IBM Skills Build: SQL Server

- Instalar SQL Server
- Instalar SQL Server management studio
- tablas, campos,
- keys primarios y externos
- normalización de bases de datos
- Diagrama entidad relación
- SELECT, INSERT, UPDATE, DELETE
- WHERE, TOP, OTHER BY, DISTINCT
- Funciones de agregado
- Operadores lógicos AND, OR, IN, LIKE, BETWEEN
- Variables y store procedures
- IF, WHILE, CASE
- Operadores aritmeticos y de comparación
- CREATE, ALTER, DROP, TRUNCATE
- LEFT, RIGHT, REPLICATE, DATE, ADD
- transacciones
- INNER JOIN, JOIN LEFT, JOIN RIGHT, UNION
- FUNCIONES DE USUARIO
- tablas temporales físicas y en memoria
- Vistas
- TRIGGERS, insert, update, delete, before after
- Shedule jobs
- Back up / Restore
- Administración usuarios, Roles, Schemas, Permi
- Integrar datos de Excel
- SQL Server Profiler

Boehringer Ingelheim

Instalación SQL Server

instalamos la edición Developer y de ella la versión custom

la instalamos en C:\SQL2022

Para la instalación elegimos la opción
New SQL Server standalone (la primera)

la edición free edition: Developer

Azure extension for SQL Server: NO

las características basicas a instalar:
SQL Server replication
Full-text and Semantic Extractions integration services

la instancia por defaulf: MSSQLSERVER

El SQL Server Agent es el encargado de ejecutar varias tareas automaticamente y para ello debe iniciarse con la maquina.
Start up type => automatic

Nos conectaremos al SQL en Mixed Mode
autenticación < windows
SQL

Establecemos la contraseña para el usuario "sa" System administrator. En mi caso la de 42.

Hay que añadir un usuario de windows

Reiniciamos la máquina

Boehringer Ingelheim

Verificación de servicios (video 10)

iniciamos la utilidad services/servicios de windows y verificamos en la columna nombre que SQL Server (MSSQLSERVER) está en estado ejecución.
la instancia instalada

Si no lo estuviera lo iniciamos desde la barra de iconos superior del programa ▷ □ ❚❚ |▶

instalación MS SQL Server Management Studio Release 20.1 (12)

se ejecuta con el comando ssms (13)

El server name empieza por . y no es necesario poner la instancia por defecto ya la encuentra.

Management studio nos permite gestionar los objetos de la base de datos (instancias) (14)

Primera impresión:

Object Explorer (F8)
muestra a que instancia estamos conectados.

Solution Explorer
Muestra todos los soluciones que gestionamos veremos en los ≠ proyectos

la carpeta miscelaneos mantiene los queries abiertos y no salvados

Dentro de una solución podemos crear un Project con el botón derecho
*.ssmssqlproj

(ctrl+n → new query
veremos en cuadro blanco Base de datos a la que esta conectado (master)
→ crea nueva solución
*.ssmssln es la extensión de las soluciones)

un Project tiene 3 carpetas: connections, queries (para los *.sql) y miscellaneous

Asociaremos a un project una conexión y así no requerirá la contraseña (video 15)

Boehringer Ingelheim

SSMS: Herramientas Video (16)

Podemos reubicar las explorer arrastrándolas a nuestro gusto. Hay una posición default en window/Reset window layout
CtrL+F5 F5
Parse ≠ Execute. Parse activa el analizador sintáctico
Se puede comentar e indentar bloques de texto

Solución Video (17)

Podemos crear una solución en blanco.
Podemos crear un proyecto dentro de una solución.
Hay un tick para crear el directorio para la solución

Atajos de teclado más usados (18)

ctrl+N	New query	F5	ejecuta script
ctrl+O	open script	F8	mostrar object explorer
ctrl+S	Guardar script	ctrl+Alt+L	" solución panel
ctrl+Shift+Upper / ctrl+Shift+Lower	cambiar texto	ctrl+R	" Result panel
ctrl+6		ctrl+D	cuadrícula
		ctrl+T	texto

S3

¿Qué es una Base de Datos? (19)

¿Que es una tabla, campos y registros (20)

¿Que es un valor NULL? (21)

¿Que es una Primary Key (22)

un identificador único de un registro
no debe admitir valores NULL
Hay primary key < compuestos / simples
una tabla solo poseerá una ~~tab~~ clave primaria

Boehringer Ingelheim

¿Que es una clave externa? (23)

consistencia de datos, integridad referencial
una Foreig key es una relación entre dos tablas.
En un registro solo se puede, en una FK, ingresar ~~regis~~ valores que hay en la tabla relacionada
una tabla puede tener más de una FK.
un campo FK puede aceptar valores NULL
la FK debe ser del mismo tipo de datos que su campo relacionado.

Normalización de Base de datos (24)

Existen 3 formas normales.

Tipos de datos - SQL Server (25)

Bit tinyint smallint int bigint

1	1	2	4	8	bytes
0	0	-32768	-2147483648	-9.223.372.036.854.775.80	
1	255	+32767	+2147483647		

money 4 decimales de precisión
Decimal (precisión, escala)
dígitos decimales

char
varchar

Datetime 1/1/1753 – 31/12/9999
00:00:00 → 23:59:59.997

Boehringer Ingelheim

Creando nuestra Base de Datos (27)

Object Explorer → Connect → Database Engine

OE → databases → New database

Database name = CentroMedico CamelCase

Hay 2 Database files

Objetos CentroMedico

Transacciones CentroMedico_log

En los paths orientar a dispositivos externos o a otras particiones para facilitar las copias de seguridad y las migraciones o actualizaciones del sistema operativo

se crean dos archivos: { CentroMedico.mdf / CentroMedico_log.ldf

El archivo MDF (Master data file) contiene tablas, índices, procedimientos y objetos

El archivo LDF (log data file) contiene registro de transacciones y fallos

Abriendo ventana Script. Estableciendo default db

menu New query | CTRL+N

Por defecto se asigna a la base de datos master pero lo podemos cambiar en el ListBox desplegable

Eso es una asignación temporal. Para que la asignación sea permanente...

OE - Security - Logins - usuario → diálogo - general

cambiamos el default database.

Boehringer Ingelheim

Definiendo la tabla pacientes (30)

Podemos definir tablas con el diseñador de tablas o con el lenguaje transact-SQL

OE → Db → CentroMedico → tables → New table

column name	Data type	Allow Nulls
idPaciente	int	campo obligatorio o no y admite nulls
nombre	varchar(50)	0..50
apellido	varchar(50)	
fNacimiento	date	
domicilio	varchar(50)	
idPais	char(3)	3 siempre
telefono	varchar(20)	0..20
email	varchar(30)	0..30
observacion	varchar(1000)	0..1000

Por defecto el campo no es obligatorio

click en el disquette para salvar.
nombre de la tabla en singular. Gracias Paciente

Aparece el nombre de la tabla recien creada en la lista de tablas de la bd centromedico antecedida por el nombre del esquema dbo.

un esquema es un espacio de nombres para almacenar tablas

Modificando una tabla existente (31)

OE - Db → CentroMedico → tables → dbo.Paciente → Design muestra el diseñador de tablas. cambiamos y salvamos

Por defecto no está permitido hacer cambios a tablas existentes, ya que exige al motor su recreación.

Se cambia esto en: menu tools / opciones / designers / table and db designers / Prevent saving changes that require table re-creation.

Boehringer Ingelheim

Como establecer una Primary key (32)

Oe-db-centroMedico-tables-dbo.Paciente-Designs

seleccionamos la fila que queremos convertir en Primary key. < boton derecho
< como set Primary key

cuando un campo se define PK, automaticamente se desmarca su opción Allow Nulls. una PK no admite valores NULL

El nombre por defecto es PK_Paciente

Activando la propiedad IDENTITY (33)

Oe-db-centroMedico -> tables -> dbo.Paciente -> Designs

En columns Properties abajo aparece:

identity Specification
(is identity) yes/no
identity increment
identity seed

Al ser numérico el campo, cada vez que se inserte un registro en la tabla, se incrementará según el "increment", Empezando en "seed".

La propiedad IDENTITY solo aplica a campos numéricos (int/bigint). IDENTITY (seed=1, increment=3)

Al aplicarlo en una PK logramos valores no duplicados para generar registros únicos.

