01 - Uvod

Baza - skup međusobno povezanih podataka.
SUBP (**DBMS**) - Softverski sistem koji omogućava korisnicima rad sa bazama podataka.
Create - Dodavanje novih podataka, **R**ead - Čitanje, **U**pdate - Promena, **D**elete - Uklanjanje.
Key - Atribut koji se koristi da identifikuje slogove u bazi. **Primary key** - -ll- jedinstveno identifikuju.
Modeli - Nivoi/pogledi na bazu (konceptualni, logički, fizički)
SUBP tipovi *(Većina DBMS sistema danas su objektno-relacioni)*

- Hijerarhijski - Podaci su organizovani kao drvo (odnos "roditelj-dete")
- Mrežni - Struktura podataka je u obliku grafa (dodati sibling)
- Relacioni - Podaci se nalaze u relacijama (tabelama)
- Objektno-orijentisani - Podaci predstavljeni kao objekti, podržano nasleđivanje, pristup preko metoda
- O-relacioni - Kombinuje prethodna dva (korisničko-definisani tipovi i funkcije, nasleđivanje).

SQL - Structured Query Language (IBM), deklarativni jezik (opisuje šta želite, a ne korake kako doći to toga)
Kategorije SQL naredbi

- **DDL** - Data Definition Language (Create, Alter, Drop) (naredbe za definisanje objekata)
- **DML** - Data Manipulation Language (Select, Insert, Update, Delete) (pretraga i modifikovanje podataka)
- **DCL** - Data Control Language (Grant, Revoke, Rollback, Commit) (naredbe za davanje prava pristupa)

Normalizacija - Proces organizovanja polja i tabela u bazi sa ciljem postizanja optimalnog dizajna baze.
1NF - Eleminiše grupe sa ponavljanjem u tabelama, kreira odvojenu tabelu za svaki skup srodnih podataka i identifikuje svaki skup srodnih podataka primarnim ključem.
2NF - Kreira zasebne tabele za skupove vrednosti koji se odnose na više redova i povezuje ih stranim ključem.
3NF - Ni jedna kolona ne sme da zavisi od neke druge kolone koja nije ključ.
Prirodan ključ - formiran na osnovu podataka koji se odnose na entitet (JBMG) ; **Surogat** - Veštački (KorisnikID)
Strani ključ - referenca (veza) između tabela. Primarni ključ jedne tabele je strani ključ druge.
Odnosi između tabela : 1:1 (jedan red jedne tabele odgovara tačno jednom druge), 1:m, m:m.

02 - SQL server
- Klijentske aplikacije i alati povezuju se preko **Tabluar Data Stream (TDS)** protokola

03 - Tipovi podataka
Tipovi podtaka određuju šta se može pamtiti, ograničavaju tip pdtk koje obejkat može da sadrži.
3 osnovna skupa tipova podataka

- Sistemski tipovi podataka
- Alijas tipovi podataka (npr. BrojTelefona kao alijas za nvarchar(16)).
- Korisnički definisani tipovi podataka (pomoću .NET)

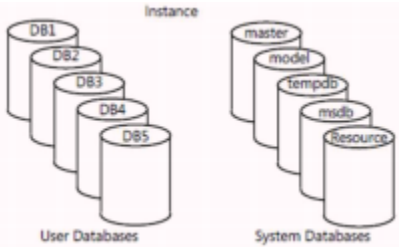
SQL server ne nudi opciju za unos datuma i vremena eksplicitno, već se oni unose kao karakteri i konvertuju.
GUID - **G**lobal **U**nique **Id**entifier - postoji tip podataka **uniqueidentifier**.
Null - Vrednost se ne mora obezbediti. **Not null** - Vrednost se mora obezbediti.
Unicode - Svetski standard za kodiranje znakova, zahteva N (National Character Set) prefix kod literala.
Non-unicode karakteri zauzimaju 1 bajt, unicode tipovi zauzimaju 2 bajta po karakteru.
DECLARE @Hello nvarchar(10); SET @Hello = N'Ćao';
Collations - kontroliše pravila koja regulišu kako SQL server sortira i poredi vrednosti
(CI/CS - Case I/Sensitive (a = A), AI/S - Accent I/Sensitive (ć = c (ako je insensitive))

CAST, **CONVERT** i **PARSE** omogućavaju promenu jednog tipa podatka u drugi.
Kada se podaci ne konvertuju eksplicitno, SQL server pokušava da implicitno konvertuje tipove.
Prioritet (niži -> viši)
CHAR > VARCHAR > NVARCHAR > TINYINT > INT > DECIMAL > TIME > DATE > DATETIME2 > XML
timestamp i **rowversion** automatski menjaju vrednost kada god se menja red.

