SQL

1.- Definición: Literalmente significa **Structured Query Language (SQL)**; es decir, **Lenguaje de Consulta Estructurado,** pero ¿qué diantres es eso?, ¿eso con qué se come?

Para comenzar a comprender que significa **SQL**, debemos ir por partes; en este caso que vamos a iniciar por comprender que significa **Lenguaje** en el contexto SQL veamos lo siguiente: Cuando dos o más personas necesitan comunicarse, deben usar una serie de símbolos y reglas en común; a esas reglas comunes se le llaman **lenguaje o idioma**. Así, si estamos en el Reino Unido, usamos Ingles; si vivimos en Holanda, pues usamos holandés; pero si vivimos en Venezuela, Colombia o Argentina usamos el español o castellano como idioma o lenguaje.

En nuestro contexto de base de datos relacionales entendemos que ese **lenguaje** son todas aquellas reglas y símbolos que nos permiten a nosotros como usuarios interactuar o comunicarnos con el interface del SMBD / SGBD y por ende con la base de datos.

Ahora bien, ¿por qué se llama **estructurado**?; aquí es una cuestión tanto de historia como acercamiento al usuario. Me explico, en los primeros momentos de la ciencia de la computación, el método para ingresar datos e interactuar con un sistema de cómputo fue a través de presionar muchos interruptores. Solo los genios pertenecían a este exclusivo club.

En este punto la necesidad de hacer que los sistemas de cómputo fueran más cercanos a las personas obligo a crear interfaces de comunicación más prácticas y fáciles con el usuario; aquí es donde el teclado se unió al PC en un hermoso matrimonio ***in saecula saeculorum*** (**"por los siglos de los siglos"**); es decir, el teclado creo una capa de abstracción entre el usuario y el lenguaje natural de los computadores que es de Ceros (o) y Unos (1) (Binario).

En este momento para lograr algún tipo de resultado de los computadores, como por ejemplo un reporte, era necesaria la mente brillante y la mano experta de un programador para que él construyera un programa de 7000 líneas de código o más, programa que era construido en algún lenguaje de programación como Cobol, Fortran o C. Este

```
/*Lectura y valodacion del Vector B*/
printf("\n\n\n\n\n\nCaptura de datos del Vector B");
for(i=0;i<=n;i++){
    do{
        printf("\nPosicion---->%d  de  %d",i,n);
        do{
            printf("\nValor B--->");
            scanf("%d",&vb[i]);
        }while(!vb[i]>=1);
        enc=0;
        k=0;
        while(k<=i-1 && enc==0){
            if(vb[i]==vb[k]){
                enc=1;
            }
            k=k+1;
        }
    }while(enc==1);
}

/*Suma de los dos Vectores en Vector C*/
for(i=0;i<=n;i++){
    vc[i]=va[i]+vb[i];
}
```

programa se comunicaba con los archivos de un o varias Tablas con el objetivo de generar un reporte para el usuario. Todavía seguía siendo un proceso engorroso y dependiente.

A Dios gracias, para todos nosotros… primero en IBM y luego en ORACLE hubo un grupo de personas que se le prendió el bombillo y construyeron toda una infraestructura basada en comandos y parámetros para administrar Bases de Datos; donde el usuario simplemente tenía que comprender qué hace el comando y qué parámetros necesita para funcionar correctamente. En este contexto, lo que antes eran programas, programas y más programas….; ahora se convirtieron en simples comandos que el usuario aplica; es decir, se convirtieron en bloques de códigos asociados a un nombre que recibían uno o varios datos para que luego dicha **estructura** realice una operación como puede ser: crear una tabla, insertar unos registros o mostrar unos datos.

