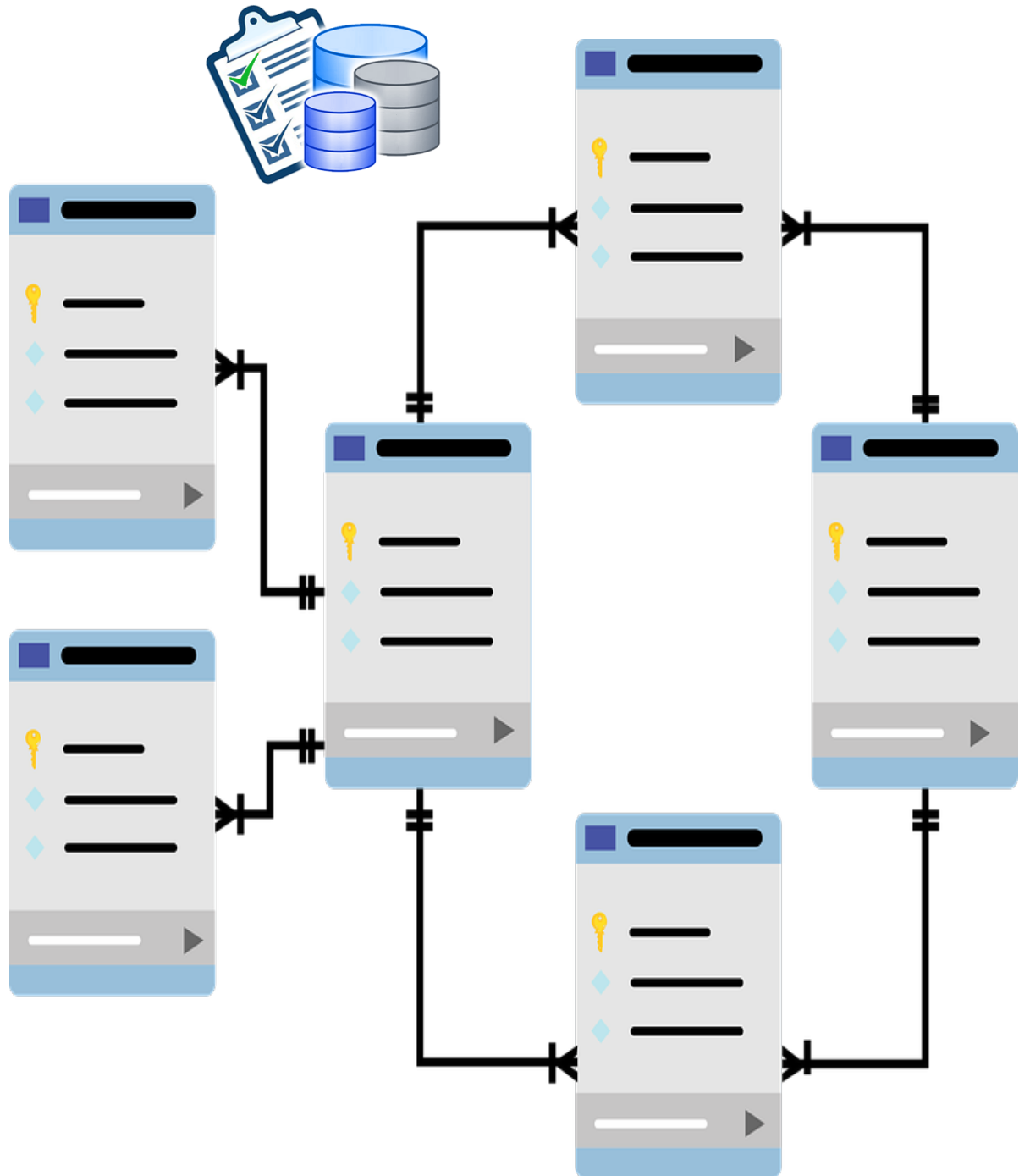


Introducción a las Bases de Datos



Introducción a las Bases de Datos

1º Edición

Lic. En Sistemas de Información Claudio Dalmasso

Profesor Universitario

*“ Que otros se enorgullecen por lo que
han escrito, yo me enorgullezco por
lo que he leído ”*

