SQL - Structured Query Language

DDL - Data Definition Language - CREATE, ALTER, DROP commands.

DML - Data Manipulation Language - SELECT, INSERT, UPDATE, DELETE commands.

DCL - Data Control Language - GRAND, REVOKE, DENY

T-SQL - Transact SQL

Entity/Relationship Diagram - E/R Diagram

Projection - когато селектираме само някои колони от таблицата.

Selection - когато селектираме точно определени редове от таблицата.

Join - комбиниране на колони от различни таблици. (Обединяване на foreign key с primery key).

Нормализация - да направим базата без повторения (всяко entity в отделна таблица), а денормализация е обратното.

Транзакциите се грижат за конкурентният достъп и да не се омазват нещата когато няколко потребителя едновременно променят едни и същи данни.

Cluster - Да стройш 50 машини да работят заедно.

In computer science, ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties that guarantee that database transactions are processed reliably.

In the context of databases, a single logical operation on the data is called a transaction.

For example, a transfer of funds from one bank account to another, even involving multiple changes such as debiting one account and crediting another, is a single transaction.

Transactional - означава стабилно!!! Данните остават консистентни след всяка една операция върху тях!!!

В базата данни има компенсаторна логика!!! - използва се при трансакции.

Базата данни разчита, че последната промяна прави валидни всички предходни промени.

Ако последната промяна не е минала има rollback. Rollback се изпълняма при рестарт на сървъра.

Трансакциите гарантират консистентност и цялост.

..................................................................................................................................

Типове Данни

bit(1bit) - 0 или 1

real - e float!!!

numeric(scale,precision) - задаваме колко да е дълго нашето число и колко знака след десетичната запетайка!!!

money - се използва за работа с пари!!!

char - е фиксиран стринг. Ако не въведем информация до края на стринга, до края се запълват празни интервали.

varbinary - последователност от битове.

image - бинарен блок

datetime – date and time starting from 1.1.1753 to 31.12. 9999, a precision of 1/300 sec.

smalldatetime – date and time (1-minute precision)

timestamp – automatically generated number whenever a change is made to the data row - автоматично се променя при промяна на текущият ред!!!

uniqueidentifier – GUID identifier - Това е уникален идентификатор за целия свят!!!

xml – data in XML format - може да запазваме xml формат в нашите колони.

................................................................................................................................

SQL основи

Projection - Когато взимам само част от колоните в една таблица.

Selection - е нещо като филтър. Взимам само тези редове който ми трябват.

SELECT

DISTINCT DepartmentID

FROM Employees

SELECT FirstName AS Name FROM Employees

UNION

SELECT LastName AS Name FROM Employees

SELECT FirstName AS Name FROM Employees

INTERSECT

SELECT LastName AS Name FROM Employees

USE SoftUni

SELECT FirstName + ' ' + LastName AS [Full Name], EmployeeID AS [No.]

FROM Employees

SELECT \* FROM customers WHERE city like '%Berlin%'

SELECT [FirstName] + '''s last name is ' + [LastName] as [FullName],[Salary],[Salary] \* 0.2 as [Bonus]

FROM Employees

SELECT FirstName, LastName, Salary, ManagerID FROM Employees

WHERE ManagerID IS NOT NULL AND LastName LIKE '%so\_'

ORDER BY ManagerID DESC

SELECT \*

FROM Towns

ORDER BY Name DESC

OFFSET 20 ROWS

FETCH NEXT 5 ROWS ONLY

SELECT LastName, Name as DepartmentName

FROM Employees e, Departments d

WHERE e.DepartmentID = d.DepartmentID

По този начин без JOIN си правим JOIN. Не се използва често!!! Нарича се Equijoin.

SELECT e.LastName,e.LastName, a.AddressText, d.Name as DepartmentName

FROM Employees e

JOIN Departments d

ON e.DepartmentID = d.DepartmentID

INNER JOIN Addresses a

ON e.AddressID = a.AddressID

WHERE d.Name = 'Marketing'

Това е INNER JOIN.

INSERT INTO Projects(Name, StartDate)

SELECT Name + ' Restructuring', GETDATE()

FROM Departments

UPDATE Employees

SET Salary = Salary \* 1.10, JobTitle = 'Senior ' + JobTitle

WHERE DepartmentID = 3

UPDATE Employees

SET JobTitle = 'Senior ' + JobTitle

FROM Employees e

JOIN Departments d

ON e.DepartmentID = d.DepartmentID

WHERE d.Name = 'Sales'

DELETE FROM Employees

FROM Employees e

JOIN Departments d

ON e.DepartmentID = d.DepartmentID

WHERE d.Name = 'Sales'

...............................................................................................................................

SQL - Функции, Групиране и DDL

SELECT FirstName, LastName, Salary, DepartmentId

FROM Employees e

WHERE Salary =

(SELECT max(Salary)

FROM Employees

WHERE DepartmentId = e.DepartmentId)

ORDER BY DepartmentId

SELECT FirstName, LastName, DepartmentID, Salary

FROM Employees

WHERE DepartmentID IN

(SELECT DepartmentID

FROM Departments

WHERE Name='Sales')

declare @md date;

set @md = (select max(HireDate) from Employees);

declare @em nvarchar(50);

set @em = (select min(FirstName) from Employees

where HireDate = @md);

select @md as MaxDate, @em as EmployerName

declare @min money;

set @min = (select min(Salary) from Employees);

declare @max money;

set @max = (select min(Salary)\*1.10 from Employees);

SELECT FirstName, LastName, Salary

FROM Employees

WHERE Salary BETWEEN @min AND @max

SELECT

AVG(Salary) [Average Salary],

MAX(Salary) [Max Salary],

MIN(Salary) [Min Salary],

SUM(Salary) [Salary Sum]

