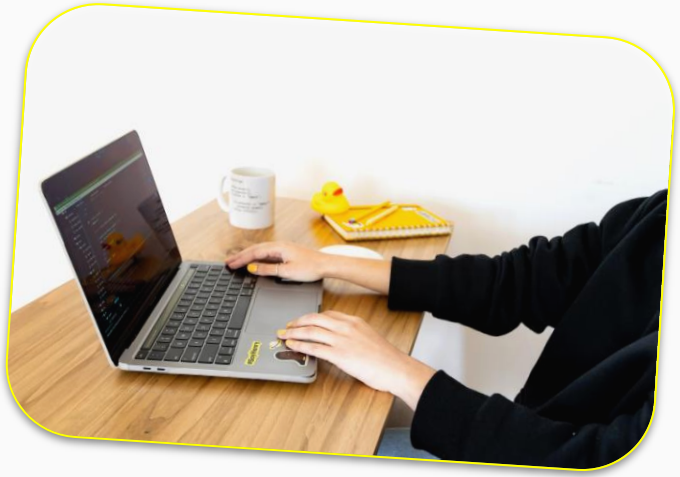
HENRY

Query optimization

Data Science

Agenda



- → Query Optimization
- → Estadísticas de consultas
- → SQL Indíces

OBJETIVOS DE LA CLASE

Al finalizar esta lecture estarás en la capacidad de...

- → **Conocer** el concepto de índices en bases de datos y Formas Normales
- → **Identificar** los diferentes esquemas de Modelos de Datos
- → **Optimizar** las consultas

Optimización de consultas

La optimización de consultas es de gran importancia para el rendimiento de una base de datos relacional, especialmente para la **ejecución de sentencias SQL** complejas. Un **optimizador de consultas** decide los mejores métodos para implementar cada consulta.

Ejemplo

Selecciona, por ejemplo, si desea o no usar índices para una consulta determinada y qué métodos de unión usar al unir varias tablas. Estas decisiones tienen un tremendo efecto en el rendimiento de SQL,

Algunas consideraciones

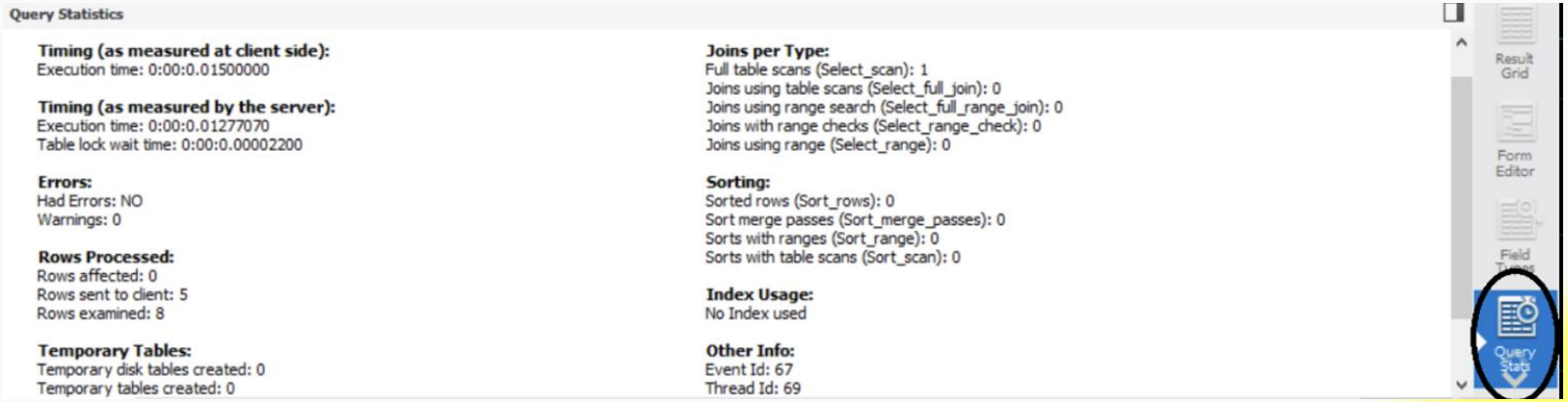
- Configuración de índices
- Minimización de análisis de tablas completas
- Investigación de la documentación del SGBD

Por lo general LEFT (Nombre) = 'Rod' es menos performante que LIKE 'Rod%'

Estadísticas de consultas

La ficha **Resultados** del editor SQL posee Estadísticas de consulta que utiliza datos del esquema de rendimiento para recopilar métricas clave para la consulta ejecutada, como la sincronización, las tablas temporales, los índices, las combinaciones y mucho más.
Las **estadísticas en MySQL** se pueden consultar en el margen derecho de la pantalla de resultados, mediante la opción "Query Stats".

