Universidad de Ingeniería y Tecnología

Base de Datos I

Profesor: Teófilo Chambilla

2020-2

Proyecto Final

Efraín Antonio Córdova Amaya 201910129

Daniela Abril Vento Bustamante 201910331

Jean Paul Huby Tuesta 201910194

13 de diciembre del 2020

Índice general

1. Requisitos **5**

1.1. Introducción 5

1.2. Descripción general de la empresa 5

1.3. Necesidad de la base de datos 6

1.4. ¿Cómo resuelve este problema hoy? 7

1.4.1. ¿Cómo se almacenan los datos hoy? 7

1.4.2. Flujo de datos 7

1.5. Descripción detallada del sistema 7

1.5.1. Objetos de información actuales 7

1.5.2. Características y funcionalidades esperadas 7

1.5.3. Tipos de usuarios necesarios 8

1.5.4. Tipos de consulta, actualizaciones 8

1.5.5. Tamaño estimado de la base de datos 10

1.6. Objetivos del proyecto 11

1.7. Referencias del proyecto 11

1.8. Eventualidades 12

1.8.1. Problemas que pudieran encontrarse en el proyecto 12

1.8.2. Límites y alcances del proyecto 12

2. Modelo Entidad-Relación **13**

2.1. Reglas semánticas 13

2.2. Modelo Entidad-Relación 14

2.3. Especificaciones y consideraciones sobre el modelo 14

3. Modelo Relacional **18**

3.1. Modelo Relacional 18

3.2. Especificaciones de transformación 19

3.2.1. Entidades 19

3.2.2. Entidades débiles 19

3.2.3. Entidades virtuales 20

3.2.4. Entidades superclase/subclase 20

3.2.5. Relaciones binarias 20

3.3. Diccionario de datos 21

4. Implementación de la base de datos **26**

4.1. Creación de Tablas en PostgreSQL (Scripts) 26

4.2. Carga de datos 29

4.3. Simulación de Datos Faltantes 31

5. Optimización y Experimentación **32**

5.1. Consultas SQL 32

5.1.1. Descripción del tipo de consultas seleccionadas 32

5.1.2. Implementación de consultas en SQL 33

5.2. Metodología del experimento 36

5.3. Optimización de consultas 37

5.3.1. Planes de índices para la consulta 1 39

5.3.2. Planes de índices para la consulta 2 40

5.3.3. Planes de índices para la consulta 3 41

5.4. Plataforma de Pruebas 42

5.5. Medición de tiempos 42

5.5.1. Sin índices 42

5.5.2. 1 000 tuplas 42

5.5.3. 10 000 tuplas 43

5.5.4. 100 000 tuplas 43

5.5.5. 1 000 000 tuplas 43

5.5.6. Con índices 44

5.5.7. 1 000 tuplas 44

5.5.8. 10 000 tuplas 44

5.5.9. 100 000 tuplas 45

5.5.10. 1 000 000 tuplas 45

5.6. Resultados 46
5.6.1. Consulta 1 46
5.6.2. Consulta 2 47
5.6.3. Consulta 3 48
5.7. Análisis y Discusión 49
5.7.1. Análisis de mejoras 49

6. Conclusiones **51**
6.1. Conclusiones 51
6.2. Reflexiones y mejoras 51

Appendices **53**

A. Anexos **54**
A.1. Modelo ER 54
A.2. Modelo Relacional 55

1. Requisitos

1.1. Introducción

Grot es una empresa la cual se dedica a la importación y distribución de productos alimenticios para varios sectores del mercado. Esta se especializa en proveer ingredientes y especialidades a varios sectores económicos (agropecuario, alimenticios, farmacéuticas, etc).

- **Razón Social:** GROT DEL PERU SOCIEDAD ANONIMA CERRADA - GROT PERU S.A.C.
- **RUC:** 20601547687.
- **Dirección:** Jr. los Mirtos Nro. 590 Int. 1406
- **Página Web:** http://www.grotperu.com/gp/index.html

1.2. Descripción general de la empresa

Grot empezó como una empresa llamada "Arqui building" la cual se centraba en la venta de materiales de construcción. Sin embargo, debido a problemas económicos, realizaron un cambio hacia un nuevo sector con mayores oportunidades: el sector de importaciones y exportaciones de productos alimentarios.

En la actualidad la empresa tiene 3 actividades principales:

- Importación: La mayoría de los productos que vende la empresa son traídos de empresas extranjeras.

- Distribución: Una vez que se tiene el producto, la empresa lo deposita en almacenes propios para luego distribuirlos a los distintos clientes. Esto incluye:
  - Tener un control solo del inventario actual.
  - El historial de ventas y compras.

1.3. Necesidad de la base de datos

Con el nuevo rumbo de Grot, el dueño está preocupado de que llegue un momento en el que la empresa crezca hasta el punto de que no sean capaces de organizar toda su información. Por ese motivo quiere una nueva herramienta que le permita gestionarla desde todos sus locales en Lima.

La información que requieren en la base de datos es extremadamente sensible. Es decir, si contratan a un tercero que guarde la información y no sepa manejarla, puede comprometer la información de otros productos. Por ello, el dueño quiere que se mantenga la autenticidad de los datos y sea fácil de navegar a través de ella. Lo segundo es también para asegurar que sea de fácil lectura para los empleados de Grot, quienes también deben tener acceso a estos datos. Esto les permitirá agilizar los procesos de la empresa. Cabe mencionar que el mercado cubierto por Grot en el Perú es muy pequeño, por lo cual también se quiere proteger sus datos de agentes externos.

Una base de datos, al facilitar el acceso y modificación de datos resulta una alternativa viable, segura y escalable para satisfacer los requerimientos de la empresa. Los datos de mayor importancia que necesitan son respecto al flujo económico de la empresa, así como el estado del inventario (stock) actual.

1.4. ¿Cómo resuelve este problema hoy?

1.4.1. ¿Cómo se almacenan los datos hoy?

La mayoría de los datos se guardan en una base de datos rústica y los papeles de documentación necesaria de movimiento económicos de la empresa (Facturas, Boletas, etc). Debido a los múltiples submenús que dispone, la navegación tiene algunos encabezados redundantes.

Se anotaba todo a mano usando Excel y a partir de eso los empleados analizaban toda la información para luego hacer reportes del estado actual.

1.4.2. Flujo de datos

Todos los datos estaban ingresados en un pequeño sistema parecido a Excel en el cual se separaba la información en distintas tablas. Todos los datos eran ingresados manualmente usando todos los documentos facturados de los movimientos de la empresa.

1.5. Descripción detallada del sistema

1.5.1. Objetos de información actuales

Actualmente, la base de datos donde se almacena la información presenta fallas estructurales. Se tomarán algunos datos pertinentes de esta para el desarrollo del nuevo sistema.

1.5.2. Características y funcionalidades esperadas

Se espera que la nueva base de datos resuelva las mayores dificultades sobre el modelo actual que el personal ha dejado en manifiesto durante la creación de la propuesta que ofrecemos. En esa línea, nuestro esquema evita las ambigüedades que el modelo actual tenía respecto a las actividades de compra/venta. Esto se

logra diferenciando a las empresas de las personas naturales, así como la compra de la venta.

El sistema actual también permite encerrar información relevante y extensa sobre la empresa transportista.

Finalmente, el stock está diseñado como una estructura dinámica, de manera que su relación con los productos deja evidente el efecto que tienen las acciones de compra/venta sobre la cantidad de productos.

1.5.3. Tipos de usuarios necesarios

Los tipos de usuarios que usarán del sistema serán los empleados administrativos de la empresa así como los vendedores. Los datos históricos de las ventas pueden servir para hacer un análisis de la situación de la empresa y la dirección a la que puede ir. De igual manera, el personal administrativo tiene que tener pendiente el estado de los inventarios y el almecén de la empresa.

1.5.4. Tipos de consulta, actualizaciones

Resulta muy importante para la empresa saber con quienes tiene conexiones, así como los productos que tiene disponibles. Por ello, se realizarán mayoritariamente consultas relacionadas a los clientes/proveedores, así como de inventario. Adicionalmente, se realizan consultas respecto a dónde se encuentran dichos productos, las compras y ventas y la logística detrás de ellas.

La base de datos debe permitir ver lo siguiente:

- **El inventario actual de productos**
  La empresa vende productos alimenticios, así como diferentes tipos de químicos, los cuales son guardados en almacenes que son proveídos por Grot. El usuario de la herramienta debería ser capaz de ver el estado actual del stock de los productos de la empresa así como obtener de manera efectiva los datos de las transacciones pasadas.

- **El estado de los almacenes**
  Al ser los almacenes las instancias donde se guardan los productos, se necesita saber cuál es el estado de esta infraestructura. Esto es de vital importancia al momento de vender productos a los clientes, ya que se necesita saber en qué almacenes se tienen estos productos para poder planificar la entrega al cliente.

- **El movimiento de compras y ventas de la empresa**
  La empresa esta interesada en tener los datos históricos de todas las ventas y compras realizadas por sus clientes debido a que es muy importante saber quienes son los provedores y compradores más importantes, así como ver los distintos precios de un producto por otros proveedores.En un determinado periodo de tiempo.

- **Las operaciones logísticas de estos materiales**
  Después de cada compra o venta por parte de un cliente, se necesita hacer un documento de transporte debido a requerimientos de la SUNAT. Este documento debe tener toda la información pedida por la SUNAT para evitar problemas legales.

