;
GO
```

```
/* =====
=====
MODELO LÓGICO
- carreras: REPLICADA en los 3 sitios
- extensiones: FRAGMENTADA horizontalmente (una por sitio)
- oferta_carrera: FRAGMENTADA horizontalmente por extensión
```

```

=====
===== */

-- Carreras (replicada)

CREATE TABLE site_huachi.carreras (
    carrera_id    INT      NOT NULL PRIMARY KEY,
    nombre        VARCHAR(100) NOT NULL,
    nivel         VARCHAR(30) NOT NULL, -- Pregrado/Postgrado
    modalidad     VARCHAR(30) NOT NULL, -- Presencial/En línea/Híbrida
    codigo_sniese VARCHAR(20) NULL,
    duracion_sem TINYINT   NOT NULL
);

CREATE TABLE site_ingahurco.carreras (
    carrera_id    INT      NOT NULL PRIMARY KEY,
    nombre        VARCHAR(100) NOT NULL,
    nivel         VARCHAR(30) NOT NULL,
    modalidad     VARCHAR(30) NOT NULL,
    codigo_sniese VARCHAR(20) NULL,
    duracion_sem TINYINT   NOT NULL
);

CREATE TABLE site_querochaca.carreras (
    carrera_id    INT      NOT NULL PRIMARY KEY,
    nombre        VARCHAR(100) NOT NULL,
    nivel         VARCHAR(30) NOT NULL,
    modalidad     VARCHAR(30) NOT NULL,
    codigo_sniese VARCHAR(20) NULL,
    duracion_sem TINYINT   NOT NULL
);

-- Semillas (idénticas en los 3 sitios)

INSERT INTO site_huachi.carreras VALUES
(1,'Ingeniería en Sistemas',    'Pregrado','Presencial','UTA-IS',10),
(2,'Ingeniería Civil',          'Pregrado','Presencial','UTA-IC',10),

```

```
(3,'Administración de Empresas', 'Pregrado','Híbrida',  'UTA-AE',8),
(4,'Enfermería',           'Pregrado','Presencial','UTA-ENF',10);
```

```
INSERT INTO site_ingahurco.carreras SELECT * FROM site_huachi.carreras;
```

```
INSERT INTO site_querochaca.carreras SELECT * FROM site_huachi.carreras;
```

```
-- Extensiones (fragmentación horizontal: una fila por sede)
```

```
CREATE TABLE site_huachi.extensiones (
    extension_id INT      NOT NULL PRIMARY KEY, -- 100-199
    nombre        VARCHAR(120) NOT NULL,
    campus        VARCHAR(40) NOT NULL,          -- Huachi / Ingahurco
    / Querochaca
    ciudad        VARCHAR(60) NOT NULL,
    direccion     VARCHAR(160) NULL,
    telefono      VARCHAR(30) NULL
);
```

```
CREATE TABLE site_ingahurco.extensiones (
    extension_id INT      NOT NULL PRIMARY KEY, -- 200-299
    nombre        VARCHAR(120) NOT NULL,
    campus        VARCHAR(40) NOT NULL,
    ciudad        VARCHAR(60) NOT NULL,
    direccion     VARCHAR(160) NULL,
    telefono      VARCHAR(30) NULL
);
```

```
CREATE TABLE site_querochaca.extensiones (
    extension_id INT      NOT NULL PRIMARY KEY, -- 300-399
    nombre        VARCHAR(120) NOT NULL,
    campus        VARCHAR(40) NOT NULL,
    ciudad        VARCHAR(60) NOT NULL,
    direccion     VARCHAR(160) NULL,
    telefono      VARCHAR(30) NULL
);
```

```
);
```

```
-- Semillas por sede
```

```
INSERT INTO site_huachi.extensiones VALUES  
(101,'Universidad Técnica de Ambato - Campus Huachi','Huachi','Ambat  
o','Av. Los Chasquis s/n','(03) 299-0001');
```

```
INSERT INTO site_ingahurco.extensiones VALUES  
(201,'Universidad Técnica de Ambato - Campus Ingahurco','Ingahurc  
o','Ambato','Av. Colombia y Chile','(03) 299-0002');
```

```
INSERT INTO site_querochaca.extensiones VALUES  
(301,'Universidad Técnica de Ambato - Campus Querochaca','Querocha  
ca','Cevallos','Vía a Quero km 2','(03) 299-0003');
```

```
-- Oferta de carreras (fragmentada por extensión/sitio)
```

```
CREATE TABLE site_huachi.oferta_carrera (  
    oferta_id    INT      NOT NULL PRIMARY KEY,  
    extension_id INT      NOT NULL,  
    carrera_id   INT      NOT NULL,  
    jornada      VARCHAR(20) NOT NULL, -- Matutina/Vespertina/Noctur  
na  
    cupo         INT      NOT NULL,  
    estado       VARCHAR(15) NOT NULL, -- Activa/Inactiva  
    CONSTRAINT FK_ofh_ext FOREIGN KEY (extension_id) REFERENCES  
site_huachi.extensiones(extension_id),  
    CONSTRAINT FK_ofh_car FOREIGN KEY (carrera_id) REFERENCES  
site_huachi.carreras(carrera_id)  
);
```

```
CREATE TABLE site_ingahurco.oferta_carrera (  
    oferta_id    INT      NOT NULL PRIMARY KEY,  
    extension_id INT      NOT NULL,  
    carrera_id   INT      NOT NULL,  
    jornada      VARCHAR(20) NOT NULL,
```

```
    cupo      INT      NOT NULL,  
    estado    VARCHAR(15) NOT NULL,  
    CONSTRAINT FK_ofi_ext FOREIGN KEY (extension_id) REFERENCES  
    site_ingahurco.extensiones(extension_id),  
    CONSTRAINT FK_ofi_car FOREIGN KEY (carrera_id) REFERENCES si  
    te_ingahurco.carreras(carrera_id)  
);
```

```
CREATE TABLE site_querochaca.oferta_carrera (  
    oferta_id   INT      NOT NULL PRIMARY KEY,  
    extension_id INT      NOT NULL,  
    carrera_id  INT      NOT NULL,  
    jornada     VARCHAR(20) NOT NULL,  
    cupo        INT      NOT NULL,  
    estado      VARCHAR(15) NOT NULL,  
    CONSTRAINT FK_ofq_ext FOREIGN KEY (extension_id) REFERENCES  
    site_querochaca.extensiones(extension_id),  
    CONSTRAINT FK_ofq_car FOREIGN KEY (carrera_id) REFERENCES  
    site_querochaca.carreras(carrera_id)  
);
```

```
-- Semillas de oferta por sede
```

```
INSERT INTO site_huachi.oferta_carrera VALUES  
(1101,101,1,'Matutina',60,'Activa'),  
(1102,101,3,'Nocturna',40,'Activa');
```

```
INSERT INTO site_ingahurco.oferta_carrera VALUES  
(2101,201,2,'Matutina',50,'Activa'),  
(2102,201,1,'Vespertina',45,'Activa');
```

```
INSERT INTO site_querochaca.oferta_carrera VALUES  
(3101,301,4,'Vespertina',35,'Activa'),  
(3102,301,3,'Nocturna',30,'Activa');  
GO
```

```
/* ======  
=====
```

```
CAPA GLOBAL – Transparencia de acceso/localización/fragmentación
```

```
=====
===== */
```

```
-- Unifica extensiones (fragmentación H)
```

```
CREATE VIEW global.extensiones AS
    SELECT * FROM site_huachi.extensiones
    UNION ALL
    SELECT * FROM site_ingahurco.extensiones
    UNION ALL
    SELECT * FROM site_querochaca.extensiones;
GO
```