Ejemplo

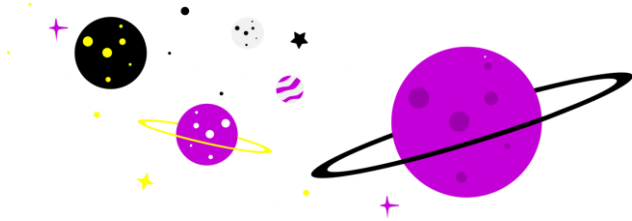

Query Statistics
Timing (as measured at client side):
Execution time: 0:00:0.01500000
Timing (as measured by the server):
Execution time: 0:00:0.01277070
Table lock wait time: 0:00:0.00002200
Errors:
Had Errors: NO
Warnings: 0
Rows Processed:
Rows affected: 0
Rows sent to client: 5
Rows examined: 8
Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0
Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0
Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0
Index Usage:
No Index used
Other Info:
Event Id: 67
Thread Id: 69
Result Grid
Form Editor
Query Stats

Plan de explicación visual

- Visualización de consultas con MySQL Workbench
- Formatos de consulta en MySQL Workbench
- Instalación del complemento
- Beneficios de la visualización y el análisis

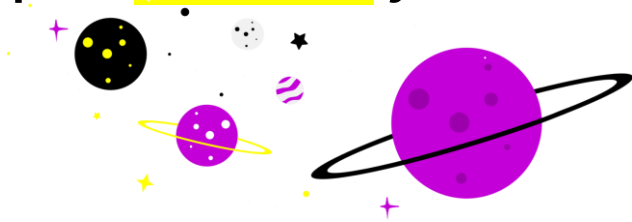
Visualización de consultas con MySQL Workbench

- MySQL Workbench ofrece una característica de explicación visual.
- Genera y muestra una representación visual de la instrucción MySQL EXPLAIN.
- Utiliza información extendida disponible en el formato JSON extendido.



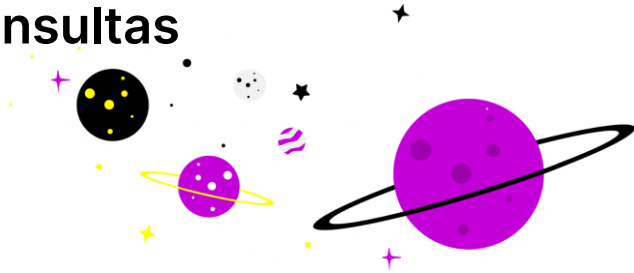
Formatos de consulta en MySQL Workbench

- MySQL Workbench proporciona varios formatos para las consultas ejecutadas.
- Incluye el JSON extendido sin procesar, el formato tradicional y el plan de consulta visual.
- Estos formatos brindan opciones flexibles para visualizar y analizar consultas.