- **Cálculo de impuestos**
  Después de cada venta se tiene se le aplica el impuesto del IGV que luego es pagado por la empresa. Sería de gran ayuda que la empresa pueda calcular los impuestos usando el gestor de la base de datos para poder agilizar ayudar a los contadores.

Tomando estos puntos en consideración, planteamos las siguientes tres consultas para este análisis:

- Obtener el nombre de los productos (o el producto) más comprado(s) por mes en un año.

- Obtener al cliente (o los clientes) que más ha(n) comprado entre ciertas fechas.

- Obtener el nombre (o los nombres) del proveedor(es) del (de los) que más se ha comprado y el nombre del producto que más se ha adquirido de este.

1.5.5. Tamaño estimado de la base de datos

Para poder determinar el tamaño de la base de datos, es importante tener algunos puntos en consideración:

- La base de datos es actualizada cada vez que se realice una compra o venta. Dada la naturaleza de la encuesta, esto es algo que ocurre frecuentemente.
- La mayoría de datos son de tipo *int*, seguido de *string*.
- Los datos de tipo *string* son de tamaños variados, pero se están estableciendo números de caracteres estimados.

Considerando las entidades tenemos:

Nombre de la tabla	**Longitud de atributos (bytes)**	**Longitud del registro (bytes)**
Producto	4+32	36
Almacen	32+42+4	78
Comprobante	4+3+4+1	12
Cliente	4+32	36
Empresa	4+32	36
Persona	4+22+22+22	70
OperacionLogistica	4+8+4+4+22+3+3+12+4+32	96
Vehiculo	8+4+4+7	23
EmpresaDeTransporte	4+32	36
Stock	4+32+4	40
CDetalle	4+32+4+4+4	48
VDetalle	4+4+4+4	12
Venta	4+4	8
Compra	4+4	8
	Total	539

Dadas las ventas de la empresa, se puede estimar que en un mes se realizan 150 operaciones. A partir de ello podemos estimar que en un mes se consumirán aproximadamente

150 x 539 b = 80850 b = 78.96 kb como máximo.

Por consiguiente, en un año se consumen aproximadamente

80850 b x 12 = 970200 b = 947.46 kb por año.

1.6. Objetivos del proyecto

El objetivo de este proyecto es realizar una base de datos para la empresa Grot la cual cumpla con su necesidad de mantener un historial de sus movimientos, así como fijar su inventario de manera efectiva.

1.7. Referencias del proyecto

El cliente llego a darnos base de datos que le ayudan para tomar desiciones para su negocio. Hace uso de herramientas de gestión como referencia, tales como:

- veritrade: https://www.veritradecorp.com/
  Veritrade es una herramienta que sirve para ver el movimiento de importación y exportación de productos. El cliente hace uso de este para ver el historial de compra actual y pasado de ciertos productos.
- Sistema de la Sunat: http://www.aduanet.gob.pe/servlet/HPSGDec10A
  El sistema de la Sunat, a diferencia de Veritrade, es más difícil de leer pero contiene una gran cantidad de información útil para el negocio para ver los productos que entran al país.

Estás base de Datos si bien sirven más que nada para ver el historial importacion y exportacion del Pais. Sin embargo, pueden ser tomados como referencia debido a que el cliente ya está familiarizado con ellos. Dándole un Enfoque más al inventario y los movimientos económicos de la empresa.

1.8. Eventualidades

1.8.1. Problemas que pudieran encontrarse en el proyecto

Los problemas que encontramos se deben a que solo vemos una parte específica del negocio, dado que el esquema gira entorno a la entidad Comprobante. En la práctica, se necesitaría un control más grande que considere ciertas cualidades de las entidades que no necesariamente eran relevantes durante la construcción de la base de datos, tales como:

- Llevar un seguimiento de las fechas de caducidad de los productos perecibles. Esto representaría enfocar a los productos más como una superclase que una clase.
- Considerar otras formas de que los productos ingresen o salgan del stock. El modelo considera que las únicas actividades que pueden modificar el stock durante una transacción son la compra y la venta. Además, estas nuevas vías de alterar el stock serían ajenas al comprobante. Habría que dar un nuevo enfoque a toda la base de datos si pensamos en más escenarios.

1.8.2. Límites y alcances del proyecto

- **Alcance:**
  - Como se explicó en el anterior apartado, este modelo considera a la compra y venta como las únicas actividades de la empresa que son significativas para el resto de elementos de la organización, como el stock.
- **Límites:**
  - El esquema propuesto es ajeno a como funciona la administración interna de la empresa. Esto puede ser un problema si se desea saber, por ejemplo, los datos del personal en X o Y servicio. De nuevo, esto sucede para centralizar el modelo en las transacciones y no en la empresa.

2. Modelo Entidad-Relación

2.1. Reglas semánticas

- Grot realiza servicios de compra/venta tanto a personas como a empresas, ambas definidas por el RUC. En el caso de las personas, es relevante también su nombre y apellidos (materno y paterno), mientras que de las empresas se almacena también su razón social. La base de datos contiene a su vez la dirección del cliente, sea persona o empresa.
- El mismo cliente puede ser comprador si en una transacción pasada fue vendedor y viceversa..
- Para las actividades de compra/venta, Grot está afiliada a una o más empresas transportistas, de las cuales se sabe su razón social y se distinguen por su RUC.
- Se conoce también el estado de la entrega. Esa información nos permite saber si el pedido fue realizado correctamente o hubo algún inconveniente durante la entrega.
- Al ocurrir una compra/venta, Grot genera un comprobante, del cual se reconoce su fecha, precio y se distingue por su número. Este comprobante debe especificar exactamente los siguientes rubros:
  - El precio unitario y la cantidad de los productos en stock que fueron adquiridos/vendidos por Grot.
  - La información relevante sobre el cliente.
- Por cada comprobante hay una operación logística de la entrega. Se necesita conocer el código del transporte, el precio por la operación, la fecha

y la dirección de entrega; así como los datos del vehículo y del chófer que participaron en la operación.

- Un producto puede ser guardado en varios almacenes y un almacén puede contener diferentes productos.
- Se desea tener información sobre los productos que se encuentran en el almacén. Cuando un producto sale o entra al stock, esta operación debe verse reflejada en la base de datos, es decir, debe estar reflejada la cantidad de productos que se encuentran en el almacén.

2.2. Modelo Entidad-Relación

Ver anexo A.1.

2.3. Especificaciones y consideraciones sobre el modelo

- **Entidad Producto**
  - **Especificaciones:** Almacena datos sobre los productos poseídos por la empresa.
  - **Consideraciones:** Sus atributos difícilmente sean modificados. Se añaden más tuplas cuando se adquieren nuevos productos. Se consultan estos productos al momento de realizar una venta. Forma una entidad virtual junto a almacén y stock: puede que exista o no un producto en una cierta cantidad en un almacén, lo cual es necesario saber al momento que la empresa haga una compra.
- **Entidad Almacén**
  - **Especificaciones:** Almacena datos sobre los almacenes de la empresa.
  - **Consideraciones:** Se añaden nuevas tuplas cuando se consiguen nuevos locales. Forma una entidad virtual junto a producto y stock: existe al menos un almacén para almacenar los productos.

- **Entidad Comprobante**
  - **Especificaciones:** Almacena los datos de los comprobantes de pago de la empresa.
  - **Consideraciones:** Se asocian a tanto compras como ventas y tienen relaciones con los detalles de dichas transacciones. Asimismo, cada una de ellas genera una operación logística.
- **Entidad Cliente**
  - **Especificaciones:** Almacena datos generales de los clientes de la empresa.
  - **Consideraciones:** Estos pueden ser tanto aquellos que compran como aquellos que venden de ellos. Requieren un identificador: el RUC, cual también nos permite distinguir el tipo de cliente con el que se está lidiando, sea empresa o persona natural.
- **Entidad Empresa**
  - **Especificaciones:** Almacena datos sobre las empresas que han sido clientes de Grot.
  - **Consideraciones:** Un tipo de cliente. Además de su identificador, se solicita su razón social.
- **Entidad Persona**
  - **Especificaciones:** Almacena datos sobre las personas naturales que han sido clientes de Grot.
  - **Consideraciones:** Un tipo de cliente. Además de su identificador, se solicita su nombre completo.
- **Entidad OperacionLogistica**
  - **Especificaciones:** Almacena datos sobre las operaciones logísticas que se producen cada vez que la empresa emite un comprobante de pago.
  - **Consideraciones:** Es débil de tanto comprobante como vehículo, puesto que requiere saberse a qué transacción está asociada y qué medio de transporte ha utilizado.