Jorge Luis Borges.-

Ciudad de Rosario, Argentina - Septiembre de 2022

ESUPCOM

TIC Departamento de la Información y las Comunicaciones

Material disponible en Biblioteca Virtual

https://esupcom.unr.edu.ar/bv_tics/



Índice general

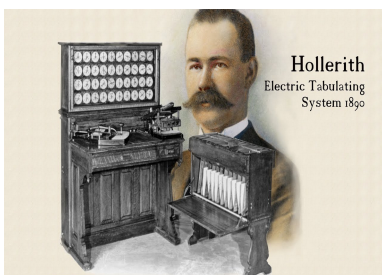
1. Conceptos de Bases de Datos.	
1.1. Introducción.....	4
1.2. Origen de las Bases de Datos.....	4
1.3. Entonces, ¿a que llamamos Bases de Datos?.....	4
1.4. Evolución de las Bases de Datos.....	5
2. Modelo Relacional.	
2.1. Manipulación de Base de Datos.....	5
2.2. Bases de Datos Relacionales.....	6
2.3. Tablas	7
2.4. Columnas o Atributos.....	7
2.5. Filas.....	8
2.6. Clave Primaria o Primary Key.....	8
2.7. Clave Secundaria o Foreign Key	8
2.8. Relaciones.....	9
2.9. Vista Entidad-Relación de las Tablas Cliente & Zona.....	10
3. Language SQL.	
3.1. Introducción a SQL.....	10
3.2. Elementos del lenguaje SQL.....	11
3.3. Cláusulas.....	12
3.4. Operadores de Comparación.....	13
3.5. Sentencia Inner Join.....	13
3.6. Funciones de Agregación.....	14
3.7. Bibliografía.....	14

1.1. Introducción

La primera vez que se escuchó sobre el término Bases de Datos fue en 1963 en un simposio (reunión de especialistas) en California, refiriéndose a ella como un conjunto de “*información relacionada*” que se encuentra agrupada o estructurada. Es importante resaltar que información ≠! dato, los datos por sí solo no tienen significado hasta que se les da un contexto.

1.2. Origen de las Bases de Datos

El ser humano siempre ha tenido la necesidad de guardar información, es por ello que los orígenes de las Bases de Datos se remontan desde la antigüedad; urgía la necesidad de organizar, administrar “existencias” como ser libros en una biblioteca, toda clase de registro de una cosecha, etc. La invención de la máquina de tarjetas perforadas de tabulación por Herman Hollerith, marca el comienzo de la era de procesamientos de datos de sistemas semiautomáticos.



Herman Hollerith creó el tabulador electromagnético de tarjetas perforadas para inicialmente poder resumir la información y posteriormente posibilitar la contabilidad de la misma.

1.3. Entonces, ¿a que llamamos Base de Datos?

Una Base de Datos es un conjunto de datos almacenados que se encuentran organizados mediante una estructura de datos. Las Bases de Datos han sido diseñadas con la intención de satisfacer los requisitos de información de una organización o una empresa; por ejemplo un hospital, una biblioteca, un supermercado, etc. Podemos decir que una BD (Base de Datos) es un gran almacén de datos que se define y se crea una sola vez pudiendo ser accedida por varios usuarios aún al mismo tiempo (*concepto de concurrencia*), de este modo se posibilita el acceso a todos los departamentos que posean los permisos necesarios en una organización. La BD no solo contiene los datos de dicha organización sino también almacena una descripción de dichos datos. Esta descripción se denomina Metadatos y se almacena en un Diccionario de Datos y es lo que permite que exista independencia de datos física y lógica.