En otras palabras, todas las operaciones para administrar los datos de una BD **se estructuraron en bloques de ordenes (Comandos)**, por eso se llama **estructurado.**

```
-- TMEMPLEADOS
SELECT 'Paso 08: Trabajando con TMEMPLEADOS.....'  AS paso, pg_sleep(05);
CREATE  TABLE   tmempleados (
                cedemple        varchar(12)  not null PRIMARY KEY,
                nomemple        varchar(40)  not null,
                fecha           date         not null,
                fkcodcar        integer      not null,
                fkcods          integer      not null DEFAULT 1,
                foreign  key(fkcodcar)  references  tmcargos(codcar)
                                on  update  cascade    on  delete  restrict,
                foreign  key(fkcods)  references  tmstatus(cods)
                                on  update  cascade    on  delete  restrict) ;

INSERT INTO  tmempleados (cedemple,       nomemple,   fecha,   fkcodcar)
       VALUES
                ('1000' , 'ROBERTO JAIMES' , '1978-03-03' , 3),
                ('2000' , 'ZILA CONTRERAS' , '1980-10-05' , 2),
                ('3000' , 'MARTHA'         , '1996-11-22' , 3),
                ('4000' , 'CARLOS'         , '2000-07-15' , 4),
                ('5000' , 'MATIAS'         , '2005-09-25' , 4),
                ('6000' , 'NATASHA'        , '2007-10-28' , 1),
                ('7000' , 'PEGGY CARTER'   , '1999-08-29' , 2),
                ('8000' , 'YSA CAPACHO'    , '1993-07-23' , 2),
                ('9000' , 'MARIA'          , '2003-07-07' , 4);

SELECT    *    FROM    tmempleados ;
```

Fe de erratas: *No he dicho que los valientes, creativos, proactivos, disciplinados trabajadores y siempre incomprendidos programadores hayan desaparecido como los dinosauros de la faz de la tierra.*

Nada más lejos de la realidad, es importante recordar, ellos(Los programadores) crearon toda la infraestructura para que el usuario trabaje simplemente invocando comandos, en una forma más cómoda, sin estrés y sin la presencia constante de un programador.

Para muestra de esta realidad vale un botón, por ejemplo: recuerda los prompt que UD escribe en chatGP, Gemini o cualquier Inteligencia Artificial (IA); detrás de ellos existen muchos programadores que analizaron como UD piensa y hace una petición al computador desde el punto de vista de un lenguaje natural. Luego, Los programadores diseñaron y desarrollaron un comando llamado "Prompt", el cual UD escribe en la pantalla de la IA para recibir una respuesta bastante condensada de su petición.

En fin, lo que sucede es que estos abnegados trabajadores tecnológicos desde hace tiempo cambiaron su lugar de trabajo a un sitio menos expuesto, ahora su oficina está detrás de las cámaras.

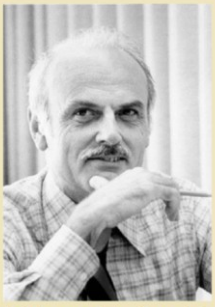
Finalmente, ¿qué se debe entender por "**de Consulta**"?; pues como su nombre lo indica el objetivo de todo este "**Lenguaje Estructurado" es hacer consultas**; en otras palabras, es presentarle al usuario una herramienta para extraer los datos de una base de datos en forma ordenada, simple y lo menos traumática posible a través de comandos.

```
bdretardados=# SELECT * FROM TMEMPLEADOS;
 cedemple | nomemple |   fecha    | fkcodcar | fkcods
----------+----------+------------+----------+--------
 1000     | ROBERTO  | 1978-03-03 |        1 |      1
 2000     | MARTHA   | 1980-10-05 |        2 |      1
 3000     | SANDRA   | 1996-11-22 |        3 |      1
 4000     | WILMARY  | 2000-07-15 |        4 |      1
 5000     | CARLOS   | 2005-09-25 |        4 |      1
 6000     | NATASHA  | 2007-10-28 |        4 |      1
 7000     | MARIA    | 1999-08-29 |        4 |      1
(7 filas)
```

2.- Historia:

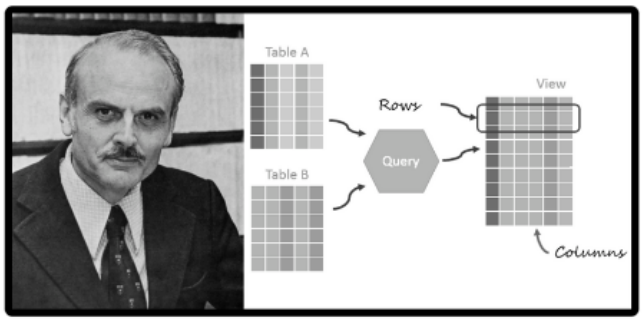
Alguien escribió en algún momento lo siguiente*: "La historia de la humanidad está llena de personas, eventos, situaciones, circunstancias y paradojas curiosas que marcan el camino";* aquí inicia uno de esos hechos curiosos de la historia, donde los protagonistas son IBM, ORACLE y el Software Libre (Open Source).

