## Tarea 1: Investigación de Terminología

* 1. **Base de datos:** 
     1. **Concepto.**
     2. **Ventajas y desventajas.**
     3. **Tipos de base de datos.**
     4. **Ejemplos de algunas bases de datos, etc.**
  2. **PostgreSQL.**
     1. **Definición.**
     2. **Características.**
  3. **Herramientas CASE.**
     1. **Definición.**

**1.3.2 Ventajas**

**1. 4 Pgmodeler.**

**1.4.1 Definición.**

**1.4.2 Beneficios**

**1.5 PgAdmin.**

**1.5.1 Definición.**

**1.5.2 Beneficios.**

**1.6 Modelo entidad Relación.**

**1.6.1 Definición.**

**1.7 Normalización.**

**1.7.1 Definición.**

**1.8 Integridad de Bases de Datos**

**1.9 Tipos de Integridad de Base de Datos**

**1.10 Transacciones en base de datos Definición**

**1.10.1 Importancia y propiedades de la transacción)**

**1.10 Funciones (definición, tipos)**

**1.11 procedimiento.**

**1.11.1 Definición.**

**1.11.2 Tipos.**

**1.11.3 Diferencia con las funciones**

**1.12 Seguridad en base de datos**

**1.13 Usuarios en base de datos**

**1.14 Esquema**

**1.14.1 Definición.**

**1.14.2 Usos.**

**1.15 Backup.**

**1.15.1 Definición**

**1.15.2 Usos.**

**1.16 Restauración:**

**1.16.1 Definición.**

**1.16.2 Usos**

**1.17 Diccionario de Datos**

**Base de Datos**

Concepto Base de Datos:

Una base de datos es un “almacén” que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrar y utilizar fácilmente.

Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más **columnas** y **filas**. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

Características

* Redundancia mínima.
* Acceso concurrente por parte de múltiples usuarios.
* Integridad de datos.
* Respaldo y recuperación.

**Ventajas y Desventajas**

**Ventajas**.

1. Control sobre la redundancia de datos.

Los sistemas de ficheros almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además provocar la falta de consistencia.

En una Base de Datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos.

1. Consistencia de datos

Eliminando y controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar solo una vez, y está disponible para todos los usuarios inmediatamente. Si uno datos están duplicados y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantienen consistentes.

1. Compartir datos.

En una empresa las Bases de Datos le pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados.

1. Mantenimiento de estándares.

Los estándares establecen el formato de los datos para facilitar su intercambio, pueden establecerse sobre el formato de los datos para facilitar su intercambio,

**Desventajas**.

1. Complejidad.

Los SGBD son conjuntos de programas que pueden llegar a ser complejos con una gran funcionalidad. Es preciso comprender su funcionamiento para poder realizar un buen uso de ellos.

1. Coste del equipamiento.

Tanto el SGBD, como la propia Base de Datos, pueden hacer que sea necesario adquirir más espacio de almacenamiento. Además, para alcanzar las prestaciones deseadas, es posible que sea necesario adquirir una máquina más grande o una que se dedique solamente al SGBD.

Todo esto hará que la implantación de un sistema de Base de Datos sea más caro.

1. Vulnerable a los fallos.

El hecho de que todo esté centralizado en el SGBD hace que el Sistema sea más vulnerable ante fallos que pueda producirse, es recomendable hacer Backus.

1. Tipos de Campos.

* Numérico: enteros ‘sin decimales’ y reales ‘decimales’.
* Booleanos: poseen dos estados: ‘Verdadero’ o ‘Falso’.
* Memos: son campos alfanuméricos de longitud ilimitada. Presentan el inconveniente de no poder ser indexados.
* Fechas: almacenan fechas de esta forma posibilita ordenar los registros por fechas o calcular los días entre una fecha y otra.
* Alfanuméricos: contienen cifras y letras. Presentan una longitud (255 caracteres).
* Autoincrementables: son campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado. Sirve de identificador ya que resultan exclusivos de un registro.

**Tipos de Base de datos**

1. **MySQL:**

MySQL es un Sistema Gestor de Bases de Datos relacionales potente y versátil, capaz de satisfacer la mayoría de los proyectos en la web.

Desarrollado inicialmente por Sun Microsystem,cayó en manos de Oracle cuando ésta adquirió a Sun. En estos momentos MySQL se mantiene con licencia dual, por una parte es GPL y código abierto , gratuito para uso por la comunidad, por otra parte tiene licencia comercial. Este motor de Bases de Datos es multiplataforma, por lo que se puede instalar en Windows, Linux y Mac. En servidores web se encuentra sobre plataformas Linux habitualmente y forma parte de la arquitectura LAMP.

1. **PostgreSQL y Oracle:**

Son sistemas de Base de Datos poderosos. Administra muy bien grandes cantidades de datos, y suelen ser utilizadas en intranets y sistemas de gran calibre.

1. **Access:**

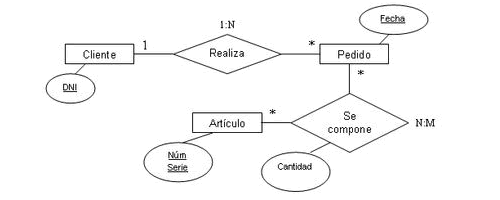
Es una Base de Datos desarrollada por Microsoft. Esta Base de Datos, debe ser creada bajo el programa access, el cual crea un archivo .mdb con la estructura ya explicada.

1. **Microsoft SQL Server:**

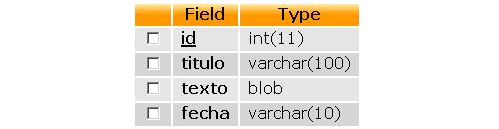
Es una Base de Datos más potente que access desarrollada por Microsoft. Se utiliza para manejar grandes volúmenes de información.

**Ejemplos**

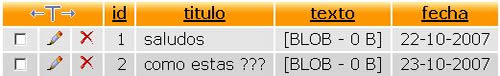
**Modelo Entidad Relación**

****

**Tabla “comentarios”**

****

**Los datos quedarían organizados como mostramos en siguiente ejemplo:**

****

SQL es universal en los sistemas de Bases de Datos.

Este lenguaje nos permite realizar consultas:

* Mostrar: muestra los registros.

Select \* From comentarios

* Insertar: borrar registros.

Insert Into comentarios (título, texto, fecha) Values ('saludos', 'como esta', '22-10-2007')

* Actualizar: actualiza los registros.

Update comentarios Set titulo='Mi Primer Comentario' Where id='1'

* Borrar: borra registros.

Delete From comentarios Where id='1'.

**PostgreSQL**

PostgreSQL es un Gestor de Bases de Datos relacional y orientado a objetos. Su licencia y desarrollo es de código abierto, siendo mantenida por una comunidad de desarrolladores, colaboradores y organizaciones comerciales de forma libre y desinteresadamente. Esta comunidad es denominada PDGD (*PostgreSQL Global Development Group).*

Es reconocido actualmente como uno de los sistemas gestores de Bases de Datos relacionales más potentes del mercado. Presenta fácil accesibilidad, es multiplataforma y está disponible para su utilización en casi todos los sistemas operativos utilizados en la actualidad.