Como crear una base de datos con transact SQL (36)

CTRL+N CREATE DATABASE Ejemplo

F5

Como crear una tabla con transact SQL (37)

```
CREATE TABLE Paciente(
idPaciente   int          NOT NULL,
nombre       varchar(50)  NOT NULL,
apellido     varchar(50)  NULL,
fNacimiento  date         NULL,
domicilio    varchar(50)  NULL,

CONSTRAINT PK_Paciente PRIMARY KEY(idPaciente)
```

Boehringer Ingelheim

Definiendo la tabla Historias clinicas (38)

PK	idHistoria	int	NOT NULL IDENTITY(1,1)
	fechaHistoria	datetime	NULL
	observacion	varchar(2000)	NULL

crear la tabla Pais (40)

```
CREATE TABLE [dbo].[Pais](
  [idPais] [char](3) NOT NULL PRIMARY KEY,
  [pais] [varchar](30) NULL)
)
GO
```

crear la tabla turnos (41)

```
CREATE TABLE [dbo].[turno](
  [idturno] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,
  [fechaturno] [datetime] NULL,
  [estado] [smallint] NULL,
  [observacion] [varchar](30)
)
GO
```

crear la tabla relacion (42)

```
CREATE TABLE [dbo].[turnoPaciente](
  [idturno] [int] NOT NULL,
  [idPaciente] [int] NOT NULL,
  [idMedico] [int] NOT NULL,
  CONSTRAINT [PK_turnoPaciente] PRIMARY KEY CLUSTERED
  (
   [idturno] ASC,
   [idPaciente] ASC,
   [idMedico] ASC
  )
)
GO
```

las tablas de relaciones no llevan la propiedad IDENTITY
P.Q. registran números ya existentes en otras tablas (43)

Boehringer Ingelheim

Crear la tabla TurnoEstado (44)

```
CREATE TABLE [dbo].[TurnoEstado](
  [idEstado] [smallint] NOT NULL PRIMARY KEY
  [descripcion] [varchar] (50) NULL )
GO
```

crear la tabla Especialidad (45)

```
CREATE TABLE [dbo].[Especialidad](
  [idEspecialidad] [int] NOT NULL IDENTITY(1,1) PRIMARY KEY
  [Especialidad] [varchar] (30) NULL)
GO
```

Definiendo la tabla pago (46)

```
CREATE TABLE [dbo].[Pago](
  idpago int IDENTITY (1,1) NOT NULL PRIMARE KEY
  concepto tinyint NOT NULL,
  fecha datetime NOT NULL,
  monto money NOT NULL,
  estado tinyint
  obs varchar(1000)
```

Tabla de relación Pago Paciente turno (47)

```
CERATE TABLE [dbo].[PagoPaciente](
  [idPago] [int] NOT NULL,
  [idPaciente] [int] NOT NULL,
  [idturno] [int] NOT NULL,
  PRIMARY KEY (idPago, idPaciente, idturno)
```

Crear la tabla Medico (48)

```
CREATE TABLE [dbo].[Medico](
  [idMedico] [int] NOT NULL IDENTITY(1,1) PRIMARE KEY,
  [nombre] [varchar] (50),
  [apellido] [varchar] (50)
)
```

Boehringer Ingelheim

Crear tabla MedicoEspecialidad (49)

```
CREATE TABLE [dbo].[MedicoEspecialidad](
  [idMedico]        [int] NOT NULL,
  [idEspecialidad] [int] NOT NULL,
  [descripción]  [varchar](50)
  CONSTRAIN PK_MedicoEspecialidad PRIMARY KEY CLUSTERED
  ( [idMedico] ASC,
    [idEspecialidad] ASC
  )
```

Esto implementa una relación muchos a muchos video 68 y 69

Definiendo la tabla concepto (50)

```
CREATE TABLE [dbo].[Concepto](
  [idConcepto] [int] NOT NULL IDENTITY(1,1) PRIMARY KEY,
  [descripción][varchar](100) NOT NULL)
```

Definiendo tipos de datos de usuario (51)

OE - DB - CentroMedico - Programmability - Types - User-Defined Data Types - New User-Defined

Importante Hacerlo, facilita SP

En el dialogo nombro el tipo de dato "Paciente" y establezco de que tipo es.

Con la instrucción SQL puedo hacer lo mismo

```
CREATE TYPE [dbo].[paciente] FROM [int] NOT NULL
```

Ahora que podemos cambiar los tipos de datos por default en las columnas de las tablas por los tipos de datos definidos por el usuario

Asignar el tipo de dato Paciente (52)

En las tablas turnoPaciente / PagoPaciente / Paciente
HistoriaPaciente

Crear y Asignar el tipo de dato Historia (53) (54) (55)

```
CREATE TYPE [dbo][Historia] FROM [int] NOT NULL
CREATE TYPE [dbo][Medico] FROM [int] NOT NULL
CREATE TYPE [dbo].[observacion] FROM [varchar](100) NULL
```

Boehringer Ingelheim

Sección 5 Relaciones y Restricciones entre Tablas:

Introducción (57)

Conceptos Relaciones entre tablas (58)

Las relaciones entre tablas no son obligatorias en una BD Relacional

La integridad referencial viene garantizada por las ~~reglas~~ Reglas

Reglas = relación PK – FK PK / FK > = tipo dato

La regla evita que en el campo FK de una tabla se pueda colocar un valor que no existe en la tabla que tiene el PK

La regla evita que borremos un registro de una tabla cuya PK es la FK de otra tabla.

Modelo Entidad Relación DER (59)

Diagrama ER

OE-DB-Centro Médico - Database Diagrams → NEW DB Diagram

Elegimos las tablas a relacionar con una regla

Creamos la FK en una tabla desde el OE: } PK BLACK / FK WHITE

OE-DB-CM-Tables-Paciente-Keys-NEW FK

En el diálogo: Tables And Columns specification ...

En el nuevo diálogo elegimos la tabla que tiene la PK de la relación (País) y la tabla que contiene la FK de la relación (Paciente) y de ambos elegimos los campos concretos. (de igual tipo)

El nombre de la FK es FK Paciente_País

GUARDAR CAMBIOS

En el DER → RIGHT CLICK → arrange tables

En la RELACION → RIGHT CLICK → Properties

Boehringer Ingelheim

Relación 1 a muchos (61)

El valor de la PK puede estar en muchos registros con la FK

Borrado de FK (62) ∃ 3 formas

a) OE - DB - CMedico → tables → Paciente - keys → FK → Right click, Delete <DELETE>

b) .

c) en el DER, Right click en la relación → Delete Relationship From DB

+ GRABAR CAMBIOS + REFRESH

Edición del DER (63)

Right click en el DER → Add table

Relacionar un Turno con su Estado (64)

PK idestado
FK estado

Relacionar un Pago con su concepto (65

PK idconcepto
FK concepto

Relación Pago con turno y Paciente (66)

Add Pago Paciente

	Pagos
FK idpago	PK idpago
	Paciente
FK idpaciente	PK idpaciente
FK idturno	PK idturno

Relación turno con Medico y Paciente (67)

creación arrastrando un campo (FK) hacia el otro (PK)

Relación Medico y especialidad (68)

Relación Muchos a Muchos (69)

} van juntos

la tabla MedicoEspecialidad es la "3ª tabla" que representa una relación muchos a muchos.

un médico tiene varias especialidades

una Especialidad tiene varios médicos

Boehringer Ingelheim

Relación Historia clínica con el paciente y el medico (70)

Relación Uno a Uno (71)

una vez creada la tabla PacienteInfo
CREATETABLE PacienteInfo (idpaciente paciente, implanta BIT, diabetes BIT,
PRIMARY KEY (idpaciente))

Establecemos una relación. Pero antes vamos a definir que la PK de la tabla PacienteInfo es TAMBIEN FK en la tabla pacientelnfo. Para ello en la tabla PacienteInfo creo una FK (59) cuya PK es el idpaciente de la tabla paciente. Esta FK de la tabla PacienteInfo es idpaciente, que también es la PK de la tabla PacienteInfo.

Hemos podido crear la relación al revés P.C. ∃ paciente en la tabla pacientes.

Eliminar una Foreig Key (72)
lo he realizado según (62-a)

Boehringer Ingelheim

Sección 6 Manipulación de registros con INSERT y SELECT (DML: Data Manipulation Language)

Insertando y seleccionando países con SSMS (73)

OE-DB-Cen.Med-tables-país-Rclic-Edit top 200 rows
muestra como si fuera un Excel

OE-DB-CM-tables-país-R.clic → Select top 1000 rows
crea automáticamente un DML (SQL) un select
SELECT TOP (1000) [Idpais], [pais] From [Ejemplo].[dbo].[pais]
los datos mostrados no son editables

Insertar registros en la tabla país (74)

comprobando la restricción FK Paciente país (75)
edit top 200 rows de la tabla paciente

Insertar nuevos pacientes (76)

Insertar nuevo medico (77)

comando: SELECT FROM (78)
Permite seleccionar uno o más registros de la tabla
SELECT <lista de campos> FROM <tabla>
asterisco para todos

comando INSERT e INSERT multiple (79)
INSERT INTO <tabla> (<lista campos>) VALUES ( ), ( )
lista valores

valores tipo varchar entre comillas simples
valores tipo fecha 'año(4)-mes(2)-dia(2)': Formato valido
un campo con la propiedad identity no puede estar en la lista de campos
NULL, es admitido en los parentesis de valores
<lista de campos> indica los campos a insertar. no tiene p.q. ser la lista completa de campos.
" comilla, comilla es la introducción de un campo vacio.
los campos con la propiedad NOT NULL debe figurar en la lista de campos

Boehringer Ingelheim

Comando INSERT y Primary key (8c)

Pk nos evita duplicados.
INSERT INTO [dbo].[pais] (idpais, pais) VALUES ('ESP', 'España')
infracción en la restricción de PRIMARY KEY 'Pk_pais'

insertar registros en tabla Turno Estado (8f1) y verificar Pk (8f2)

El mensaje de alerta indica el valor de la clave duplicada

insertar nuevos turnos (8f3)

Hacer select * en Pacientes, Medicos y turnos para ver las Pk disponibles.

usar un INSERT INTO TurnoPaciente para insertar más, más rápido.

Comprobar restricciones FK turno, paciente Medico (8f4)

He intentado insertar una terna con 3 FK inexistentes en las tablas de las Pk. Se ha quejado una a una, en un orden diferente al orden de las columnas.

tras arreglar las FKs, se queja de la duplicidad de valores. Se queja de una terna duplicada aun que la Pk fue definida CLUSTERED

Boehringer Ingelheim

Seccion 7: clausulas SQL

TOP | ORDER BY | TOP con ORDER BY | DISTINCT | WHERE | GROUP B

la clausula TOP (87)

SELECT TOP (num) <lista campos> FROM <tabla>
num: indica el número de registros deseados

la clausula ORDER BY (88)

SELECT <lista campos> FROM <tabla> ORDER BY <campo> [DESC / ASC]
default

TOP y ORDER BY (89)

SELECT TOP 1 * FROM PACIENTE ORDER BY fnacimiento DESC
nos devuelve el + joven.