La historia de SQL comenzó en 1969, cuando el **investigador de IBM, el Dr. Edgar F. Codd** definió el modelo de base de datos relacional. Este modelo se basó en la asociación de «claves» entre tablas. Por ejemplo: un ID se puede asociar con un nombre real, una dirección, un número telefónico o una fecha de nacimiento de una persona.

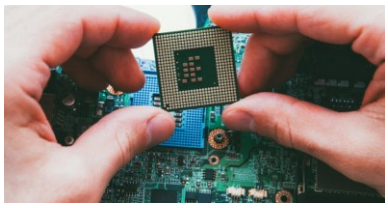

Edgar F. Codd

Para **1970** el **Dr. Edgar F. Codd** publicó en la revista IBM Journal of Research and Development (Revista de Investigación y Desarrollo de IBM) un artículo llamado *"Modelo relacional de datos para grandes bancos de datos compartidos (A Relational Model of Data for Large Shared Data Banks)".* En este artículo, **Codd** presentó las bases teóricas del modelo relacional y describió una versión preliminar del lenguaje SQL. En otras palabras, Cood expuso que los datos se deben guardar en varias tablas y dichas tablas se asocian por medio de relaciones definidas por el diseñador. En retrospectiva lo brillante de Codd fue que aplico con extrema maestría la teoría de

conjuntos de la matemática a los datos; en la práctica los datos los convirtió en tablas y relaciones.



En 1977 aparece comercialmente la Computadora Personal APPEL II (Microsoft), la cual fue basada en **microprocesadores.** El microprocesador, esta pequeña pasta de



silicio con el tiempo permitió que las BDs y todos sus aplicativos llegaran a la estratosfera; pues puso al alcance de todos nosotros los humildes mortales una herramienta capaz de procesar y entregar resultados en segundos; lo que a la mente humana le podía llevar días, semanas, meses e incluso hasta años.

El Microprocesador, también permitió a las almas de espíritu libre desarrollar programas y soluciones creativas a problemas complejos desde la comodidad de su hogar, donde muchas de esas almas de espíritu libre compartieron con la comunidad sus soluciones sin pedir nada a cambio(Los primeros pasos del Software libre).

Paralelamente en 1977; en Silicon Valley los ingenieros: **Larry Ellison, Ed Oates y Bob Miner**, **fundan** una empresa de consultoría llamada ***Software Development Laboratories*** (***SDL, léase ORACLE***); ellos desarrollaron un sistema especial de bases de datos **basaron en las ideas y conceptos presentados por Edgar F. Codd en su artículo de la revista IBM del año 1970**. Hasta este momento nada extraordinario.

Ahora bien, ***Software Development Laboratories*** (***SDL*****, léase ORACLE)** desde el principio centro la visión y misión de empresa en:

1.- Desarrollar proyectos que fueran compatibles con las ideas del modelo relacional SQL de IBM que presentó **Edgar F. Codd** en su artículo de 1070.

2.- Orientar las ventas al mercado de las Minicomputadoras y Microcomputas (Las PC). Este mercado, para las décadas de 1070 y 1980 estaba naciendo. (*Aquí es donde viene uno de los eventos paradójicos en la historia de las BDs*); IBM considero en ese momento como poco prometedor el mercado de las Microcomputadoras. OJO OJO, no dije que IBM abandonara el mercado, dije “poco prometedor”; es más, IBM desarrollo el modelo de computadoras personales (PC) pero decidió no desarrollar todo su potencial, IBM prefirió concentrarse en la venta de las supercomputadoras a gobiernos y grandes empresas.

SDL (ORACLE) aprovecho esos puntos **para desarrollar y comercializar sin ninguna competencia a nivel de computadoras medianas y personales (PC) un Sistema Gestor/Manejador de Base de Datos (SGBD/SMBD) altamente operativo**, fácil de usar y robusto.