**Características**

* Presenta un sistema de alta concurrencia: Presenta un sistema denominado MVCC, el cual permite que mientras un proceso escribe una tabla, otros puedan acceder a la misma tabla sin necesidad de verse bloqueados, y cada usuario obtiene una visión consistente.
* Sistema “Hot Standby”: Este proceso permite a los usuarios poder conectarse con el servidor y ejecutar búsquedas en la Base de Datos mientras la misma está en modo de recuperación o “stand by”. También se puede pasar de este modo, a modo normal sin detener el flujo de búsquedas o consultas de los usuarios, manteniendo las conexiones abiertas. Esto es posible únicamente cuando la Base de Datos se encuentra en modo de solo-lectura.
* Uso de formato JSON: El formato JSON se convierte en el punto débil de muchos sistemas de Bases de Datos relacionales. Sin embargo, PostgreSQL presenta buenas herramientas con las que es posible indexar elementos y realizar búsquedas en dicho formato. Aunque no se recomienda manejar toda la toda la Base de Datos en JSON, y se utiliza para el guardado de información de algunos elementos e indexar sus propiedades.
* Notificaciones a tiempo real: A pesar de que PostgreSQL no fue diseñada para ser una BD que trabaje al 100% en tiempo real, si es posible mantener sincronizado en varios dispositivos un sistema de notificación para cuando se hacen cambios específicos en la base de datos, gracias a las funciones LISTEN, UNLISTEN y NOTIFY.
* **Registro y guardado de transacciones:** Una de las características más interesantes de PostgreSQL, es su capacidad de registrar cada transacción en un WAL (*Write-Ahead-Log*). Esto permite restaurar la base de datos a cualquier punto previamente guardado, una especie de *"Checkpoint".* Esto permite que no sea necesario realizar respaldos completos de forma frecuente, en especial para los casos en los que se trabaja con una bd que es muy grande o que contiene mucha cantidad de datos.

1. Nombre del disparador.
2. Momento de arranque definido.
3. Evento del disparador.
4. Tabla dónde se ejecuta.
5. Frecuencia de ejecución.
6. Función llamada / Función correcta o incorrecta.

**Ventajas y desventajas PostgreSQL**

**Ventajas**

1. Instalación y uso gratuito: **PostgreSQL** es un gestor de Base de Datos de código libre y completamente gratuito, por lo que podemos instalarlo y utilizarlo las veces que queramos y en todos los dispositivos que queramos.
2. Sistema disponible multiplataforma: Es compatible con prácticamente todas las tecnologías y sistemas operativos de la actualidad.
3. Estabilidad: **PostgreSQL** hace más de 20 años en el mercado con un constante desarrollo y soporte de sus versiones. Incluso presenta un sistema de alta disponibilidad mientras los servidores están en modo de suspensión o recuperación, por lo que los usuarios pueden acceder en modo de solamente lectura sin bloquear de forma completa el sistema.
4. Escalabilidad y configuración: Es posible configurar de forma individual **PostgreSQL** según los recursos de Hardware disponibles en nuestro sistema, por lo que podemos ajustar el número de CPU y cantidad de memoria disponible para un funcionamiento óptimo.
5. Estándar SQL: Implementa la mayor parte de las funcionalidades principales del estándar SQL, por lo que se puede realizar de forma sencilla el incluir consultas y scripts de otros motores de Bases de Datos.
6. Herramienta gráfica: Incorpora una herramienta gráfica para la administración de las Bases de Datos de forma fácil e intuitiva, por la cual podemos ejecutar sentencias SQL, realizar copias de seguridad o tareas de mantenimiento.
7. Robustez y fiabilidad: **PostgreSQL** cumple con las características y protocolo ACID, lo que significa Atomicidad, Consistencia, Aislamiento y Durabilidad. Por ello, se garantiza la información de la Base de Datos y fiabilidad del sistema.
8. Soporte y ayuda: A pesar de no contar con soporte telefónico o en línea, existe una infinidad de foros y páginas para nuestra ayuda. Además, la comunidad de **PostgreSQL** es una de las más activas.

**Desventajas**

* **PostgreSQL** está diseñado específicamente para ambientes con alto volumen de datos, por lo que puede parecer lento en implementaciones para Bases de Datos de pequeño y mediano tamaño.
* No presenta una facilidad en comandos o sintaxis, por lo que se puede dificultar su uso si no poseemos conocimientos medios en lenguajes SQL.

**Herramientas CASE**

Son diversas Aplicaciones informáticas destinadas a aumentar la productividad en el Desarrollo de Software reduciendo el coste de las mismas en términos de tiempo y dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo de software en tareas como el diseño de proyectos, cálculos de coste, implementación de parte del código automáticamente con el diseño dado, Compilación automática, documentación o detección de errores entre otras.

**Componentes de una herramienta CASE**

De una forma esquemática podemos decir que una herramienta CASE se compone de los siguientes elementos:

**1. Repositorio (diccionario)** donde se almacenan los elementos definidos o creados por la herramienta, y cuya gestión se realiza mediante el apoyo de un Sistema de Gestión de Base de Datos (SBBD) o de un Sistema de Gestión de ficheros.

**2. Metamodelo (no siempre visible),** este modelo parte del principio que toda información para ser registrada en un sistema de información debe ser estructurada en uno o varios formatos.

**3. Comprobación de errores,** facilidades que permiten llevar a cabo un análisis de la exactitud, integridad y consistencia de los esquemas generados por la herramienta.

**Estructura general de una herramienta CASE**

* CASE de alto nivel son aquellas herramientas que autorizan o apoyan las fases finales o superiores del ciclo de vida del desarrollo de sistemas como la planificación de sistemas, el análisis de sistemas y el diseño de sistemas.
* CASE de bajo nivel, son aquellas herramientas que automatizan o apoyan las fases finales o inferiores del ciclo de vida como el diseño detallado de sistema, la implantación de sistemas y el soporte de sistemas.
* CASE cruzado de ciclo de vida se aplica a aquellas herramientas que apoyan actividades que tienen lugar a lo largo de todo el ciclo de vida, se incluyen actividades como la gestión de proyectos y la estimación.

**Clasificación**

Las herramientas CASE se pueden clasificar teniendo en cuenta los siguientes parámetros:

1. Las plataformas que soportan.
2. Las fases del ciclo de vida del desarrollador de sistemas que cubren
3. La arquitectura de las aplicaciones producen.
4. Su funcionalidad.

La clasificación basada en las fases del ciclo de desarrollo cubre:

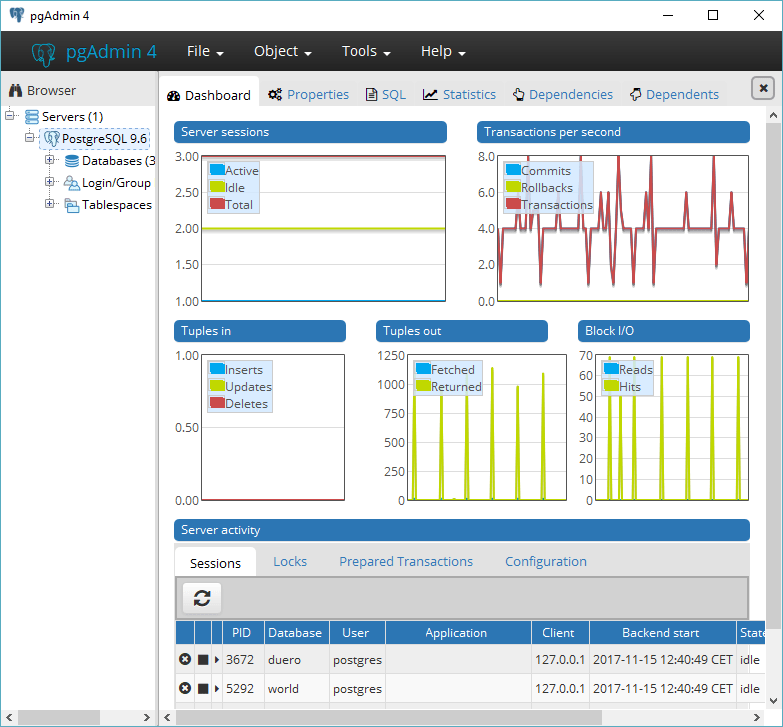
* Upper Case (U-CASE), herramientas que ayudan en las fases de planificación, análisis de requisitos y estrategia del desarrollo, usando, entre otros diagramas UML.
* Middle CASE (M-CASE), herramientas para automatizar tareas en el análisis y diseño de la aplicación.
* Lower CASE (L-CASE), herramientas que semi-automatizan la generación de código, crean programas de detección de errores, soportan la depuración de programas y pruebas.