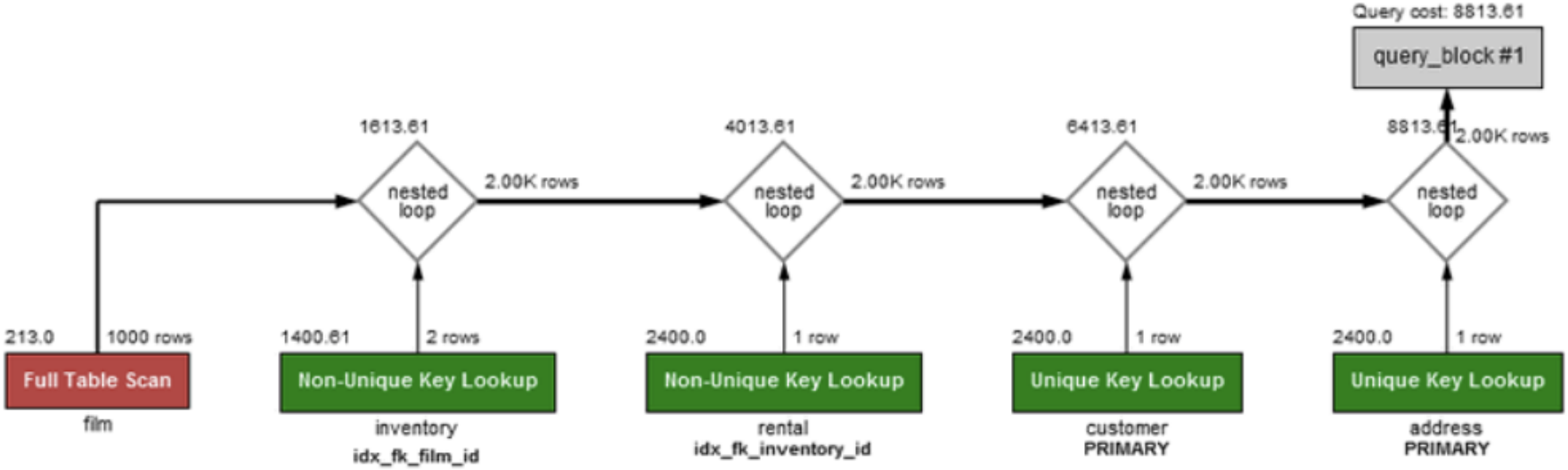
Instalación del complemento

- El complemento para la visualización de **EXPLAIN** en JSON extendido debe ser instalado en MySQL Workbench.
- Proporciona **funcionalidades adicionales** para aprovechar la información extendida.
- **Mejora** la comprensión y análisis de las consultas ejecutadas.



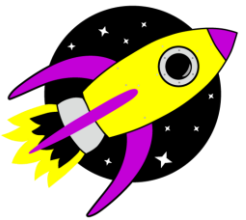
Beneficios de la visualización y el análisis

- **Visualización** clara y comprensible de la estructura y el plan de ejecución de consultas.
- **Identificación** de posibles cuellos de botella y áreas de optimización.
- **Mejora** de la eficiencia y el rendimiento en la optimización de consultas.

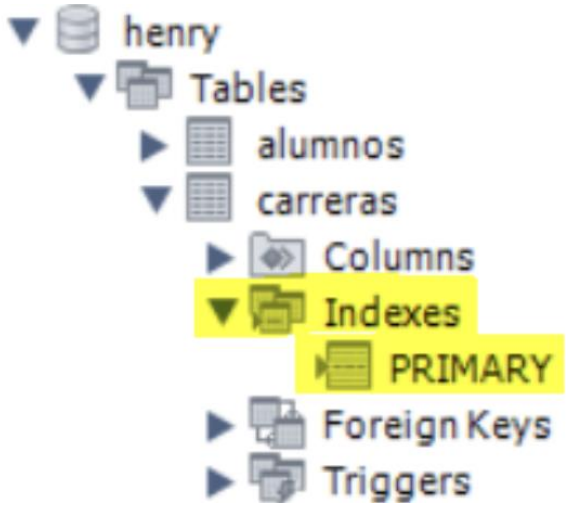

Query cost: 8813.61
query_block #1
1613.61
4013.61
6413.61
8813.61
2.00K rows
nested loop
2.00K rows
nested loop
2.00K rows
nested loop
2.00K rows
nested loop
213.0
1000 rows
Full Table Scan
film
1400.61
2 rows
Non-Unique Key Lookup
inventory
idx_fk_film_id
2400.0
1 row
Non-Unique Key Lookup
rental
idx_fk_inventory_id
2400.0
1 row
Unique Key Lookup
customer
PRIMARY
2400.0
1 row
Unique Key Lookup
address
PRIMARY

SQL-INDÍCES

- Los **índices de SQL** son la principal herramienta de rendimiento, por lo que generalmente se aplican si una base de datos se incrementa.
- El **motor SQL** reconoce varios tipos de índices, pero uno de los más comunes es el índice agrupado. Esta clase de índice se crea automáticamente con una PK. En el momento en el que se ejecuta la consulta, el motor SQL creará automáticamente un índice agrupado en la columna especificada.



```sql
CREATE TABLE carrera (
        idCarrera INT NOT NULL AUTO_INCREMENT,
        nombre VARCHAR (20) NOT NULL,
        PRIMARY KEY (idCarrera) --Aquí al crear una PK, SQL además crea un índice agrupado.
);
```