DISTINCT (90)

SELECT DISTINCT <campo> From <tabla>
obtenemos valores unicos/diferentes que hay en esa columna.
El profesor usa la expresión "valores agrupados por el mismo valor"

GROUP BY (91)

SELECT <campos> FROM <tabla> GROUP BY <columna>
si la lista de campos es unaria, el campo debe estar despues del Group By
si la lista de campos no es unaria, todos los campos que no estan en GROUP BY deben llevar una función de agregación.

WHERE (92)

filtraremos registros segun los valores de determinados campos

SELECT * FROM <tabla> WHERE <columna> = <Valor>

si tipo columna es varchar / fecha valor comilla simple

Boehringer Ingelheim

Sección 8 Manipulación de Registros

UPDATE DELETE

comando UPDATE (94)

actualizamos campos y registros de una tabla

UPDATE <tabla> SET <columna> = <valor>

Pone el valor en ~~todos~~ el campo columna en TODOS los registros de la tabla

Para reducir riesgos hay que vincular el comando UPDATE con la clausula WHERE <col> = <val>

UPDATE <tabla> SET <col> = <val>, <col> = <val> WHERE.

para actualizar varios campos a la vez

comando DELETE (95)

Elimina registros de las tablas

DELETE FROM <tabla> WHERE <col> = <val>

cuidado con olvidarse la clausula WHERE en el comando DELETE

comando DELETE y FK (96)

No puedo, por las reglas de integridad referencial, borrar un registro de una tabla cuya PK esta en la FK de otra tabla. (violación de restricción REFERENCE FK

1.- Debo borrar el registro de la tabla con la FK

2.- puedo borrar el registro de la tabla con la PK.

Boehringer Ingelheim

Sección 9 Ejercicios complementarios

Establecer propiedad ALLOW NULLS (99)

OE-DB-CM-tables → Medico → R.clic → Design
column properties allow Nulls X

Establecer propiedad IDENTITY (100)

OE-DB-CM-tables → turnoEstado → R clic → Design → select row idEstado → column properties → identity specification

como tenía 7 valores en la tabla (0..6) he definido seed=7 para que fuera el próximo valor a generar. tras insertar 2 (7 y 8) y borrar los dos, la siguiente inserción dio 9.
concluyo que identity no aprovecha los códigos de los registros borrados.

Renombrar campo (101)

OE-DB-CM-tables-Pago → R clic → Design → cambiar

Eliminar campo (102)

OE-DB-CM-tables-paciente-R clic → Design → click column
R click → Delete column

SELECT ORDER BY (103)

SELECT * FROM [ejemplo].[dbo].[Pagos] ORDER BY fecha

Devolver el paciente más joven (104)

SELECT TOP 1 nombre, apellido FROM paciente order BY fnacim DES

Boehringer Ingelheim

Seccion 10: Funciones de Agregado

Funcion MAX y MIN (105)

Si aplicadas a varchar usa el orden alfabético

SELECT MAX(fecha) From pago → 2019-09-27

SELECT FUNC(<col>) From <table>

Funcion SUM (106)

Solo aplica a campos numéricos

SUM puede sumar varios campos simultaneamente

por ese motivo no devuelve un nombre de columna en los resultados

SELECT SUM (<col> o <Expression) AS alias
FROM <tabla>

SUM (monto + 20)

SUM (monto) + 20

Funcion AVG (107)

igual que el anterior

Funcion COUNT (108)

cuenta el numero de registro

Funcion HAVING (109)

SELECT estado FROM turno GROUP BY estado HAVING COUNT(estado)=2

1º Agrupa los registros de Turno por estado

2º Para cada valor de estado cuenta los registros que hay

3º Identifica los grupos con 2 registros

4º) Presenta en el select los valores de estado identificados

Boehringer Ingelheim

Sección 11 Operadores lógicos

Operador AND (110) OR (111) IN (112)

Son operadores para la cláusula WHERE
nos ayudan a establecer criterios para filtrar
IN nos permite filtrar por un conjunto de valores
SELECT * FROM turno WHERE estado IN (2,1,3)
si el campo es varchar, los valores de IN irán entre comillas simples

Operador LIKE (113)

se aplica a campos de cadena de caracter
Hay 2 metacaracteres % => ó más caracter
subrayado _ => 1

SELECT * FROM paciente WHERE fnacimiento LIKE '-----2-
2024-02-
seleccione los nacidos en Febrero

Operador NOT (114)

Niega los operadores IN o LIKE o el resultado de una expresión logica

Operador BETWEEN (115)

Filtra registros cuyos valores de un campo están en un rango de valores, fechas o palabras
cuidado al filtrar fechas si hay 'time' guardado en la fecha de tipo datetime

combinando operadores (116)

Avanzamos con los paréntesis

Boehringer Ingelheim

Sección 12 Stored Procedures, Variables Subconsultas

Que es un stored procedure? (118)

un store procedure es un conjunto de instrucciones SQL almacenado en el interior de la base de datos para ser ejecutado a voluntad.

Los stored procedure tienen que tener un objetivo tienen que estar diseñados para un objetivo concreto.

Es recomendable tener una nomenclatura definida para la creación de los procedimientos

```
CREATE [OR ALTER] {PROC|PROCEDURE} <nombre>
   (parametros)
   AS
 [BEGIN]
     Sentencias;
 [END]
 GO
```

Si el SP no tiene parámetros no hay que poner paréntesis

```
CREATE PROCEDURE [dbo].[S_paciente]
   (@codigo [dbo].[paciente] = 1)
AS
BEGIN
  SELECT * FROM [ejemplo].[dbo].[Paciente] WHERE idpaciente = @codigo;
END
```

ANSI_NULLS especifica si los operadores de comparación = e <> siguen la norma ISO cuando encuentra valores NULL (ON) o si no siguen la norma ISO (OFF)

ISO trata el NULL como desconocido

WHERE Nombre = NULL;
WHERE Nombre <> NULL; } nunca devuelve nada con el ON ya que desconocido no es ni TRUE ni FALSE

QUOTED_IDENTIFIER (ON|OFF) para seguir las normas ISO referentes a entrecomillar identificadores y literales

Boehringer Ingelheim

que es una variable? (119)

las variables se declaran en el cuerpo de un lote o un procedimiento. Su ámbito de visibilidad y vida es lo que dura el procedure

DECLARE @<local_var_name> [AS] <data_type> [= value]

se les asigna un valor con SET

SET @<local_var_name> = <value>

las variables declaradas no inicializadas tienen valor NULL

ISNULL(expresión, valor_de_substitución)

Del tipo adecuado

Ejemplos declaraciones (120)

DECLARE @entero INT
SET @ entero = NULL
SELECT ISNULL(@entero, 0) AS Valor

DECIMAL(10,2) DATETIME

0.00 GETDATE()

Programando subconsultas y uso de alias de tablas (121)

Una subconsulta es una sentencia select dentro de una sentencia select.

Tenemos que asegurar que la subconsulta devuelva un único valor.

Usaremos los alias de tabla para poder diferenciar columnas de igual nombre en diferentes tablas.

SELECT apellido, nombre, idPais, observación,
(SELECT ps.pais from pais AS ps WHERE ps.idpais = pa.idpais) AS nombrepais
FROM paciente AS pa WHERE idpaciente = @idpaciente

El alias de columna es el nombre de la subconsulta

Boehringer Ingelheim

Sección 13 Estructuras de control

Introducción (122) IF | EXISTS | WHILE | RETURN | BREAK | TRY CATCH

El condicional IF ELSE y Estructura BEGIN END (123)

```
IF @idpaciente =1
BEGIN select * from paciente WHERE idpaciente=@idpaciente
      print @idpaciente.
END
ELSE
BEGIN
      select
      print 'que pasó'
END
```

La función EXISTS (124)

La función nos permite determinar si existen registros como resultado de una consulta

EXISTS (Subquery)

El Subquery es un SELECT restringido. Sin INTO

```
SELECT a.name, a.surname FROM Paciente as a
WHERE EXISTS( SELECT *
              From empleados as b
              where b.algo = a.algo;)
```

El bucle WHILE (125)

```
WHILE <condición>
BEGIN
END <cambiar condición>
```

Si el WHILE entra en bucle infinito

Reinicializar Servidor

El condicional CASE (126)

```
CASE [input_expresion]
  WHEN when_expresion THEN Result 1
  ⋮                        ⋮      ⋮
  ELSE resultexpresion n
END
```

Se puede usar como una expresión válida

Boehringer Ingelheim

Las instrucciones RETURN y BREAK (127)

BREAK sale de un bucle while
CONTINUE reanuda un bucle while
RETURN [n] sale incondicionalmente de una query o procedimiento

La Estructura TRY CATCH (128)

Nos permite manejar errores en tiempo de ejecución. Errores no controlados.

BEGIN TRY
END TRY
BEGIN CATCH
END CATCH

Pueden anidarse
Solo captura errores de severidad > 10 que no cierran la conexión con la BD.
Con funciones ERROR_xxx() recuperamos información del error.