```
-- Unifica oferta (fragmentación H)
```

```
CREATE VIEW global.oferta_carrera AS
    SELECT * FROM site_huachi.oferta_carrera
    UNION ALL
    SELECT * FROM site_ingahurco.oferta_carrera
    UNION ALL
    SELECT * FROM site_querochaca.oferta_carrera;
GO
```

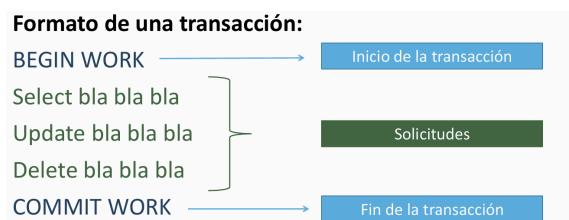
```
-- Carreras replicada: elegimos copia de HUACHI como autoritativa
```

```
CREATE VIEW global.carreras AS
    SELECT * FROM site_huachi.carreras;
GO
```

Transparencia de Transacción

Permite que una transacción actualice datos en varios sitios de la red garantizando la integridad de los datos.

Una transacción es un proceso (conjunto de operaciones).



Solicitud Remota, solicitada en A, pero
hecha en B.

Se valida al ejecutar la sentencia y viendo que se devuelvan los datos correctos.

Transacción No Distribuida:	Transacción Distribuida:	Transacción Remota:
Se ejecuta en un solo nodo y los datos están en la misma BD local.	Actualiza y solicita datos de varios sitios (nodos) remotos en una RED.	Compuesta de varias solicitudes remotas, pero la transacción se hace en LOCAL.

- **Solicitud remota:** Una única operación SQL que se ejecuta en un nodo diferente.
 - **Transacción remota:** Conjunto de varias operaciones (lectura/escritura), todas ejecutadas en un nodo remoto.
 - **Solicitud distribuida:** Es una consulta que accede a datos en varios sitios.
 - **Transacción distribuida:** es un conjunto de operaciones (lectura/escritura) que deben cumplirse atómicamente en varios sitios.

La conexión remota se hace mediante la solicitud del cliente a un sitio A , pero los datos requeridos están en el sitio B por ello el sitio B devuelve los datos al sitio A y este sitio A le devuelve al cliente.



Una base de datos distribuidas es cuando existe transparencia de los datos.

Transparencia de Replicación

El acceso a los datos se percibe como único y consistente, aunque haya varias copias.

- Se refiere a que el usuario no debe preocuparse por cuántas copias de un mismo dato existen ni dónde están almacenadas.
- El sistema se encarga de mantener las copias actualizadas y consistentes entre los distintos nodos.

Transparencia de Falla

El usuario no nota la falla de un nodo y las operaciones continúan normalmente.

- Permite que el sistema siga funcionando aunque un nodo falle.
- Las funciones y los datos que se pierden en el nodo afectado son recuperados automáticamente por otros nodos de la red.



Consumo de recursos es una desventaja de la replicación.

Transparencia de Desempeño

Permite que cuando los Objetos de la BD estén fragmentados o replicados mantenga un rendimiento eficiente optimizando la ejecución de consultas o transacciones.

Optimización de consultas: Reducir al MINIMO el costo TOTAL al ejecutar una solicitud, buscando una ruta eficiente en:

- Tiempo de acceso (E/S) a los datos
- Comunicación en transmisión de datos entre nodos
- Procesamiento de manejar transacciones distribuidas

Optimización de consulta automática: el DDBMS localiza la ruta de acceso más barata sin la intervención del usuario.

Optimización de consulta manual: requiere que la optimización sea seleccionada y programada por el usuario o programador.

Optimización de consulta estática ⇒ Ocurre cuando la consulta es compilada.

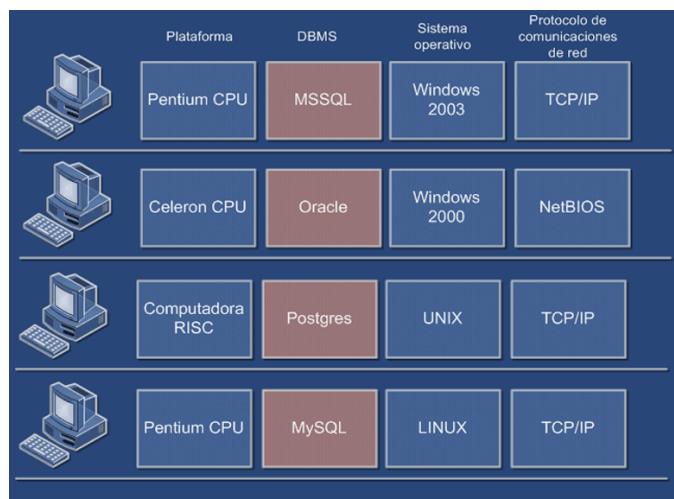
Optimización de consulta dinámica ⇒ Ocurre en tiempo de ejecución al ejecutar el programa.

Técnicas de optimización:

- Basado en estadísticas
- Basado en reglas

Transparencia de Heterogeneidad

Permite la integración de varios sistemas de administración de bases de datos locales diferentes (relacional, de red, jerárquicos, multimedia, etc.) conforme un esquema global común y que funcione de manera uniforme.



Leer artículo: Fragmentación de datos en bases de datos distribuidas

Semana 3

08/Sep/2025

⇒ Preguntas

1. En UTA_Distribuida, una consulta a global.extensions devuelve filas de Huachi, Ingahurco y Querochaca sin que el usuario conozca dónde vive cada fragmento. ¿Qué tipo de transparencia es la principal en juego?

Transparencia de Distribución

2. global.oferta_carrera une (UNION ALL) las tablas site_* .oferta_carrera y el usuario no escribe manualmente múltiples SELECT por sede. ¿Qué nivel de transparencia se ilustra?

Transparencia de Ubicación

3. Para simplificar consultas, se crean sinónimos carreras, extensiones, oferta que apuntan a las vistas globales; así el usuario no necesita prefijos de esquema. ¿Qué transparencia describe mejor este caso?

Heterogeneidad

4. Una transacción inserta oferta en Querochaca y actualiza cupos en Huachi como una unidad atómica; si falla un sitio se revierte todo. ¿Qué transparencia se busca demostrar?

Transacción

Arquitectura de los SABDD (DDBMS)

La Arquitectura de los Sistemas Manejadores de Bases de Datos ANSI-SPARC divide a un sistema en tres niveles: **interno, conceptual y externo**.

El comité ANSI-SPARC (American National Standard Institute- Standards Planning and Requirements Committee) propuso una arquitectura de tres niveles para los sistemas de bases de datos por los años 1975.

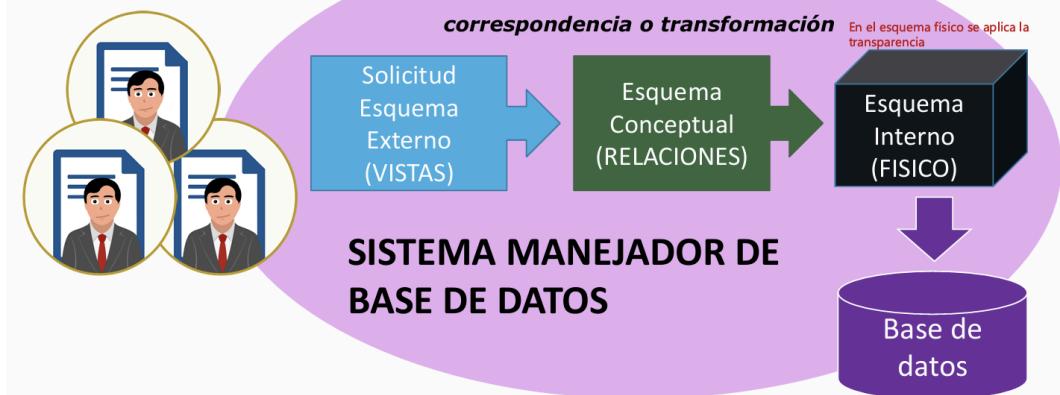
Esquema interno → Esquema Físico

Esquema conceptual → Diseño de tablas

Esquema externo → Replicaciones, Consultas, Vistas

- En el nivel interno: Describe la estructura física de la base de datos mediante un esquema interno.
- Nivel conceptual: Se concentra en describir entidades, atributos, relaciones, operaciones de los usuarios y restricciones.
- Nivel externo: Describe varios esquemas externos o vistas de usuario.

Clase 8. Tema 2. Bases de Datos Distribuidas (BDD). Arquitectura ANSI-SPARC de los sistemas de administración de base de datos



PostgreSQL → Relacionales

MongoDB → No relacionales

Arquitectura ANSI/SPARC de un sistema distribuido

Basado en datos. Se identifican los diferentes tipos de descripción de datos.

Define las unidades funcionales que realizarán o usarán los datos de acuerdo con las diferentes vistas.

- El **esquema de fragmentación** describe la forma en que las relaciones globales se dividen entre las bases de datos locales.
- El **esquema de asignamiento** especifica el lugar en el cual cada fragmento es almacenado.



Se agregan al Esquema Conceptual: Esquema de fragmentación y de asignación.

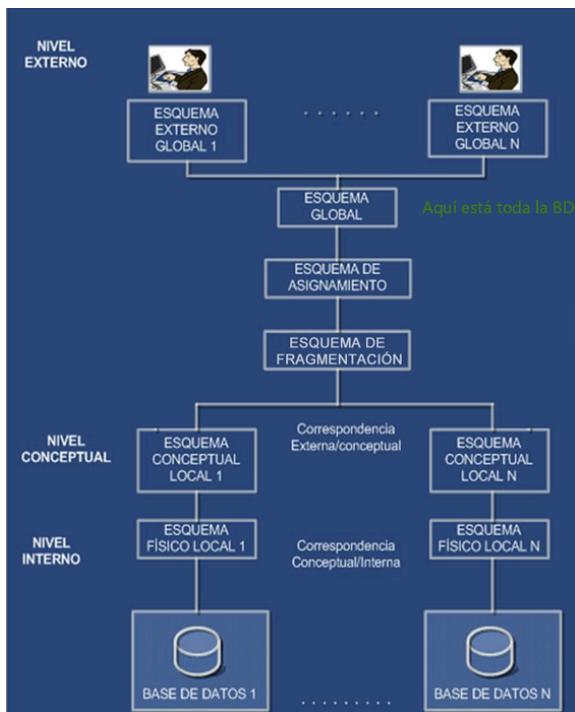
DDBMS Homogéneo

- Tiene múltiples colecciones de datos
- Integra múltiples recursos de datos

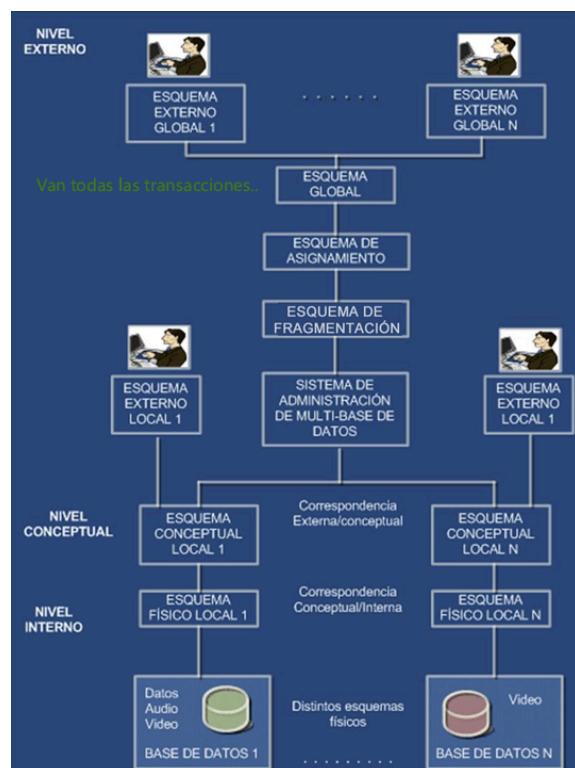
DDBMS Heterogéneo

- Se caracterizan por manejar diferentes sistemas de administración de bases de datos en los nodos locales.

- Similar a un centralizado. Datos en varios sitios comunicados por la red