Todo un hito en la historia de los gestores de bases de datos, pues al pasar de los años; el enfoque, el diseño y modelos de trabajo presentados por ***SDL*** **(ORACLE)** se convirtieron en el gran referente de las bases de datos relacionales del mundo; es decir, **SDL (ORACLE)** desarrolló y **presentó el primer SGDB / SMBD del mundo**, prácticamente sin competencia, lo comercializo en un mercado relativamente virgen; este hecho convirtió a **SDL (ORACLE)** en una de las mayores referencias en BD del mundo y una empresa con mucho éxito.

Ahora bien; para la década de **1980**, **IBM presentó un lenguaje** para los sistemas de gestión de bases de datos relacionales basado en los trabajos de **Codd**. Dicho lenguaje se llamó **SEQUEL (*Structured English Query Language***). Después de varias implementaciones y revisiones pasó a llamarse **SQL**, como lo conocemos hoy en día.

En **1981**, IBM empezó la comercialización del modelo: DS/SQL (1981) y DB2/SQL (1983), los cuales incluyeron a SQL como parte del software de trabajo para manipular datos. Además, en **1981** IBM presentó al mundo la **Computadora Personal (PC)**, la cual se basó en un **microprocesador** tan cual como la conocemos hoy en día. Las PCs fueron muy

populares, al punto que su diseño se convirtió en el estándar del mundo informático e implementado por la gran mayoría de fabricantes en todo el mundo, en esa época era común escuchar a cualquier vendedor de computadoras decir: “Esto es un **clon IBM**”.

Para la década de los años **1980** las computadoras personales basadas en microprocesadores eran lo más popular del mercado. Era también la época donde Bill Gates y Steve Jobs se jalan los cabellos, hasta el punto de que Steve Jobs se va de Apple y le deja el trono la computación personal a Bill Gates.

Bill Gates

Steve Jobs

En **1982**, la empresa ***Software Development Laboratories*** (***SDL)*** cambio de nombre a “**Relational Software Incorporated (RSI)”,** para ajustarse mejor a la visión de la empresa. Luego, en **1984** la empresa cambia nuevamente de nombre a “**Oracle System Corporation”** y al poco tiempo acorto su nombre a “**Oracle Corporation**”, con el tiempo simplemente todos la conocemos como: “**ORACLE**”.

En **1986** la **ANSI** (American National Standards Institute - Instituto Nacional Estadounidense de Normalización) **certifica SQL**. Desde entonces se convirtió en el estándar del mercado para la gestión de bases de datos relacionales; fue tal su éxito que SQL se volvió indispensable

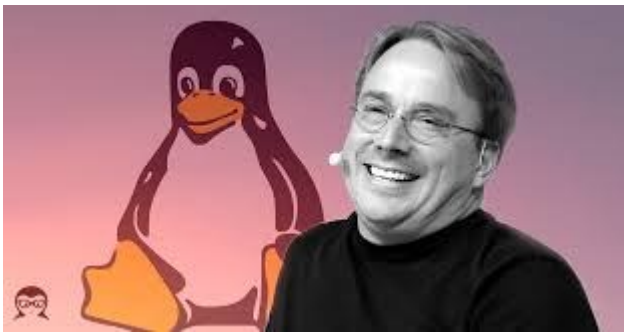
para las empresas que trabajan con grandes volúmenes de datos en el mundo.

Microsoft en **1989** lanza **SQL Server** como respuesta al **SMBD de Oracle**; es más, SQL Server sigue siendo el SGBD relacionales que comercializa Microsoft, el cual está basado en el sistema operativo Windows.

Los años **1990**, eran los años de la explosión de la web y del open source; a partir de este momento se comenzó a hablar de:

- Software bajo licencia o propietario, cuyo mayor exponente sigue siendo la empresa MicroSoft o IBM
- Software libre (open source / de código abierto) cuyo mayor referente es Linux.

El **17/sept/1991** Linus Torvalds público la primera versión de Linux, con lo cual se



abrió la puerta para la implementación de la sinergia en el desarrollo de aplicativos, que van desde la construcción del kernel de sistema operativo; pasando por codificación de software para la edición de texto, imágenes y videos; implementación de nuevos lenguajes de programación como python; y por supuesto, también incursionaron en el mundo de los SMBD. En resumen, todo el anterior desarrollo y mucho más bajo la licencia de software libre; es decir, aplicativos que UD puede instalar en su PC, tener acceso al código fuente para realizar mejoras y volver a compilar sin necesidad de pagar licencia(s) privativas.