Sección 14 Operadores aritméticos y de comp (

+ − * / %
+= −= *= /= %=
+ también con texto (129)

= > < >= <= <> !< !>
!= (130)

Sección 15 Practica 2: Stored procedures insert (131)

inserción paciente (132 y 133)

Sin generar registros duplicados.
Verificamos que no exista el dni del cliente en la tabla.

Los valores por defecto de los argumentos de la SP deben ser una constante. No puedo usar GETDATE() como default value. Tengo que usar GETDATE() en el cuerpo del SP para hacer la fecha de alta del paciente

```
CREATE/ALTER PROCEDURE ALTA_paciente (
  @dni VARCHAR(20),
  :
  @fechaAlta varchar(14)=NULL
)
AS
IF NOT EXISTS (SELECT TOP 1 nombre FROM paciente WHERE dni=@dni)
  INSERT INTO paciente (...) VALUES (..., GETDATE())
ELSE
  PRINT 'El paciente '+@dni+' ya existe.'
RETURN
```

Todos los campos de la tabla

+velocidad

Inserción Turnos (134 y 135)

1º Hay que validar que el turno Existe ya que es una FK de la tabla turnoPaciente

El idturno de la tabla turno es identity. No lo tengo que insertar. Verifico la existencia del turno por fechaturno

Si para la fecha turno ∄ turno, lo creo y el idturno generado por la identity lo usaré en la inserción del turno paciente.

El turno generado por la restricción identity de la tabla turno lo obtengo en la variable del sistema @@IDENTITY

Boehringer Ingelheim

CREATE/ALTER PROCEDURE ALTA_turnoPaciente(
@fechaturno VARCHAR(14),
@idpaciente paciente
@idmedico medico
@observacion observacion="")
IF NOT EXIST (SELECT TOP 1 idturno FROM turno WHERE fechaturno=@fe
BEGIN
INSERT INTO turno (fechaturno estado, observacion)
VALUES (@fechaturno @idpaciente @observacion)

...ement Studio\Solucion_1\Proyecto_1\SP_ALTA_MEDICO_2.sql 1

```sql
ALTER PROCEDURE [dbo].[ALTA_Medico_2](
    @nombre VARCHAR(50),
    @apellido VARCHAR(20),
    @idespecialidad int,
    @descripcion VARCHAR(50)
    )
AS

SET ANSI_NULLS OFF
DECLARE @idMedico medico;
SELECT @idMedico = (SELECT TOP 1 idMedico FROM [ejemplo].[dbo].[Medico] WHERE [dbo].[Medico].[nombre] = @nombre AND [dbo].[Medico].[apellido] = @apellido);

IF @idMedico = NULL
BEGIN
    INSERT INTO [ejemplo].[dbo].[Medico]
    (nombre, apellido)
    VALUES(@nombre, @apellido);
    SET @idMedico = @@IDENTITY;
    PRINT 'EL medico' + SPACE(1) + @nombre + SPACE(1) + @apellido + SPACE(1) + 'Ha sido creado con id ' + CAST(ISNULL(@idMedico, -1) as NVARCHAR(5)) +'.';
END
ELSE
BEGIN
    PRINT 'El medico' + SPACE(1) + @nombre + SPACE(1) + @apellido + SPACE(1) + 'Ya Existe. Tiene el id ' + CAST(ISNULL(@idMedico, -1)as NVARCHAR(5)) + '.';
END

IF EXISTS(SELECT TOP 1 especialidad FROM [ejemplo].[dbo].[Especialidad] WHERE idEspecialidad = @idespecialidad)
BEGIN
    IF NOT EXISTS(SELECT TOP 1 [descripcion] from [ejemplo].[dbo].[MedicoEspecialidad] WHERE [idMedico] = @idMedico AND [idEspecialidad] = @idespecialidad)
    BEGIN
        INSERT INTO [ejemplo].[dbo].[MedicoEspecialidad]
        (idMedico, idEspecialidad, descripcion)
        VALUES (@idMedico, @idespecialidad, @descripcion)
        PRINT 'He asignado al médico' + SPACE(1) + @nombre + SPACE(1) + @apellido + SPACE(1) + ' la especialidad ' + CAST(@idespecialidad as NVARCHAR(5))+'.'
    END
    ELSE
        PRINT 'El médico' + SPACE(1) + @nombre + SPACE(1) + @apellido + SPACE(1) + 'Pertenece la especialidad ' + CAST(@idespecialidad as NVARCHAR(5))+', no hago nada.'
END
ELSE
    PRINT 'Especialidad inexistente.'
RETURN
GO
```

```
CREATE/ALTER PROCEDURE ALTA_turnoPaciente(
  @fechaturno VARCHAR(14),
  @idpaciente ~~VAR~~ paciente
  @ idmedico medico
  @observacion observacion='')
  IF NOT EXIST (SELECT TOP 1 idTurno FROM turno WHERE fechaturno=@fe
  BEGIN
    INSERT INTO turno (fechaturno estado, observacion)
    VALUES (@fechaturno, 0, observacion)

    DECLARE @idturno turno
    SET @idturno = @@identity
    INSERT INTO turnopaciente (idturno, idPaciente, idMedico)
    VALUES (@idturno, @idpaciente, @idmedico)
  END
  ELSE
    PRINT 'El turno' + CONVERT(VARCHAR(4), @idturno + 'Existe'
```

Inserción Médico (136)

Medico		Medico Especialidad	Especialidad
id	identity	idMedico	idEspecialidad Idem
nombre		idEspecialidad	Especialidad
apellido		descripcion	

1. no debo duplicar el médico si no existe, inserto el medico, si ∃ no lo inserto.
2. verifico si Existe la Especialidad, si no existe no añado registro en la tabla Médico Especialidad.
3. Si Existe la especialidad, debo verificar que no exista ya en la relación muchos a muchos. la combinación idMedico, idEspecialidad

Sección 16 Herramientas adicionales

Proceso SP_HELPTEXT (138)

SP_help [[@objname =] 'name'] da información sobre un objeto, un tipo de dato ALT+F1

Si no indicamos objeto da información de todos.

SP_helptext [@objname =] 'name'

muestra información de SP or funciones

CTRL+T lo mostrará en modo texto
CTRL+A selecciona todo el texto
CTRL+C lo copia
CTRL+N crea un nuevo query
CTRL+V lo pega para editarlo

Agregando atajos de teclado (139)

Menu tools/options Environment/Keyboard/query shortcuts

Elegimos una libre y ponemos el nombre de la SP.

Esto es efectivo en los nuevos ventanas de query

Sección 17 Sentencias DDL (Data Definition Language)

Sentencias ALTER TABLE, ALTER COLUMN, DROP COLUMN (140)

Alter modifica una estructura de una tabla. Uno de cada vez

ALTER TABLE tabla { ADD COLUMN
ALTER COLUMN
ADD CONSTRAIN
DROP COLUMN
DROP CONSTRAIN

Cómo generar un script de tabla CREATE TABLE (141) y (142)

OE-DB-Ejemplo-tables-Paciente-RCLIC-script table as CREATE TO - New query editor window

y ablandremos la DDL de creación de la tabla

Boehringer Ingelheim

PAD_INDEX = {ON|OFF} activa el relleno de las páginas de índices para futuro uso, según el parámetro fillfactor. En el b-tree del índice se dejan libres para futuro uso algunos nodos. Es una opción a considerar cuando esperamos muchas altas bajas modificaciones en una tabla.

STATISTICS_NORECOMPUTE = {ON|OFF}

con ON evitamos durante la actualización de un índice se recalculen estadísticas. Eso puede ser interesante contrastado con tablas muy grandes y horas pico de trabajo.

IGNORE_DUP_KEY = {ON|OFF}

Afecta solamente a operaciones insert una vez el índice ha sido creado.

ON nos permitiría duplicar un email en una columna, siempre que las condiciones de unicidad no se violen.

ALLOW_ROW_LOCKS
ALLOW_PAGE_LOCKS = {ON|OFF}

impacta en el desempeño del servidor cuando tenemos muchos usuarios accediendo a la misma tabla.

ALTER TABLE ADD FOREIGN KEY (143)

Definimos las FK con transact SQL

```
ALTER TABLE Paciente
  ADD FOREIGN KEY (idpais) REFERENCES Pais(idpais);
ALTER TABLE Paciente
  ADD FOREIGN KEY (idpaciente) REFERENCES HistoriaPaciente (idPaci
```

Me da error p.q. la referencia hacia una PK (la de HistoriaPaciente) compuesta de varias columnas no es posible tenemos que invertir la sentencia

```
ALTER TABLE HistoriaPaciente
  ADD FOREIGN KEY (idpaciente) REFERENCES Paciente (idPaciente)
```

Boehringer Ingelheim

Crear y designar tipo de dato Especialidad (144)

OE-DB-Ejemplo - Programmability - types - User-defined data types
Rclick - New user-defined data type

La sentencia CREATE FUNCTION (145) UDF USER DEFINED FUNCTION

CREATE FUNCTION funcname (parameters list)
RETURNS <datatype> @var int
int
AS
BEGIN
RETURN
END

La sentencia DROP TABLE (146)

DROP permite eliminar una tabla o una basededatos completa, cuidado!!!
no podemos eliminar una tabla si esta referenciada desde una FK.