FROM Employees

WHERE JobTitle = 'Production Technician'

select count(distinct FirstName)

from Employees

SELECT COUNT(DISTINCT m.EmployeeID) AS [Number of Managers]

FROM Employees e

JOIN Employees m

ON e.ManagerID = m.EmployeeID

SELECT e.FirstName, e.LastName, e.HireDate, d.Name as Dept

FROM Employees e JOIN Departments d

ON e.DepartmentID = d.DepartmentID

WHERE e.HireDate =

(SELECT MIN(HireDate) FROM Employees

WHERE DepartmentID = d.DepartmentID)

select d.Name as DepartmentName,JobTitle, sum(salary) as SumSalary, count(EmployeeID) as CountEmp

from Employees e join Departments d

on e.DepartmentID = d.DepartmentID

group by e.DepartmentID ,d.Name, JobTitle

order by e.DepartmentID

SELECT CAST(FirstName + ', ' AS nvarchar(max))

FROM Employees

FOR XML PATH ('')

SELECT STUFF((SELECT '; ' + FORMAT(Date, 'yyyyMMdd') FROM Ads FOR XML PATH('')),1,2,'') AS AllDates

SELECT DepartmentID, COUNT(EmployeeID) as EmpCount,

AVG(Salary) as AverageSalary

FROM Employees

GROUP BY DepartmentID

HAVING COUNT(EmployeeID) BETWEEN 3 AND 5

SELECT e.FirstName + ' ' + e.LastName AS FullName ,

CASE

WHEN m.FirstName + ' ' + m.LastName IS NULL THEN 'No manager'

ELSE m.FirstName + ' ' + m.LastName

END

AS Manager

FROM Employees e

LEFT OUTER JOIN Employees m

ON e.ManagerID = m.EmployeeID

SELECT e.FirstName + ' ' + e.LastName AS FullName , ISNULL( m.FirstName + ' ' + m.LastName, 'No manager') AS Manager

FROM Employees e

LEFT OUTER JOIN Employees m

ON e.ManagerID = m.EmployeeID

select FirstName, LastName, ISNULL(CAST(ManagerID as nvarchar(max)),'no manager')

from Employees

order by ManagerID

SELECT FirstName, LastName

FROM Employees

WHERE LEN(LastName) = 5

SELECT TOP 1 t.Name AS Name, COUNT(a.TownID) AS [Number of employees]

FROM Employees e

JOIN Addresses a

ON e.AddressID = a.AddressID

JOIN Towns t

ON a.TownID = t.TownID

GROUP BY a.TownID, t.Name

ORDER BY COUNT(a.TownID) DESC

SELECT t.Name AS Town, COUNT(e.ManagerID) AS [Number of managers]

FROM Employees e

JOIN Addresses a

ON e.AddressID = a.AddressID

JOIN Towns t

ON a.TownID = t.TownID

WHERE e.EmployeeID IN

(SELECT DISTINCT ManagerID FROM Employees)

GROUP BY t.Name

ORDER BY t.Name

SELECT LastName, LEN(LastName) AS LastNameLen,

UPPER(LastName) AS UpperLastName

FROM Employees

WHERE RIGHT(LastName, 3) = 'son'

select dateadd(day,1,datefromparts(2012,2,28))

select FirstName, LastName,

DATEDIFF(year,HireDate,GETDATE()) \* salary/1000

as [Annual Bonus]

from Employees

SELECT convert(varchar, getdate(), 104)+' '+convert(varchar, getdate(), 114)

SELECT CAST(ROUND(5.86, 0) AS INT)

NEWID() - ми генерира уникален номер

<> - Това е операторът различно (!)

GROUP BY - групира по колона и разделя редовете на подмножество от техните редове??? и след това може да правим SUM, MAX и т.н.

...........................................................................................................................

T-SQL Stored Procedures

DECLARE @n tinyint

SET @n = 5

IF (@n BETWEEN 4 and 6)

BEGIN

WHILE (@n > 0)

BEGIN

SELECT @n AS 'Number',

CASE

WHEN (@n % 2) = 1

THEN 'EVEN'

ELSE 'ODD'

END AS 'Type'

SET @n = @n - 1

END

END

ELSE

PRINT 'NO ANALYSIS'

GO

.....................................................................

Stored Procedure

USE SoftUni

GO

CREATE PROC dbo.usp\_SelectEmployeesBySeniority

AS

SELECT \*

FROM Employees

WHERE DATEDIFF(Year, HireDate, GETDATE()) > 5

GO

Така си създавам процедурата!!!

INSERT INTO Customers

EXEC usp\_SelectEmployeesBySeniority - А така я изпълнявам процедурата!!!

..........................................................................................................................

Redis

SET users:count 5

RPUSH users:names peter

HSET users:peter name "Петър Иванов"

HSET users:peter email "peter@gmail.com"

HSET users:peter age 25

HKEYS users:peter

1) "name"

2) "email"

3) "age"

HVALS users:peter

1) "Петър Иванов"

2) "peter@gmail.com"

3) "25"

HGETALL users:peter

1) "name"

2) "Петър Иванов"

3) "email"

4) "peter@gmail.com"

5) "age"

6) "25"

HGET users:peter name

"Петър Иванов"

RPUSH users:names maria

HSET users:maria name "Мария Кирчева"

HSET users:maria email "maria@yahoo.com"

HSET users: maria age 25

Publish - Subscribe модел!!!

Първи клиент:

subscribe goals

Reading messages...

1) "subscribe"

2) "goals"

Втори клиент:

publish goals "Manchester - Chelse 0:0"

Първи клиент:

1) "message"

2) "goals"

3) "Manchester - Chelse 0:0"