**Ejemplos de Herramientas CASE más utilizadas**

* ERwin: es una herramienta de diseño de Base de Datos. Brinda productividad en diseño, generación, y mantenimiento de aplicaciones. ERwin permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la Base de Datos. Genera automáticamente las tablas y miles de líneas de stored procedure y triggers para los principales tipo de Bases de Datos.
* EasyCASE: herramienta que permite automatizar las fases de análisis y diseño dentro del desarrollo de una aplicación, para poder crear las aplicaciones eficazmente, desde procesamiento de transacciones a la aplicación de bases de datos de cliente/servidor, así como sistemas de tiempo real.
* Oracle Designer: fue la herramienta CASE de Oracle para diseñar un sistema de información y generarlo. Después de generar el sistema de información, uno puede editar el código generado con Oracle Developer Suite.

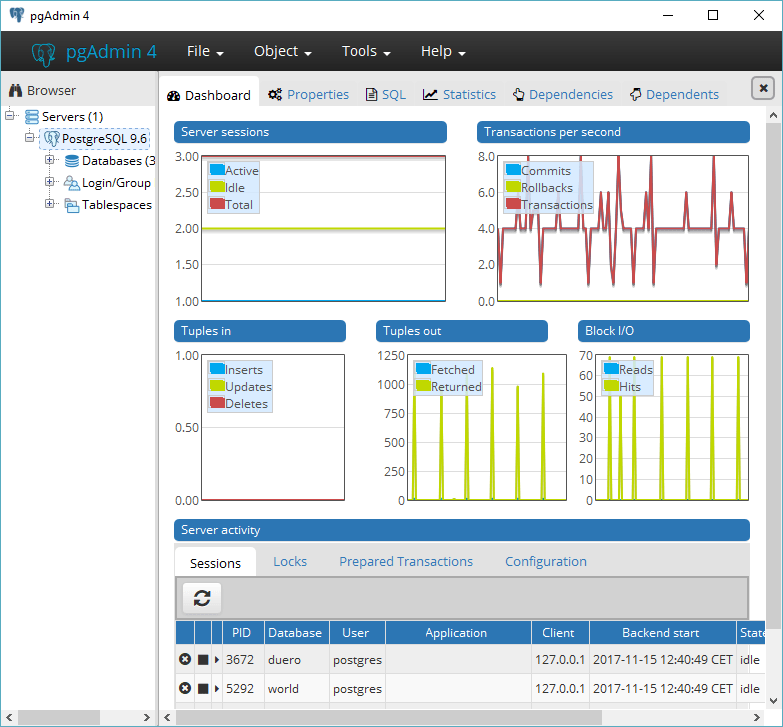
**PgAdmin**

Es una herramienta indispensable para gestionar y administrar PostgreSQL, la base de datos de código abierto más avanzada del mundo.



**Beneficios**

* La interfaz de usuario es más flexible, permite que las pestañas se acoplen y reorganicen en más formas que anteriormente. Ahora es posible disponer de pestañas desmontables, lo que permite que la Herramienta de consulta (*Query Tool*) y el Depurador se abren en pestañas nuevas y luego se puedan mover a pantallas alternativas.
* Destaca el completo Panel de control (*dashboard*), que sirve para **monitorizar el estado del servidor y de las bases de datos.** La pestaña Panel de control se puede cerrar si se desea, para minimizar el tráfico de consultas resultante de la actualización del gráfico. Además desde el Panel de control se pueden cancelar las consultas.



**Modelo Entidad - Relación**

Es una herramienta que permite representar de manera simplificada los componentes que participan en un proceso de negocio y el modo en el que estos se relacionan entre sí.

1. Entidades. Las entidades representan *cosas* u *objetos* (ya sean reales o abstractos), que se diferencian claramente entre sí.

Ejemplo.

**Coches** (objeto físico): contiene la información de cada taller.

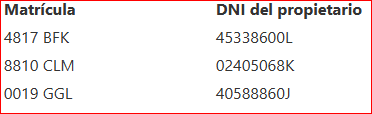
**Empleado** (*objeto* físico): información de los trabajadores.

**Cargo del empleado** (*cosa* abstracta): información de la función del empleado.



1. Atributos. Los atributos definen o identifican las características de la entidad (**es el contenido de esta entidad**).

Unos posibles atributos serían los siguientes: *número de chasis*, *matrícula*, *DNI del propietario*, *marca*, *modelo* y muchos otros que complementen la información de cada coche.

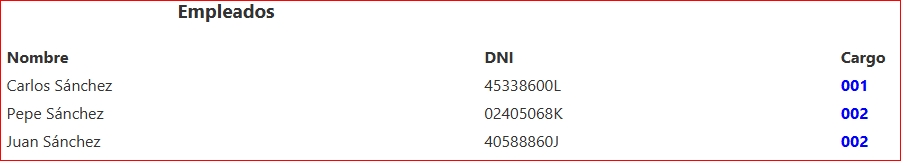


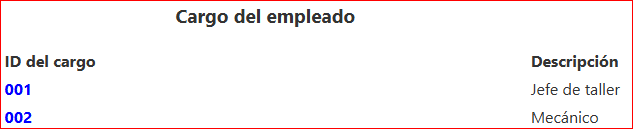
1. Relaciones. Es un vínculo que nos permite definir una dependencia entre varias entidades, es decir, nos permite exigir que varias entidades comparten ciertos atributos de forma indispensable.

Ejemplo: los empleados del taller (de la entidad "**Empleados**") tienen un cargo (según la entidad "**Cargo del empleado**"). Es decir, un atributo de la entidad "*Empleados*" especificará que cargo tiene en el taller, y tiene que ser idéntico al que ya existe en la entidad "*Cargo del empleado*".



El diagrama entidad relación es la expresión gráfica del modelo entidad relación.

****

****

****

#### **Relaciones de cardinalidad**

* **Uno a uno:** una entidad se relaciona únicamente con otra y viceversa.

Ejemplo: Entidad ‘chasis’, un chasis solo le pertenece una ‘matrícula’. y sola una matrícula le pertenece a un ‘chasis’



* **Uno a varios o varios a uno:** Determina que un registro de una entidad puede estar relacionado con varios de otra entidad, pero en esta entidad existir solo una vez.

Ejemplo: de cero a muchos ‘Empleados’ trabajan o están a cargo del empleo pero sólo un cargo está destinado a un ‘empleado’.



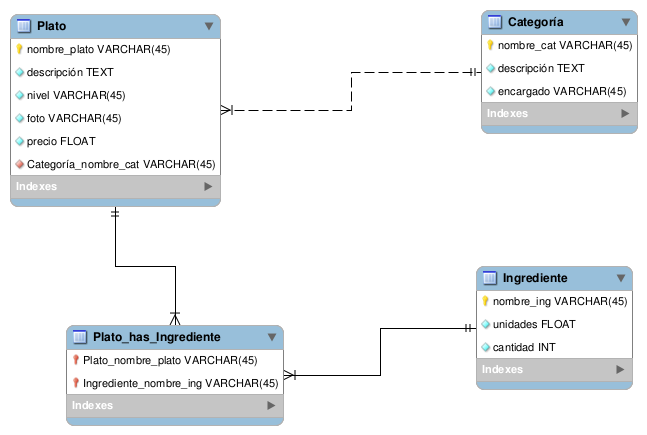
* **Varios a varios:** Determina que una entidad puede relacionarse con otra con ninguno o varios registros y viceversa.