- No existe usuarios locales y todos ellos acceden a la base de datos a través de una interfaz global
- El esquema global es la unión de todas las descripciones de datos locales



- Una subclase son los sistemas de administración de multi-bases de datos.
- Un sistema de multi-bases integra todos ellos mediante subsistemas de software.
- Existen usuarios locales y globales.



Diseño de una base de datos distribuidas

1. Diseño conceptual de la base de datos: Se definen entidades y las relaciones.
2. Diseño lógico de la base de datos: Se transforma el diseño conceptual en un esquema lógico global.
3. Diseño físico de la base de datos: Se determina cómo y dónde se almacenarán los datos.

Objetivos

- Procesamiento local: SO
- Distribución de la carga de trabajo: SGBD
- Costo de almacenamiento y disponibilidad: Red



Que es?

Catalogo de Base de Datos: es un inventario detallado de los activos de datos dentro de una organización. Ayuda a los usuarios a descubrir, comprender, gestionar, seleccionar y acceder fácilmente a los datos.

Diseño físico de la base de datos

1. Como dividir la base de datos en fragmentos
2. Que fragmentos replicar
3. Donde localizar estos fragmentos

El problema de la fragmentación

Se refiere al particionamiento de la información para distribuir cada parte en los diferentes sitios de la red.

- a. Por que hacer una fragmentación de datos?
- b. Como realizar la fragmentación?
- c. Que tanto se debe fragmentar?
- d. Como probar la validez de una fragmentación?
- e. Como realizar el asignamiento de fragmentos?
- f. Como considerar los requerimientos de la información?

La información de la fragmentación de los datos se guarda en un Catalogo de Datos Distribuidos, desde donde es accedida por el procesador de transacciones para procesar las solicitudes de los usuarios.

La tabla se divide en fragmentos logicos:

1. Fragmentación horizontal

2. Fragmentación vertical
3. Fragmentación hibrida/mezclada

Fragmentación horizontal

Se refiere a la división de una relación en fragmentos de tuplas (filas).

Cada fragmento se guarda en un nodo diferente y cada uno de ellos tiene filas únicas.

Todas las filas únicas tienen los mismos atributos(columnas).

Fragmentación vertical

Se refiere a la división de una relación en fragmentos de atributos (columnas).

Cada fragmento se guarda en un nodo diferente, y cada fragmento tiene columnas únicas, excepto la columna clave, la cual es común en todos los fragmentos.

Fragmentación hibrida

Se refiere a la combinación de estrategias horizontales y verticales.

Una tabla puede dividirse en varios fragmentos horizontales y cada una tiene fragmentos de los atributos

Taller envió de archivó de MAQUINA LOCAL a VIRTUAL y viceversa.

| 12/Sep/2025

⇒ Preguntas

1. (Opción múltiple) ¿Qué describe el nivel interno?
 - A) Vistas de usuario y reportes
 - B) Estructura física mediante un esquema interno
 - C) Entidades, atributos y restricciones lógicas
 - D) Reglas de negocio en la aplicación

2. El nivel conceptual se centra en entidades, atributos, relaciones, operaciones de usuario y restricciones.

Verdadero

3. ¿Qué nombre reciben las "vistas de usuario" y en qué nivel se ubican?

Vistas, Externo

4. ¿Cuál de estos enunciados corresponde al nivel externo?

- A) Define índices y estructuras de almacenamiento en disco
- B) Proporciona múltiples esquemas orientados a distintos usuarios
- C) Define claves foráneas a nivel físico
- D) Determina direcciones de bloques en el disco

5. Relacione cada nivel con su enfoque principal:

- I. Interno ————— a) Vistas de usuario
- II. Conceptual —— b) Estructura física
- III. Externo ——— c) Modelo lógico global

1b, 2 c,3a

4. En qué nivel se documentan "entidades, atributos, relaciones y restricciones" del modelo global?

- A) Externo
- B) Conceptual
- C) Interno
- D) Físico-operativo

7. Menciona dos ejemplos de decisiones típicas del nivel interno.

Como dividir la BD en fragmentos

Donde Localizar los fragmentos

8. El nivel externo presenta un único esquema para todos los usuarios.

Falso

9. ¿Qué afirmación es correcta respecto a la separación entre niveles?

- A) Los usuarios finales acceden directamente al nivel interno
- B) El nivel conceptual actúa como puente entre externo e interno

- C) El nivel externo controla los buffers de disco
D) El nivel interno define las vistas de seguridad lógicas
10. Caso breve) Una analista necesita una vista con solo Cliente(nombre, ciudad) para un grupo comercial, sin modificar la base física.
- ¿Qué nivel crea esa vista?
 - ¿Qué nivel permanece estable aunque cambien índices y archivos?

Externo, Conceptual



En conexiones remotas hacia una tabla temporal se hace con FROM



Esquema conceptual: cuando se presenta el diagrama de tablas.

Fundamentos de Subconsultas SQL

RESUMEN: Una subconsulta SQL es una sentencia **SELECT** incrustada en otra **sentencia SELECT**. Podemos considerar las subconsultas como bloques que conforman consultas complejas que nos permiten descomponer tareas complejas en partes más pequeñas y simplificar la lectura del código.

Casos de uso comunes para subconsultas SQL:

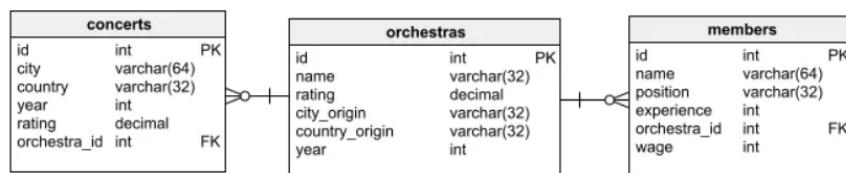
- **Filtrado de datos :** Utilice subconsultas en la cláusula **WHERE** para filtrar datos según condiciones específicas, lo que hará que sus consultas sean más dinámicas. Se explica en los ejercicios prácticos de subconsultas 1, 2, 3, 4, 8 y 9.
- **Agregaciones anidadas :** Utiliza subconsultas para realizar agregaciones dentro de otras agregaciones, lo que permite cálculos más complejos. Se aborda en los ejercicios prácticos de subconsultas 5, 6 y 7.
- **Comprobación de existencia:** Determinar si un valor específico existe en otra tabla utilizando subconsultas con **EXIST** o **IN**. Tratado en los ejercicios de práctica de subconsultas 1, 2 y 14.
- **Subconsultas correlacionadas :** Crea subconsultas que hacen referencia a columnas de la consulta externa, lo que permite el filtrado contextual. Se aborda en los ejercicios prácticos de subconsultas 10, 11, 12 y 13.

- **Subconsulta en SELECT la cláusula** : Incluya una subconsulta en la cláusula SELECT para recuperar un valor único o un conjunto de valores que puedan usarse en la consulta principal. Se aborda en los ejercicios de práctica de subconsultas 10 y 13.
- **Subconsulta en FROM la cláusula** : Use una subconsulta en la cláusula FROM para crear una tabla temporal, lo que permite uniones más complejas. Se explica en los ejercicios de práctica de subconsultas 14 y 15.

Taller

Realizar la consulta y una discusión sobre fragmentación horizontal, vertical o hibrida.

▼ Base de datos Orquesta



-- BD