La sentencia TRUNCATE (147)

TRUNCATE table nametable, borra los registros de una tabla como lo hacia DELETE sin la clausula WHERE
no funciona si la tabla esta referenciada por un FK
reinicia los campos IDENTITY

Seccion 18: Funciones de conversion texto

LEFT y RIGHT (148) (149)
select LEFT(Nombre, 1)+LEFT(Apellido, 1) from Paciente;

LEN (150) (151)

LOWER y UPPER (152) upper(left(@var, 1))+lower(right(var, len()-1))

REPLACE (153) (var, encontrar, sustituir)

REPLICATE (154) (string, numeroveces) 8000 max char

Boehringer Ingelheim

LTRIM RTRIM (155)

CONCAT (156) Dado mas valores

GETDATE() y GETUTCDATE() (157)

para grabar fecha y hora del momento en que se insertan registros en una tabla

14:17:52.547 ESPAÑA
MI SERVIDOR ESTA

12:17:52.547 UTC

DATEADD (datepart { year, quarter, month, day, week, day of year }, number, date) (158)

DATEDIFF (datepart, start date, end date) (159)

DATEPART (datepart { dw, dd }, date) (160)

ISDATE (Expresion) (161)

CAST y CONVERT (162)

cast convierte tipos de datos entre si, siempre que sean de la misma familia

CAST (@numero as INT) no cambia el tipo de la variable, lo altera a modo de visualizacion

CONVERT(data_type [(length)], expression [, style])

CAST (Expresion AS data_type [(length)])

Hay q ver la Ref guide

Boehringer Ingelheim

Sección 19 transacciones

La Instrucción BEGIN TRAN (163)(164)(163)

Permitirá ejecutar una instrucción y si se verifican las comprobaciones se podrá hacer un COMMIT o bien un ROLLBACK.

Mientras que estamos actualizando o borrando datos sobre tablas, también con insert, estamos en medio de una transacción. La transacción bloquea las tablas que usa hasta que ejecuta el COMMIT o el ROLLBACK.

Si no se ejecuta el commit o el Rollback nadie puede usar la tabla.

La variable @@ROWCOUNT indica cuántos registros han sido afectados por la transacción anterior.

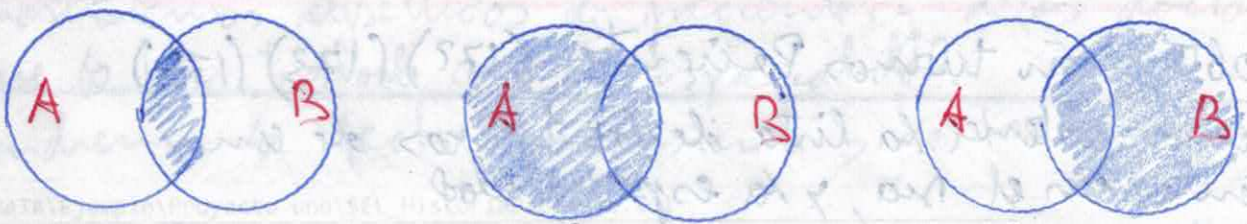
Controlando @@ROWCOUNT podemos verificar si la ejecución ha sido la esperada o no y entonces ejecuta el commit.

```
BEGIN TRANSACTION
  UPDATE paciente SET telefono=444 WHERE idpaciente=8
  IF@@ROWCOUNT = 1
     COMMIT TRAN
  ELSE
     ROLLBACK TRAN
```

```
BEGIN TRAN
  DELETE FROM Turno WHERE estado=1
  IF@@ROWCOUNT = 1
     COMMIT TRAN
  ELSE
     ROLLBACK TRAN
```

Boehringer Ingelheim

Seccion 20 JOINS y UNIONS entre tablas



INNER JOIN — LEFT JOIN — RIGHT JOIN

```
SELECT * FROM Paciente AS P
                                   TurnoPacien
ON P.idpaciente = T.idpaciente;
```

mal orden

- Asegurarse que la cláusula ON se refiere a PK si es posible. La respuesta será más rapida
- tambien hace mas veloz el orden de los campos que hay en la clausula ON. Se recomienda que el primer campo de la clausula ON sea el de la tabla inmediata con la que estoy haciendo la conexion.
  ON T.idpaciente = P.idpaciente orden q da + velocidad
- usar alias de tablas para evitar ambigüedades de campos

La clausula UNION y UNION ALL (171)

Une los resultados de varias consultas en una tabla. Los estructuras de las diferentes consultas deben ser igual. mientras que sea igual las une.

```
SELECT idtiene FROM turno UNION SELECT idpaciente FROM paciente
```

une los codigos siendo un sin sentido

UNION ⟶ quita los duplicados
UNION ALL ⟶ NO quita los duplicados

Boehringer Ingelheim

Sección 21: Práctica 3 Programando SP de SELECT UPDATE DELETE

SP obtención turnos Paciente (172) (173) (174)

queremos obtener la lista de los turnos de un paciente con el día, y la especialidad

1.- Estudiar las relaciones entre las tablas que contienen las informaciones buscadas

2.- Planificar el orden de los inner joins de forma que las cláusulas ON tengan acceso a los campos implicados

3.- En las cláusulas ON la primera columna pertenece a la última tabla referenciada y la segunda columna a tablas referenciadas con anterioridad.

4.- Los parámetros del SP incluidos en la cláusula WHERE

...rso\SQLDATA\Ejemplo\Proyecto uno\SEL_TurnosPaciente.sql

```sql
CREATE PROCEDURE SEL_TurnosPaciente (
    @idpaciente paciente


AS
set nocount on

SELECT
    P.apellido + ',' + P.nombre AS Paciente,
    CONVERT(date,T.fechaTurno,101) AS Dia,
    E.especialidad AS Especialidad,
    'Dr@. ' + M.apellido AS Doctor
FROM Paciente AS P
INNER JOIN TurnoPaciente AS TP
ON TP.idPaciente = P.idPaciente
INNER JOIN Turno AS T
ON T.idTurno = TP.idTurno
INNER JOIN MedicoEspecialidad AS ME
ON ME.idMedico = TP.idMedico
INNER JOIN Medico AS M
ON M.idMedico = TP.idMedico
INNER JOIN Especialidad AS E
ON E.idEspecialidad = ME.idEspecialidad
WHERE P.idPaciente = @idpaciente;
```

	Paciente	Dia	Especialidad	Doctor
1	Ramírez,Jorge	2023-03-15	Cardiología	Dr@. Castelo
2	Ramírez,Jorge	2023-03-15	Medicina interna	Dr@. Casado
3	Ramírez,Jorge	2023-03-15	Traumatología	Dr@. Blasco
4	Ramírez,Jorge	2023-03-15	Cardiología	Dr@. Castelo
5	Ramírez,Jorge	2023-03-15	Medicina interna	Dr@. Casado
6	Ramírez,Jorge	2023-03-15	Traumatología	Dr@. Blasco
7	Ramírez,Jorge	2019-01-22	Cardiología	Dr@. Castelo
8	Ramírez,Jorge	2019-01-22	Medicina interna	Dr@. Casado
9	Ramírez,Jorge	2019-01-22	Traumatología	Dr@. Blasco

Boehringer Ingelheim

SP Obtener la Historia clínica de un Paciente (175)

Mostraremos apellidos de paciente y delas doctores que lo han tratado con su especialidad.
Pondremos la fecha nacimiento y de tratamiento

...o\SQLDATA\Ejemplo\Proyecto uno\SEL_HistoriaPaciente.sql 1

```sql
ALTER PROCEDURE SEL_HistoriaPaciente (
    @idpaciente paciente
)
AS

/*
Muestra la historia clinica de un paciente con sus doctores y especialidades
*/

BEGIN
set nocount on

select
    P.nombre, P.apellido, p.fNacimiento,    DATEDIFF(year,
      P.fNacimiento,GETDATE()) AS 'Años edad',
    h.fechaHistoria, h.observacion,
    M.apellido AS Doctor,
    E.especialidad,
    ME.descripcion
FROM Paciente AS P
INNER JOIN HistoriaPaciente AS HP
ON HP.idPaciente = P.idPaciente
INNER JOIN Historia AS H
ON H.idHistoria = HP.idHistoria
INNER JOIN Medico AS M
ON M.idMedico = HP.idMedico
INNER JOIN MedicoEspecialidad AS ME
ON ME.idMedico = M.idMedico
INNER JOIN Especialidad AS E
ON E.idEspecialidad = ME.idEspecialidad
WHERE P.idPaciente = @idpaciente;

END
```

nombre	apellido	fNacimiento	Años edad	fechaHistoria	observacion	Doctor	especialidad	descripcion
Jorge	Ramirez	2019-01-18	5	2020-10-10 20:30:15.0...	Observacion...	Castelo	Cardiología	sin observacio
Jorge	Ramirez	2019-01-18	5	2020-02-11 20:32:15.0...	Observacion...	Casado	Medicina inter...	Sin Observacion
Jorge	Ramirez	2019-01-18	5	2020-03-11 20:33:15.0...	Observacion...	Blasco	Traumatología	Sin Observacion
Jorge	Ramirez	2019-01-18	5	2020-04-11 20:34:15.0...	Observacion...	Batalla	Cardiología	Una obaservaci...
Jorge	Ramirez	2019-01-18	5	2020-05-11 20:35:15.0...	Observacion...	Batalla	Cardiología	Una obaservaci...
Jorge	Ramirez	2019-01-18	5	2020-06-11 20:36:15.0...	Observacion...	Batalla	Cardiología	Una obaservaci...

SP Obtener Especialidades (176)

```sql
CREATE PROCEDURE SEL_Especialidades
AS
IF EXISTS(SELECT * FROM Especialidad)
  SELECT E.especialidad FROM Especialidad AS E;
ELSE
  SELECT Ø AS Resultado;
```

Sin paréntesis
∄ Argumentos

Boehringer Ingelheim

SP Actualizar turno (177) (178)

Solamente, en este ejercicio, actualizamos estado y observación de un turno.
Sin un control de que hay una fecha disponible para el especialista no deberíamos actualizar el campo fecha para evitar colisiones.