1.4. Evolución de las Bases de Datos.

Recordemos la necesidad que origina el surgimiento de las BD; el sistema vigente hasta ese momento era el de “ficheros”. Los ficheros consistían en fichas almacenadas en mobiliarios, las mismas generalmente eran organizadas en orden alfabético para agilizar su ubicación. Cabe destacar, que estos procedimientos eran manuales, esta forma de trabajo impedía primeramente la concurrencia a la información, es decir, cuando mas de una persona necesitaba acceso simultáneo, en segundo orden la distribución física de los espacios, en muchos casos los departamentos organizacionales se encontraban ubicados en distintos pisos lo cual imposibilitaba la accesibilidad y en tercer lugar los imprevistos y siniestros como ser roturas de fichas, extravíos, deterioro, o episodios mas graves como incendios, inundaciones, etc.

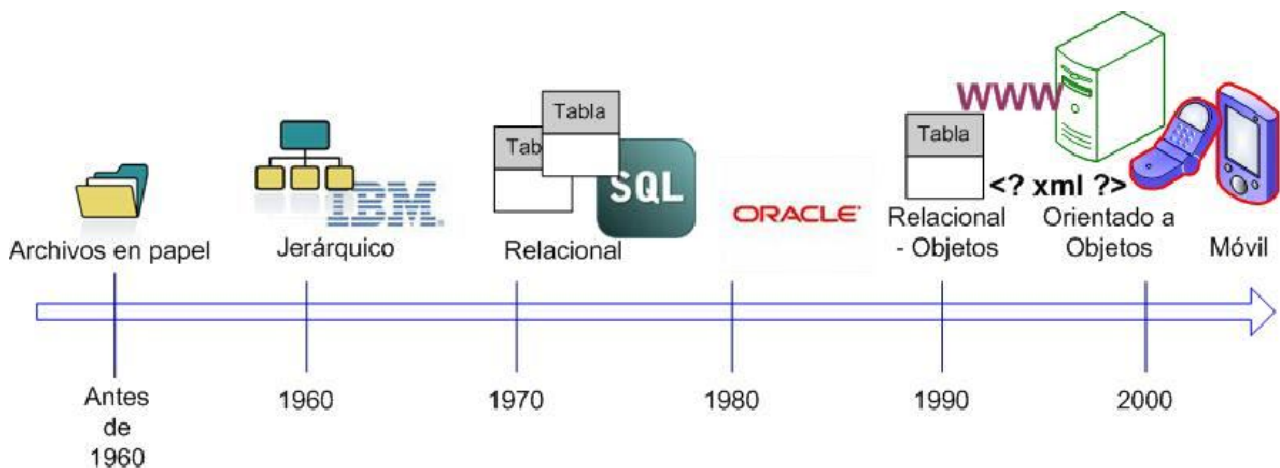


Gráfico evolutivo del formato papel al formato digital.

2.1. Manipulación de Base de Datos.

Existen cuatro niveles jerárquicos que desempeñan distintos roles en cuanto a la intervención de una BD.

* *Administrador de Base de Datos (BDA)*. Se encarga de la administración física de la BD, establece el tipo de datos e índices a implementar, su ubicación, toma decisiones en cuanto a la ubicación física de los datos, es el encargado de la política de seguridad y del acceso concurrente de los datos.

* *Diseñador de Base de Datos*. Realizan el diseño de la BD debiendo identificar los datos, las relaciones entre ellos y las restricciones sobre los datos y sobre sus relaciones.

Es deber del del Diseñador de Base de Datos tener un profundo conocimiento de los datos de la Organización o Empresa y también debe conocer sus reglas de negocio. Las reglas de negocio describen las características principales sobre el comportamiento de datos y como las ve la empresa u organización.

* *Los Programadores.* Se encargan de implementar las aplicaciones desarrolladas a través de diversos lenguajes de programación elegido según el criterio y experiencia del desarrollador, también de acuerdo al tipo de necesidad requerida. Finalmente luego deben conectar el software a la BD. Estos programas serán utilizados por usuarios finales.