```
create database Orquesta;
```

```
create table orchestras (
    id int primary key,
    name varchar(32) not null,
    rating decimal not null,
    city_origin varchar(32) not null,
    country_origin varchar(32) not null,
    year int not null);
```

```
create table members (
    id int primary key,
    name varchar(64) not null,
    position varchar(32) not null,
    experience int not null,
    orchestra_id int not null references orchestras(id),
```

```
wage int not null);

create table concerts (
id int primary key,
city varchar(64) not null,
country varchar(32) not null,
year int not null,
rating decimal not null,
orchestra_id int not null references orchestra(id));
```

Inserciones

```
-- 5 ORCHESTRAS
INSERT INTO orchestras (id, name, rating, city_origin, country_origin, year) VALUES
(1, 'Sinfonietta Quito', 4.5, 'Quito', 'Ecuador', 1990),
(2, 'Filarmónica Guayaquil', 4.8, 'Guayaquil', 'Ecuador', 1985),
(3, 'Orquesta Andes', 4.2, 'Cuenca', 'Ecuador', 2000),
(4, 'Orquesta Nacional de Quito', 4.9, 'Quito', 'Ecuador', 1975),
(5, 'Orquesta del Pacífico', 4.3, 'Guayaquil', 'Ecuador', 1995);

-- 20 MEMBERS distribuidos entre 5 orquestas
INSERT INTO members (id, name, position, experience, orchestra_id, wage) VALUES
(1, 'Juan Pérez', 'Violin', 10, 1, 1500),
(2, 'Ana Torres', 'Cello', 12, 1, 1600),
(3, 'Carlos López', 'Flute', 8, 1, 1400),
(4, 'María Fernández', 'Viola', 15, 1, 1700),
(5, 'Luis García', 'Trumpet', 7, 2, 1300),
(6, 'Sofía Castro', 'Timpani', 9, 2, 1350),
(7, 'David Morales', 'Oboe', 11, 2, 1550),
(8, 'Lucía Ramos', 'Clarinet', 13, 2, 1650),
(9, 'Enrique Ruiz', 'Bass', 6, 3, 1250),
(10, 'Carla Gómez', 'Horn', 8, 3, 1400),
(11, 'Fernando Díaz', 'Violin', 14, 3, 1750),
(12, 'Valentina Rojas', 'Cello', 10, 3, 1500),
(13, 'Miguel Suárez', 'Flute', 9, 4, 1450),
(14, 'Isabel León', 'Trombone', 7, 4, 1300),
```

```
(15, 'Diego Paredes', 'Percussion', 12, 4, 1600),  
(16, 'Camila Ortega', 'Harp', 6, 4, 1200),  
(17, 'José Cabrera', 'Violin', 15, 5, 1800),  
(18, 'Natalia Vega', 'Clarinet', 11, 5, 1550),  
(19, 'Ricardo Andrade', 'Trumpet', 13, 5, 1650),  
(20, 'Paula Méndez', 'Oboe', 9, 5, 1450);
```

-- 5 CONCERTS, uno por cada orquesta

```
INSERT INTO concerts (id, city, country, year, rating, orchestra_id) VALUES
```

```
(1, 'Quito', 'Ecuador', 2023, 4.5, 1),  
(2, 'Guayaquil', 'Ecuador', 2023, 4.6, 2),  
(3, 'Cuenca', 'Ecuador', 2023, 4.2, 3),  
(4, 'Quito', 'Ecuador', 2023, 4.8, 4),  
(5, 'Guayaquil', 'Ecuador', 2023, 4.3, 5);
```

```
select * from orchestras;
```

```
select * from members;
```

```
select * from concerts;
```

Ejercicio 1: Seleccionar orquestas con ciudad de origen donde se realizó un concierto en 2013

Seleccione los nombres de todas las orquestas que tienen la misma ciudad de origen que cualquier ciudad en la que alguna orquesta actuó en 2013.

```
select name from orchestras  
where city_origin = (select city from concerts  
where year = 2013);
```

Explicación de la solución:

Nuestro objetivo es seleccionar nombres de orquestas que cumplan una condición, por lo que comenzamos con SELECT name FROM orchestras. Luego, la condición se aplicará a la city_origin columna, como se indica en las instrucciones.

Queremos seleccionar sólo las orquestas cuya ciudad de origen pertenece al grupo de ciudades donde se realizaron conciertos en el año 2013. Para crear esta condición en la WHERE cláusula, utilizamos la subconsulta SQL.

Creemos una (sub)consulta que seleccione todas las ciudades donde se realizaron conciertos en 2013: SELECT city FROM concerts WHERE year = 2013. Devuelve una columna que contiene los nombres de las ciudades.

Para garantizar que la ciudad de origen pertenezca a las ciudades devueltas por la subconsulta, utilizamos el IN operador .

Ejercicio 2: Seleccionar miembros que pertenezcan a orquestas de alto nivel

Seleccione los nombres y posiciones (es decir, instrumento tocado) de todos los miembros de la orquesta que tengan más de 10 años de experiencia y que no pertenezcan a orquestas con una calificación inferior a 8.0.