04 - Kreiranje baza podataka i tabela

Logički, baza je uvek jedan entitet. Fizički može biti smeštena u više datoteka :
.mdf (Primary Data File), .ndf (Secondary Data File), .ldf (Transaction Log File)
Postoje sistemske i korisničke baze (korisničke se prave sa CREATE DATABASE).

Sistemska baza	Opis
master	Sadrži informacije o: drugim bazama, pravima pristupa, sistemske tabele i procedure Ključna je za rad SQL servera i ne može se brisati
msdb	Čuva informacije o job-ovima, operatorima i alert-ima
model	Šablon po kom se kreiranju nove, korisničke baze
tempdb	Čuva privremene podatke, kao što su: privremene tabele, varijable tipa table, podaci koji se sortiraju, ...
resource	Sakrivena, read-only baza Čuva definicije svih sistemskih objekata



Šeme su kontejneri za objekte tipa tabela, procedura, pogleda... definišu bezbednosne granice.
Server.Database.Schema.Object
*SELECT ProductID FROM **Production.Product**;*
Ako je ime šeme izostavljeno, obično se **dbo** šema smatra kao podrazumevana. Prvo se vrši pretraga objekta u korisnikovoj podrazumevanoj šemi, ako se ne nađe tu onda se dbo šema pretražuje.
Tabele predstavljaju tip objekta ili entitet, a svaki njihov red predstavlja jedan objekat ili entitet.
CREATE / ALTER / DROP TABLE
Session privremene tabele vidljive su jedino njihovim kreatorima u istoj sesiji, kreiraju se **# prefiksom** i nestaju kada se korisnik diskonektuje. **##** za globalne privremene tabele, brišu se kada se svi koji ih koriste diskonektuju.

05 - SQL upitni jezik

Deklarativni, a ne proceduralni jezik. Microsoft implementacija se naziva T-SQL (Transact). DML, DDL, DCL.
Imena varijable počinju sa jednim @ znakom (@@ je rezervisan za sistemske funkcije).
DECLARE @VAR INT = 30;
Izrazi predstavljaju kombinaciju identifikatora, vrednosti i operatora koji daju rezultat (kod SELECT i WHERE)

Elementi	Predikati i operatori
Predikati	IN, BETWEEN, LIKE, ALL, SOME, ANY
Operatori poređenja	=, >, <, >=, <=, <>, !=, !>, !<
Logički operatori	AND, OR, NOT
Aritmetički operatori	+, -, *, /, % (modulo)
Nadovezivanje	+
Dodeljivanje	=

Kontrola toka	Upravljanje greškama	Upravljanje transakcijama
• IF...ELSE • WHILE • BREAK • CONTINUE • BEGIN...END	• TRY...CATCH	• BEGIN TRANSACTION • COMMIT TRANSACTION • ROLLBACK TRANSACTION

06 - SELECT naredba

SELECT *kolona* FROM *tabela/pogled*

Skalarni izrazi vraćaju jednu vrednost - SELECT (unitprice*qty) FROM sales;

Redosled	Element	Izraz	Uloga
5	SELECT	<select list>	Definiše kolone koje se vraćaju
1	FROM	<table source>	Definiše tabele koje se pretražuju
2	WHERE	<search condition>	Filtrira redove koristeći uslov
3	GROUP BY	<group by list>	Uređuje redove po grupama
4	HAVING	<search condition>	Filtrira grupe koristeći uslov
6	ORDER BY	<order by list>	Sortira rezultat

DISTINCT određuje da se u rezultujućem setu mogu naći samo jedinstveni redovi, uklanja duplikate.

Alijas (prva i treća metoda rade i u FROM delu na tabelama)

- SELECT qty **AS** quantity
- SELECT quantity **=** qty
- SELECT qty quantity

CASE izrazi vraćaju jednu vrednost, poredi jednu vrednost sa listom mogućih vrednosti i vraća prvo pronađenu. Ukoliko ne pronađe vrednost, vraća vrednost iz else dela. Ako else nije definisan onda NULL.

```
SELECT productid, productname, categoryid,
CASE categoryid
        WHEN 1 THEN 'Beverages'
        WHEN 2 THEN 'Condiments'
        WHEN 3 THEN 'Confections'
        ELSE 'Unknown Category'
END AS categoryname
FROM Production.Categories
```

07 - JOIN naredba

Kartezijanski proizvod kreira sve moguće kombinacije dva skupa (svaki sa svakim, npr. fudbal liga)

Tip JOIN-a	Opis
CROSS	Kombinuje sve redove u obe tabele (kreira Kartezijanski proizvod).
INNER	Počinje kao Kartezijanski proizvod; primenjuje filter za izbor redova na osnovu predikata.
OUTER	Počinje kao Kartezijanski proizvod; svi redovi iz jedne tabele se čuvaju, odgovarajući redovi iz druge tabele se preuzimaju. Dodatne NULL vrednosti se ubacuju.