(cuidado NO OLVIDAR WHERE en el update. Si lo olvidamos actualiza toda la columna)

```
CREATE PROC UPD_turno (
  @idturno turno
  @estado
  @observación observacion)
AS
IF EXISTS (SELECT * FROM Turno WHERE IdTurno=@IdTurno)
  UPDATE Turno SET
     estado = @estado,
   observación = @observacion
  WHERE idturno = @idturno;
ELSE
  SELECT 0 AS Resultado;
```

IF EXISTS (SELECT * FROM TurnoEstado WHERE IdEstado=@estado)

SP Actualizar un paciente (179)

```
CREATE PROCEDURE (
  todos los parametros
  )
AS
IF EXISTS (SELECT * FROM Paciente WHERE idpaciente = @idPaciente)
AND EXISTS (SELECT * FROM Pais WHERE idPais = @nac_pai)
  UPDATE Paciente SET
    =
    =
  WHERE idpaciente = @idpaciente
ELSE
  SELECT 'paciente '+cast(idpaciente as VARCHAR(2)+
      'o pais '+CAST(@nac_pai AS VARCHAR(3)+
      'no existe' AS RESULTADO
```

Boehringer Ingelheim

SP Eliminar un turno (180)

idturno es FK en dos tablas Pago Paciente y turnoPaciente
Para borrar un registro de turno, primero debo eliminar los registros en TurnoPaciente que en su FK hacen referencia al turno que quiero Borrar.

```
CREATE PROCEDURE DEL_TURNO (@idturno turno)
AS
IF EXISTS (SELECT * FROM turno WHERE idturno = @idturno)
BEGIN
    DELETE FROM TURNOPACIENTE WHERE idturno = @idturno;
    DELETE FROM Pago Paciente WHERE idturno = @idturno;
    DELETE FROM turno WHERE idturno = @turno;
END
ELSE
    SELECT 0 AS RESULT;
```

Seccion 22 Funciones definidas por el usuario (UDF)

Funciones de tipo Escalares (181) (182)

Reciben varios parametros, devuelven UN valor

```
CREATE FUNCTION [schema_name] function_name
( parameter declaration
)
RETURNS return_data_type
[AS]
BEGIN
    function body
    RETURN scalar_expression
END
```

```
CREATE FUNCTION MYCONC(
@nom VARCHAR(50)
@ape VARCHAR(50)
RETURNS VARCHAR(102)
BEGIN
    DECLARE @r VARCHAR(107);
    SET @r = @ape+', '+@nom;
    RETURN @r;
END
```

```
CREATE FUNCTION dbo.pais(
  @idpaciente Paciente)
RETURNS VARCHAR(50)
BEGIN
  DECLARE @pais VARCHAR(50)
  SET @pais = (SELECT PA.Pais FROM
  Paciente AS P INNER JOIN Pais AS PA
  ON PA.idpais = P.idpais
  WHERE P.idpaciente = @idpaciente
  RETURN @pais
END
```

Boehringer Ingelheim

Funciones tipo tabla (183)

Devuelve una tabla, un conjunto de registros.

```
CREATE FUNCTION name()
RETURNS @paises TABLE (idpais CHAR(3)
                       Pais VARCHAR(30))
BEGIN
  INSER INTO @paises SELECT * FROM Pais;
  RETURN
END
```

SECTION 23 tablas temporales { memoria / Física (184)

Definición y uso de tabla temporal en memoria (185)

Las tablas temporales mejoran la performance.
La tabla temporal en memoria dura mientras dura el script en ejecución.

```
DECLARE @mitabla TABLE (columns definition)
  insert into @mitabla VALUES (      );
SELECT * FROM @mitabla;
```

Definición y uso de tabla temporal Física (186)

Permanece en el servidor hasta que se reinicie el servicio de base de datos.
Se diferencia con # una tabla física de la tabla en memoria.

```
CREATE TABLE #mitabla (columns definition)
  insert into #mitabla VALUES (      );
```

Boehringer Ingelheim

Implementación de una tabla temporal (187) (188)

Buscamos los turnos que tiene un determinado paciente. Sin tablas temporales los INNER JOIN tienen que tratar con todos los registros de las tablas. Si las tablas son grandes las consultas demoran.
usamos las tablas temporales para poner en ellas el subconjunto de filas que nos interesan de una tabla que vamos a joinear

```
DECLARE @turnos TABLE (id INT IDENTITY(1,1)
                       idturno Turno
                       idPaciente Paciente);

INSERT INTO @Turnos (idturno, idPaciente)
SELECT TP.idturno, TP.idPaciente FROM turnoPaciente AS TP
WHERE TP.idPaciente = 1;

SELECT * FROM PACIENTE AS P
INNER JOIN @turnos AST
ON T.idPaciente = P.idPaciente;
```

Tecnica para recorrer una tabla temporal (188)

Suponemos que la tabla temporal @turnos contiene todos los registros de turnoPaciente. Vamos a recorrer la tabla temporal y borrar los registros de los pacientes no deseados

```
DECLARE @i INT = 1
DECLARE @numreg INT = (SELECT COUNT(idturno) FROM @turnos
DECLARE @nointeresa INT = 1;

WHILE @i <= @num_reg
BEGIN
    IF (SELECT idPaciente FROM @turno WHERE id=@i) <> @nointeresa
        DELETE FROM @turnos WHERE id=@i;
    SET @i = @i + 1;
END
```

Boehringer Ingelheim

Sección 24 Vistas (189, 190, 191)

una vista es una consulta almacenada en VIEWS.
Se actualiza cada vez que la ejecutamos.
una vez creada, la trataremos como si fuera una tabla.
La podemos crear con TRANSACT SQL o desde el explorador de objetos
OE-DB-centromedico → VIEWS → R. click → New View.

```
CREATE VIEW view_name
AS SELECT... statement.
```

La podemos usar como interfaz de compatibilidad con una tabla cuyo esquema ha cambiado.
también es una medida de seguridad para no dar acceso a los usuarios a las tablas subyacentes a la vista
Rescato de este video el uso del isnull en la cláusula
WHERE ISNULL (estado, 0) = 0

Sección 25 Triggers

¿Que es un trigger? tipos (192, 193)

un trigger es un tipo especial de SP que se ejecuta automaticamente cuando hay un evento en el servidor.
Hay DML triggers q se activan con { INSERT, UPDATE, DELETE
Hay DDL triggers q se activan con CREATE, ALTER, DROP
Hay LOGON triggers q se activan con logins de usuarios
Usamos los triggers para monitorear acciones que analizaremos a posteriori.

Boehringer Ingelheim

Creación de trigger tipo INSERT (194)

Creamos un trigger que se activará cada vez que se inserte un paciente mejicano. La acción del trigger es insertar un registro de log en una tabla Pacientes_log

Los triggers usan dos tablas especiales, temporales, residentes en memoria: "inserted" "deleted".

Estas tablas especiales copian los registros afectados en la tabla modificada.

deleted los recibe antes de borrarlos o actualizarlos

inserted los recibe después de ser insertados o modificados

Esto nos permite comparar los valores originales con los modificados

```
CREATE TRIGGER PacienteMejicanos ON Paciente
                                    Triggered table
AFTER INSERT
AS
  IF (SELECT idPais FROM inserted) = 'MEX'
    INSERT INTO Pacientes_log (idPaciente, idPais, fechaAlta)
      SELECT i.idPaciente, i.idPais, GETDATE() FROM inserted AS i
```

Creación de trigger tipo UPDATE (196)

Una transacción de actualización es similar a una de borrado seguida de una de inserción.

1. La fila original se copia a deleted table
2. La tabla se actualiza con los nuevos valores
3. La fila modificada se copia a inserted table.

```
CREATE TRIGGER PacienteModificado ON Paciente
                                     Triggered table
AFTER UPDATE
AS
BEGIN
DECLARE @paciente paciente = (SELECT idPaciente FROM inserted, deleted;
IF EXISTS(SELECT * FROM PACIENTE_log WHERE idPaciente = @paciente)
  UPDATE Paciente_log SET fechaMod = GETDATE() FROM (inserted, deleted) AS I, D
ELSE
  INSERT INTO Pacientes_log (idpaciente, idPais, fechaMód)
    SELECT I.idpaciente, I.idPais, GETDATE() FROM (inserted AS I
```

Boehringer Ingelheim

Crear un trigger de tipo DELETE (196) (197)

1. Agregar campo fechaBaj en la tabla Pacientes_log
2. Insertar un nuevo paciente que luego borraremos para probar el trigger
3. Crear el trigger que anote la fecha de baja de un paciente

ALTER TABLE Pacientes_log ADD fechaBaja datetime;

INSERT INTO Paciente (- - - -)
Values ('999', 'Mario', 'Rodriguez', '20001116', 'Gaumlica 12', 'BRL', '607727376', 'pepe@hotmail', 'Nada que observar');
→ Reg 13

```
...\SQLDATA\Ejemplo\Proyecto uno\TRI_PacientesBorrados.sql
ALTER TRIGGER PacienteBorrado ON Paciente
AFTER DELETE
AS
BEGIN
DECLARE @paciente paciente =(SELECT idPaciente FROM deleted);

IF EXISTS(SELECT * FROM Pacientes_log WHERE idPaciente = @Paciente)
    UPDATE [Pacientes_log] SET fechaBaja = GETDATE() WHERE idPaciente =
      @Paciente;
ELSE
    INSERT INTO [Pacientes.log] (idPaciente, idPais, FechaBaja)
        SELECT D.idPaciente, D.idPais, GETDATE() FROM deleted AS D;
END
```

Sección 26: Backup y Restore

Desde Management Studio (198) (199)

Hay Backup completo y hay Backup differencial
OE-DB-Centro Medico → R clic → Tasks → Backup →
En el diálogo es autoexplicativo. La ubicación por defecto C:\Program Files\MS SQL Server\MSSQL 16...\Backup...
Una vez elegida la ruta, se hace la copia de seguridad.