```
select name, position from members
where experience > 10 and
orchestra_id not in (select id from orchestras
where rating <8);
```

Ejercicio 3: Seleccionar miembros que ganen más que los violinistas

Ejercicio: Mostrar el nombre y puesto de los miembros de la orquesta que ganan más que el salario promedio de todos los violinistas.

```
select name, position from members
where wage > (select avg(wage) from members
where position = 'Violin');
```

Ejercicio 4: Seleccione orquestas de alta calificación más nuevas que la orquesta de cámara Ejercicio: Mostrar los nombres de las orquestas que se crearon después de la 'Orquesta de Cámara' y que tienen una calificación mayor a 7.5.

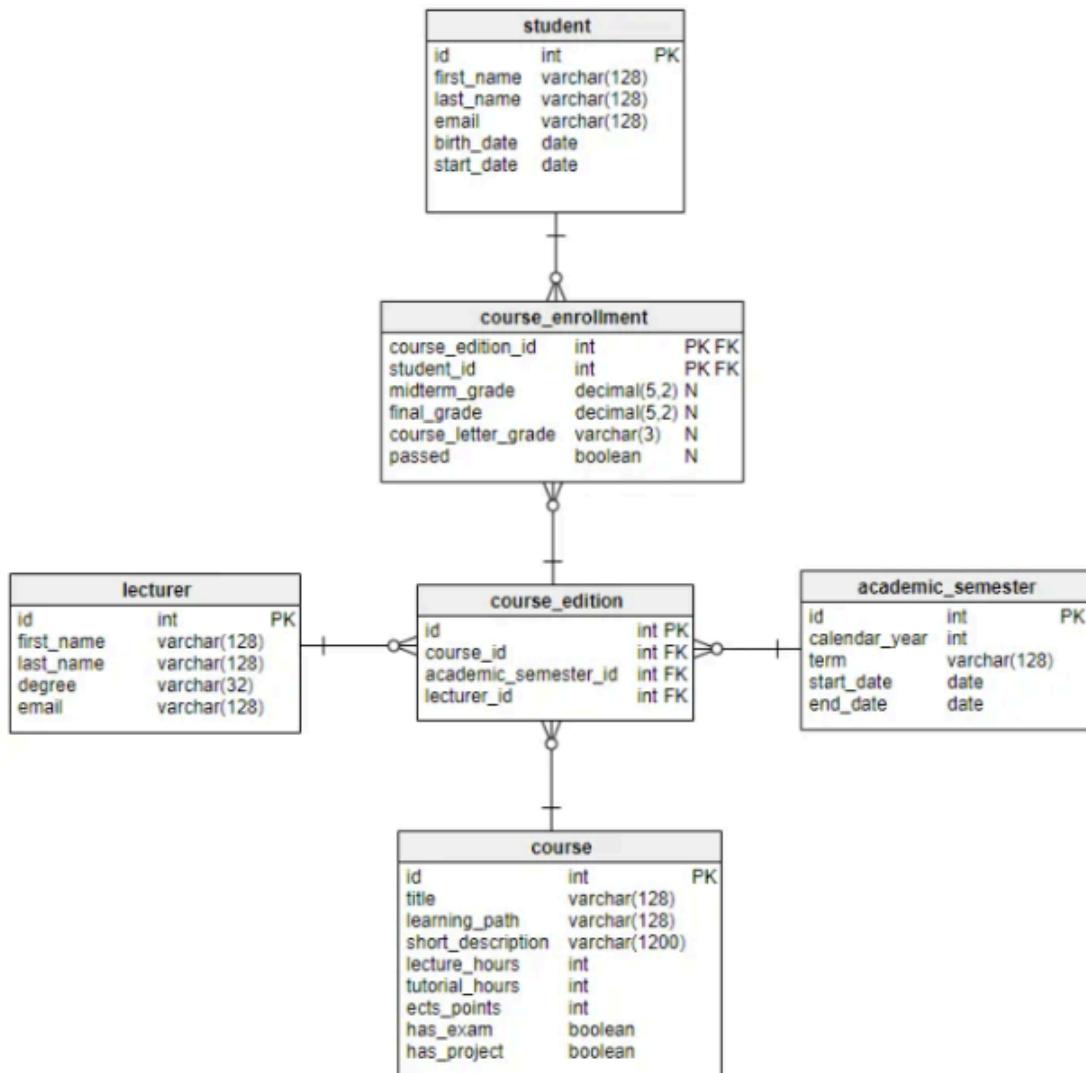
```
select name from orchestras
where year > (select year from orchestras
where name = 'Orquesta de Cámara') and
rating > 7.5;
```

Ejercicio 5: Seleccionar intérpretes en grandes orquestas Ejercicio :Muestre el nombre y el número de miembros de cada orquesta que tenga más

miembros que el promedio de miembros de todas las orquestas en la tabla.

```
select o.name, count(o.id) num_mem
from orchestras o
join members m on o.id = m.orchestra_id
group by o.name
having count(o.id) > (select avg(cant_mem) prom from
(select count(*) cant_mem from members
group by orchestra_id));
```

▼ Base de datos Universidad



--BD

```
create database universidad
```

```
create table student (
    id int primary key,
    first_name varchar(128) not null,
    last_name varchar(128) not null,
    email varchar(128) not null,
    birth_date date not null,
    start_date date not null);

create table academic_semester (
    id int primary key,
    calendar_year int not null,
    term varchar(128) not null,
    start_date date not null,
    end_date date not null);

create table course (
    id int primary key,
    title varchar (128) not null,
    learning_path varchar(128) not null,
    short_description varchar(1200),
    lecture_hours int not null,
    tutorial_hours int not null,
    ects_points int not null,
    has_exam boolean not null,
    has_project boolean not null);

create table lecturer (
    id int primary key,
    first_name varchar(128) not null,
    last_name varchar(128) not null,
    degreee varchar(32) not null,
    email varchar(128));

create table course_edition (
    id int primary key,
    course_id int not null references course(id),
    academic_semester_id int not null references academic_semester(id),
```

```
lecturer_id int not null references lecturer(id));  
  
create table course_enrollment (  
course_edition_id int not null references course_edition(id),  
student_id int not null references student(id),  
midterm_grade decimal(5,2),  
final_grade decimal(5,2),  
course_letter_grade varchar(3),  
passed boolean);
```

Inserciones:

```
-- SEMESTRES  
INSERT INTO academic_semester (id, calendar_year, term, start_date, end_date) VALUES  
(1, 2024, 'Spring', '2024-01-15', '2024-06-30'),  
(2, 2024, 'Fall', '2024-08-15', '2024-12-20'),  
(3, 2025, 'Spring', '2025-01-15', '2025-06-30'),  
(4, 2025, 'Fall', '2025-08-15', '2025-12-20');  
  
-- CURSOS  
INSERT INTO course (id, title, learning_path, short_description, lecture_hours, tutorial_hours, ects_points, has_exam, has_project) VALUES  
(1, 'Databases', 'Computer Science', 'Intro to relational databases', 40, 20, 6, true, false),  
(2, 'Algorithms', 'Computer Science', 'Data structures and algorithms', 50, 20, 7, true, true),  
(3, 'Networks', 'Computer Science', 'Basics of computer networks', 40, 15, 5, true, false),  
(4, 'Math I', 'Mathematics', 'Linear algebra and calculus', 60, 30, 8, true, false),  
(5, 'AI Fundamentals', 'Computer Science', 'Introduction to AI', 40, 20, 6, true, true);  
  