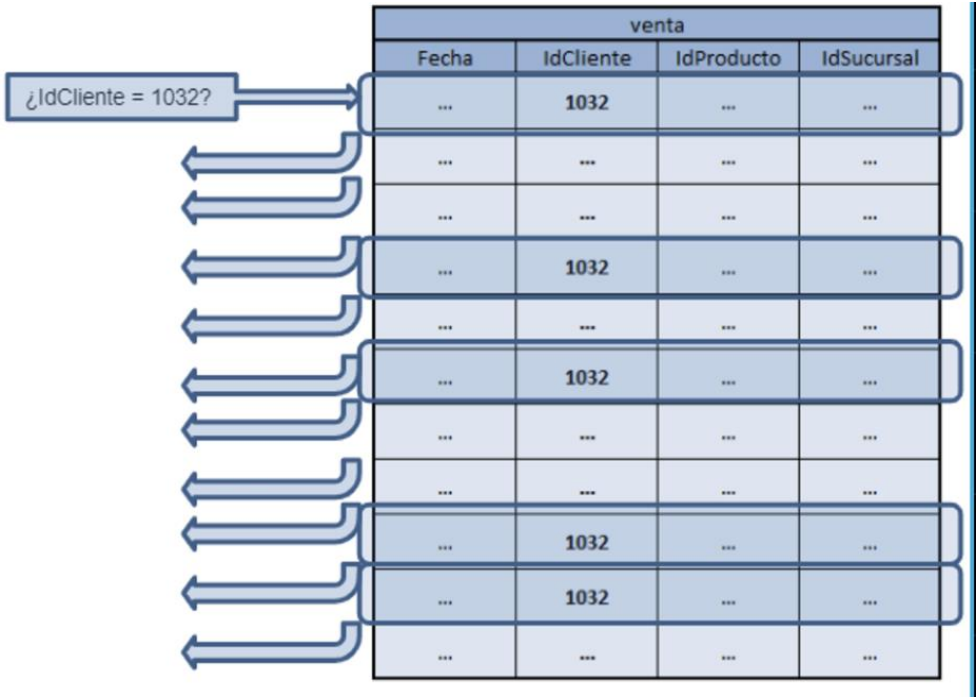


Los **índices de las tablas** ayudan a indexar el contenido de diversas columnas para facilitar la búsquedas de contenido de cuando se ejecutan consultas sobre esas tablas.

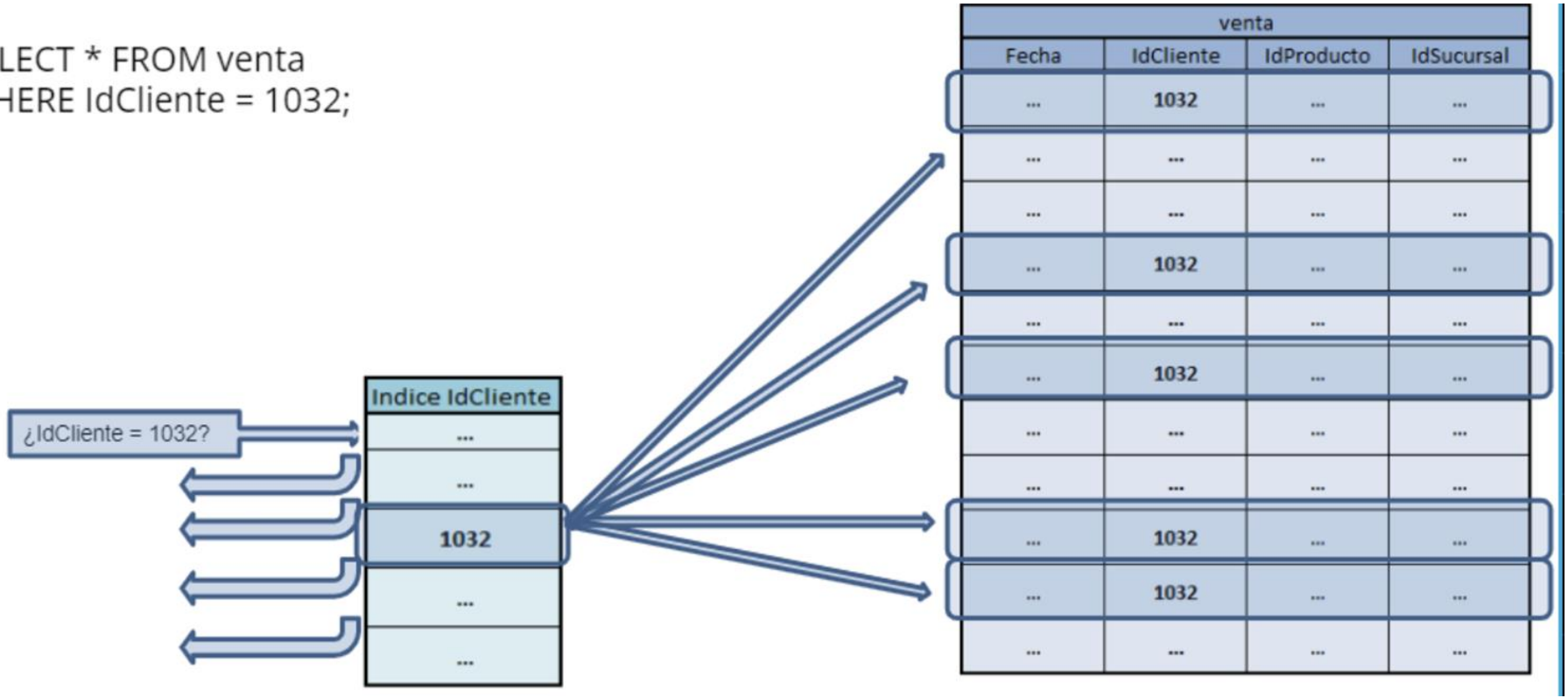
En MySQL puede utilizarse **CREATE INDEX** para crear o añadir índices en las tablas de una base de datos.

```mysql
CREATE INDEX nombre_indice ON nombrede_tabla(columna [columna2...]);
```

```
SELECT * FROM venta
WHERE IdCliente = 1032;
```



SELECT * FROM venta
WHERE IdCliente = 1032;

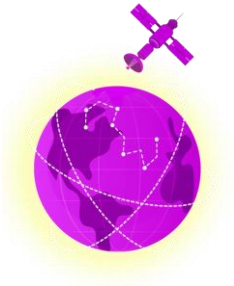


Podemos tener los siguientes tipos de índices en una tabla de MySQL:

- Únicos.
- Primarios.
- Ordinarios.
- De texto completo.
- Parte de campos o columnas.