* *Los Usuarios.* Son los “clientes” en términos informáticos, de la BD, los usuarios finales intervienen cuando la BD ha sido diseñada e implementada.

2.2. Bases de Datos Relacionales

Reciben este nombre ya que permiten “relacionar” tablas y de este modo se posibilita la vinculación entre los datos. La mayor parte de los sistemas de BDR, emplean de forma embebida el lenguaje SQL (Structured Query Language) su traducción en español es Lenguaje de Consulta Estructurado. Mas adelante trataremos su implementación y los beneficios que ofrece a la hora de ejecutar consultas = reportes = informes.



Compañías de Bases de Datos Relacionales.

2.3. Tablas

Las Tablas al final de cuentas son los medios donde se depositan los datos, es importante comprender que no surgen de la nada misma, existe un modelo de álgebra que permite relacionarlas aplicando procesos normalizadores. La normalización (solo haremos mención) es el proceso de análisis y estudio previo a la construcción de una tabla y también el más importante por que de allí depende si su creación es adecuada o no para el sistema en estudio. Un análisis pertinente tiene en cuenta todos los requerimientos por los cuales van tomando forma las Tablas y las Relaciones. Esta labor la llevan adelante diversos profesionales de sistemas, los mas especializados son los DBA (Data Basic Admin) en su traducción Administradores de Bases de Datos, aunque en los tiempos actuales muchos desarrolladores ya manejan los conocimientos para poder intervenir en la producción de Bases de Datos.

La necesidad de almacenar datos digitalmente tiene que ver con la posibilidad concreta de agilizar las consultas, poder efectuar modificaciones a través de la edición, cargar nuevos datos, y también eliminarlos. Ahora bien, introduciéndonos directamente en las tablas citaremos un ejemplo y haremos un abordaje de la misma:

Tablas los nombres de las tablas suelen estar en singular y hacen mención al contenido que almacenan. Las vamos a encontrar en diversas fuentes también como Entidad. En este caso nuestra tabla o entidad se llama *Cliente*.

Cliente - TIC - LibreOffice Base: vista de datos de tabla

Archivo Editar Ver Insertar Datos Herramientas Ventana Ayuda

	codCliente	nombreCliente	calleCliente	ciudadCliente	codPostal	cel	email	codZona
▶	1000	Juan Carlos Gomez	Lima 4563	Resistencia	3500	362345	carlitozgo	1
	1001	María Laura Serra	Daniel Damiano	Pergamino	2700	247762	marivende	1
	1002	María Inés Cano	Belgrano 723 Pi	Rosario	2000	341223	lachinita3	2
	1003	Mirco Ferreyra	Salta 2112 PB	Rosario	2000	341-32	mircoferr	2
	1004	Hector Alcorta	Sarmiento 454	Arrecifes	2740	247862	hector34@	1
★	<Campo autom							

2.4. Columnas o Atributos los atributos se disponen de forma vertical y suministran datos de distintos tipo como ser números, nombres, direcciones, email, etc. En la tabla *Cliente* visualizamos atributos como ser **codPostal** quien contiene el código postal de la localidad a la que hace referencia.

codPostal
3500
2700
2000
2000
2740


Una observación..... es muy común encontrar los nombres de los atributos escritos en forma de camello (codPostal) se comienza en minúscula y cuando surge otra referencia se cambia la primer letra a mayúscula. Cuando hay muchas tablas, este tip's permite ubicar rápidamente el atributo en cuestión y facilita la administración de los DBA.

2.5. Filas = Registros = Tuplas una fila, registro o tupla almacena una sección individual de información, es decir, si estamos en la tabla *Cliente* y deseamos acceder al registro cuyo **codCliente** = 1004, nos mostrará toda la información que contiene esa propia fila.