-- PROFESORES  
INSERT INTO lecturer (id, first_name, last_name, degreee, email) VALUES  
(1, 'Alice', 'Smith', 'PhD', 'alice.smith@uni.edu'),
```

```
(2, 'Bob', 'Johnson', 'MSc', 'bob.johnson@uni.edu'),  
(3, 'Clara', 'Brown', 'PhD', 'clara.brown@uni.edu');
```

-- ESTUDIANTES

```
INSERT INTO student (id, first_name, last_name, email, birth_date, start  
_date) VALUES  
(1, 'Juan', 'Perez', 'juan.perez@uni.edu', '2002-03-10', '2021-09-01'),  
(2, 'Maria', 'Lopez', 'maria.lopez@uni.edu', '2001-07-20', '2021-09-01'),  
(3, 'Carlos', 'Ramirez', 'carlos.ramirez@uni.edu', '2003-11-05', '2022-09  
-01'),  
(4, 'Ana', 'Torres', 'ana.torres@uni.edu', '2002-05-14', '2021-09-01'),  
(5, 'Diego', 'Morales', 'diego.morales@uni.edu', '2000-09-22', '2020-0  
9-01');
```

-- EDICIONES DE CURSO (ligadas a semestre y profesor)

```
INSERT INTO course_edition (id, course_id, academic_semester_id, lect  
urer_id) VALUES  
(1, 1, 1, 1), -- Databases en Spring 2024 con Alice  
(2, 2, 1, 2), -- Algorithms en Spring 2024 con Bob  
(3, 3, 2, 2), -- Networks en Fall 2024 con Bob  
(4, 4, 3, 3), -- Math I en Spring 2025 con Clara  
(5, 5, 4, 1); -- AI Fundamentals en Fall 2025 con Alice
```

-- INSCRIPCIONES (con notas y si pasó o no)

```
INSERT INTO course_enrollment (course_edition_id, student_id, midter  
m_grade, final_grade, course_letter_grade, passed) VALUES  
(1, 1, 70, 'B', true), -- Juan aprobó Databases  
(1, 2, 60, 50, 'D', false), -- Maria reprobó Databases  
(1, 3, 85, 90, 'A', true), -- Carlos aprobó  
(2, 1, 55, 65, 'C', true), -- Juan aprobó Algorithms  
(2, 2, 40, 45, 'F', false), -- Maria reprobó Algorithms  
(2, 4, 88, 92, 'A', true), -- Ana aprobó  
(3, 5, 75, 80, 'B', true), -- Diego aprobó Networks  
(3, 3, 50, 55, 'D', false), -- Carlos reprobó Networks  
(4, 2, 90, 95, 'A', true), -- Maria aprobó Math I  
(4, 4, 65, 70, 'C', true), -- Ana aprobó  
(4, 5, 45, 50, 'E', false), -- Diego reprobó
```

```
(5, 1, 78, 82, 'B', true), -- Juan aprobó A!  
(5, 3, 92, 96, 'A', true); -- Carlos aprobó A!
```

Ejercicio 6: Seleccionar cursos del semestre de primavera. Muestra los ID y títulos de todos los cursos que se llevaron a cabo durante cualquier período de primavera.

```
select id, title from course  
where id in (select course_id from course_edition  
    where academic_semester_id in (select id from academic_semester  
        where term = 'Spring'));
```

Ejercicio 7: Seleccionar todos los estudiantes que aprobaron al menos un curso. Seleccione los ID y nombres de los estudiantes que aprobaron al menos un curso.

```
-- consulta simple  
select id, first_name, last_name from student  
where id in (select student_id from course_enrollment  
    where passed = true);  
-- consulta mas detallada  
select id, first_name, last_name from student  
where id in (select student_id from course_enrollment  
    group by student_id  
    having count(passed)>1);
```

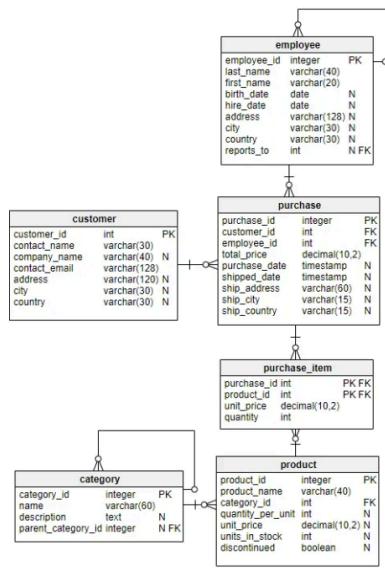
Ejercicio 8: Seleccione el/los profesor(es) que imparten menos cursos. Encuentra al/los profesor(es) con menos cursos impartidos. Muestra su nombre y apellidos, así como el número de cursos que imparte (como no_of_courses).

```
select l.first_name, l.last_name, count(ce.id) no_of_courses  
from lecturer l  
join course_edition ce on l.id = ce.lecturer_id  
group by l.id  
having count (ce.id) = (select min(num_cur)  
    from (select count(id) num_cur from course_edition  
        group by lecturer_id) );
```

Ejercicio 9: Seleccionar estudiantes matriculados en la mayor cantidad de cursos. Encuentra a los estudiantes matriculados en el mayor número de ediciones del curso. Muestra su ID, nombre y apellidos, y el número de ediciones del curso en las que ha estado matriculado (como no_of_course_ed).

```
select e.id, e.first_name, e.last_name, count(ce.course_edition_id) no_o
f_course_ed
from student e
join course_enrollment ce on e.id = ce.student_id
group by e.id
having count(ce.course_edition_id) = (select max(num_cur) from
(select count(course_edition_id) num_cur
from course_enrollment
group by student_id));
```

▼ Base de datos Tienda



```
-- BD
create database tienda_almacen;

-- Importante revisar la conexion
create table category (
    category_id int primary key,
    name varchar(60) not null,
```

```
description text,  
parent_category_id int,  
foreign key (parent_category_id) references category(category_id)  
);
```

```
create table product (  
product_id int primary key,  
product_name varchar(40) not null,  
category_id int not null references category(category_id),  
quantity_per_unit int,  
unit_price decimal(10,2),  
units_in_stock int,  
discontinued boolean  
);
```

```
create table employee (  
employee_id int primary key,  
last_name varchar(40) not null,  
first_name varchar(20) not null,  
birth_date date,  
hire_date date,  
address varchar(128),  
city varchar(30),  
country varchar(30),  
reports_to int references employee(employee_id)  
);
```

```
create table customer (  
customer_id int primary key,  
contact_name varchar(30) not null,  
company_name varchar(40),  
contact_email varchar(128) not null,  
address varchar(120),  
city varchar(30),  
country varchar(30)  
);
```

```
create table purchase (
```

```

purchase_id int primary key,
customer_id int not null references customer(customer_id),
employee_id int not null references employee(employee_id),
total_price decimal(10,2) not null,
purchase_date timestamp,
shipped_date timestamp,
ship_address varchar(60),
ship_city varchar(15),
ship_country varchar(15)
);

create table purchase_item (
purchase_id int not null references purchase(purchase_id),
product_id int not null references product(product_id),
unit_price decimal(10,2) not null,
quantity int not null
);

```

Inserciones:

```

INSERT INTO employee (employee_id, last_name, first_name, birth_date, hire_date, reports_to) VALUES
(1, 'García', 'Juan', '1985-05-15', '2010-03-01', NULL),
(2, 'Pérez', 'Ana', '1990-08-20', '2015-06-10', 1),
(3, 'López', 'Carlos', '1988-11-25', '2012-09-15', 1),
(4, 'Rodríguez', 'María', '1992-02-12', '2018-01-22', 2),
(5, 'Fernández', 'Luis', '1987-07-30', '2014-04-05', 3),
(6, 'Díaz', 'Paola', '1995-03-01', '2020-05-20', 1),
(7, 'Sánchez', 'Andrés', '1993-09-19', '2019-11-11', 2),
(8, 'Romero', 'Julia', '1989-12-05', '2016-08-28', 3),
(9, 'Suárez', 'Pedro', '1991-04-18', '2017-07-07', 4),
(10, 'Castro', 'Gabriela', '1996-06-22', '2021-02-14', 5),
(11, 'Navarro', 'Jorge', '1984-01-30', '2013-09-01', 1),
(12, 'Jiménez', 'Sofía', '1997-07-03', '2022-03-10', 6),
(13, 'Ruiz', 'Daniel', '1994-11-14', '2019-06-25', 7),
(14, 'Ortiz', 'Valeria', '1998-08-08', '2023-01-05', 8),
(15, 'Mendoza', 'Ricardo', '1990-10-21', '2018-04-12', 9),
(16, 'Guerrero', 'Natalia', '1992-05-09', '2017-10-30', 10),

```