- **Entidad Vehiculo**
  - **Especificaciones:** Almacena datos sobre los vehículos de terceros utilizados por Grot para sus operaciones.
  - **Consideraciones:** Cada uno pertenece a una sola empresa de transporte.
- **Entidad EmpresaDeTransporte**
  - **Especificaciones:** Almacena datos sobre las empresas de transporte que han proveído vehículos a Grot. Cada vehículo pertenece únicamente a una empresa. Se reconocen por el RUC de la empresa a la que corresponden y su placa.
  - **Consideraciones:** Hay al menos una empresa que ha proveído vehículos a Grot para la realización de sus operaciones. Se reconocen por su RUC.
- **Relación Stock**
  - **Especificaciones:** Almacena datos sobre la cantidad de productos disponibles en un almacén.
  - **Consideraciones:** Contiene el ID del producto y la dirección del almacén con el objetivo de identificar
- **Relación CDetalle**
  - **Especificaciones:** Almacena datos sobre el producto adquirido desde un almacén específico para una compra.
  - **Consideraciones:** Relaciona a comprobante con la entidad virtual compuesta por Producto-Stock-Almacén. Guarda las llaves de Stock y Comprobante y el precio unitario y la cantidad en stock del producto comprado.
- **Relación VDetalle**
  - **Especificaciones:** Almacena datos sobre la venta de un producto específico.

- **Consideraciones:** Relaciona a producto y comprobante. Guarda las llaves de ambas relaciones y el precio untario y la cantidad en stock del producto vendido.

- **Relación Venta**
  - **Especificaciones:** Almacena datos sobre los comprobantes de venta.
  - **Consideraciones:** Relaciona a comprobante y cliente. Por cada venta hay exactamente un comprobante.

- **Relación Compra**
  - **Especificaciones:** Almacena datos sobre los comprobantes de compra.
  - **Consideraciones:** Relaciona a comprobante y cliente. Por cada compra hay exactamente un comprobante.

3. Modelo Relacional

3.1. Modelo Relacional

Ver anexo A.2 para el diagrama.

- Producto(id:int, nombre:varchar(30))
- Almacen(direccion:varchar(30), descripcion:varchar(40), capacidad:int)
- Comprobante(numero:int, fecha:date, precio:float, tipo:bit)
- Cliente(ruc:int, direccion:varchar(30))
- Empresa(Cliente.ruc:int, razonSocial:varchar(30))
- Persona(Cliente.ruc:int, nombre:varchar(20), apellidoMaterno:varchar(20), apellidoPaterno:varchar(20))
- OperacionLogistica(Comprobante.numero:int, Vehiculo.placa:int, Vehiculo.conductor:int, EmpresaDeTransporte.ruc:int, codigo:int, fecha:date, estado:varchar(10), precioTransporte:float, direccion:varchar(30))
- Vehiculo(EmpresaDeTransporte.ruc:int, placa:int, conductor:int, categoria:varchar(5))
- EmpresaDeTransporte(ruc:int, razonSocial:varchar(30))
- Stock(Productos.id:int, Almacen.direccion:varchar(30), cantidad:float)

- CDetalle(Productos.id:int, Almacen.direccion:varchar(30), Comprobante.numero:int, precioUnitario:float, cantidad:int)
- VDetalle(Producto.id:int, Comprobante.numero:int, precioUnitario:float, cantidad:int)
- Venta(Comprobante.número:int, Cliente.ruc:int)
- Compra(Comprobante.número:int, Cliente.ruc:int)

3.2. Especificaciones de transformación

3.2.1. Entidades

Existen 10 entidades, de las que se distinguen **8 entidades fuertes**. A este grupo pertenecen las entidades Comprobante, Producto, Almacén y Empresa de Transporte. También se incluye a una entidad superclase, sus dos entidades subclase, y una entidad virtual. De estas últimas entidades hablaremos detalladamente más adelante, pues, además de ser entidades fuertes, también presentan una clasificación adicional.

Los atributos y llaves de todas las entidades mencionadas se encuentran ampliamente definidos en apéndices anteriores.

3.2.2. Entidades débiles

Del total de 10 entidades, existen **2 entidades débiles:** Operación logística y vehículo.

La primera es débil tanto de Comprobante como de Vehículo, de ahí que su llave sea una combinación de las llaves de ambas entidades y su llave parcial: código.

Por otro lado, la entidad Vehículo es débil de la entidad fuerte Empresa de Transporte, de ahí que su llave sea una combinación del RUC de la Empresa en particular y su llave parcial: placa.

3.2.3. Entidades virtuales

Se tiene únicamente **1 entidad virtual:** Producto-Stock-Almacén. Esta entidad nos permite hablar de la relación entre el producto y el almacén al momento de la compra, algo que evita ambigüedades, puesto que una compra puede venir de varios almacenes. Se podría optar por convertir la relación Compra en una relación ternaria, pero se ha terminado por no considerar esta opción ya que se desea mantener el diseño simple.

3.2.4. Entidades superclase/subclase

El esquema presenta **1 superclase**, Cliente, que cuenta con **2 Subclases**, Empresa y Persona.

Gracias a esta relación de parentesco es posible indexar a los clientes bajo su rol jurídico, algo que a Grot le interesa al momento de analizar la data.

La llave de Cliente es RUC. Sus subclases extienden sus propios atributos, pero siguen diferenciándose únicamente por la llave de su superclase.

3.2.5. Relaciones binarias

El modelo cuenta con **8 relaciones binarias:**

- *Stock*, que relaciona a Producto y Almacén (1-N a 0-N). Tiene como atributo descriptivo a Cantidad(kg).
- *Compra*, que relaciona a Comprobante y Cliente (1-1 a 1-N).
- *Venta*, que relaciona a Comprobante y Cliente (1-1 a 1-N).
- *CDetalle*, que relaciona a la entidad virtual Producto-Stock-Almacén y Comprobante (0-N a 1-N). Tiene como atributos descriptivos a Precio Unitario y Cantidad.
- *VDetalle*, que relaciona a Producto y Comprobante (0-N a 1-N). Tiene como atributos descriptivos a Precio Unitario y Cantidad.

- *Pertenece*, que relaciona a Empresa de Transporte y su entidad débil, Vehículo (1-1 a 1-N).
- *Recorrido*, que relaciona a Vehículo con su entidad débil Operación Logística (0-N a 1-1)
- *Entrega*, que relaciona a Comprobante con su entidad débil, Operación Logística (0-N a 1-1). Tiene como atributo descriptivo a Estado.

3.3. Diccionario de datos

- Producto

PRODUCTO				
Nombre del campo	Tipo de dato	PK	FK	Descripción
id	int	X		ID del producto
nombre	varchar (30)			Nombre del producto

- Almacen

ALMACEN				
Nombre del campo	Tipo de dato	PK	FK	Descripción
direccion	varchar (30)	X		Dirección del almacén
descripcion	varchar (40)			Oración que describe el almacén
capacidad	int			Cantidad de productos que puede almacenar

- Comprobante

COMPROBANTE				
Nombre del campo	Tipo de dato	PK	FK	Descripción
numero	int	X		Número identificador del comprobante
fecha	date			La fecha en la que el comprobante fue emitido
precio	float			El precio con el cual se facturó el comprobante
tipo	bit	X		Tipo de comprobante 0: físico, 1: electrónico

- Cliente - Superclase

CLIENTE				
Nombre del campo	Tipo de dato	PK	FK	Descripción
ruc	int	X		La identificación de una persona natural o jurídica. Si empieza con 10 es natural, si empieza con 20 es jurídica
direccion	varchar(30)			La dirección del cliente

- Empresa - Subclase de Cliente

EMPRESA				
Nombre del campo	Tipo de dato	PK	FK	Descripción
ruc	int	X	X	La identificación de una persona natural o jurídica
razonSocial	varchar(30)			El nombre con el que se conoce a la empresa

- Persona - Subclase de Cliente

PERSONA				
Nombre del campo	Tipo de dato	PK	FK	Descripción
ruc	int	X	X	La identificación de una persona natural o jurídica
Nombre	varchar (20)			Nombre de la persona
apellidoPaterno	varchar (20)			Apellido paterno de la persona
apellidoMaterno	varchar (20)			Apellido materno de la persona

- OperacionLogistica

OPERACIONLOGISTICA				
Nombre del campo	Tipo de dato	PK	FK	Descripción
numero	int	X	X	Número identificador del comprobante
placa	varchar(6)	X	X	Placa del Vehiculo
conductor	int	X	X	DNI del conductor
ruc	int	X	X	La identificación de una empresa de transporte
codigo	varchar (20)	X		Código de la operación
iFecha	date			Fecha de inicio de la operación
fFecha	date			Fecha de fin de la operación
estado	varchar(10)			Estado de la operación
precioTransporte	float			Costo del transporte
direccion	varchar(30)			Dirección de la operación

- Vehiculo

VEHICULO				
Nombre del campo	Tipo de dato	PK	FK	Descripción
ruc	int	X	X	La identificación de una empresa de transporte
placa	varchar(6)	X		Placa del Vehiculo
conductor	int	X		DNI del conductor
Categoria	varchar(3)			Categoría del vehículo

- EmpresaDeTransporte

VEHICULO				
Nombre del campo	Tipo de dato	PK	FK	Descripción
ruc	int	X		La identificación de una empresa de transporte
razonSocial	Varchar(30)			El nombre con el que se conoce a la empresa

- Stock

STOCK				
Nombre del campo	Tipo de dato	PK	FK	Descripción
id	int	X	X	ID del producto
direccion	varchar(30)	X	X	Dirección del almacén
cantidad	float			Cantidad del producto (kg)

- CDetalle

CDETALLE				
Nombre del campo	Tipo de dato	PK	FK	Descripción
id	int	X	X	ID del producto
dirección	varchar(30)	X	X	Dirección del almacén
numero	int	X	X	Número identificador del comprobante
precioUnitario	float			Precio de la unidad del producto
cantidad	float			Cantidad del producto (kg)