1004	Hector Alcorta	Sarmiento 454	Arrecifes	2740	247862	hector34@1
------	----------------	---------------	-----------	------	--------	------------

Hasta el momento podemos inferir que una base de datos tiene similitudes con una planilla de cálculos y estamos en lo cierto, la gran diferencia radica en que una BD contiene un volumen de datos robusto, no lo puede administrar cualquier tipo de usuario, se requieren conocimientos específicos y profesionales para su manipulación y por ende aportan mayor seguridad en cuanto a la integridad de los mismos.

Vista en Modo Diseño de la tabla Cliente.

Cliente	
	codCliente
	nombreCliente
	calleCliente
	ciudadCliente
	codPostal
	cel
	email
	codZona
	provCliente

→ **2.6. Clave Primaria o Primary Key** toda tabla debe contener para poder ser representada una clave primaria o candidata. Esta clave identifica de forma unívoca cada fila de una entidad. Debe ser elegida adecuadamente de forma tal que no pueda repetirse. Por ej: en la imagen vemos que se eligió el atributo **codCliente** como clave primaria. De este modo nos garantizamos de que no van a existir dos clientes con el mismo código.

→ **2.7. Clave Secundaria o Foreign Key** es una columna o combinación de columnas en una tabla, cuyo/s valor/es es/son el valor de una clave primaria para otra otra tabla. La clave foránea debe ser del mismo tipo de dato que la clave

primaria de la tabla con la cual se relaciona. En el ejemplo de la tabla *Cliente* tenemos como clave foránea **codZona** quien es clave primaria en la tabla *Zona*. Se pueden relacionar unicamente si se dá esta paridad de vincular clave primaria con clave foránea del mismo tipo de datos.

2.8. Relaciones

Como el término lo indica, una relación nos permite vincular, en este caso tablas, y lo enriquecedor de contar con relaciones es que podemos extender la información de un registro, es decir, tener mas datos por ende mayor información. De no existir las relaciones, las tablas se mantendrían aisladas con sus contenidos sin tener la posibilidad de cruzar datos. Lo importante en una BDR es que las relaciones permiten evitar datos redundantes y garantiza la integridad referencial de los datos (esto significa que si una clave foránea contiene un valor, ese valor se refiere a un registro existente en la tabla relacionada). La redundancia tiene que ver con datos repetidos o duplicados en tablas donde no los necesitamos.

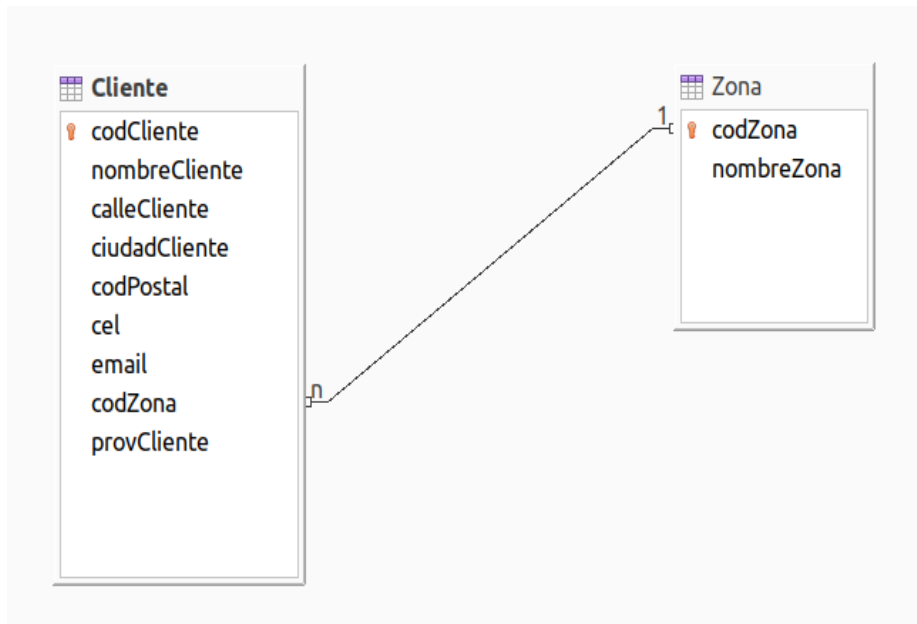
Ahora bien, las relaciones pueden darse de:

- Uno a Uno (1 a 1) - este tipo de relación se establece entre una entidad de una tabla y otra entidad de otra tabla. Por ej: las tablas *Presidente* – *País* un país solo puede tener un presidente y un presidente solo puede gobernar un país.
- Uno a Muchos (1 a ∞) - esta posibilidad se da entre varias entidades de una tabla y una entidad de otra tabla. Por ej: las tablas *Cliente* – *Pedido* un cliente puede solicitar muchos pedidos.
- Muchos a Muchos (∞ a ∞) - esta relación se produce cuando varios registros de una tabla se asocian a varios registros de otra tabla. Por ej: *Cliente* – *Producto* donde muchos clientes pueden comprar varios productos y los productos pueden ser comprados por varios clientes.

Al momento de crear las relaciones debemos tener en cuenta dos procedimientos importantes:

- ✓ Actualización en Cascada. Significa que las filas de referencia se van actualizar en la tabla secundaria cuando la fila referenciada se actualiza en la tabla principal que tiene una clave primaria.
- ✓ Exigir Integridad Referencial. Son reglas que utilizan la mayoría de las BDR para asegurarse que los registros de tablas relacionadas sean válidos y que no se borren o cambien datos relacionados de forma accidental produciendo errores de integridad.

2.9. Vista Entidad-Relación de las Tablas Cliente & Zona.



Analicemos el ejemplo presentado anteriormente:

1. Identificamos **codZona** como clave primaria en la tabla *Zona*.
2. Encontramos en la tabla *Cliente* su foránea **codZona** para que se pueda dar la relación.
3. Podemos identificar su cardinalidad de 1 a ∞ (1 a muchos) donde una zona puede tener muchos clientes y muchos clientes pertenecer a una sola zona.
4. No lo identificamos visualmente por estar en Vista diseño pero garantizamos que **codZona** es foránea de la tabla *Cliente* siendo del mismo tipo de datos que **codZona** es clave primaria de la tabla *Zona*. El tipo de dato es un Entero - Integer.
5. Esta Relación nos permitirá en un futuro consultar la zona a la que pertenece un cliente. Lo veremos con la ejercitación.

3.1. Introducción a SQL

SQL es el lenguaje fundamental de los Sistemas de Gestión de Base de Datos Relacionales (SGBD). Es uno de los lenguajes más utilizados en la historia de la informática y además sigue siendo de aprendizaje casi obligatorio para cualquier profesional que se desempeñe en computación.

SQL es declarativo, y se centra en definir en lo que se quiere hacer por encima de como hacerlo. Las razones particulares de por que se procede así, tiene que ver con los lenguajes

declarativos que se parecen más al lenguaje natural humano y parecen más apropiados para trabajar con Bases de Datos especialmente las relacionales. Pretende cumplir con la quinta regla de Cood (Edgard Frank Cood) pionero en Bases de Datos; el sostenía que el lenguaje de base de datos debería ser capaz de realizar cualquier instrucción sobre la misma. Por ejemplo en Oracle esta condición se cumple a la perfección ya que toda la gestión y administración del sistema de base de datos se puede realizar utilizando solo el lenguaje de SQL.