También se pueden **eliminar** índices mediante la sentencia DROP INDEX, siempre teniendo presente que la utilización de la sentencia DROP puede llevar a consecuencias indeseadas.

```mysql
DROP INDEX 'segundo_apellido' ON clientes
DROP INDEX 'PRIMARY' ON clientes;
```



Optimización en WHERE

Un tema en la optimización es un enfoque constante en la cláusula WHERE. **¡Cuanto más rápido podamos dividir nuestro conjunto de datos a solo las filas que necesitamos, más eficiente será la ejecución de la consulta!**

Ejemplo:

```
#Utilizamos IN para crear un conjunto en la segmentación
SELECT *
FROM instructores
WHERE idInstructor IN (1,2,3,4,5)

#Utilizamos OR para crear el mismo conjuntos
SELECT *
FROM instructores
WHERE idInstructor = 1 OR idInstructor = 2 OR idInstructor = 3
OR idInstructor = 4 OR idInstructor = 5
```

Formas Normales

Las **formas normales** son **conjuntos de criterios** que utilizamos para diseñar la estructura de las bases de datos. Para mejorar el desempeño de una base de datos, así como evitar redundancia en la información que contiene y, en consecuencia, generar condiciones para un **mejor diseño,** se deben conocer las formas de normalización y condiciones en las que la desnormalización es recomendable.

FN1

Una relación se encuentra en 1FN sólo si cada uno de sus atributos contiene un único valor para un registro determinado.

Codigo	Nombre	Apellido	Sucursal
3467	Daniel Arnoldo	Abib	Avellaneda, Caseros y Córdoba Quiroz
3243	Hugo Briz	Abilio	Quilmes y Velez
3124	Jaime Gaston	Acevedo Salinas	Caballito, Caseros y Quilmes

Para corregirlo, se crean dos tablas, donde vamos a poder ver que cada registro guarda un solo valor. De esta manera, el esquema cumple la 1FN.

Codigo	Nombre	Apellido
3467	Daniel Arnoldo	Abib
3243	Hugo Briz	Abilio
3124	Jaime Gaston	Acevedo Salinas

Codigo	Sucursal
3467	Avellaneda
3467	Caseros
3467	Córdoba Quiroz
3243	Quilmes
3243	Velez
3124	Caballito
3124	Cabildo
3124	Mendoza
3124	Moron

FN2

Una relación se encuentra en 2FN sólo si se cumple 1FN y todos sus atributos no clave dependen en forma completa de la clave.
Supongamos que tenemos una tabla donde guardamos cuántas ventas se hizo a cada cliente en cada sucursal, y contamos con un esquema como el de la imagen.
La clave de esta tabla, está formada por los campos Codigo_Cliente y Codigo_Sucursal y la relación se encuentra en 1FN, pero:

- Apellido_Nombre sólo depende de Codigo_Cliente.
- Sucursal sólo depende de Codigo_Sucursal.
- Ventas depende de la clave completa Codigo_Cliente + Codigo_Sucursal.
  Por lo que ocurre en 1 y 2 no se cumple con la 2FN.

FN2

Codigo_Cliente	Codigo_Sucursal	Apellido_Nombre	Sucursal	Ventas
3467	101	Abib, Daniel Arnoldo	Avellaneda	3
3467	103	Abib, Daniel Arnoldo	Caseros	2
3467	106	Abib, Daniel Arnoldo	Córdoba Quiroz	5
3243	107	Abilio, Hugo Briz	Quilmes	6
3243	110	Abilio, Hugo Briz	Velez	2
3124	109	Acevedo Salinas, Jaime Gaston	Caballito	3
3124	103	Acevedo Salinas, Jaime Gaston	Caseros	1
3124	107	Acevedo Salinas, Jaime Gaston	Quilmes	2

FN2

Para corregirlo, debemos llegar a un esquema de 3 tablas como el que se puede observar. De esta manera sí se cumple la 2FN para el esquema.