- VDetalle

VDETALLE				
Nombre del campo	Tipo de dato	PK	FK	Descripción
id	int	X	X	ID del producto
numero	int	X	X	Número identificador del comprobante
precioUnitario	float			Precio de la unidad del producto
cantidad	float			Cantidad del producto (kg)

- Venta

VENTA				
Nombre del campo	Tipo de dato	PK	FK	Descripción
numero	int	X	X	Número identificador del comprobante
ruc	int		X	La identificación de una persona natural o jurídica. Si empieza con 10 es natural, si empieza con 20 es jurídica

- Compra

COMPRA				
Nombre del campo	Tipo de dato	PK	FK	Descripción
numero	int	X	X	Número identificador del comprobante
ruc	int		X	La identificación de una persona natural o jurídica. Si empieza con 10 es natural, si empieza con 20 es jurídica

4. Implementación de la base de datos

4.1. Creación de Tablas en PostgreSQL (Scripts)

Se crearon 4 esquemas con las cantidades de datos solicitados para las consultas de prueba. En el siguiente script, se mostrará el código para la tabla con 1k de registros.

```
SET search_path = "$user", test1k;

create TABLE producto(
    id integer,
    nombre varchar(30),
    PRIMARY KEY (id)
);

create TABLE almacen(
    direccion varchar(30),
    descripcion varchar(40),
    capacidad integer,
    PRIMARY KEY (direccion)
);

create TABLE comprobante(
    numero integer,
```

```
    fecha date,
    precio float,
    tipo bool,
    PRIMARY KEY (numero)
);

create TABLE vDetalle(
    id integer,
    numero integer,
    cantidad float,
    precioUnitario float,
    primary Key (id, numero),
    FOREIGN key (id) references producto(id),
    FOREIGN key (numero) references comprobante(numero)
);

create TABLE cDetalle(
    id integer,
    direccion varchar(30),
    numero integer,
    cantidad float,
    precioUnitario float,
    primary Key (id, numero, direccion),
    FOREIGN key (id) references producto(id),
    FOREIGN key (numero) references comprobante(numero),
    FOREIGN key (direccion) references almacen(direccion)
);

CREATE TABLE stock (
    id integer,
    almacen varchar(30),
    cantidad float,
    primary Key (id, almacen),
    FOREIGN KEY (id) REFERENCES producto(id),
    FOREIGN KEY (almacen) REFERENCES almacen(direccion)
```

```
);

CREATE TABLE Cliente (
    ruc integer PRIMARY KEY,
    direccion varchar(30)
);

CREATE TABLE Empresa (
    razonSocial integer,
    CONSTRAINT p_child_pkey_emp PRIMARY KEY (ruc)
) INHERITS (Cliente);

CREATE TABLE Persona (
    nombre varchar(20),
    apellidoPaterno varchar(20),
    apellidoMaterno varchar(20),
    CONSTRAINT p_child_pkey_per PRIMARY KEY (ruc)

) INHERITS (Cliente);

CREATE TABLE Venta (
    numero integer PRIMARY KEY,
    ruc integer,
    FOREIGN KEY (numero) REFERENCES Comprobante(numero)
);

CREATE TABLE Compra (
    numero integer PRIMARY KEY,
    ruc integer,
    FOREIGN KEY (numero) REFERENCES Comprobante(numero)
);
```

4.2. Carga de datos

Para la creación de data de prueba, utilizamos el lenguaje **Python** para elaborar consultas y ejecutarlas mediante una conexión dada por la librería **psycopg2** con la base de datos montada en **PostgresSQL**.
La gramática de nuestras inserciones, naturalmente, tienen la forma:

```
INSERT INTO (table_name) VALUES (<list_of_atributes>)
```

junto con las validaciones respectivas solicitadas por la conexión con la base de datos SQL.
Un buen ejemplo de esto es la función generate_comprobante

```
def generate_comprobante(size):
comprobante_cache = []
for i in range(size):
    with conn:
        with conn.cursor(cursor_factory=psycopg2.extras.DictCursor) as curr:
            numero = random.randint(1000, 9999)
            while numero in comprobante_cache:
                numero = random.randint(1000, 9999)
            comprobante_cache.append(numero)
            random_date = gen_random_date()
            random_cost = random.randint(100, 1000000)
            random_type = (random.randint(1,2) == 1)
            curr.execute(""" INSERT INTO Grot.comprobante
            (numero, fecha, precio, tipo) VALUES (%s, %s, %s, %s);
            """,(numero,random_date,random_cost,random_type))
            conn.commit()
#End session
curr.close()
```

En esta consulta pueden verse explícitamente los pasos de la operación:

1. Iniciar la conexión
2. Generar un cursor
3. Generar un cache para guardar llaves que se usaran en otras tablas
4. Generar, mediante las bondades de la librería *random*, los datos de prueba
5. Ejecutar el Insert
6. Hacer el *commit* respectivo
7. Cerrar el cursor.

Otra función útil durante el desarrollo fue get_keys:

```
def get_keys(keys, table, rows = "", prob = 1):
row = []
if(str(rows) != ""):
    rows = "Limit " +  str(rows)

with conn:
    with conn.cursor(cursor_factory=psycopg2.extras.DictCursor) as curr:
        curr.execute("SELECT " + str(keys)  +" FROM Grot." + str(table)
        + " TABLESAMPLE BERNOULLI( " + str(prob) + ") " + rows)
        row = curr.fetchall ()
        conn.commit()
return row
```

Esta función permite la obtención de llaves fóraneas necesarias para las relaciones o las entidades hijas.
Recuerde que puede encontrar los scripts para generar estas y las demás tablas en el repositorio, cuyo link esta referenciado en **??**.

4.3. Simulación de Datos Faltantes

No existe una lógica en especial para la creación de parámetros aleatorios. Los atributos generalmente son generados aleatoriamente mediante la función random, los cuales son parseados por la propia librería de forma que la base de datos pueda consumirlos como uno de sus tipos primtivos (integer, datetime, etc).
Nuestros módulos de creación de data son líneas de código genéricas que solicitan información al contexto de la librería y la retornan apropiadamente para su uso en las tablas:

```
def genText():
return ''.join(random.choices(string.ascii_letters + string.digits, k=16))

def gen_random_date():
start_date = datetime.date(2010, 1, 1)
end_date = datetime.date(2020, 12, 1)
time_between_dates = end_date - start_date
days_between_dates = time_between_dates.days
random_number_of_days = random.randrange(days_between_dates)
return start_date + datetime.timedelta(days=random_number_of_days)
```

5. Optimización y Experimentación

5.1. Consultas SQL

Se poblaron 10 de las 15 tablas de la base de datos para generar tres consultas cuyo coste temporal es de importancia a largo plazo para Grot.

5.1.1. Descripción del tipo de consultas seleccionadas

Las consultas planteadas son:

Consulta 1:
Obtener el nombre de los productos (o el producto) más comprado(s) por mes en un año.
Consulta 2:
Obtener al cliente (o los clientes) que más ha(n) comprado entre ciertas fechas.
Consulta 3:
Obtener el nombre (o los nombres) del proveedor(es) del (de los) que más se ha comprado y el nombre del producto que más se ha adquirido de este

5.1.2. Implementación de consultas en SQL

Consulta 1:

```
SELECT nombre
from Producto NATURAL JOIN
(
    SELECT id, mes, SUM(cantidad) cantidadProductos
    FROM VDetalle NATURAL JOIN
    (
        SELECT numero, EXTRACT(MONTH FROM fecha) as mes
        FROM Comprobante
        WHERE fecha BETWEEN '2020-01-01' AND '2020-12-31'
    ) cantidadesMenusales
    GROUP BY id, mes
    HAVING SUM(cantidad) =
    (
        SELECT MAX(cantidadProductos) FROM
        (
            SELECT id, mes, SUM(cantidad) cantidadProductos
            FROM VDetalle NATURAL JOIN
            (
                SELECT numero, EXTRACT(MONTH FROM fecha) as mes
                FROM Comprobante
                WHERE fecha BETWEEN '2020-01-01' AND '2020-12-31'
            ) comprobantesYSuMes
            GROUP BY id, mes
        ) comprasDeProductoPorMes
    )
) temp
;
```

Consulta 2:

```
SELECT ruc
FROM (
    SELECT ruc, COUNT (numero) as comprobantes
    FROM venta NATURAL JOIN (
        SELECT numero
        FROM Comprobante
        WHERE fecha BETWEEN '2019-12-29' AND '2020-11-27'
        ) tempFecha
    GROUP BY ruc
    HAVING COUNT(numero) = (
        SELECT MAX(comprobantes)
        FROM (
        SELECT ruc, COUNT (numero) as comprobantes
        FROM venta NATURAL JOIN (
            SELECT numero
            FROM Comprobante
            WHERE fecha BETWEEN '2019-12-29' AND '2020-11-27'
            ) tempFecha
        GROUP BY ruc
        )temp)
)clientesMax
;
```