3.2. Elementos del lenguaje SQL

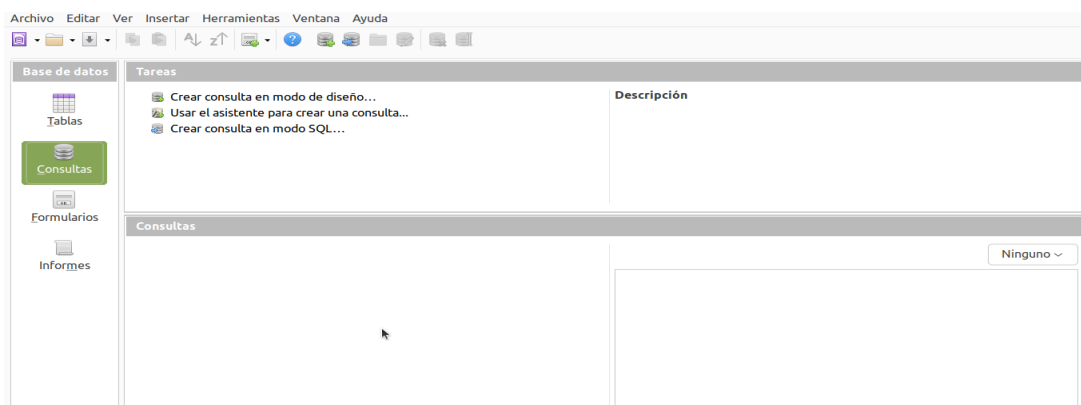
Como solo nos centraremos en la introducción al lenguaje, simplemente incorporaremos los comandos de consulta que son más útiles a la hora de generar informes.

DML (Data Manipulation Language) su traducción es Lenguaje Manipulador de Datos. Lo forman los comandos que modifican filas y por lo tanto datos de las tablas. Lo forman las instrucciones:

- ✓ **Insert** → Se utiliza cuando quieres añadir o insertar nuevos datos.
- ✓ **Update** → Se utiliza cuando quieres cambiar o actualizar datos existentes.
- ✓ **Delete** → Se utiliza cuando quieres eliminar o borrar datos existentes.
- ✓ **Select** → Este comando permite realizar consultas sobre los datos de las Bases de Datos. Aunque esta operación forma parte de la función de manipulación resulta tan importante que en la actualidad se le asocia toda una función, la función *Consulta*.

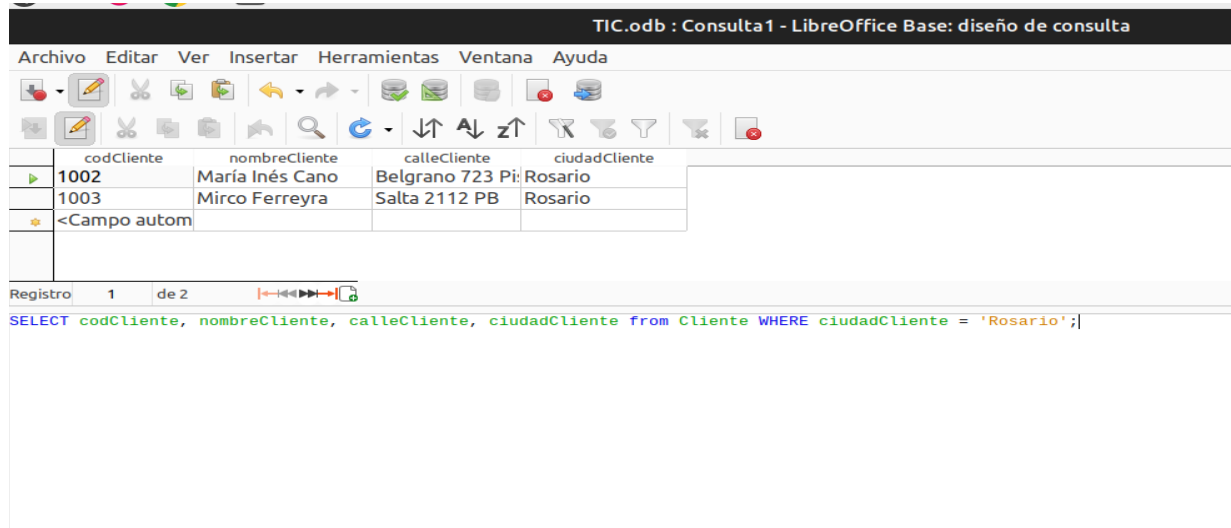
Ejemplo.