Ejemplo: En el ‘taller’ un ‘coche’ puede reparar varios ‘coches’ distintos.



### **Claves**

**Superclave:** El atributo o combinación de atributos que identifica de forma exclusiva una fila o registro en una relación se conoce como clave primaria.



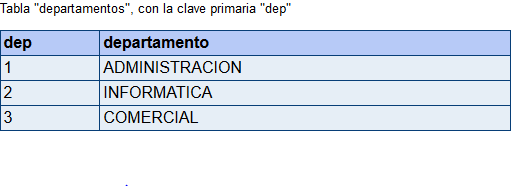
**Clave primaria:** La clave secundaria es un campo no único. Un valor de clave secundaria puede referirse a muchos registros.

Ejemplo PRIMARY KEY

CREATE TABLE personas { identificador int NOT NULL, nombre varchar(255) NOT NULL, apellido1 varchar(255) NOT NULL, PRIMARY KEY (identificador) }

**Clave externa o clave foránea:** o FOREIGN KEY, es una columna o varias columnas, que sirven para señalar cuál es la clave primaria de otra tabla.

Ejemplo: FOREIGN KEY



**CREATE TABLE departamentos**

**{ dep int NOT NULL,**

**departamento varchar(255),**

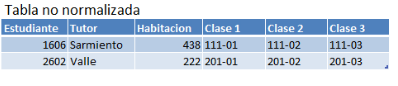
**PRIMARY KEY (dep)**

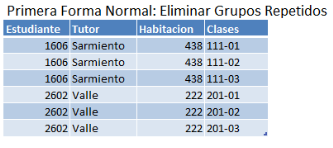
**}**

**NORMALIZACIÓN**

Es el proceso de organizar los datos siguiendo una serie de buenas prácticas tras el paso del modelo entidad-relación al modelo relacional. La normalización, tiende a eliminar la información redundante en las bases de datos que cumplen estas normas. Si el negocio exige un cambio en un dato, éste propagará o afectará a otras tablas solo donde sea necesario.

Primera forma normal: La primera forma normal eliminamos los grupos repetidos quedándonos con una sola columna de clases y repitiendo los datos del estudiante tutor y habitación. Ahora no tenemos grupos repetidos porque se aplica la primera forma normal.



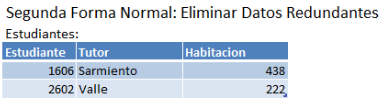


Segunda forma normal: Asegura que cada atributo describe la entidad. Crear tablas separadas para el conjunto de valores y los registros múltiples, estas tablas se deben relacionar con una clave externa. Los registros no deben depender de otra cosa que la clave principal de la tabla, incluida la clave compuesta si es necesario.

Ejemplo: Se elimina los datos redundantes, y para lograrlo vamos a crear dos tablas.

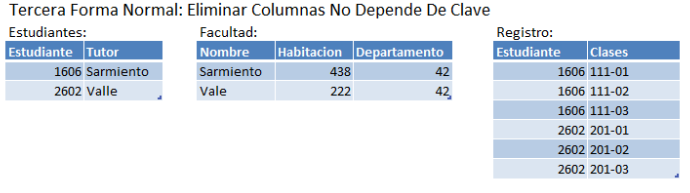
1° tabla: Estudiantes; donde eliminaremos los datos redundantes quedándonos con los datos únicos (Estudiante, Tutor, y Habitación)

2° tabla: Registro; para el número de estudiante y las clases que llevará en el ejemplo el estudiante 1606 y 2603 llevará cada uno tres clases.

Tercera forma normal: esta forma normal debemos diseñar los datos de manera que, en un mismo registro de una tabla, todos sus campos dependen de la clave primaria, elimina aquellos campos que no dependen de la clave.

Ejemplo: Eliminar los campos de No Dependen de la Clave y para lograrlo dividimos la tabla estudiante en dos tablas y creamos la tabla Facultad donde trasladaremos la columna habitación que No Depende de la Clave que es la columna estudiante, el nombre del tutor será el enlace con la tabla estudiante, aunque también podría ser la columna estudiante.



Para que se usa la normalización

Las bases de datos relacionales se normalizan para: Evitar la redundancia de los datos. Evitar problemas de actualización de los datos en las tablas

**Integridad de las Bases de Datos**

La integridad en una base de datos es la corrección y exactitud de la información contenida. Además de conservar la seguridad en un sistema de bases de datos que permite el acceso a múltiples usuarios en tiempos paralelos.

Los tres defensores de la integridad son:

* Restricciones de integridad: Protegen a la base de datos contra los daños accidentales.

Tipos de restricciones de integridad:

•Declaración de claves (primarias, candidatas).

•Cardinalidad de la relación – de varios a varios, de uno a varios, de uno a uno.

•Participación mínima/máxima.

•Restricciones de los dominios.

•Dependencias funcionales.

•Dependencias multivalorado

* Restricciones de dominio: Una definición adecuada de las restricciones de los dominios no sólo permite verificar los valores introducidos en la base de datos sino también examinar las consultas para asegurarse de que tengan sentido las comparaciones que hagan.

Ejemplo: Normalmente no se considerará que la consulta “Hallar todos los clientes que tengan el nombre de una sucursal” tenga sentido. Por tanto, nombre-cliente y nombre-sucursal deben tener dominios diferentes.

* Restricciones de existencia: Esta restricción evita la aparición de valores nulos en las columnas.
* Restricciones de unicidad: Esta restricción evita la aparición de valores duplicados en las columnas.

CREATE TABLE Sucursales (

nombre-sucursal VARCHAR(20),

ciudad-sucursal VARCHAR(20) NOT NULL,

PRIMARY KEY(nombre-sucursal) UNIQUE (ciudad-sucursal))

* Cuando un administrador desea agregar, modificar o eliminar la información de la base de datos, la integridad de la base primero se debe comprobar.
* La inclusión o el cambio de cualquier información puede ser desastrosa, ya que los datos pueden tener efectos de largo alcance en otras tablas y secciones de la base de datos.
* Para mitigar el riesgo de esto, y para mantener la información que ya está en la base de datos sea precisa, la integridad siempre se comprueba antes de que cualquier tipo de cambio sea ejecutado.

**Funciones**

El lenguaje SQL tiene funciones incorporadas para hacer cálculos sobre los datos. Las funciones se pueden dividir en dos grupos (existen muchas más, que dependen del sistema de bases de datos que se utilice):

Funciones agregadas SQL, devuelve un sólo valor, calculado con los valores de una columna.

1. **AVG() -** La media de los valores
2. **COUNT() -** El número de filas
3. **MAX() -** El valor más grande
4. **MIN() -** El valor más pequeño
5. **SUM() -** La suma de los valores
6. **GROUP BY -** Es una sentencia que va muy ligada a las funciones agregadas

**Funciones escalares SQL, devuelve un sólo valor basándose en el valor de entrada.**

1. **UCASE() -** Convierte un campo a mayúsculas
2. **LCASE() -** Convierte un campo a minúsculas
3. **MID() -** Extrae caracteres de un campo de texto
4. **LEN() -** Devuelve la longitud de un campo de texto
5. **NOW() -** Devuelve la hora y fecha actuales del sistema
6. **FORMAT() -** Da formato a un formato para mostrarlo

**Funciones agregadas SQL**

**1. AVG()**

La función *AVG()* devuelve la media de valores de una columna numérica.