En 1995 **Sun MicroSystem** lanzo MySQL el cual fue desarrollado por **Michael "Monty" Widenius. Mysql** fue el primer SGDB de la línea del open source, donde lo más importante de **MySQL** fue que se complementó en forma perfecta con PHP.

En este punto es importante señalar: PHP es un lenguaje embebido en HTML, que permite manipular datos en una forma muy simple; donde desde su aparición se integró en forma perfecta con el SGBD de MySQL.

Es más, hoy en día sigue siendo uno de los matrimonios más estable de la web; aquí simplemente hay que decir: “Es una historia de idilio desde el principio, al estilo de los cuentos de hadas de las películas de Disney”.

Este matrimonio permitió la octava maravilla de la programación, el desarrollo de sitio dinámicos y el crecimiento exponencial de la web. Para verlo en contexto, permitió que UD desde una página web lograra registrar un retiro bancario, registrar el pago de un producto/servicio o recibir el reporte de los movimientos bancarios. Antes del matrimonio **MySQL & PHP** una página web era simplemente una estructura estática; es decir, era una hoja donde UD escribía una vez, donde si UD deseaba cambiar algo en dicha hoja, se debía escribir una nueva hoja y desechar la antigua.

MySQL & PHP permitieron que cualquier cristiano con conocimientos básicos de programación en PHP y HTML, desarrollarán **páginas web dinámicas**, sitios web donde los usuarios se podían registrar, solicitar productos o servicios, hacer seguimiento a pedidos, pedir reportes; permitió la **construcción de sitios web dinámicos** como Amazon, Mercado Libre, Temu, Shein o la página web de su banco de confianza.

Aquí es importante señalar: La explosión de la web fue gracias al desarrollo, compromiso y visión de oportunidad de los millones de programadores de todo el mundo que aprovecharon las ventajas que tiene el software libre, entre ellas tenemos: para UD desarrollar cualquier aplicativo no era necesario pagar ningún tipo de licencia a nadie; pues, las herramientas utilizadas para codificar, como por ejemplo PHP & MySQL eran de uso libre y distribución gratuita. Aquí simplemente hay que decir: “Mucha gente creció, hizo plata y llevo al pan a la mesa de su hogar gracias a estas ventajas”.

En 1996 aparece **PostgreSQL**, el SGBD más robustos de open source hasta la fecha, pues cumple la mayoría de los estándares del mundo de las bases de datos relaciones, e incluso está inspirado en Oracle.

PostgreSQL incorpora a la gestión de tablas y base de datos herramientas para programar funciones, procedimientos, vistas, disparadores, gestión de usuarios, entre muchas otras herramientas de trabajo.

En 2000, cuando aparece SQLite, que es una base de datos relacional muy liviana.

(Aquí viene otro evento paradójico de la historia)

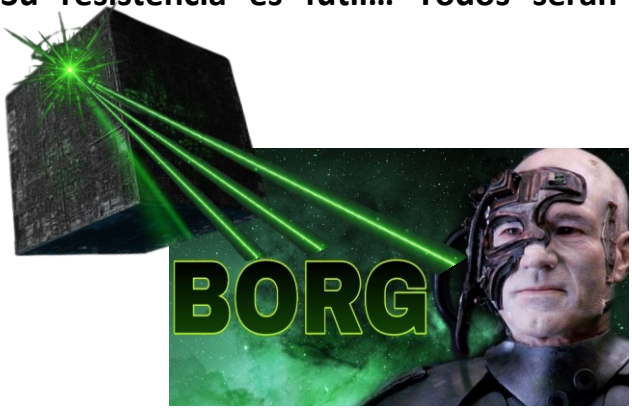
Como ya se mencionó, **MySQL** fue desarrollado por la empresa **Sun Microsystems en 1995**; pero, en abril 2009 se anunció un acuerdo de compra de **Sun** por parte de **Oracle.** Luego, el 27 de enero de 2010 se concretó la firma de la compra de **Sun Microsystems** parte de Oracle. Está *inocente compra* creo "**un disturbio en la fuerza**". En esa época, esa compra – venta abrió la puerta a un tsunami de preocupación en todas las personas que desarrollaban proyectos web dinámicos con MySQL & PHP; pues en el mundo de los programadores de páginas web dinámicas que capturaban, procesaban y almacenaban datos en MySQL nació una onda y seria preocupación.