Consulta 3:

```
SELECT ruc, nombre
FROM Producto NATURAL JOIN (
    SELECT ruc, id
    FROM CDetalle NATURAL JOIN (
        SELECT ruc, numero
        FROM Comprobante NATURAL JOIN (
            SELECT ruc, numero
            FROM Compra NATURAL JOIN (
            SELECT ruc
            FROM
                (SELECT ruc, count (ruc) compras
                FROM Compra
                GROUP BY ruc
                HAVING COUNT(ruc) = (
                    SELECT MAX(compras)
                    FROM (
                        SELECT ruc, COUNT(ruc) compras
                        FROM Compra
                        GROUP BY ruc
                    ) CompradorCount
                ))CompradorMaxCompras
            )CompradorMax
        )ComprasMax
    )ComprobantesMax
    WHERE cantidad = (
        SELECT MAX(cantidad)
        FROM(
            SELECT ruc, id, cantidad
            FROM CDetalle NATURAL JOIN (
            SELECT ruc, numero
            FROM Comprobante NATURAL JOIN (
                SELECT ruc, numero
                FROM Compra NATURAL JOIN (
```

```
                SELECT ruc
                FROM
                    (SELECT ruc, count (ruc) compras
                    FROM Compra
                    GROUP BY ruc
                    HAVING COUNT(ruc) = (
                        SELECT MAX(compras)
                        FROM (
                            SELECT ruc, COUNT(ruc) compras
                            FROM Compra
                            GROUP BY ruc
                        ) CompradorCount
                    ))CompradorMaxCompras
                )CompradorMax
            )ComprasMax
        )ComprobantesMax)cntMax
    )
)ProductoMax
;
```

5.2. Metodología del experimento

Para poder tomar una medición exacta de acuerdo a los objetivos de este experimento seguiremos los siguientes pasos:

1. Creamos una base de datos para cada una de las métricas solicitadas (1k, 10k, 100k y 1M) con el fin de agilizar los test.
2. Ya con la base de datos, primero se realizan las consultas sin índices.
    a) Se realizan 5 iteraciones por métrica contando a partir de la segunda iteración debido a que esta va ha tener un tiempo mucho más alto debido al caché. Cada vez que se vuelve a usar una tabla con otra consulta o distinta tabla limpiaremos el cache.
    b) Por cada métrica, tomamos promedio simple a las iteraciones.

3. Luego, realizamos los pasos anteriores con las consultas indexadas usando, en síntaxis de psql:

```
SET enable_<index_name>TO on/off
```

Ademas de limpiar el cache utilizado anteriormente con el comando;

```
DISCARD ALL;
```

5.3. Optimización de consultas

Tras analizar los atributos empleados en cada consulta, se procedió a construir los siguientes índices para agilizarlas:

Tabla	**Columna**	**Nombre**	**Tipo**
Compra	ruc	*ind_compra_ruc*	Hash
Compra	numero	*ind_compra_num*	Hash
Comprobante	numero	*ind_comprobante_num*	Hash
Comprobante	fecha	*ind_comprobante_fecha*	B-tree
CDetalle	numero	*ind_cdetalle_num*	Hash
CDetalle	id	*ind_cdetalle_id*	Hash
VDetalle	numero	*ind_vdetalle_num*	Hash
VDetalle	id	*ind_vdetalle_id*	Hash

El código utilizado para crear los índices es el siguiente:

```
CREATE INDEX ind_compra_ruc ON compra USING hash (ruc);
CREATE INDEX ind_compra_num ON compra USING hash (numero);

CREATE INDEX ind_comprobante_num ON comprobante USING hash (numero);
CREATE INDEX ind_comprobante_fecha ON comprobante USING btree (fecha);

CREATE INDEX ind_cdetalle_num ON cdetalle USING hash (numero);
CREATE INDEX ind_cdetalle_id ON cdetalle USING hash (id);
```

```
CREATE INDEX ind_vdetalle_num ON vdetalle USING hash (numero);
CREATE INDEX ind_vdetalle_id ON vdetalle USING hash (id);
```

La mayoría trabaja con búsquedas directas, motivo por el cual se emplean índices hash. Debido a que los atributos empleados son mayoritariamente los mismos para cada consulta, no se vio mayor necesidad de especificar cuál se usaba para cada parte. Colocamos a continuación el plan de ejecución, el cual muestra el uso de los índices, en cada una de las consultas.