Errores Frecuentes en la restauración (200)

- Exclusive access could not be obtained because the database is in use
  La BD está en uso. Desconecta de la BD y del servidor y abre sesión nueva

Boehringer Ingelheim

• Base de datos en modo Restoring. Texto junto al nombre de la base de datos "(Restoring...)" no permite iniciar sesión.

En una nueva query (CTRL+N) ejecutar:

RESTORE DATABASE <dbname> WITH RECOVERY;

Pero esto debemos hacerlo en una nueva sesión en la que "entramos" conectados a la BD "master".

En connect properties / Connect to database indicamos "master"

¿Cómo restaurar un backup? (201)

OE-DB-RCLIC → RESTORE DATABASE...

En el dialogo @ Device y en ... Buscamos el backup media, que en nuestro caso es un file una vez elegido el "media" le hace un chequeo.

Habla de tail-log backup: va a capturar los registros log que aún no han sido llevados a una copia de seguridad para prevenir una perdida de trabajo en la restauración. De esta forma la cadena de logs quedará intacta

En Destination, escribimos el nombre de la BD que queremos restaurar. En el ejercicio hemos puesto uno diferente.

¿Cómo hear un Backup desde un script? (202) (203) (204)

Las copias de seguridad creadas por versiones recientes de SQL Server no pueden ser restauradas en versiones previas de SQL Server

Para saber el espacio que ocupa la base de datos:

```
USE <Database name>;
GO
EXEC sp_spaceused @oneresultset=1; => 16.00 MB
GO
```

En el dialogo del Backup desde el OE hay la posibilidad de obtener el Script SQL que va a ejecutarse al hacer OK.

Boehringer Ingelheim

BACKUP DATABASE
TO {DISK, TAPE, URL} = N'c:\ruta hasta\backupfilename.bak'
WITH <options>
NAME = N'name_of_backup_set_media'
NO_COMPRESSION 7668 kB | COMPRESSION 1572 kB
20'50%

...Build\SQL\curso\SQLDATA\Ejemplo\Proyecto uno\Backup.sql 1

```sql
USE [Ejemplo]
GO

DECLARE @fecha CHAR(13)
DECLARE @path VARCHAR(100);
DECLARE @name VARCHAR(20);
SET @fecha = convert(CHAR(8), GETDATE(), 112) +'_'+ replace(convert(CHAR(5),
    GETDATE(), 108),':','')
SET @path = N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL
    \Backup\Ejemplo_' + @fecha +'.bak'
SET @name = N'Ejemplo'+@fecha +'-Full Database Backup'


BACKUP DATABASE [Ejemplo]
TO  DISK = @path
WITH
NO_COMPRESSION,
NAME = @name,
NOFORMAT,
NOINIT,
NOREWIND,
NOUNLOAD,
SKIP,
STATS = 10
GO
```

Seccion 27 Scheduled Jobs

¿Que es un SQL Job? opciones (205)

Tareas programadas o scheduled jobs, estan administradas por el SQL Server Agent. Son procesos que ejecutan scripts de SQL en determinados momentos del dia.

OE- SQL Server Agent - Jobs - New Job

En el dialogo hay varias pestañas

General (nombre, descripcion, habilitacion, categoria)
steps (relacion de scripts a ejecutar)
schedules (frecuencias, fechas, repeticiones..)
notifications (e mail, Pager, log)

Boehringer Ingelheim

Preparando un Job para realizar una tarea (206)

La version SQL Server Express, gratuita, no tiene el SQL Server Agent. Lo haremos con Windows scheduler

El Agente SQL Server, es un servicio separado del propio SQL Server y puede que tenga ajustes de inicio como servicio diferentes del propio servidor SQL.

Se puede inicializar OE - SQL Server Agent - RCLIC -> Start

Vamos a preparar un JOB para hacer copias de seguridad. Usaremos el SQL de la Sección 26.

OE-SQL Server Agent - Jobs - RCLIC - New

GENERAL
- Name: Back_up_Ejemplo
- Owner: El q. tenga permiso para acceder a la BD.
- Category: Elegir una (o crearla) para ayudar en una futura clasificación, Busqueda de Jobs.
- Description: Una Explicación
- Enable: OK

NEW STEP:
- Name:
- Type: Integration Services | Powershell | ActiveX ...
- Runas: Especifica el contexto de seguridad en el que se va a ejecutar el Job.
- Command: Pegamos el Script de la Sección 26
- On success: Go to next | Quit report success | quit Report failure |
- On failure: " " "
- Retry attempts <num>: Retry interval <minutes>

NEW SCHEDULE
- Name: Turno mañana
- Schedule type: Recurring | si CPU IDLE | When Agent Starts
- Rellena los campos...

NOTIFICATION
- E-MAIL OK (1º hacer un Job Operator)

Boehringer Ingelheim

Tip SQL Server Express: Emula JOB scheduler (207..212)

1º Verificar con un terminal que tenemos instalado SQLCMD llamando al comando
#> SQLCMD -?

2º Descargar SQLCMD utility del portal de MS

3º Identificar la IP del servidor con ipconfig.

4º Guardar el script SQL que se desea ejecutar

5º Crear un archivo x.bat con este comando

SQLCMD -S .\<IP> -i <script.sql>

servidor — input file

de ruta si tiene espacios en blanco debe ir entre doble comilla "ruta"

6 Programar una "tarea básica" con el programador de tareas de Windows que se ejecute diariamente a una hora y que tome el archivo x.bat anterior y empiece en la carpeta adecuada.

Sección 28: Practica y Programando Funciones, Scheduled Jobs y Vistas (213)

Crear una función convierte fecha en letra (214) (215)

Buscamos mostrar un formato:

<diasemana>, <diames> de <nombre mes> de <año>

Boehringer Ingelheim

```sql
/*
Function FCN_fecha2txt() converts a datetime argument into
a text wiht this format
<weekdayname>, <monthday> de <monthname> de <yearnumber>
Considering spanish names for days, 'miercoles is the
longest wiht 9 chars.
The longest month name is 'Septiembre' wiht 10 chars
09 miercoles
02 ,_
02 monthday
04 _de_
10 septiembre
04 _de_
04 2024
35 Total

The funciotn returns a VARCHAR(35)
*/
ALTER FUNCTION FCN_fecha2txt(@f datetime)
RETURNS VARCHAR(35)
AS
BEGIN
DECLARE @result VARCHAR(35);
DECLARE @weekdayname VARCHAR(9);
DECLARE @monthname VARCHAR(10);
DECLARE @monthday INT = DATEPART(dd,@f);
DECLARE @weekday INT = DATEPART(dw, @f);
DECLARE @monthnum INT = DATEPART(mm ,@f);
DECLARE @year CHAR(4) = CONVERT(CHAR(4), DATEPART(year, @f));

SET @weekdayname = (CASE WHEN @weekday = 7 THEN 'Domingo'
                         WHEN @weekday = 1 THEN 'Lunes'
                         WHEN @weekday = 2 THEN 'Martes'
                         WHEN @weekday = 3 THEN 'Miércoles'
                         WHEN @weekday = 4 THEN 'Jueves'
                         WHEN @weekday = 5 THEN 'Viernes'
                         WHEN @weekday = 6 THEN 'Sábado'
                         END);
SET @monthname = (CASE
                         WHEN @monthnum = 1 THEN 'enero'
                         WHEN @monthnum = 2 THEN 'febrero'
                         WHEN @monthnum = 3 THEN 'marzo'
                         WHEN @monthnum = 4 THEN 'abril'
                         WHEN @monthnum = 5 THEN 'mayo'
                         WHEN @monthnum = 6 THEN 'junio'
                         WHEN @monthnum = 7 THEN 'julio'
                         WHEN @monthnum = 8 THEN 'agosto'
                         WHEN @monthnum = 9 THEN 'septiembre'
                         WHEN @monthnum = 10 THEN 'octubre'
                         WHEN @monthnum = 11 THEN 'noviembre'
                         WHEN @monthnum = 12 THEN 'diciembre'
END);
SET @result = @weekdayname +', '+ convert (VARCHAR(2), @monthday) + ' de ' +
    @monthname+ ' de ' + @year;
RETURN @result
END
```

Creación Job cancele turnos vencidos no atendidos (216 (217

Se trata de hacer un select en la tabla turnos actualizar los estado con valor = 0 (pendiente) y al valor 2 (cancelado) para

Boehringer Ingelheim

los turnos cuya fechaturno sea menor que la fecha actual.
comparamos fechas datetime como CHAR(8) en formato aaaammdd.
convertimos los datetime con el formato 112

```
update turno SET Estado = 2 WHERE Estado = 0
AND CONVERT(CHAR(8), fechaturno, 112) < CONVERT(CHAR(8), GETDATE(), 112);
```

Este script SQL, salvado en la solución, es el que abrimos con el SQL Server Agent al crear un nuevo JOB cuando tenemos que definir el command a ejecutar.