Todos en el mundo del desarrollo sitios web dinámicos al ver esta ***simple e inocente compra*** sentimos rodar por la espalda un sudor frio…. el cual en la medida que rodaba por la espalda crecía sin parar un profundo escalofrió de terror con los pelos de punta; pues dicha compra, implicaba que para usar MySQL era necesario pagar una licencia; lo que antes era gratis, posiblemente mañana no lo sería.

Para ver esta historia en contexto; cierre los ojos e imagine por un instante que está en un salón oscuro. De pronto de la nada escucha una respiración tosca, metálica y profunda…. luego observa, como va apareciendo muy lentamente un sable de luz **rojo…** poco a poco se va presentado una silueta negra y tenebrosa …. es **Drak Vader (Guerra de las Galaxias)**.

Diga sus últimas oraciones … UD está muerto.

Es la misma sensación que se sintió en el **Universo cinematográfico de Marvel** (MCU: Marvel Cinematic Universe) cuando todos comenzamos a ver los efectos del famoso "**chasquido**" de los dedos de Thanos. O en la película Star Trek, cuando en el comando de la federación en San Francisco se escuchó en los parlantes una voz metálica que decía: "**Somos los Borgs… Su resistencia es fútil… Todos serán asimilados".**



En resumen, esa *inocente compra* de **Sun Microsystems** por parte de **Oracle** representó para todos en el mundo de la programación de **sitios webs dinámicos**, una llamada directa y personal de Oracle en estos términos: "**Mire mijo … no importa si UD se queja, llora, quema cauchos, protesta, pelea, suplica, declara la guerra … a partir de este momento UD debe pagar hasta para hacer el programa 'Hola Mundo' cuando use MYSQL**".

Sin embargo, este "**disturbio en la fuerza**" también llegó a la alta gerencia de **Oracle**; es más, ellos eran conscientes de su responsabilidad en **ese disturbio**. Para **Oracle** este "**disturbio**" representaba un problemita de personal, la mayoría de los genios que programaban en su empresa y en la recién adquirida (Sun Microsystems) eran

pro sotfware libre; es decir, tenían una revolución en puerta, pues ellos veían con muy malos ojos está compra.

Para evitar esa revolución o separación al estilo Martín Lutero, que posiblemente llevara a grandes problemas existenciales o a la quiebra de la empresa Oracle; se optó por una solución salomónica, la cual fue dejar dos versiones de MySQL: la primera, una versión propietaria la cual Oracle manejaría a su antojo; y la segunda, una versión software libre (Open source), la cual dicho sea de paso todavía existe y puede ser usada libremente.

Sin embargo, el daño y la duda todavía persistían; así que el 22 enero 2009 **Michael "Monty" Widenius.** creó una Fundación sin fines de lucro, cuyo objetivo era construir un **SGBD** lo más alejado posible de Oracle – MySQL. Así, en octubre del 2009 nació la primera versión del SMBD llamado **MariaDB**, con lo cual se puso tierra y mar de por medio entre Oracle y cualquier mala intención con el código fuente de MariaBD.

Esta historia también tiene una anécdota medio cicuta; y como ya mencioné "**Paradójica**", pues muchos de los programadores de **ORACLE** que trabajaron en **Sun MicroSystem** en el desarrollo de **MySQL** donaron su tiempo e ingenio para construir

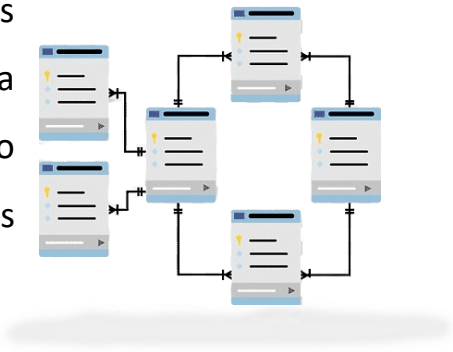
el motor de **MariaBD**. Definitivamente La historia está llena de situaciones, circunstancias, personas y paradojas capciosas.