5.3.1. Planes de índices para la consulta 1

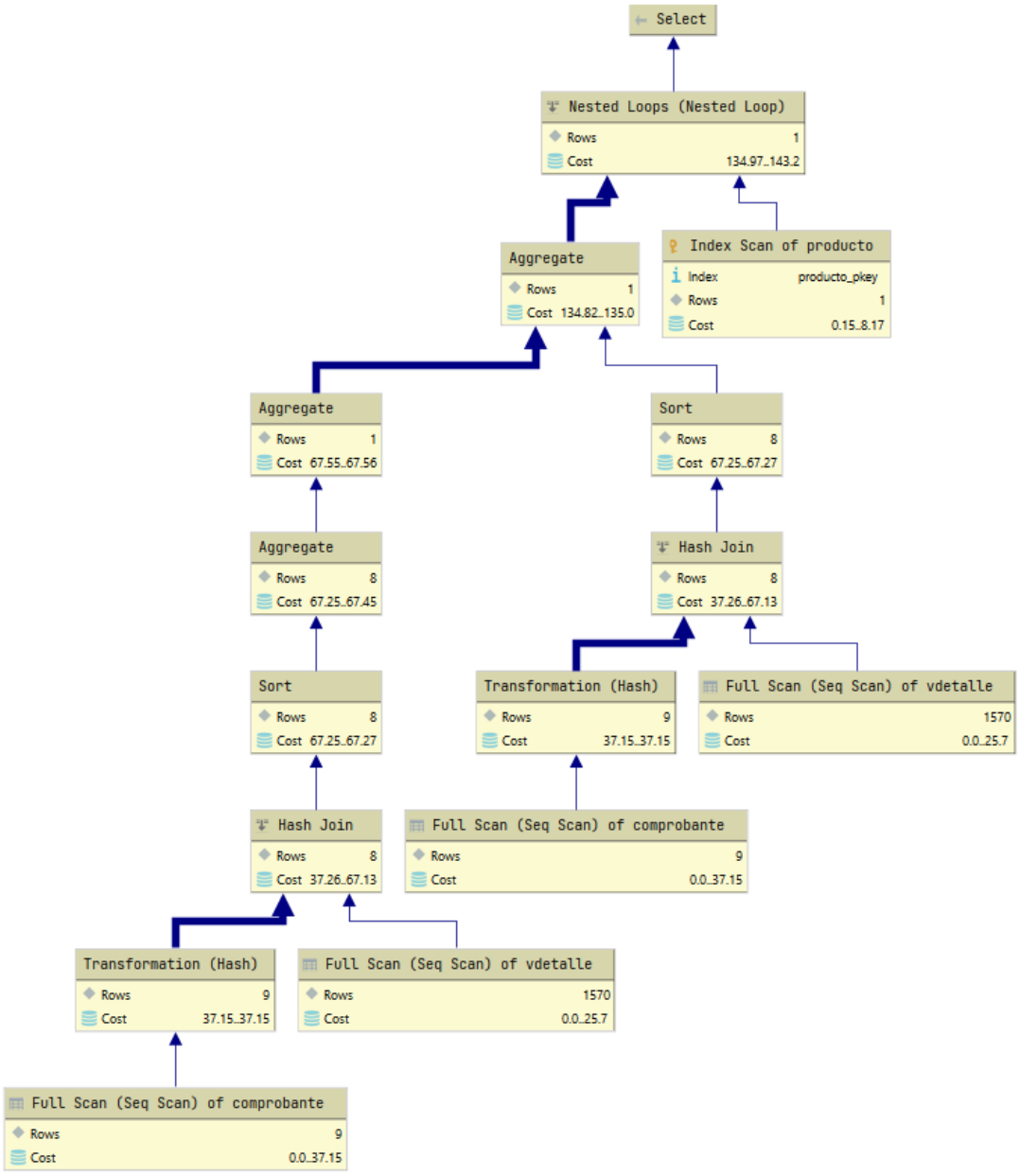


Figura 5.1: Árbol de consultas generado en DataGrip

5.3.2. Planes de índices para la consulta 2

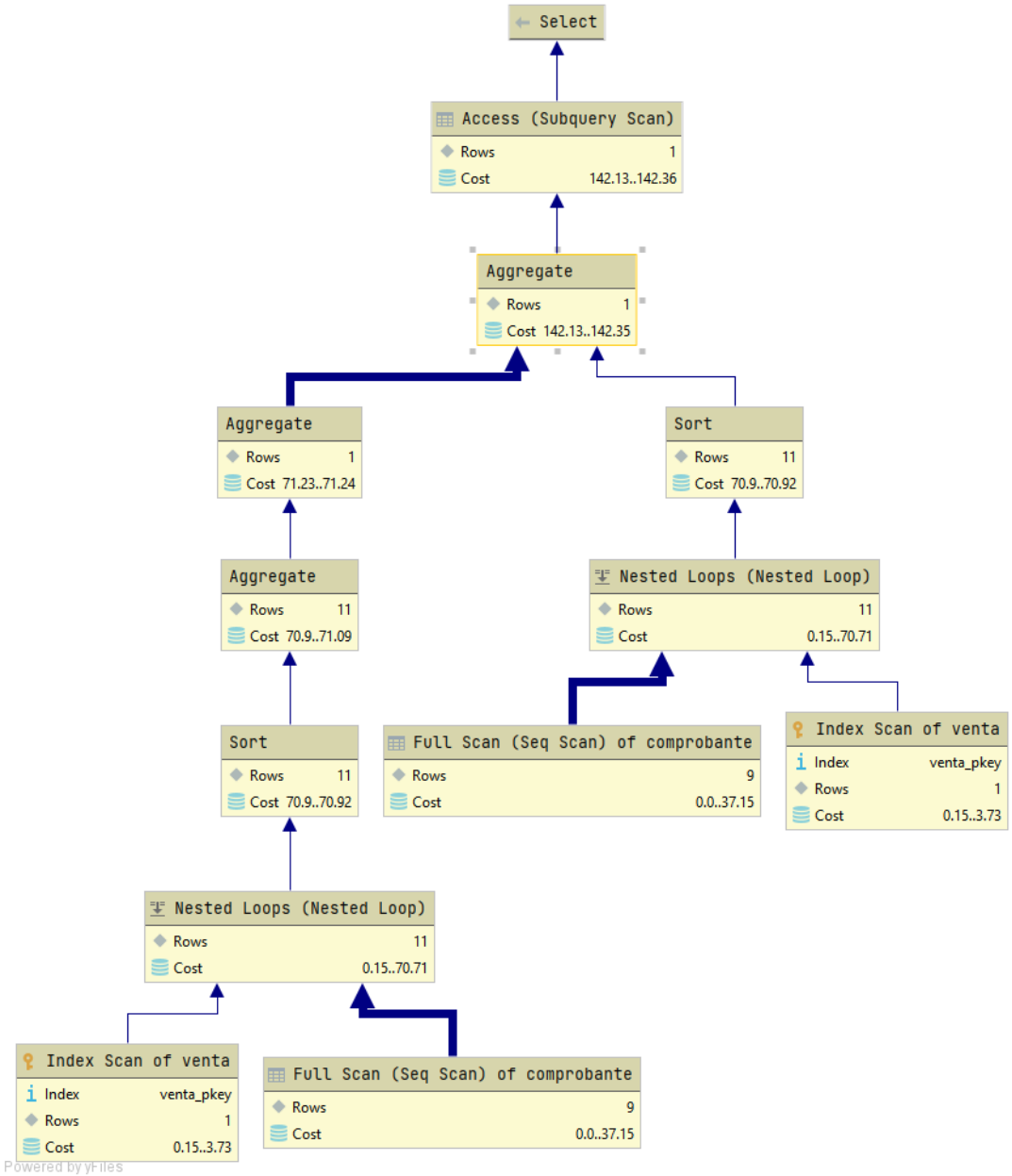


Figura 5.2: Árbol de consultas generado en DataGrip

5.3.3. Planes de índices para la consulta 3

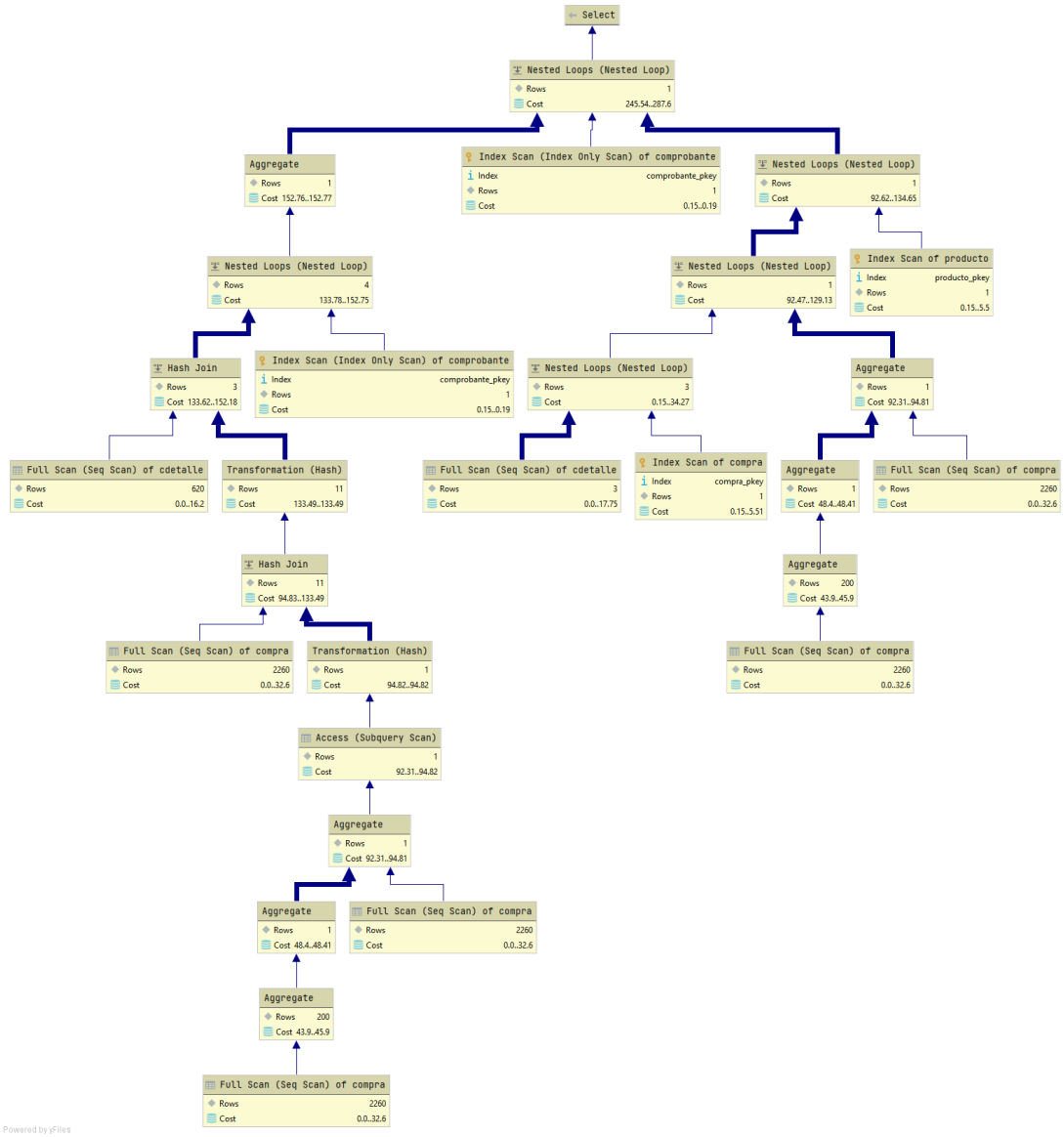


Figura 5.3: Árbol de consultas generado en DataGrip

5.4. Plataforma de Pruebas

Sistema Operativo	Windows 10 home, version 20H2
RAM	8,00 GB
CPU	Procesador Intel(R) Core(TM) i7-8750H
[SSD] Protocolo trans.	NTFS
[SSD] Capacidad	118 GB
[SSD] Tasa trans(Lectura / Escritura)	927 MB/s y 386 MB/s
PostgreSQL	PostgreSQL 13
pgADMIN	pgADMIN 4.26

5.5. Medición de tiempos

5.5.1. Sin índices

5.5.2. 1 000 tuplas

Medición de tiempos (msec) para 1 000 tuplas

Iteración	Consulta 1	Consulta 2	Consulta 3
Iteración 1	73	64	95
Iteración 2	67	85	133
Iteración 3	90	89	447
Iteración 4	72	71	80
Iteración 5	74	84	74
Promedio	75,2	78,6	165,8

5.5.3. 10 000 tuplas

Medición de tiempos (msec) para 10 000 tuplas

Iteración	Consulta 1	Consulta 2	Consulta 3
Iteración 1	116	77	103
Iteración 2	157	84	202
Iteración 3	127	108	165
Iteración 4	168	84	118
Iteración 5	123	125	134
Promedio	138,2	95,6	144,6

5.5.4. 100 000 tuplas

Medición de tiempos (msec) para 100 000 tuplas

Iteración	Consulta 1	Consulta 2	Consulta 3
Iteración 1	515	254	774
Iteración 2	444	247	781
Iteración 3	442	222	832
Iteración 4	560	211	909
Iteración 5	503	232	879
Promedio	492,8	233,2	835

5.5.5. 1 000 000 tuplas

Medición de tiempos (msec) para 1 000 000 tuplas

Iteración	Consulta 1	Consulta 2	Consulta 3
Iteración 1	2501	966	6228
Iteración 2	2494	825	6106
Iteración 3	2444	805	6202
Iteración 4	2397	854	6431
Iteración 5	2367	784	6626
Promedio	2440,6	862,5	6318,6

5.5.6. Con índices

5.5.7. 1 000 tuplas

Medición de tiempos (msec) para 1 000 tuplas

Iteración	Consulta 1	Consulta 2	Consulta 3
Iteración 1	109	54	70
Iteración 2	95	55	119
Iteración 3	133	55	94
Iteración 4	177	63	119
Iteración 5	156	56	132
Promedio	134	56,6	106,8

5.5.8. 10 000 tuplas

Medición de tiempos (msec) para 10 000 tuplas

Iteración	Consulta 1	Consulta 2	Consulta 3
Iteración 1	120	108	113
Iteración 2	132	99	171
Iteración 3	132	107	102
Iteración 4	163	97	97
Iteración 5	100	118	121
Promedio	129,4	105,8	120,8