SELECT AVG (nombreColumna) FROM nombreTabla;

La siguiente sentencia SQL devuelve la media del stock que hay en el almacén:

SELECT AVG (Stock) FROM Productos;

La siguiente sentencia muestra el NombreProducto y el Precio de los registros que tienen un Precio por encima de la media:

SELECT NombreProducto, Precio FROM Productos

WHERE Precio > (SELECT AVG (Precio) FROM Productos);

**2. COUNT()**

La función *COUNT()* devuelve el número de filas que cumplen con un determinado criterio:

Número de valores en una columna (los valores NULL no se cuentan):

SELECT COUNT (nombreColumna) FROM nombreTabla;

Número de registros en una tabla

SELECT COUNT(\*) FROM nombreTabla;

Número de valores distintos de una columna

SELECT COUNT (DISTINCT nombreColumna) FROM nombreTabla;

**3. MAX()**

La función *MAX()* devuelve el mayor valor de la columna seleccionada:

SELECT MAX (nombreColumna) FROM nombreTabla;

De la tabla productos vamos a coger el producto más caro:

SELECT MAX (Precio) AS ProductoMasCaro FROM Productos;

**4. MIN()**

La función *MIN()* devuelve el menor valor de la columna seleccionada:

SELECT MIN (nombreColumna) FROM nombreTabla;

De la tabla productos vamos a coger el producto más barato:

SELECT MIN (Precio) AS ProductoMasBarato FROM Productos;

**5. SUM()**

La función *SUM()* devuelve la suma de una columna numérica:

SELECT SUM (nombreColumna) FROM nombreTabla;

De la tabla productos vamos a coger el número total de productos en stock:

SELECT SUM (Stock) AS ProductosTotales FROM Productos;

**6. GROUP BY**

La sentencia GROUP BY se utiliza junto con las funciones agregadas para agrupar en un *result-set* una o más columnas.

SELECT nombreColumna, funcion\_agregada(nombreColumna)

FROM nombreTabla

WHERE nombreColumna operador valor

GROUP BY nombreColumna;

**Funciones escalares SQL**

**7. UCASE()**

La función *UCASE()* convierte el valor de un campo a mayúsculas.

SELECT UCASE (nombreColumna) FROM nombreTabla;

Si queremos obtener todos los nombres de los clientes en mayúsculas:

SELECT UCASE (NombreCliente) AS Cliente FROM Clientes;

**8. LCASE()**

La función *LCASE()* convierte el valor de un campo en minúsculas:

SELECT LCASE (nombreColumna) FROM nombreTabla;

Si queremos obtener todos los nombres de los clientes en minúsculas:

SELECT LCASE (NombreCliente) FROM Clientes;

**9. MID()**

La función *MID()* se usa para extraer caracteres desde un campo de texto.

SELECT MID (nombreColumna, inicio [, longitud]) AS unAlias FROM nombreTabla;

Los campos obligatorios son *nombreColumna* e *inicio* (especifica la posición donde comenzar la extracción). El campo opcional es *longitud*, que especifica el número de caracteres a devolver. Si se omite, devuelve todos los caracteres hasta el final del texto.

Vamos a extraer por ejemplo los primeros cuatro caracteres de NombreProducto de la tabla Productos:

SELECT MID (nombreProducto, 1, 4) AS ProductoCorto

From Productos;

**10. LEN()**

La función *LEN()* devuelve la longitud de un campo de texto.

SELECT LEN (nombreColumna) FROM nombreTabla;

Si queremos obtener nombreProducto y la longitud de las descripciones de los productos de la tabla Productos:

SELECT NombreProducto, LEN(Descripcion) as LongitudDescripcion FROM Productos;

**11. NOW()**

La función *NOW()* devuelve la hora y fecha actuales.

SELECT NOW() FROM nombreTabla;

**Si queremos devolver el nombreProducto y Precio de hoy de la tabla Productos:**

SELECT NombreProducto, Precio, Now() AS PrecioProductosHoy FROM Productos;

**12. FORMAT()**

**La función *FORMAT()* se usa para formatear cómo se mostrará un campo.**

SELECT FORMAT(nombreColumna, formato) FROM nombreTabla;

**Vamos a mostrar un formato de fecha del ejemplo anterior:**

SELECT NombreProducto, Precio, FORMAT(Now(), 'YYYY-MM-DD') AS PrecioProductosHoy FROM Productos;

**12 Índice**

Un índice es una estructura de disco asociada con una tabla o una vista que acelera la recuperación de filas de la tabla o de la vista. Un índice contiene claves generadas a partir de una o varias columnas de la tabla o la vista. Dichas claves están almacenadas en una estructura (árbol b) que permite que SQL Server busque de forma rápida y eficiente la fila o filas asociadas a los valores de cada clave.

## **Índice Único**

Un índice único es usado para mantener la integridad de la información de las columnas sobre las cuales es creado al asegurar que no hay valores duplicados en el índice clave, y las filas de la tabla, sobre la cual el índice es creado. Esto aseguró que la información será única basada en el índice clave, dependiendo de las características de la información que es almacenada en la columna del índice clave o la lista de columnas. Si el índice Único clave consiste en una columna, el SQL Server garantizará que cada valor en el índice clave es único. Por otro lado, el índice Único clave consiste en múltiples columnas, cada combinación de valores en ese índice clave debería ser único. Puedes definir ambos el índice Agrupado y No Agrupado para ser único, siempre que la información en esos índices clave sean únicos.

Un índice Único será creado automáticamente cuando defines una restricción de CLAVE PRIMARIA o CLAVE ÚNICA en las columnas específicas. En todos los casos, crear un índice Único en la información única, en vez de crear un índice no único en la misma información, es muy recomendable, ya que ayuda al Optimizador de Consultas SQL Server a generar el plan de ejecución más eficiente basado en la información adicional útil provista por el índice.

**CREATE TABLE** sales.visits (

visit\_id INT PRIMARY KEY IDENTITY (1, 1),

first\_name VARCHAR (50) NOT NULL,

last\_name VARCHAR (50) NOT NULL,

visited\_at DATETIME,

phone VARCHAR(20),

store\_id INT NOT NULL,

FOREIGN KEY (store\_id) REFERENCES sales.stores (store\_id)

);

**14) Seguridad en base de datos**

## ¿Qué es seguridad de datos?

La seguridad de datos, también conocida como seguridad de la información o [seguridad informática](https://www.powerdata.es/el-valor-de-la-gestion-de-datos/bid/234639/C-mo-evaluar-el-riesgo-de-la-seguridad-inform-tica-de-una-empresa). Se trata de un aspecto que tiene que ver con la protección de datos contra accesos no autorizados y para protegerlos de una posible corrupción durante todo su ciclo de vida.

Seguridad de datos incluye conceptos como encriptación de datos, tokenización y prácticas de gestión de claves que ayudan a proteger los datos en todas las aplicaciones y plataformas de una organización.

**15) Usuario base de datos**

Usuario: Es una persona que utiliza una computadora o un servicio de red. Los usuarios de sistemas informáticos y productos de software generalmente carecen de la experiencia técnica necesaria para comprender completamente cómo funcionan

# **TIPOS DE USUARIOS EN BASE DE DATOS**

Tipos de Usuarios en Base de Datos

* Usuario Final: es la persona que utiliza los datos, esta persona ve datos convertidos en información:
* Desarrollador de Aplicaciones: es la persona que desarrolla los sistemas que interactúan con la Base de Datos.
* DBA: es la persona que asegura integridad, consistencia, redundancia, seguridad este es el Administrador de Base de Datos quien se encarga de realizar el mantenimiento diario o periódico de los datos.

**16) Esquema de una base de datos**

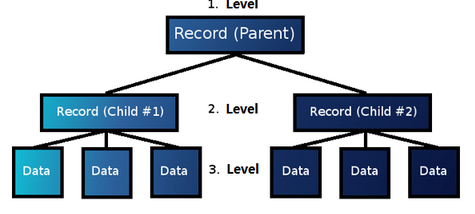
(*database schema*) describe la estructura de una base de datos, en un lenguaje formal soportado por un [sistema de gestión de base de datos](https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos) (DBMS).