Después de este recorrido por la historia, surge el siguiente dilema: en esta época

donde ya tenemos cálculos y computadoras cuánticas; programación orientada a objetos; vemos que la inteligencia artificial (IA) nos ayuda a resolver muchos problemas de la vida diaria; es más, donde tenemos la impresión de estar sentados en primera fila para ver el nacimiento de una nueva era en el mundo de la informática y la programación.

Entonces, **¿Debo perder espacio de almacenamiento en mí CPU estudiando SQL?**. La respuesta es: **¡¡¡un claro y rotundo: Sí !!!**; aquí algunas razones de ello:

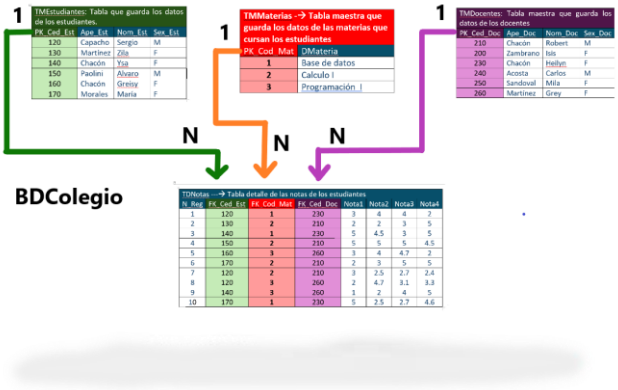
- **Todo el trabajo y operaciones del SQL, se basa en el modelo relacional**; el cual aplica la teoría de conjuntos de la matemática a los datos de un sistema. En este punto simplemente hay decir: la matemática, sus teorías de conjuntos y modelos son lo único constante en el universo, con lo cual los datos modelados allí tendrán larga vida.



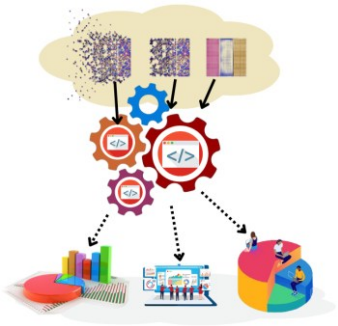
- **Este modelo de trabajo cuida mucho las relaciones y la integridad de los datos**; dicho de otro modo; SQL y el modelo relacional sobre el cual descansa, cuidan en extremo que los datos y sus asociaciones no se pierdan. UD se imagina el caos que surgiría a la velocidad de la luz, si su banco de confianza pierde las relaciones entre los clientes y sus depósitos bancarios…. los gerentes serán las primeras víctimas de este estallido social; a los días, veremos en el rio la sangre del DBA y de los programadores.

- **Para los especialistas en ciencia de datos** los SMBD / SGBD y el lenguaje SQL son herramientas muy importantes pues:
    - Desde su nacimiento **fueron orientadas al almacenamiento y procesamiento de enormes volúmenes de datos**, con lo cual desde hace años son usadas por los analistas para estudiar tendencia, realizar proyecciones, caracterizar datos, obtener reporte y finalmente extrapolar conclusiones que permiten a los gerentes tomar decisiones lo más asertivas posibles.

    - **Los datos almacenados en un SMBD/SGBD relacional están altamente estructurados**, lo cual facilita enormemente su procesamiento y estudio. Si vemos el mundo real, esta rebosante de datos; sin embargo, ellos están desordenados, sin ajustarse a ningún criterio o condición, sin definición de tipo (Entero/ Real/ Carácter), sin forma; de decir, sin estructura. Entonces, ¿qué significa estructurado?; es simplemente, que los datos están almacenados en tablas, columnas, filas y relaciones; como por ejemplo la BDColegio. .



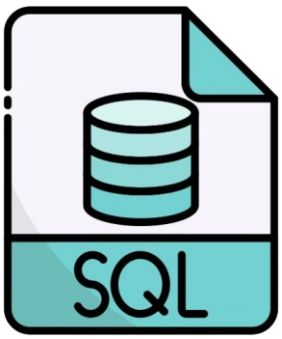
- **Los analistas pueden <u>determinar más rápido los patrones</u>** que existen en los datos que están almacenados en una BD relacional; ello se debe a que dicha data está altamente estructurada, permitiendo a los analistas ganar tiempo, dinero y velocidad en las actividades inherentes en la determinación de los patrones y/o tendencias que están ocultas entre la bruma de los datos de la BD.