5.5.9. 100 000 tuplas

Medición de tiempos (msec) para 100 000 tuplas

Iteración	Consulta 1	Consulta 2	Consulta 3
Iteración 1	265	138	267
Iteración 2	171	195	382
Iteración 3	195	157	404
Iteración 4	248	169	343
Iteración 5	267	151	401
Promedio	229,2	162	359,4

5.5.10. 1 000 000 tuplas

Medición de tiempos (msec) para 1 000 000 tuplas

Iteración	Consulta 1	Consulta 2	Consulta 3
Iteración 1	929	760	4168
Iteración 2	962	787	4168
Iteración 3	816	782	4170
Iteración 4	6917	744	4113
Iteración 5	705	699	4622
Promedio	820,8	754,4	4251,8

5.6. Resultados

En esta parte del informe, se contabilizará el costo temporal de las consultas en total de 5 interacciones (con y sin índices):

5.6.1. Consulta 1

Medición de tiempos (ms) con respecto a los índices y tamaño de la muestra

Tamaño	Sin índices	Con índices	Mejora
1k	75,2	134	-78,19 %
10k	138,2	129,4	6,37 %
100k	492,8	229,2	53,49 %
1 000k	2440,6	820,6	66,38 %

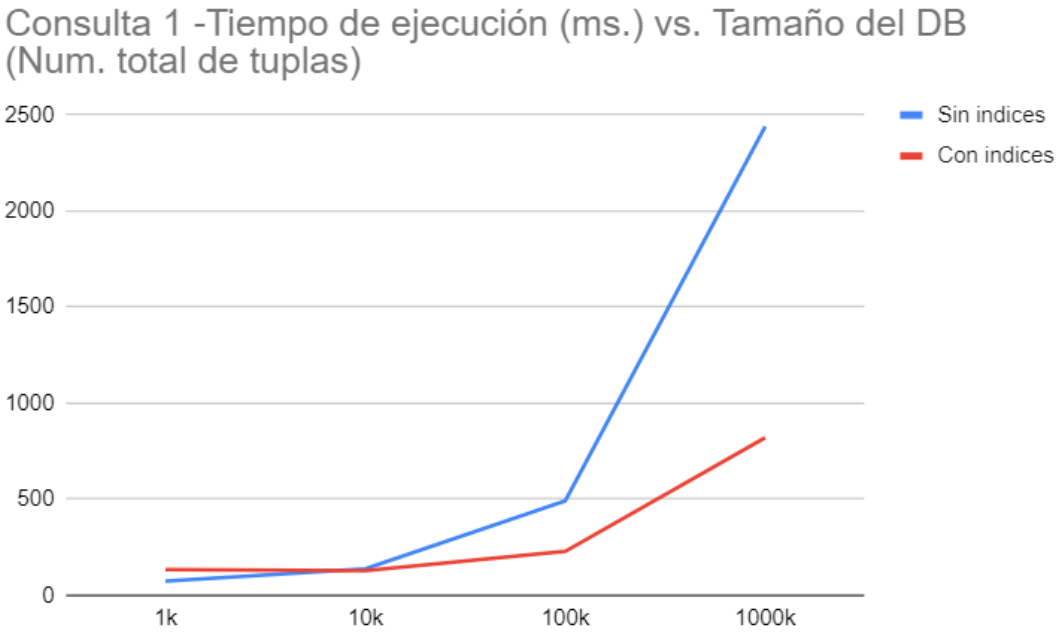


Figura 5.4: Diagrama de resultados sobre la consulta 1

Se observa que, si bien con 1000 elementos, la consulta fue más rápida al trabajar sin índices, hubo una mejora significativa al trabajar con 100 000 y 1 000

000 de datos, llegando a restarle más de la mitad del tiempo a la consulta original. Esto indica que, a final de cuentas, es una opción escalable.

5.6.2. Consulta 2

Medición de tiempos (msec) con respecto a los índices y tamaño de la muestra

Tamaño	Sin índices	Con índices	Mejora
1k	78,6	56,6	27,99 %
10k	95,6	61,4	35,77 %
100k	233,2	102	56,26 %
1 000k	862,5	601	30,32 %

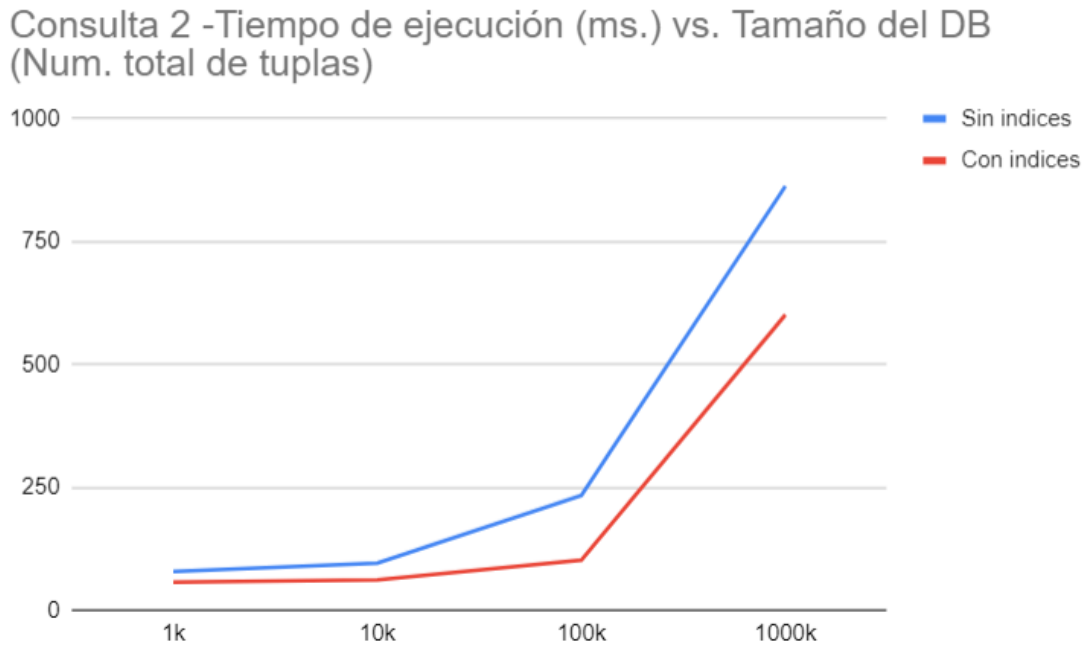


Figura 5.5: Diagrama de resultados sobre la consulta 2

En el caso de esta consulta, no se ve mucha diferencia al trabajar con o sin índices para una cantidad de datos menor a 10k. Sin embargo, el coste se mantiene en constante mejora, Esto se hace más notable cada vez que se aumenta el número

de registros. Entre los 100k y 1000k registros se observa que la pendiente se vuelve mucho más empinada, pero la parábola que se va formando no parece acercarse a una exponencial.

5.6.3. Consulta 3

Medición de tiempos (msec) con respecto a los índices y tamaño de la muestra

Tamaño	Sin índices	Con índices	Mejora
1k	165,8	106,8	35,59 %
10k	144,6	120,8	16,46 %
100k	835	359,4	56,96 %
1 000k	6318,6	4251,8	32,71 %

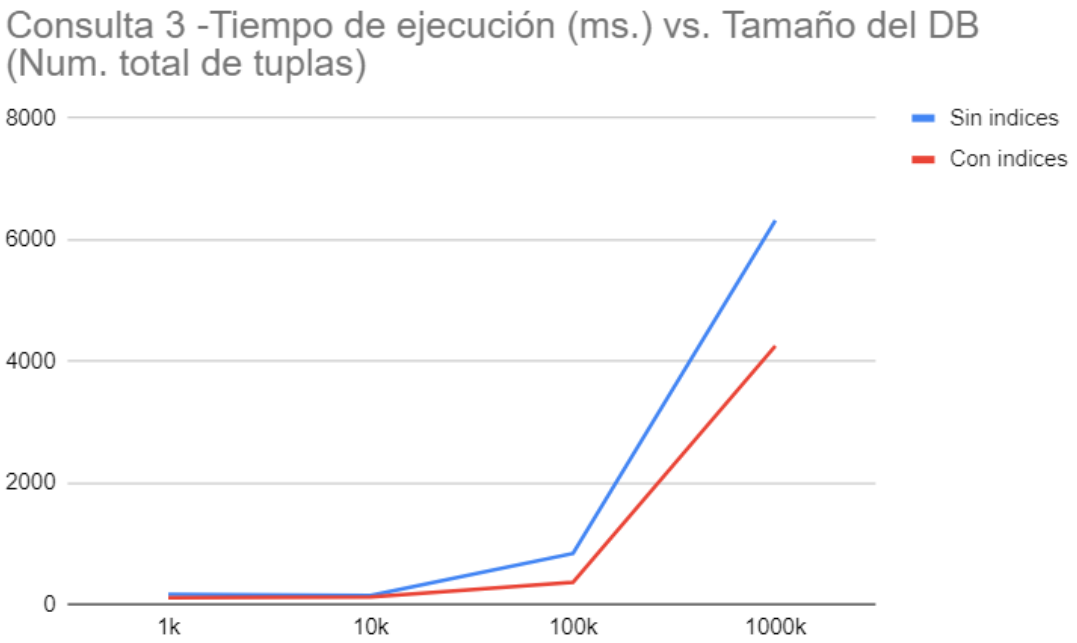


Figura 5.6: Diagrama de resultados sobre la consulta 3

Esta consulta es interesante en el sentido que los tiempos se mantienen considerablemente bajos hasta los 100k registros tanto para las consultas con índices y

sin estos. Pasada esta marca, el tiempo de ejecución se eleva considerablemente, pero manteniendo una distancia a tener en cuenta entre las consultas sin índices y con ellos. Esto denota que los índices están ayudando a la consulta.