```
(17, 'Silva', 'Alejandro', '1995-02-28', '2020-11-19', 11),
(18, 'Vargas', 'Lucía', '1999-09-11', '2023-08-21', 12),
(19, 'Morales', 'Eduardo', '1994-04-04', '2019-03-03', 13),
(20, 'Torres', 'Cristina', '1996-12-16', '2021-06-06', 14);
```

```
INSERT INTO customer (customer_id, contact_name, company_name, contact_email, address, city, country) VALUES
(101, 'Martín Ruiz', 'Innovación Digital S.A.', 'martin.r@innova.com', 'Call e Mayor 123', 'Madrid', 'Spain'),
(102, 'Elena Torres', 'Tech Solutions Ltda.', 'elena.t@techsol.cl', 'Av. Libe rtador 456', 'Santiago', 'Chile'),
(103, 'Jorge Vidal', 'Grupo Empresarial S.A.', 'jorge.v@ge.com', 'Avenida Central 789', 'Bogotá', 'Colombia'),
(104, 'Sofía Castro', 'Logística Global S.L.', 'sofia.c@logistica.es', 'Paseo del Sol 10', 'Barcelona', 'Spain'),
(105, 'David Ramos', 'Negocios Unidos', 'david.r@negociosunidos.com', 'Calle de la Luna 5', 'CDM', 'Mexico'),
(106, 'Lorena Gómez', 'Comercializadora del Sur', 'l.gomez@cosur.co m', 'Carrera 7, 34-56', 'Medellín', 'Colombia'),
(107, 'Hugo Salazar', 'Distribuciones Norte', 'hugo.s@distribunorte.cl', 'A v. Providencia 200', 'Valparaíso', 'Chile'),
(108, 'Carla Ibáñez', 'Market Place', 'carla.i@market.es', 'Plaza de Catal uña 1', 'Sevilla', 'Spain'),
(109, 'Felipe Mendoza', 'Soluciones Tecnológicas MX', 'felipe.m@solute c.mx', 'Insurgentes Sur 300', 'Guadalajara', 'Mexico'),
(110, 'Andrea Ríos', 'Todo para la Oficina', 'andrea.r@todoofi.com', 'Call e 50, 2-10', 'Bogotá', 'Colombia'),
(111, 'Pablo Giménez', 'Futuro Digital', 'pablo.g@futuro.es', 'Gran Vía 5', 'Madrid', 'Spain'),
(112, 'Laura Soto', 'Global Express', 'laura.s@globalexp.com', 'Calle Larg a 10-20', 'Quito', 'Ecuador'),
(113, 'Roberto Pinto', 'Innovación en Productos', 'roberto.p@innovaprod. com', 'Avenida Brasil 500', 'Lima', 'Peru'),
(114, 'Diana Castillo', 'Compu Mundo', 'diana.c@compumundo.com', '25 de Mayo 1000', 'Buenos Aires', 'Argentina'),
(115, 'Gabriel Orozco', 'Materiales y Suministros', 'gabriel.o@mym.com', 'Libertad 15', 'Montevideo', 'Uruguay'),
(116, 'Mónica López', 'Electro Hogar', 'monica.l@electro.com', 'Avenida
```

```
Central 345', 'San José', 'Costa Rica'),  
(117, 'Ricardo Paz', 'El Descuento', 'ricardo.p@eldescuento.com', 'Calle  
del Águila 88', 'Panamá', 'Panama'),  
(118, 'Ana Pérez', 'Tecno Store', 'ana.p@tecnostore.com', 'Calle 5ta, 3-2  
1', 'Guatemala', 'Guatemala'),  
(119, 'Juan Molina', 'Súper Ofertas', 'juan.m@superofertas.com', 'Avenid  
a Las Américas 123', 'Asunción', 'Paraguay'),  
(120, 'Isabel Ramos', 'Servicios Integrales', 'isabel.r@servicios.com', 'R  
ua da Paz, 99', 'Lisboa', 'Portugal');
```

```
INSERT INTO category (category_id, name, description, parent_categor  
y_id) VALUES
```

```
(1, 'Electrónicos', 'Dispositivos y gadgets electrónicos.', NULL),  
(2, 'Computadoras', 'Portátiles, de escritorio y accesorios.', 1),  
(3, 'Smartphones', 'Teléfonos inteligentes y sus accesorios.', 1),  
(4, 'Hogar', 'Artículos para el hogar y la cocina.', NULL),  
(5, 'Alimentos', 'Comida y bebidas.', NULL),  
(6, 'Muebles', 'Mobiliario para oficina y hogar.', 4),  
(7, 'Herramientas', 'Herramientas manuales y eléctricas.', NULL),  
(8, 'Jardinería', 'Productos para el cuidado del jardín.', 7),  
(9, 'Lácteos', 'Leche, quesos y yogures.', 5),  
(10, 'Frutas y Verduras', 'Productos frescos del campo.', 5),  
(11, 'Limpieza', 'Productos de limpieza para el hogar.', 4),  
(12, 'Audio', 'Auriculares, altavoces y sistemas de sonido.', 1),  
(13, 'Video', 'Televisores y proyectores.', 1),  
(14, 'Cereales', 'Granos y productos derivados.', 5),  
(15, 'Carnes', 'Productos cárnicos frescos y procesados.', 5),  
(16, 'Accesorios de Cocina', 'Utensilios y gadgets para la cocina.', 4),  
(17, 'Ropa', 'Prendas de vestir para todas las edades.', NULL),  
(18, 'Calzado', 'Todo tipo de zapatos y zapatillas.', NULL),  
(19, 'Libros', 'Libros de ficción, no ficción y académicos.', NULL),  
(20, 'Juguetes', 'Artículos para el entretenimiento de niños.', NULL);
```

```
INSERT INTO product (product_id, product_name, category_id, quantity  
_per_unit, unit_price, units_in_stock, discontinued) VALUES
```

```
(1001, 'Laptop UltraPro', 2, 1, 1200.00, 50, FALSE),  
(1002, 'Smartphone X1', 3, 1, 850.50, 150, FALSE),
```

```

(1003, 'Mouse inalámbrico', 2, 1, 25.00, 300, FALSE),
(1004, 'Cafetera Expresso', 4, 1, 75.99, 80, FALSE),
(1005, 'Snack de avena', 5, 100, 2.50, 200, FALSE),
(1006, 'Altavoz Bluetooth', 12, 1, 59.99, 120, FALSE),
(1007, 'Televisor SmartTV 4K', 13, 1, 650.00, 30, FALSE),
(1008, 'Silla Ergonómica de Oficina', 6, 1, 150.00, 45, FALSE),
(1009, 'Set de destornilladores', 7, 6, 19.50, 200, FALSE),
(1010, 'Leche entera 1L', 9, 1, 1.80, 500, FALSE),
(1011, 'Manzanas Red', 10, 1, 3.20, 150, FALSE),
(1012, 'Limpiador Multiusos', 11, 500, 4.10, 250, FALSE),
(1013, 'Auriculares con cancelación de ruido', 12, 1, 199.99, 90, FALSE),
(1014, 'Monitor curvo 27"', 2, 1, 299.00, 75, FALSE),
(1015, 'Yogur natural', 9, 150, 1.10, 400, FALSE),
(1016, 'Brócoli fresco', 10, 1, 2.30, 85, FALSE),
(1017, 'Aspiradora robot', 11, 1, 350.00, 20, FALSE),
(1018, 'Mesa de centro', 6, 1, 89.99, 60, FALSE),
(1019, 'Sierra eléctrica', 7, 1, 120.50, 40, FALSE),
(1020, 'Filete de salmón', 15, 200, 15.00, 100, FALSE);

```