En una base de datos relacional, el esquema define sus tablas, sus campos en cada tabla y las relaciones entre cada campo y cada tabla.

## 

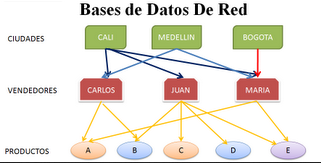
## **Evolución de los modelos de bases de datos**

### **Modelo jerárquico de base de datos:** Este es el modelo más antiguo, XML utiliza este sistema para guardar datos y algunas compañías de seguros y bancos recurren a las bases de datos jerárquicas sobre todo en las aplicaciones más antiguas de base de datos. El sistema de base de datos jerárquico más conocido es IMS/DB de IBM.



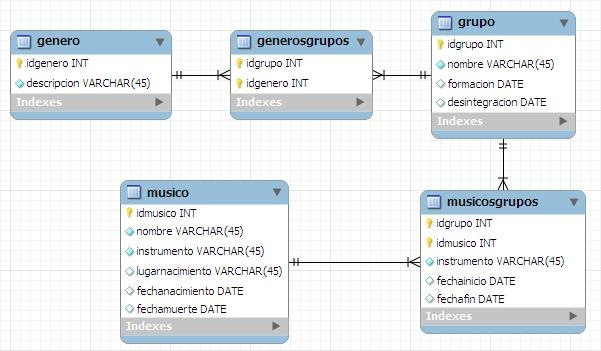
### 

### Base de datos en red: los registros o *récords* no revelan relaciones padre-hijo estrictas, sino que cada registro puede tener múltiples precedentes, lo que le da la **estructura en red** de su nombre. Para acceder a un registro tampoco hay, por eso mismo, un camino único e invariable.

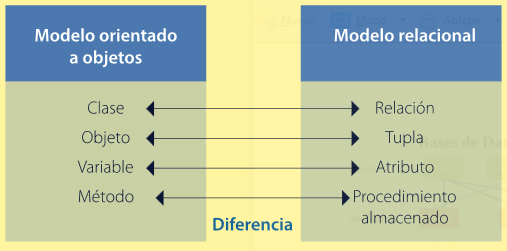


Hoy el modelo de base de datos en red se utiliza, sobre todo, en los **grandes ordenadores**. En otros campos se sigue confiando en el modelo jerárquico (clientes de IBM, sobre todo) o se ha dado el paso hacia el modelo relacional, mucho más flexible y fácil de utilizar.

### Modelo de base de datos relacional: En el modelo relacional se utiliza un grupo de tablas para representar los datos y las relaciones entre ellos.



### Modelo de base de datos orientado a objetos: es un sistema de gestión de base de datos mediante el cual representamos la información en forma de objetos que son utilizados en programación orientada a objetos.



### Modelo de base de datos orientado a documentos: también llamada una base de datos orientada a documentos u tienda de documentos, es un subconjunto de un tipo de base de datos NoSQL.

### A diferencia de las bases de datos relacionales tradicionales, el modelo de datos en una base de datos de documentos no está estructurado en un formato de tabla de filas y columnas.



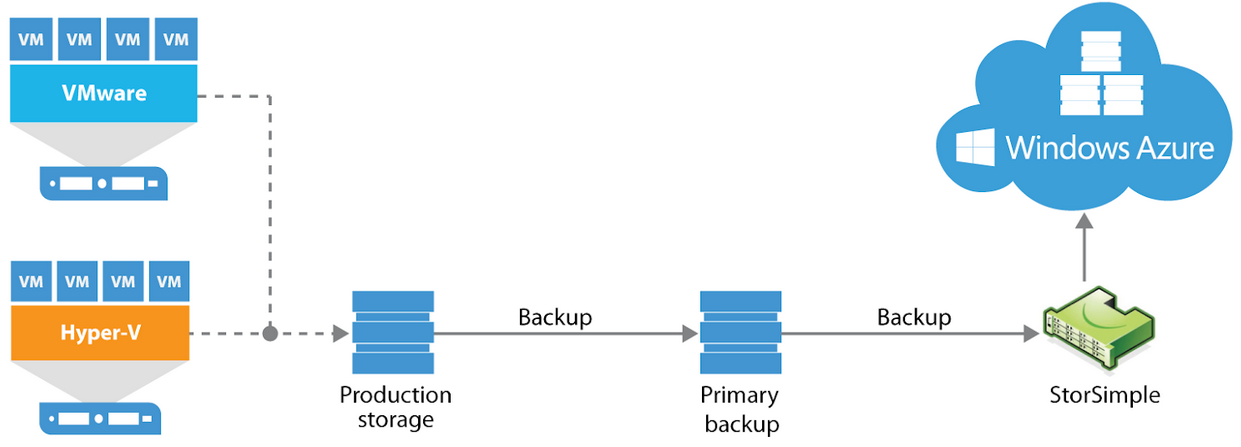
**Backup**

Es una copia de seguridad o el [proceso](https://conceptodefinicion.de/proceso/) de copia de seguridad. Backup se refiere a la copia y archivo de [datos](https://conceptodefinicion.de/datos/) de la computadora de modo que se puede utilizar para restaurar la información original después de una eventual pérdida de datos.

**Propósito Backup**

* Protección de datos.
* Insuficiencia de medios.
* Errores del usuario
* Errores de aplicación
* La preservación de datos y retención histórica.
* La transferencia de datos.

El propósito de los backups y la recuperación es restaurar una base de datos que ha fallado. Los backups protegen la base de datos contra problemas como el fallo de hardware la falta de medios errores de usuario a nivel de hardware un controlador o disco duro defectuoso puede introducir errores. Los usuarios también pueden causar errores de datos simplemente mediante la emisión de comandos que no deben ser emitidos. Esos mismos tipos de errores pueden ser causados por una aplicación con un error.



# 

# 

# 

# 

# 

# 

# 

# 

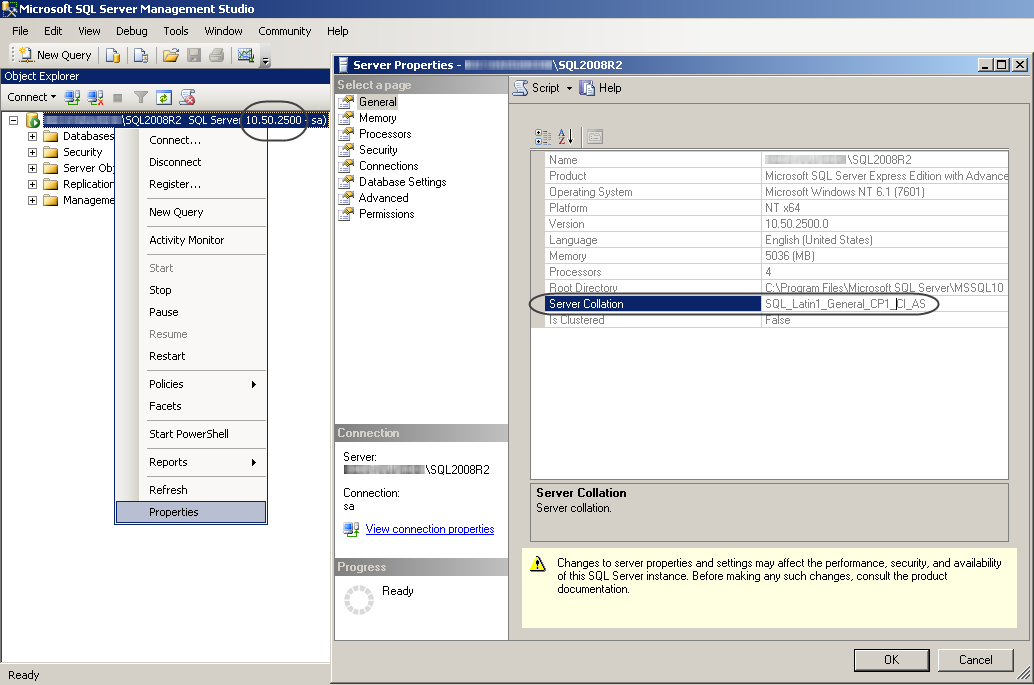
# **Restaurar backups en SQL Server**

Antes de comenzar. Para restaurar un backup se debe considerar:

**1. Correspondencia en la intercalación (collation).**

Asegúrese de que la instancia de SQL Server sobre la cuál va a restaurar un backup, tenga la misma configuración de intercalación de la instancia SQL Server de origen sobre la cual se extrajo el backup.