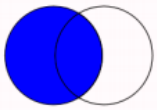
INNER vraća samo redove koji odgovaraju uslovu pronađene u obe tabele (presek skupova)

Redosled tabela nije važan. *FROM t1 INNER JOIN t2 ON t1.c1 = t2.c2*

OUTER vraća sve redove iz jedne tabele i redove iz druge tabele koji odgovaraju uslovu. U rezultat se uključuju i redovi koji nemaju odgovarajuće redove iz druge tabele (NULL vrednosti sa druge strane)

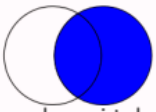
Vraća sve redove iz prve (LEFT) tabele, i samo odgovarajuće redove iz druge tabele:

```
FROM t1 LEFT OUTER JOIN t2 ON
   t1.c1 = t2.c2
```



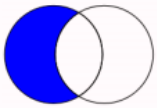
Vraća sve redove iz druge (RIGHT) tabele, i samo odgovarajuće redove iz prve tabele :

```
FROM t1 RIGHT OUTER JOIN t2 ON
   t1.c1 = t2.c2
```



Vraća samo redove iz prve tabele, koji nemaju odgovarajuće redove u drugoj tabeli:

```
FROM t1 LEFT OUTER JOIN t2 ON
   t1.col = t2.col
WHERE t2.col IS NULL
```



CROSS je kartezijanski (svaki sa svakim)

SELF JOIN poredi redove iz iste tabele jedne sa drugima (npr. podaci o zaposlenima i imena supervizora)

Vraća sve zaposlene sa imenom šefa, ukoliko ga zaposleni ima (inner join):

```
SELECT  e.empid, e.lastname,
        e.title, e.mgrid, m.lastname
FROM    HR.Employees AS e
INNER JOIN HR.Employees AS m
ON e.mgrid=m.empid;
```

Vraća sve zaposlene sa šifrom šefa (outer join). Kada prikazuje podatke o šefu, vraća NULL za njegovog šefa:

```
SELECT  e. empid, e.lastname,
     e.title, m.mgrid
FROM HR.Employees AS e
LEFT OUTER JOIN HR.Employees AS m
ON e.mgrid=m.empid;
```

08 - Sortiranje i filtriranje podataka

Redosled sortiranja se određuje sa ASC (rastući) ili DESC (opadajući)

```
SELECT <select_list>
FROM   <table_source>
ORDER BY <order_by_list> ASC|DESC;
```

```
WHERE <search_condition>
```

```
WHERE <column> <operator> <expression>

Operatori: AND, OR, NOT, IN, BETWEEN, LIKE, >,
<, >=, <=, <>, !=, !>, !<
```

ORDER BY sa imenima kolona:

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
ORDER BY orderdate;
```

Filtrira redove za kupce iz Španije

```
SELECT contactname, country
FROM Sales.Customers
WHERE country = N'Spain';
```

ORDER BY sa alijasima za kolone:

```
SELECT orderid, custid, YEAR(orderdate)
AS orderyear
FROM Sales.Orders
ORDER BY orderyear;
```

Filtrira redove za narudžbine posle 1. januara 2007.

```
SELECT orderid, orderdate
FROM Sales.Orders
WHERE orderdate > '20070101';
```

ORDER BY u opadajućem redosledu:

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
ORDER BY orderdate DESC, orderid ASC;
```

Filtrira redove za zadati opseg datuma

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
WHERE orderdate >= '20070101' AND orderdate <
'20080101';
```

SELECT TOP (5) ili **SELECT TOP** 5 **PERCENT** <kolona> FROM <tabela> ORDER BY <order>

OFFSET-FETCH obezbeđuje mehanizam za straničenje rezultata, odrediti broj redova koji će biti preskočen, kao i broj redova koji će biti vraćen.

Omogućeno je i upravljanje sa **NULL**.