De la tabla *Cliente* deseamos conocer que clientes son de la localidad de Rosario. Procedemos a efectuar una consulta en SQL y mostramos los datos relevados, nuestra BDR es Basic de Libre Office.



La consulta quedaría diseñada del siguiente modo:

```
SELECT codCliente, nombreCliente, calleCliente, ciudadCliente FROM Cliente WHERE
ciudadCliente = 'Rosario';
```



Como vemos en la imagen superior, SQL nos devuelve los datos que son de nuestro interés. Encontró entre los registros que 2 personas residen en Rosario.

3.3. Cláusulas

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular. Algunas de ellas:

- ✓ From → se utiliza para especificar la tabla de donde se van a seleccionar los registros.
- ✓ Where → se utiliza para especificar las condiciones que deben reunir los registros que se van a seleccionar.
- ✓ Group By → se utiliza para separar los registros seleccionados de acuerdo con un orden específico.
- ✓ Having → utilizada para expresar la condición que debe satisfacer cada grupo.
- ✓ Order By → utilizada para ordenar los registros de acuerdo con un criterio específico.

3.4. Operadores de Comparación

- ✓ > (Mayor que).
- ✓ < (Menor que).
- ✓ <> (Distinto que).
- ✓ >= (Mayor o igual que).
- ✓ <= (Menor o igual que).
- ✓ = (Igual que).
- ✓ **Between** (Se utiliza para especificar un intervalo de valores).
- ✓ **Like** (Utilizado en la comparación de un modelo).
- ✓ **In** (Se utiliza para especificar registros de una base de datos).

3.5. Sentencia Inner Join

La consulta Inner Join permite consultar datos de dos o mas tablas relacionadas. Su sintaxis se expresa como en el ejemplo del siguiente modo:

```
SELECT codCliente, nombreCliente, codZona
FROM Cliente cli
INNER JOIN Zona zo ON cli.codZona = zo.codZona
WHERE codZona = '13';
```

Esta consulta interviene en dos tablas *Cliente* y *Zona* ya que necesita obtener datos de ambas, para ello se crean dos alias (*cli* para la tabla *Cliente* y *zo* para la tabla *Zona*). A través del **From** se declara la primera tabla en cuestión en este caso *Cliente* y se imbocha el alias *Cliente cli*, luego **INNER JOIN** para lograr la unión con la segunda tabla *Zona* también con su respectivo alias *Zona zo*. Seguidamente **ON** habilita la unión entre la clave primaria y foránea con sus respectivos alias *cli.codZona = zo.codZona*. Finalmente se establece la condición que necesitamos en la consulta mediante **WHERE codZona = '13'**;

3.6. Funciones de Agregación

- ✓ **Count:** devuelve el número total de filas seleccionadas por la consulta.
- ✓ **MIN:** devuelve el valor mínimo del campo que especifiquemos.
- ✓ **MAX:** devuelve el valor máximo del campo que especifiquemos.
- ✓ **SUM:** suma los valores del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.
- ✓ **AVG:** devuelve el valor promedio del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.

3.7. Bibliografía

- [1] A. Silberschatz – H. Korth – S. Sudarshan (2006)
Fundamentos de Bases de Datos. Quinta Edición. MC Graw Hill.
- [2] C. Batini, S. Ceri, S. B. Navathe (1994)
Diseño Conceptual de Bases de Datos. Un enfoque de entidades–interrelaciones. Addison–Wesley /
Díaz de Santos.
- [3] R. Elmasri, S. B. Navathe (2002)
Fundamentos de Sistemas de Bases de Datos. Tercera Edición. Addison–Wesley.
- [4] M. Marquez – (2011)
Bases de Datos. Primera Edición. Sapiencia.
- [5] A. Opperl – R. Sheldon (2009)
Fundamentos de SQL. Tercera Edición. MC Graw Hill.