3.- Características de SQL

SQL prevalece como el lenguaje de consultas de datos más popular durante las últimas décadas, debido a:

A.- Implementa una sintaxis realmente fácil en sus comandos, lo cual permite a cualquier usuario interpretar y aplicar sus órdenes en forma sencilla y rápida, independientemente del volumen de datos almacenados en el repositorio; es decir, este lenguaje es usado por analistas, diseñadores, programadores, gerentes, administradores, estadísticos, matemáticos, especialistas en marketing, vendedores, especialistas en finanzas, entre otros usuarios debido a su sencillez y comprensibilidad.



B.- Flexibilidad: Los usuarios pueden adaptar los comandos a varias necesidades y consultas, lo que permite encontrar y construir soluciones fáciles a problemas complejos.

C.- Se orienta a mantener la integridad de datos: Busca proteger y garantizar que las relaciones(asociaciones) entre los datos no se rompan.

D.- Es un lenguaje certificado por la ANSI y la ISO (International Organization for Standardization): El sello de aprobación de estas entidades internacionales garantiza calidad y seguridad de uso.

E.- Es el lenguaje usado por excelencia para consultar datos, pues les permite a los científicos de datos simplificar el código usado para extraer los datos, ahorra tiempo de trabajo, reducir costos, y sobre todo ayudar a los analistas a centrar sus esfuerzos en las actividades más importante que un gerente debe asumir, como lo son:

- El análisis de los resultados.
- La búsqueda de oportunidades de negocio.
- La definición de planes y programas de crecimiento.
- Tomar las decisiones lo más asertivas posibles.
- La supervisión y control de procesos.

F.- SQL permite desarrollar aplicaciones fáciles de escalar; es decir, SQL permite construir soluciones fáciles de migrar y crecer en el tiempo; soluciones que pueden ser ajustas a diferentes áreas y operar de manera rápida e independiente.

G.- SQL funcionan con tablas de datos que sobre las cuales se ejecutan consultas.

4.- Categoría

Ahora bien, SQL organiza los comandos en los siguientes bloques o categorías:

A.- El lenguaje de definición de datos (DDL :Data Definition Language DDL): Son aquellos comandos utilizados para la creación de una BD y todos sus componentes como: tablas, índices, relaciones, funciones, disparadores (triggers), procedimientos almacenados, vistas, entre otros.

Aqui tenemos los siguientes comandos: CREATE, ALTER y DROP.

B.- El lenguaje de manipulación de datos (DML: Data Manipulation Language): Estas sentencias son utilizadas para insertar, modificar y borrar datos en la BD.

Aquí tenemos los siguientes comandos: INSERT, UPDATE y DELETE.

C.- El lenguaje de consulta (SQL: Structured Query Language): Son los comandos usados por el DBA o los analistas para extraer información de la base de datos; donde **el comando más usado es: SELECT y todas sus variantes.**

D.- Lenguaje de control de datos (DCL: Data Control Language): Son los comandos usados para controlar el acceso de los usuarios, programadores y especialistas de ciencia de datos a los datos almacenados en las BD. Fundamentalmente este lenguaje establece privilegios; es decir, los permisos y derechos de acceso a la DB según el criterio o caracterización hecha por el DBA a los usuarios del sistema.

Aquí tenemos los siguientes comandos: GRANT (Brinda acceso a un usuario a la base de datos) y **REVOKE** (Elimina el acceso a la base de datos de un usuario).

E.- Lenguaje de control de transacciones (TCL: Transactional Control Language):

Son los comandos usados para gestionar las transacciones en una base de datos. Una transacción es un conjunto de acciones vinculadas a una única acción ejecutable, donde todas las acciones asociadas a la transacción deben funcionar correctamente para que la transacción se considere un éxito; por ejemplo: registrar una compra o registrar una devolución.

Aquí tenemos los siguientes comandos: TCL TRANSACCIÓN (Realizar una operación), **ROLL-BACK** (Restablecer una operación si alguna acción no se ejecuta), **SAVEPOINT** (Se establece un punto en la transacción para guardar los datos) y **COMMIT** (Empleado para guardar el trabajo hecho).