Codigo_Cliente	Apellido_Nombre
3467	Abib, Daniel Arnoldo
3243	Abilio, Hugo Briz
3124	Acevedo Salinas, Jaime Gaston

Codigo_Cliente	Codigo_Sucursal	Ventas
3467	101	3
3467	103	2
3467	106	5
3243	107	6
3243	110	2
3124	109	3
3124	103	1
3124	107	2

Codigo_Sucursal	Sucursal
101	Avellaneda
103	Caseros
106	Córdoba Quiroz
107	Quilmes
110	Velez
109	Caballito
103	Caseros
107	Quilmes

FN3

Una relación se encuentra en 3FN sólo si se cumple 2FN y los campos no clave dependen únicamente de la clave o los campos no clave no dependen unos de otros.
Supongamos que tenemos una tabla donde guardamos datos filiatorios de clientes que tienen que ver con la Localidad y Provincia en que viven y tenemos la estructura de la imagen.
Como se puede observar, surgen las siguientes dependencias:

- Codigo_Cliente Apellido_Nombre.
- Codigo_Cliente Localidad.
- Codigo_Cliente Provincia.
  Aunque cumple con la 2FN, la Provincia está también ligada a la Localidad, con lo que no se cumple la 3FN.

Codigo_Cliente	Apellido_Nombre	Localidad	Provincia
3467	Abib, Daniel Arnoldo	Cordoba	Cordoba
3243	Abilio, Hugo Briz	Quilmes	Buenos Aires
3124	Acevedo Salinas, Jaime Gaston	CABA	Buenos Aires

Para corregirlo, debemos llegar a un esquema de 2 tablas como el que se puede observar. De esta manera se cumple la 3FN para el esquema.

Codigo_Cliente	Apellido_Nombre	Localidad
3467	Abib, Daniel Arnoldo	Cordoba
3243	Abilio, Hugo Briz	Quilmes
3124	Acevedo Salinas, Jaime Gaston	CABA

Localidad	Provincia
Cordoba	Cordoba
Quilmes	Buenos Aires
CABA	Buenos Aires

FN4

Una relación se encuentra en **4FN** sólo si se cumple **3FN** y no posee dependencias multivaluadas no triviales.

Supongamos que tenemos un esquema como el de la imagen, donde guardamos que una sucursal vende un determinado producto mediante un canal de venta.
Notemos que debido a que la tabla tiene una clave única y ningún atributo no clave, no viola ninguna forma normal hasta la **3FN**. Pero debido a que los canales de venta de una sucursal son independientes de los productos que vende, hay redundancia en la tabla: por ejemplo vemos tres veces que Caballito vende OnLine. Esto se describe como que **Canal de Venta** está teniendo una dependencia multivalor en Sucursal e impide que se cumpla con la 4FN en la relación.

Sucursal	Canal de Venta	Producto
Avellaneda	OnLine	Parlante Jbl Go Blue Bluetooth
Avellaneda	Presencial	Parlante Jbl Go Blue Bluetooth
Caseros	Presencial	Parlante Kingta So-101 Bluetooth/ Waterproof Red
Caseros	Presencial	Parlante Jbl Go Blue Bluetooth
Caballito	OnLine	Parlante Jbl Go Blue Bluetooth
Caballito	OnLine	Parlante Kingta So-101 Bluetooth/ Waterproof Red
Caballito	OnLine	Parlante Jbl Flip 4 Gray Bluetooth

Para corregirlo, debemos poner los hechos sobre los Canales de Ventas por los que se vende en una tabla diferente a los hechos sobre los productos que se vende. De esta manera se cumple la **4FN** para el

Sucursal	Canal de Venta
Avellaneda	OnLine
Avellaneda	Presencial
Caseros	Presencial
Caballito	OnLine

Sucursal	Producto
Avellaneda	Parlante Jbl Go Blue Bluetooth
Caseros	Parlante Kingta So-101 Bluetooth/ Waterproof Red
Caseros	Parlante Jbl Go Blue Bluetooth
Caballito	Parlante Jbl Go Blue Bluetooth
Caballito	Parlante Kingta So-101 Bluetooth/ Waterproof Red
Caballito	Parlante Jbl Flip 4 Gray Bluetooth

FN5

Una relación se encuentra en 5FN sólo si se cumple 4FN y cada dependencia de unión en ella es implicada por las claves candidatas.