Para revisar la intercalación de su instancia SQL Server, puede hacer clic derecho sobre la instancia y acceder a sus propiedades:

****

**2. No tener conexiones activas.**

Para completar una restauración, SQL Server requieres que no existan conexiones activas en el momento.

Se recomienda explícitamente que este tipo de tareas se coordine con otros miembros del equipo de manera que el equipo conozca que se va a restaurar la base de datos de un proyecto, y que por ejemplo, no esté el proyecto en uso.

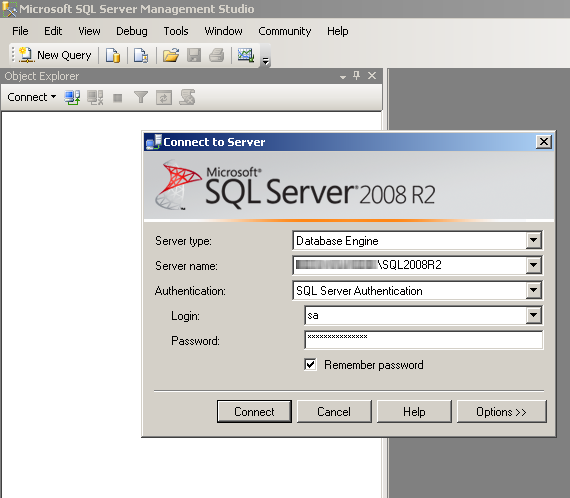
Si usted está ejecutando sus procesos sobre una plataforma de .NET, tenga en cuenta que es muy probable que el servicio de Bizagi llamado Scheduler, esté iniciado y con una conexión a la base de datos.

Para detener este servicio, y cerrar esa conexión, utilice el panel de servicios de Windows.

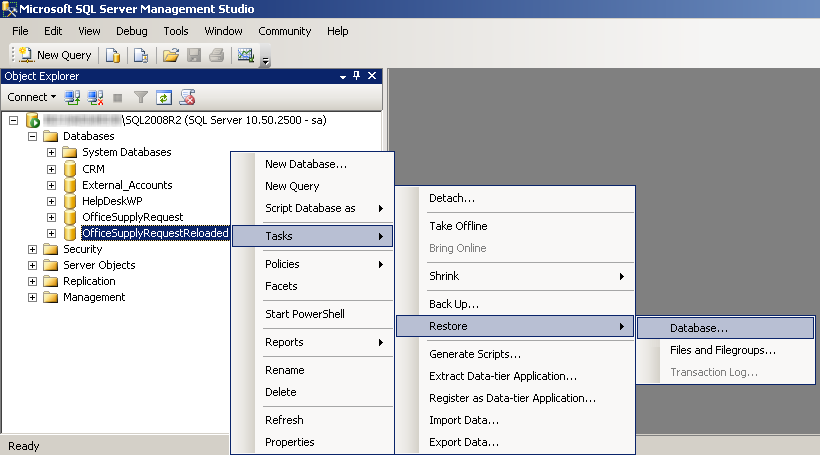
**Restaurar un Backup**

Restaure un backup .bak con los siguientes pasos:

1. Autentíquese en su instancia de SQL Server (login) a través de SQL Server Management Studio.

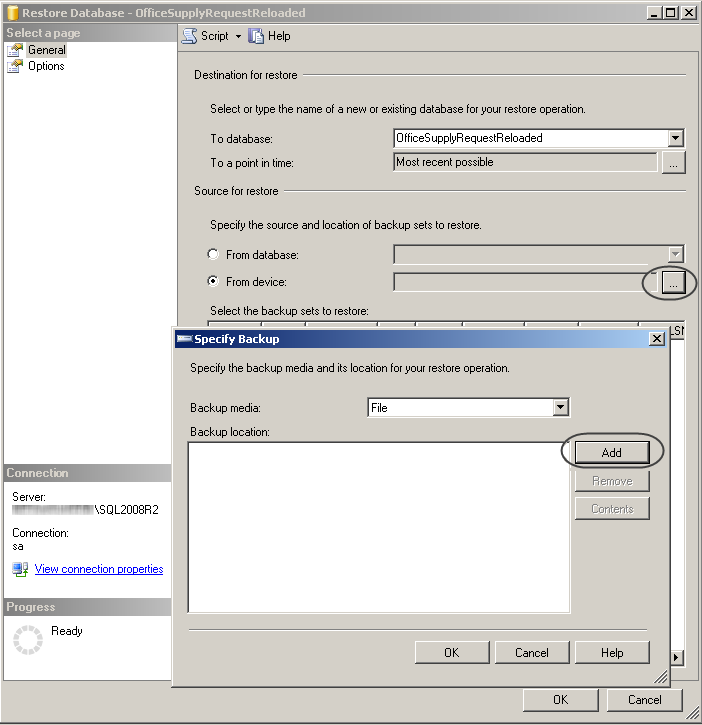
****

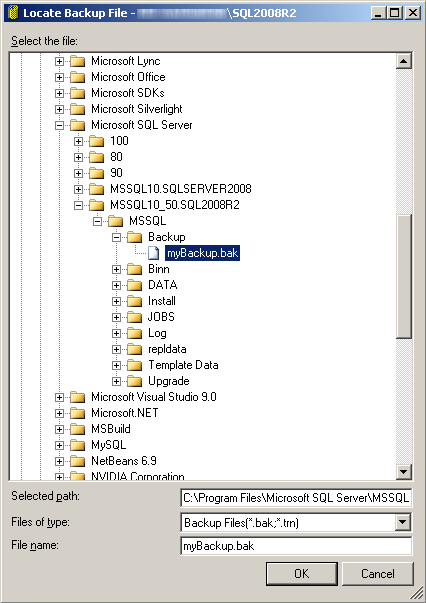
2. Ubique la Base de datos y dé clic derecho sobre ésta. Seleccione la opción de Restaurar -> Base de datos:

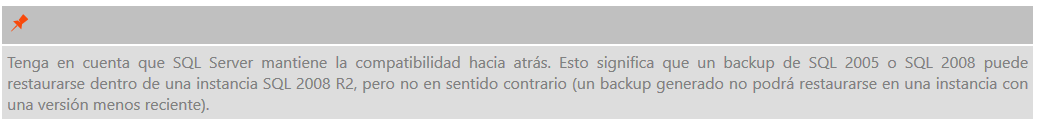
****

3. Especifique que la Base de datos será restaurada desde un dispositivo.

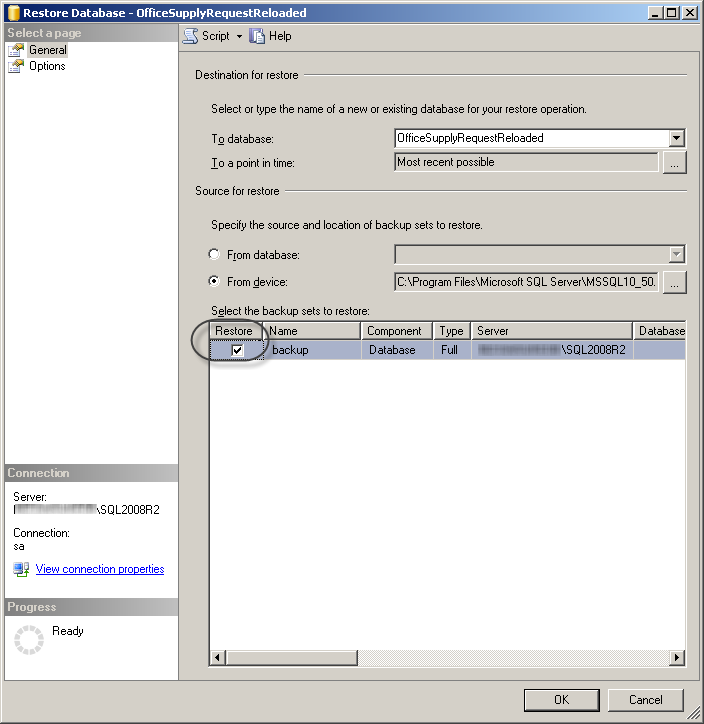
Navegue hasta seleccionar el archivo .bak de origen:

****

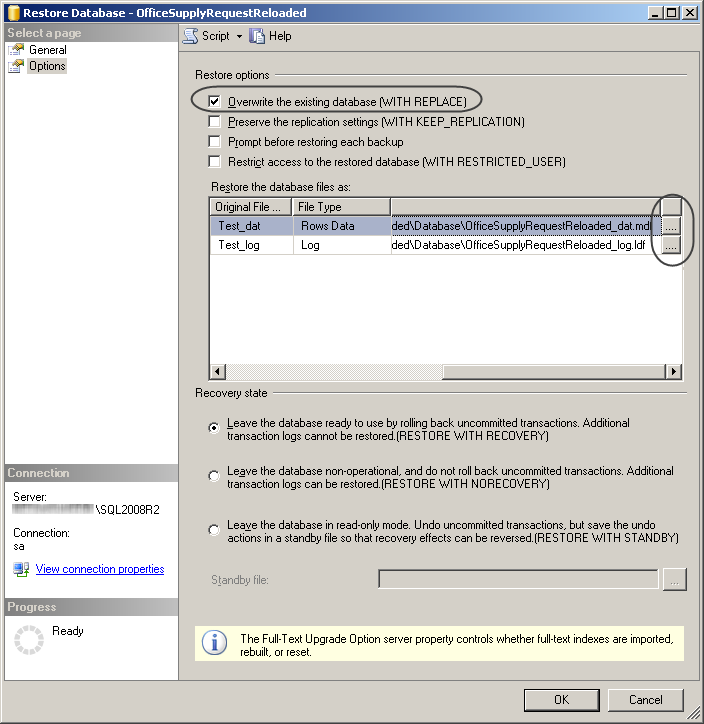
****

****

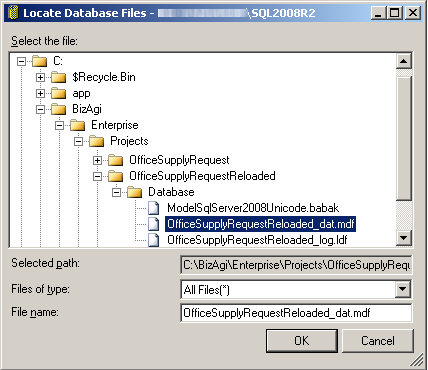
4. Marque el archivo con la opción de Restaurar:

****

5. Vaya al tab de Opciones, y marque la opción de Sobrescritura (Overwrite the existing database).



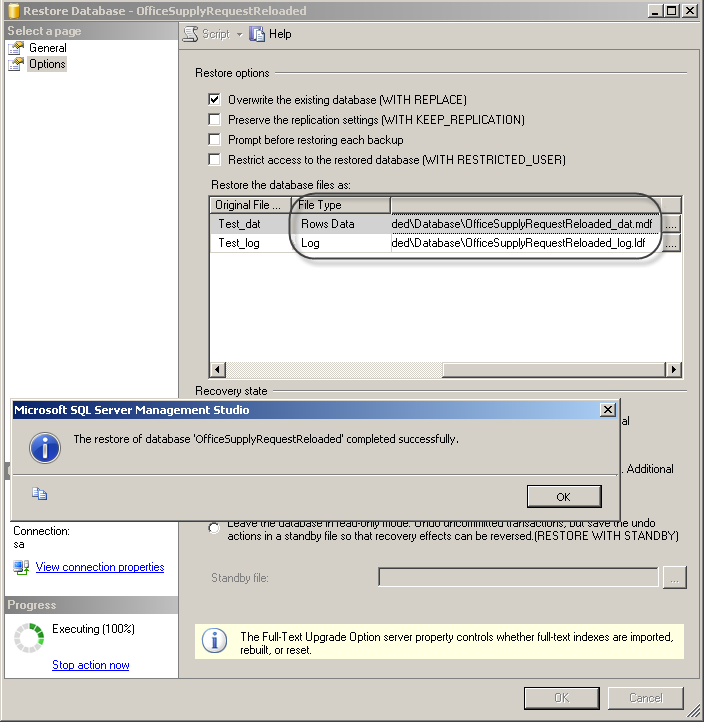
Asegúrese de seleccionar el destino de los archivos usados por la Base de datos (.dat y .log).



Nótese que estos archivos se ubican por defecto en la siguiente ruta:

* "*C:\Bizagi\Projects\[su\_proyecto]\Database\*", si el servidor de base de datos es el mismo servidor de Bizagi (el proyecto usa una base de datos local).
* En la ruta de la instancia SQL Server (por defecto "C:\Program Files (x86)\Microsoft SQL Server\[instancia]\MSSQL\Data\"), si el servidor de base de datos no es el mismo servidor de Bizagi.

6. Haga clic en OK cuando la operación de restauración se haya completado.



Importante

Después de restaurar un backup en un proyecto de Bizagi, se debe refrescar la memoria caché del servidor Web esto se logra ejecutando un IISReset, y reiniciando el servicio Scheduler.

**Diccionario de Datos**

El diccionario de datos es un repositorio de metadatos que contiene las definiciones de los objetos de datos, descripciones y relaciones entre sí. El diccionario es muy importante en el desarrollo de aplicaciones, ya que permite a los involucrados en el desarrollo conocer a detalle las bases de datos.

**Terminología**

1. **SGBD**

Sistema de Gestor de Base de Datos - Database Management System. El gestor de datos es un sistema de software invisible para el usuario final, compuesto por un lenguaje de definición de datos, un lenguaje de manipulación y de consulta, que puede trabajar a distintos niveles.

1. ***LAMP***

Siglas de linux Apache MySQL y PHP.

**Bibliografía**

* <http://www.maestrosdelweb.com/que-son-las-bases-de-datos/>
* <https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/que-es-un-gestor-de-datos-y-para-que-sirve>
* <https://desarrolloweb.com/home/mysql>
* <https://www.ecured.cu/Herramienta_CASE>
* <https://www.ecured.cu/Metamodelado>
* <https://hostingpedia.net/postgresql.html>
* <http://dblearner.com/el-diccionario-de-datos/>
* <https://www.ibm.com/support/knowledgecenter/es/SS8JFY_9.2.0/com.ibm.lmt.doc/Inventory/admin/t_sql_restore.html>
* <https://conceptodefinicion.de/backup/>
* <https://www.lucidchart.com/pages/es/que-es-un-esquema-de-base-de-datos>
* <https://www.tecnologias-informacion.com/documentosbd.html>