5.7. Análisis y Discusión

Se debatirán sobre las políticas de mejoramiento; en general, sobre la implementación de la base de datos y los resultados de ponerla en marcha en la fase de testeo.

5.7.1. Análisis de mejoras

En su mayoría, se ha mostrado una mejora considerable tras hacer uso de índices, en especial cuando se está lidiando con una cantidad amplia de datos, lo que indica un grado de escalabilidad.

Sin embargo, algunas consultas tienen una mejora negativa, es decir, el *performance* de la consulta empeora en calidad de tiempo. Esto se puede asociar a dos factores: el número de registros y el uso de funciones de agregación.

En el caso de las consultas con menor número de registros, lo que sucede es que, al haber menos opciones de toparse con elementos distintos, se producen más colisiones. Al producirse colisiones, la complejidad del hash aumenta, pues tiene que hacer una búsqueda más amplia para poder localizar el elemento requerido para la consulta.

Por otra parte, las consultas que realizamos utilizan funciones de agregación, las cuales no permiten una configuración manual de ínidces en PostgreSQL más allá de la preferida por el DBMS, como si tienen otros sistemas de gestión de base de datos, por ejemplo, MongoDB. Cabe mencionar, sin embargo, que existe una forma de optimizarlas en PSQL gracias a la agregación paralela. Las especificaciones detrás de ello es algo con lo que se puede experimentar en un futuro proyecto.

Otro punto interesante por mencionar es que, tras realizar algunas pruebas

adicionales que realizamos por ocio y curiosidad, las cuales no registramos en su totalidad, observamos que hubieron mejoras en algunos de los tiempos de ejecución. Esto se realizó un día después de las pruebas registradas, en el cual al computador empleado por las pruebas no se le había exigido tanto, por lo cual podría ser temerario arrojar conclusiones. Verificar esto es cuestión de investigar o realizar pruebas con otros ordenadores.

6. Conclusiones

6.1. Conclusiones

Al finalizar el proyecto, MangoDB logro garantizarle a Grot una base de datos con las siguientes características:

1. Que sea capaz de guardar de forma óptima el historial de los principales movimientos de la empresa, tales como la compra y venta de productos, así como la contratación de servicios de transporte.
2. Que se encuentre correctamente indexada a fin de lograr una escabilidad sobresaliente en aquellas consultas cuyo resultado es escencial para la toma de decisiones en la misión y visión de la empresa.
3. Que registre todos los movimientos durante el transporte de las entregas, así como el estado del envío, de forma que la empresa pueda analizar la calidad de este servicio.

Con estas conclusiones y todo lo evidenciado en este informe, MangoDB ha cumplido con el objetivo planteado en la sección 1.6.

6.2. Reflexiones y mejoras

- Si bien los índices pueden ayudar bastante al momento de hacer consultas, en una base de datos real también hay que tener en cuenta el coste que tienen estas a la hora de modificar una tabla. No tenerlo en cuenta puede

dar grandes problemas en tablas que son modificadas por transacciones, tales como Stock.

- Más allá de que algunas consultas indexadas tengan un mayor tiempo con una muestra pequeña, lo importante es fijarse en que tan escalable es la consulta con una mayor cantidad de datos. Esto se visualiza al fijarse no solo en los registros pequeños, sino también en los grandes.

- Si se quiere hacer una implementación más realista de la base de datos, se tiene que chequear todas las restricciones que requiere una compañíam. Algunas verificaciones como que el ruc de las empresas empiece con ciertos dígitos (por temas de formato) y/o que los productos coincidan con aquellos que son vendidos por Grot son reglas semánticas que, a fin de resaltar la implementación, no se han considerado en este proyecto académico.

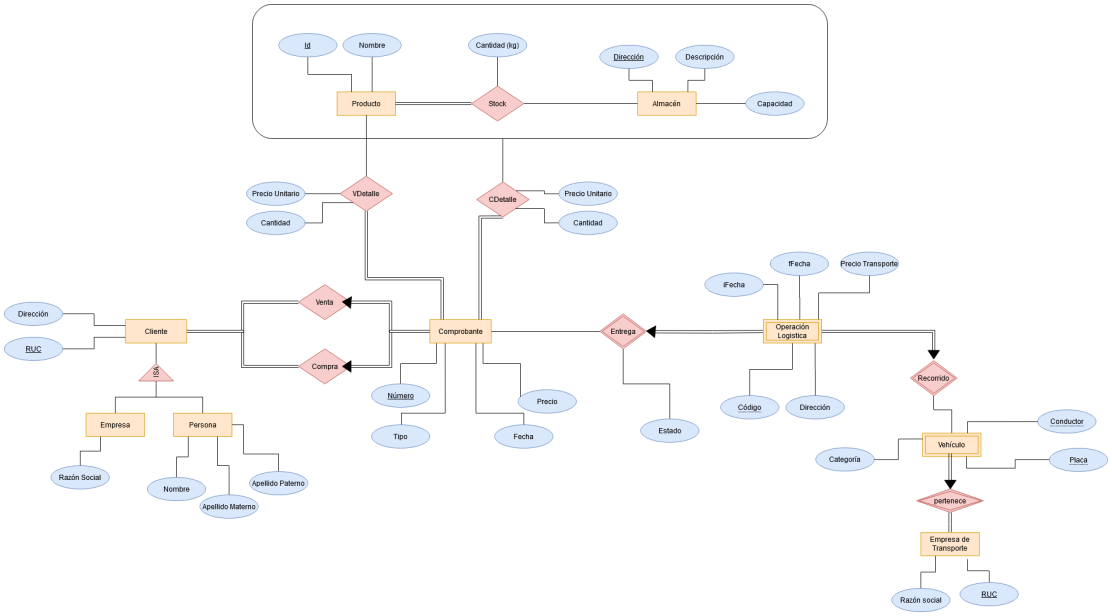
- Las observaciones apuntadas en la sección 5.7 resultan interesantes para analizar más adelante. En conjunto con algunas de las limitaciones observadas en la sección 1.8.2, se da fruto a investigaciones que se podrían llevar a cabo en un próximo estudio.

- El uso de consultas no optimizadas es solo recomendable para usos didácticos.

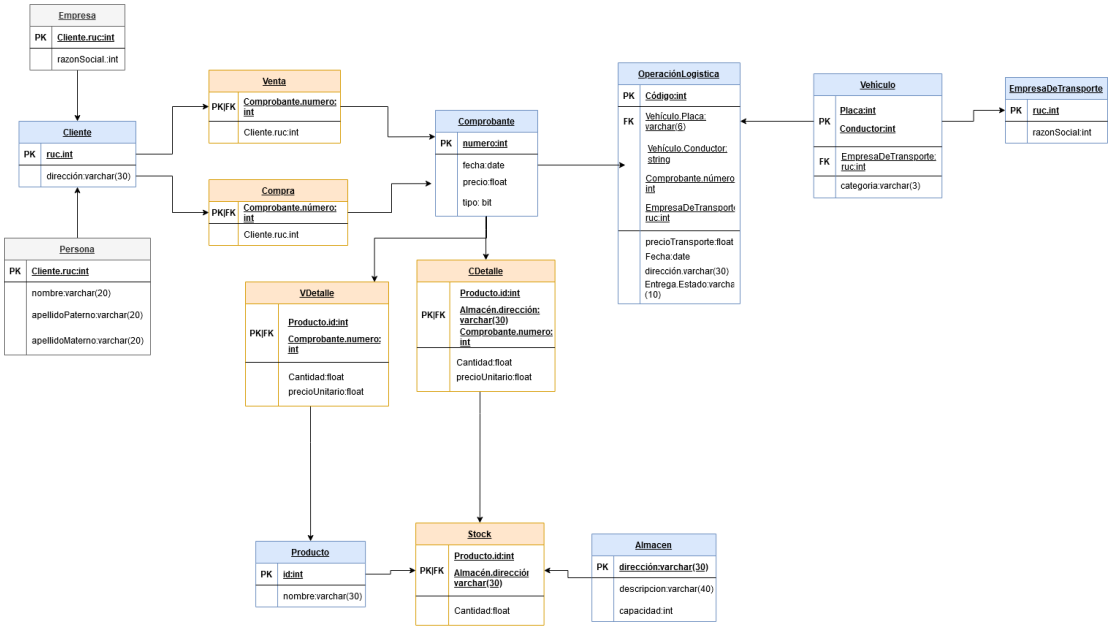
Appendices

A. Anexos

A.1. Modelo ER



A.2. Modelo Relacional



Bibliografía

[1] Ramakrishnan, R., Gehrke, J. (2000). Database Management Systems (2nd Bk & Cdr ed.). Mcgraw-Hill College.