Sucursal	Canal de Venta	Producto
Avellaneda	OnLine	Parlante Jbl Go Blue Bluetooth
Avellaneda	Presencial	Parlante Jbl Go Blue Bluetooth
Caseros	Presencial	Parlante Kingta So-101 Bluetooth/ Waterproof Red
Caseros	Presencial	Parlante Jbl Go Blue Bluetooth
Caballito	OnLine	Parlante Jbl Go Blue Bluetooth
Caballito	OnLine	Parlante Kingta So-101 Bluetooth/ Waterproof Red
Caballito	OnLine	Parlante Jbl Flip 4 Gray Bluetooth

Siguiendo con el ejemplo anterior, la relación también será válida para la 5FN si existe una regla en el escenario real que limite una relación de una Canal de Venta con un Producto al no ser vendido ese Producto por la Sucursal. El hecho de que no haya forma de limitar las combinaciones inválidas del mundo real, **limita el cumplimiento de la 5FN.**

Para corregirlo, además de poner en tablas diferentes, las relaciones posibles Sucursal-Canal de Venta y Sucursal-Producto, debemos hacer lo propio con la relación Canal de Venta-Producto. De esta manera se cumple la 5FN para el esquema.

Sucursal	Producto
Avellaneda	Parlante Jbl Go Blue Bluetooth
Caseros	Parlante Kingta So-101 Bluetooth/ Waterproof Red
Caseros	Parlante Jbl Go Blue Bluetooth
Caballito	Parlante Jbl Go Blue Bluetooth
Caballito	Parlante Kingta So-101 Bluetooth/ Waterproof Red
Caballito	Parlante Jbl Flip 4 Gray Bluetooth

Sucursal	Canal de Venta
Avellaneda	OnLine
Avellaneda	Presencial
Caseros	Presencial
Caballito	OnLine

Canal de Venta	Producto
OnLine	Parlante Jbl Go Blue Bluetooth
Presencial	Parlante Jbl Go Blue Bluetooth
Presencial	Parlante Kingta So-101 Bluetooth/ Waterproof Red
OnLine	Parlante Kingta So-101 Bluetooth/ Waterproof Red
OnLine	Parlante Jbl Flip 4 Gray Bluetooth

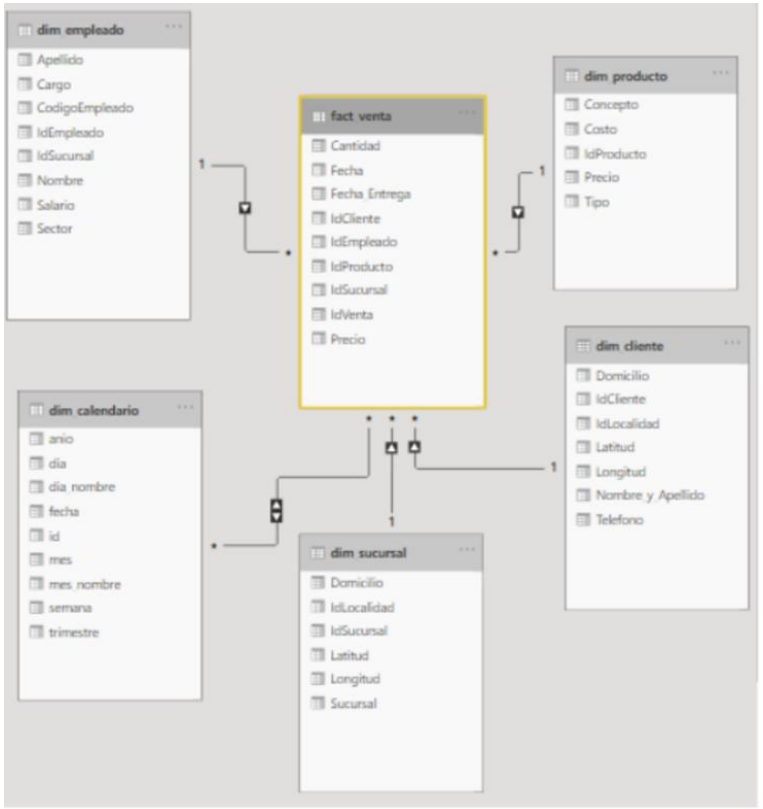
Modelos de Datos

Los **modelos de datos** buscan representar una realidad que es posible representar mediante las entidades que la conforman. Esas entidades quedan representadas en tablas, y pueden ser de dos tipos:

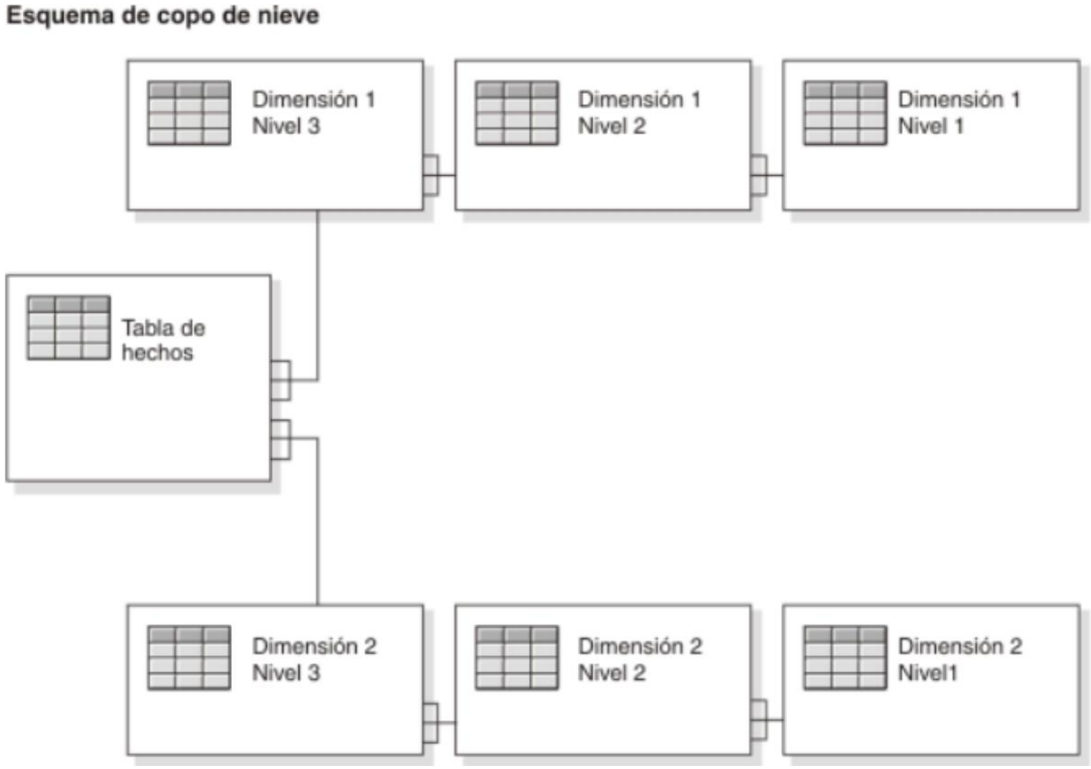
Modelos de estrella

Modelos de copos de nieve

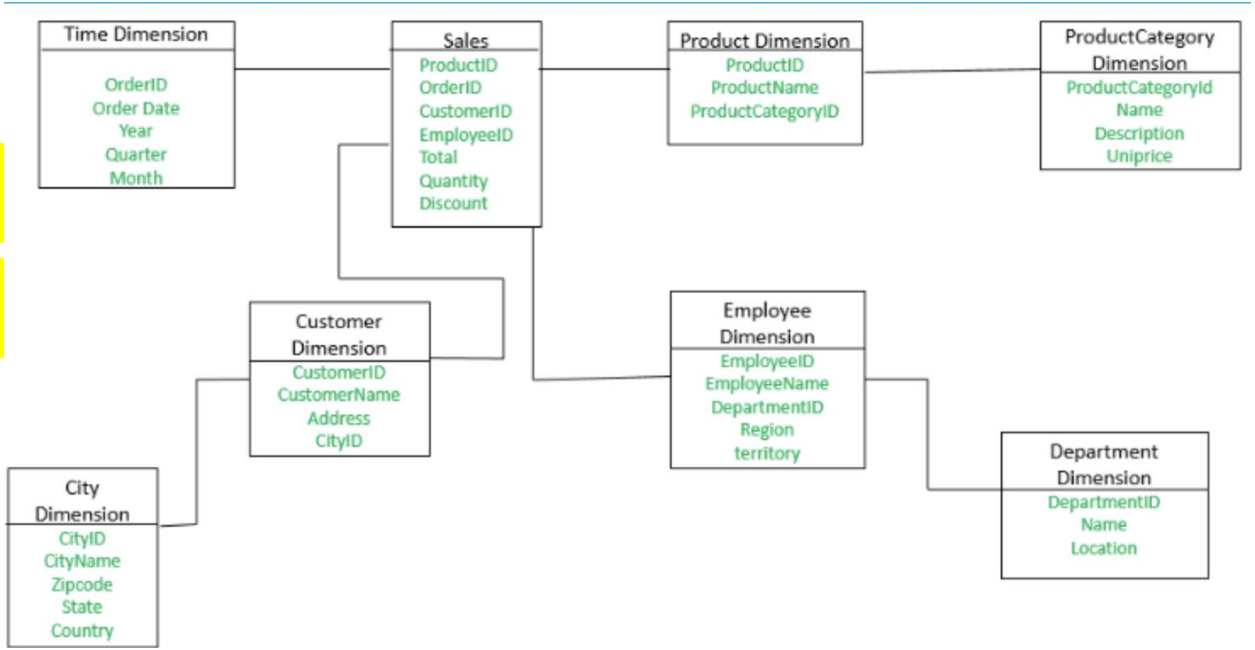
Modelo Estrella



Modelo Copo de nieve



Modelo Copo de nieve




HENRY