```

INSERT INTO purchase (purchase_id, customer_id, employee_id, total_price, purchase_date, shipped_date, ship_city, ship_country) VALUES
(1, 101, 2, 1225.00, '2023-10-25 10:00:00', '2023-10-26 14:00:00', 'Madrid', 'Spain'),
(2, 102, 3, 850.50, '2023-10-25 11:30:00', '2023-10-26 15:00:00', 'Santiago', 'Chile'),
(3, 103, 4, 75.99, '2023-10-26 09:00:00', '2023-10-27 12:00:00', 'Bogotá', 'Colombia'),
(4, 104, 2, 2.50, '2023-10-26 14:00:00', '2023-10-27 10:00:00', 'Barcelona', 'Spain'),
(5, 105, 5, 1200.00, '2023-10-27 09:30:00', '2023-10-28 11:00:00', 'CDM', 'Mexico'),
(6, 106, 6, 150.00, '2023-10-28 10:30:00', '2023-10-29 11:00:00', 'Medellín', 'Colombia'),
(7, 107, 7, 650.00, '2023-10-28 12:45:00', '2023-10-29 13:00:00', 'Valparaíso', 'Chile'),
(8, 108, 8, 59.99, '2023-10-29 09:15:00', '2023-10-30 10:30:00', 'Sevilla', 'Spain'),
(9, 109, 9, 19.50, '2023-10-29 14:00:00', '2023-10-30 11:00:00', 'Guadalajara', 'Mexico');

```

```
ajara', 'Mexico'),  
(10, 110, 10, 1.80, '2023-10-30 08:00:00', '2023-10-31 09:00:00', 'Bogotá', 'Colombia'),  
(11, 111, 11, 3.20, '2023-10-30 11:00:00', '2023-10-31 10:00:00', 'Madrid', 'Spain'),  
(12, 112, 12, 4.10, '2023-10-31 09:45:00', '2023-11-01 11:00:00', 'Quito', 'Ecuador'),  
(13, 113, 13, 199.99, '2023-11-01 13:20:00', '2023-11-02 12:00:00', 'Lima', 'Peru'),  
(14, 114, 14, 299.00, '2023-11-01 15:00:00', '2023-11-02 14:00:00', 'Buenos Aires', 'Argentina'),  
(15, 115, 15, 1.10, '2023-11-02 10:00:00', '2023-11-03 10:30:00', 'Montevideo', 'Uruguay'),  
(16, 116, 16, 2.30, '2023-11-02 14:30:00', '2023-11-03 12:00:00', 'San José', 'Costa Rica'),  
(17, 117, 17, 350.00, '2023-11-03 09:00:00', '2023-11-04 11:00:00', 'Panamá', 'Panama'),  
(18, 118, 18, 89.99, '2023-11-03 11:45:00', '2023-11-04 12:30:00', 'Guatemala', 'Guatemala'),  
(19, 119, 19, 120.50, '2023-11-04 10:10:00', '2023-11-05 10:00:00', 'Asunción', 'Paraguay'),  
(20, 120, 20, 15.00, '2023-11-04 15:00:00', '2023-11-05 14:00:00', 'Lisboa', 'Portugal');
```

```
INSERT INTO purchase_item (purchase_id, product_id, unit_price, quantity) VALUES  
(1, 1001, 1200.00, 1),  
(1, 1003, 25.00, 1),  
(2, 1002, 850.50, 1),  
(3, 1004, 75.99, 1),  
(4, 1005, 2.50, 1),  
(5, 1001, 1200.00, 1),  
(6, 1008, 150.00, 1),  
(7, 1007, 650.00, 1),  
(8, 1006, 59.99, 1),  
(9, 1009, 19.50, 1),  
(10, 1010, 1.80, 1),  
(11, 1011, 3.20, 1),
```

```
(12, 1012, 4.10, 1),  
(13, 1013, 199.99, 1),  
(14, 1014, 299.00, 1),  
(15, 1015, 1.10, 1),  
(16, 1016, 2.30, 1),  
(17, 1017, 350.00, 1),  
(18, 1018, 89.99, 1),  
(19, 1019, 120.50, 1),  
(20, 1020, 15.00, 1);
```

```
SELECT * FROM category;  
SELECT * FROM customer;  
SELECT * FROM employee;  
SELECT * FROM product;  
SELECT * FROM purchase;  
SELECT * FROM purchase_item;
```

Ejercicio 10: Calcular el porcentaje que gasta el cliente en cada compra. Para cada cliente que realizó al menos una compra, muestre el ID de cada compra y el porcentaje del dinero gastado en esa compra con respecto al total. Redondee los porcentajes a números enteros. Muestre tres columnas: contact_name, purchase_id y percentage.

```
select c.contact_name, pu.purchase_id, round((pu.total_price*100)/ (select sum(total_price)  
from purchase),0) as percentage  
from customer c  
join purchase pu on c.customer_id = pu.customer_id;
```

Ejercicio 11: Encuentra el número de productos caros en cada categoría. Muestra los nombres de las categorías y la cantidad de productos de esta categoría cuyo precio unitario es mayor que el precio promedio de un producto de esta categoría. Muestra solo las categorías que contienen dichos productos. Muestra dos columnas: nombre (el nombre de la categoría) y expensive_products cantidad de productos que cuestan más que el precio promedio de un producto de esta categoría.

```

SELECT
    c.name,
    COUNT(*) AS expensive_products
FROM category c
JOIN product p ON c.category_id = p.category_id
WHERE p.unit_price > (
    SELECT AVG(unit_price)
    FROM product
    WHERE category_id = c.category_id
)
GROUP BY c.name;

```

Ejercicio 12: Mostrar los productos comprados con su cantidad máxima comprada. Para cada producto adquirido, muestre su nombre, la cantidad máxima comprada y el número de compras realizadas. Muestre tres columnas: product_name, quantity, y purchases_number.

```

SELECT
    p.product_name,
    MAX(pi.quantity) AS quantity,
    COUNT(pi.purchase_id) AS purchases_number
FROM product p
JOIN purchase_item pi ON p.product_id = pi.product_id
GROUP BY p.product_name
ORDER BY p.product_name;

```

```

SELECT
    p.product_name,
    MAX(pi.quantity) AS quantity,
    (
        SELECT COUNT(*)
        FROM purchase_item pi2
        WHERE pi2.product_id = p.product_id
    ) AS purchases_number
FROM product p
JOIN purchase_item pi ON p.product_id = pi.product_id

```

```
GROUP BY p.product_name  
ORDER BY p.product_name;
```

Ejercicio 13: Enumere los productos discontinuados, continuados y totales en cada categoría.

Para cada categoría, mostrar:

- Su nombre.
- La cantidad de productos discontinuados (es decir, que ya no están disponibles) en esta categoría (nombre esta columna discontinued_products).
- El número de productos continuados (es decir, actualmente disponibles) en esta categoría (nombre esta columna continued_products).
- El número de todos los productos en esta categoría (nombre esta columna all_products).

Ejercicio 14: Contar las compras gestionadas por cada empleado en Houston. Muestra el ID del empleado y el número total de compras que gestionó. Utiliza una subconsulta para obtener información sobre el número de pedidos que gestionó cada empleado por cliente y haz que la consulta principal seleccione FROM esta subconsulta. Considera solo a los empleados que viven en Houston.

Ejercicio 15: Encuentra el mayor número de categorías de productos en una compra. Utilice una subconsulta para seleccionar el ID de compra y el número de categorías distintas que contiene. En la consulta principal, seleccione el número máximo de categorías de esta subconsulta.