Creación de una vista (218, 219, 220)

una vista es una tabla virtual, es una consulta predefinida que se usa para recuperar datos de varias tablas.
la vista se consulta como si fuera una tabla real.
las vistas ayudan a restringir el acceso a datos confidenciales.
creamos una vista para mostrar los medicos y la especialidad que tienen.

```
CREATE VIEW MedicosEspecialidades
AS
SELECT M.apellido, M.nombre, E.especialidad, ME.descripcion
FROM Medico AS M
INNER JOIN MedicoEspecialidad AS ME
ON ME.IdMedico = M.IdMedico
INNER JOIN Especialidad AS E
ON E.IdEspecialidad = ME.IdEspecialidad;
```

Boehringer Ingelheim

Sección 29 Administración de usuarios
Roles, Schemas y Permisos

usuarios y logins (222)

Hay 2 niveles de seguridad. uno a nivel de instancia y otro a nivel de BD.

OE - SECURITY - Logins - RCLIC - New Login

OE - DB - Centromedico - SECURITY - users - RCLIC - New User

Antes de crear un usuario en una base de datos, debemos crearle un login para que pueda acceder a la instancia del servidor. En caso contrario habremos generado un usuario HUERFANO de su login.

NEW LOGIN (Page user mapping) users en cada BD (225) automatizará la creación de

login name: función del modo autenticación elegido

Windows / SQL Server — authentication { + segura ⇒ Kerberos / - segura

Default language para los mensajes de error

Asignamos una BD por defecto para ese login

NEW USER (En la BD definida default para el login)

GENERAL
- user type: coherente con el login { SQL user with login / " " without " / window user
- user name:
- login name: (un login, no acepta dos users) (la variable SESSION_US nos da el user name correspondiente al login name)
- schema

owned schemas

Membership: Roles a los q pertenece

securables: Permisos a otorgar al usuario en los diferentes objetos de la BD. Hay 24 ≠ tipos objetos. Cada tipo de objeto tiene diferentes tipos de permisos. tablas (11) stored Procedures (5), Functions scalar (6) UD Data types (6)

Podemos GRANT permisos o DENY permiso.

WITH GRANT el usuario puede otorgar ese permiso a otro usuario.

Boehringer Ingelheim

las ventanas de las queries nos informan entre paréntesis el login al que pertenecen. Esto es importante ante el hecho de que podemos estar conectados a una misma instancia del servidor con logins diferentes SIMULTANEAMENTE

Creando un Nuevo Rol Admin Turnos (223) (224)

Un Rol es como un grupo al que se apuntan los usuarios para que tengan permisos sobre los objetos de la Base de datos.

Hay 3 tipos de roles

- Server Roles OE-Security-Server Roles
  Son específicos para administrar la base de datos

SysAdmin: Cualquier miembro puede realizar cualquier acción en el servidor SQL.
ServerAdmin: Cualquier miembro puede establecer opciones de configuración en el servidor SQL.
SetupAdmin: Cualquier miembro puede administrar servidores remotos y opciones de inicio de SQL Server.
Security Admin: Cualquier miembro puede administrar la seguridad del servidor SQL.
ProcessAdmin: Cualquier miembro puede finalizar un proceso en ejecución en el servidor SQL.
DbCreator: Cualquier miembro puede crear, modificar, eliminar y restaurar Bases de Datos.
DiskAdmin: Cualquier miembro puede administrar archivos de datos del servidor SQL. (228)
BulkAdmin: Cualquier miembro puede ejecutar el comando BULK INSERT.

- Database Roles OE-DB-Centro Medico-Security-Roles-DB Roles

db_owner: Miembros tienen permisos totales.
db_accessadmin: Miembros pueden administrar grupos de Windows y Logins de SQL Server.
db_datareader: Miembros tienen permiso de lectura. solamente
db_datawriter: Miembros pueden agregar, eliminar, o modificar datos en las tablas.
db_ddladmin: Miembros pueden ejecutar librerías DLL.
db_securityadmin: Miembros pueden modificar la membresía de Roles y administrar permisos.
db_backupoperator: Miembros pueden realizar un backup de la BD. (228)
db_denydatareader: Miembros no pueden visualizar datos dentro de la BD.
db_denydatawriter: Miembros no pueden modificar o eliminar datos de tablas o vistas.

- Roles de usuario son los definidos por nosotros
  Se habla de un nuevo DATABASE ROLE

GENERAL
- Role name: Administrador
- Owner: o un usuario / o un Role : ∀ user ∈ Role puede modificarlo
  usamos dbo q es el usuario default.

SECURABLES: Elegimos la tabla / Turno GRANT ALL / Paciente DENY ALL

Creado el Rol, se lo asigno al usuario en la página 'MEMBERSHIP'

Boehringer Ingelheim

Crear y asignar un Rol a un usuario (225)

CREATE ROLE pagos AUTHORIZATION dbo;
<RoleName> <OWNER>

ALTER ROLE pagos ADD MEMBER Emilio;

Que es un schema? (226) (227)

Es un espacio de nombres que agrupa objetos.
dbo es el esquema default que se crea al crear la base de datos dbo = "database owner"
hay un usuario default de la base de datos: dbo
se crea automaticamente al crear el login sa "sistem administrator"
en la instancia del servidor sa -> dbo
cada BD en la instancia tiene un schema dbo cuyo usuario propietario es a su vez dbo

login	user	schema	role
sa	dbo	dbo	db_owner

usaremos los schemas para aislar tablas de grupos de usuarios.
creamos el esquema en:
OE - DB - Centro Medico - Security - schemas - RCLIC - New schema
Schema name:
Schema owner:

Asignamos el schema al usuario de la BD en la page owned schema de las propiedades del usuario
Apartir de este momento el usuario, cuando se conecta, solo ve las tablas de su schema.

ALTER SCHEMA Turnos TRANSFER Turnos;
" " " " TurnoEntrad;
" " " " TurnoPaciente;
Es la forma de asignar una tabla a un schema.

Boehringer Ingelheim

Sección 30: Importación de registros mediante Excel (229 .. 233)

Nos ayudamos de Excel para crear una cláusula INSERT INTO <table> (values), entrecomillando los valores.

Copiamos y pegamos los resultados de Excel en una nueva query.

Sección 31: SQL Server Profiler

Conociendo la herramienta (234, 235)

Menu tools - SQL Server Profiler

una traza es un monitoreo de todos los procesos que se ejecutan en la instancia del servidor.

Se puede guardar en un fichero externo o en una tabla con el nombre de la traza en el propio servidor que estamos monitoreando

La traza tiene varios columnas de las que destacamos

CPU: msec used by statment of CPU ONLY

Duración: msec taken by statement to complet

READS: } # of pages { Read I/O
WRITES: } # of pages { Write I/O } caused by statement

Como monitorear procesos por tiempo ejecución (236)

Se trata de poner un filtro en la traza sobre el campo duración para que superado un valor, el proceso sea mostrado por el SQL Server Profiler.

En una traza podemos seleccionar los eventos en la pestaña "Event Selection".

Hay eventos preseleccionados por default y para cada tipo de evento seleccionamos las columnas relevantes.

Tenemos 22 categorías de eventos para 118 Eventos predefinidos.

Boehringer Ingelheim

Así el evento SP_complete de la categoría de eventos Stored Procedures captura los SP que se hayan ejecutado completamente.
Para el evento a monitorizar elegimos las columnas relevantes de entre los 65 posibles.
No todos los eventos tienen todas las columnas disponibles.
Podemos poner filtros a las columnas. Para ello editaremos la traza que previamente hemos detenido
Creamos un filtro para detectar eventos 'SP_complete' cuya 'duration' sea 'Greater than or Equal' a 1000 milisegundos.
Podríamos filtrar los eventos por el contenido del campo textData con un filtro del tipo
like %palabra_clave%

Definiendo templates para el monitoreo (737)

En el menú de SQL Server Profiler file/templates/new nos muestra un diálogo
Server type: nuestro tipo de instancia MSSQLServer 2022
name : <name> esto aparecerá en el dropbox "use the template" al crear una nueva traza

En el Event selection ~~se~~ seleccionamos los eventos y las columnas que deseamos figuren en el template.

Herramientas del menu. Import/Export trazas (73

Las trazas se pueden ser salvadas a ficheros (*.trc) que sólo reconoce el SQL Server Profiler.
También a una tabla en el propio server.
Esto puede definirse en la creación de la traza o al finalizar la traza.

Boehringer Ingelheim

SQL Server Profiler nos ofrece herramientas de búsqueda de valores en las trazas.
También el marcado con bookmarks de determinados eventos CTRL2+F2 para luego navegar entre los eventos marcados con F2
Hay una opción Auto scroll en los iconos de la barra superior que nos visualiza la última línea generada en la traza.

Sección 32: Buenas prácticas: (239)

1.- Definir nombres de campos cortos pero significativos.
2.- Intentar que FK y PK tengan el mismo nombre.
3.- Elegir los tipos de datos más adecuados a los valores a representar
4. Elegir sabiamente los nombres de los SP para que su nombre nos ayude a saber lo que hace.
5. Normalizar las tablas al menos hasta la tercera forma normal. modelo Snowflake
6.- Pensar en seguridad de la BD. No todos los usuarios deben poder acceder a todo. Trabajar bien usuarios, roles y permisos.
7. En las consultas preferiblemente filtrar por las PK. En los JOINS que la cláusula ON tenga las PK
8 Planificar Scheduled jobs para la noche o en momento de baja actividad del servidor
9 El abuso de triggers puede afectar al desempeño
10 Realizar Backups

Sección 33 Fin del curso (240, 241)

https://marianopuglisi.com

Boehringer Ingelheim