```
SELECT TOP (5) WITH TIES
orderid, custid, orderdate
FROM Sales.Orders
ORDER BY orderdate DESC;
```

Vraća redove od 51-100

```
SELECT orderid, custid, empid, orderdate
FROM Sales.Orders
ORDER BY orderdate, orderid DESC
OFFSET 50 ROWS FETCH NEXT 50 ROWS ONLY;
```

```
SELECT custid, city, region, country
FROM Sales.Customers
WHERE region IS NULL;
```

09 - Modifikovanje podataka

```
INSERT INTO Sales.OrderDetails
   (orderid, productid, unitprice, qty, discount)
VALUES (11077, 72, 34.80, DEFAULT, DEFAULT);
```

DEFAULT će ubaciti defaultnu vrednost definisanu pri kreiranju baze, ako je nema ubacuje NULL.

INSERT...SELECT se koristi da ubaci rezultujući set u postojeću tabelu

```
INSERT INTO Sales.OrderHist(
   orderid,custid,empid,orderdate)
SELECT orderid,custid,empid,orderdate
FROM Sales.Orders
WHERE orderdate < '20080101';
```

SELECT... INTO kreira novu tabelu svaki put kada se tabela izvršava, kopira imena i tipove kolona ali ne i ograničenja i indekse.

```
SELECT orderid, custid, empid, orderdate, shippeddate
INTO Sales.OrderArchive
FROM Sales.Orders
WHERE orderdate < '20080101';
```

IDENTITY svojstvo automatski generiše sekvencijalne brojeve kod dodavanja novog reda u tabelu. (vrednost, pomeraj)

```
CREATE TABLE Production.Products(
productid int IDENTITY(1,1) NOT NULL,
productname nvarchar(40) NOT NULL,
categoryid int NOT NULL,
unitprice money NOT NULL)
```

UPDATE (menja), **DELETE** (uklanja), **MERGE** menja podatke (IUD) na osnovu jednog ili više uslova.
TRUNCATE briše sve redove u tabeli odjednom. Neće se izvršiti ako ima constraint-a.

```
TRUNCATE TABLE dbo.Nums;
```

10 - Grupisanje i agregacija podataka

Agregatne funkcije vraćaju skalrnu vrednost i ignorišu NULL (izuzev u COUNT(*))
SUM, MIN, MAX, AVG, COUNT.

```
SELECT  AVG(unitprice) AS avg_price,
           MIN(qty) AS min_qty,
           MAX(discount) AS max_discount
FROM Sales.OrderDetails;
```

```
avg_price min_qty max_discount
--------- ------- ------------
26.2185   1         0.250
```

DISTINCT sume eliminišu samo duple vrednosti, a ne i duple redove (nalik SELECT DISTINCT)
SELECT COUNT(DISTINCT custid)

```
SELECT empid, YEAR(orderdate) AS orderyear,
    COUNT(custid) AS all_custs,
    COUNT(DISTINCT custid) AS unique_custs
FROM Sales.Orders
GROUP BY empid, YEAR(orderdate);
```

```
empid       orderyear   all_custs   unique_custs
----------- ----------- ----------- ------------
1           2006        26          22
1           2007        55          40
1           2008        42          32
2           2006        16          15
```

GROUP BY kreira grupe za koje se izračunava sumarna vrednost

```
SELECT empid, COUNT(*) AS cnt
FROM Sales.Orders
GROUP BY empid;
```

HAVING klauzula obezbeđuje uslov za pretraživanje, koji svaka grupa mora da zadovolji

HAVING se obrađuje posle GROUP BY

```
SELECT custid, COUNT(*) AS count_orders
FROM Sales.Orders
GROUP BY custid
HAVING COUNT(*) > 10;
```

WHERE filtrira redove pre nego što se kreiraju grupe - Kontroliše koji redovi će biti u grupama
HAVING filtrira grupe - Kontroliše koje grupe će biti prosleđene sledećoj logičkoj fazi obrade.

11 - Kreiranje podupita

Podupiti su SELECT naredbe ugnježdene u drugi upit. Unutrašnji upit je podupit, spoljašnji upit je upit u okviru koga se piše podupit. Rezultati unutrašnjeg upita se predaju spoljašnjem upitu.
Podupiti mogu da vraćaju jednu vrednost (skalarni) ili više vrednosti kao rezultat (više-vrednosni)
Podupit može biti samostalan (ne zavisi od spoljašnjeg upita) i korelativan (zavisi od spoljašnjeg upita)

```
SELECT orderid, productid, unitprice, qty
FROM Sales.OrderDetails
WHERE orderid =
    (SELECT MAX(orderid) AS lastorder
    FROM Sales.Orders);
```

```
SELECT custid, orderid
FROM Sales.orders
WHERE custid IN (
    SELECT custid
    FROM Sales.Customers
    WHERE country = N'Mexico');
```

```
SELECT custid, orderid
FROM Sales.orders
WHERE custid IN (2,3,13,58,80);
```

Korelativni podupiti dobijaju vrednosti iz spoljašnjeg upita

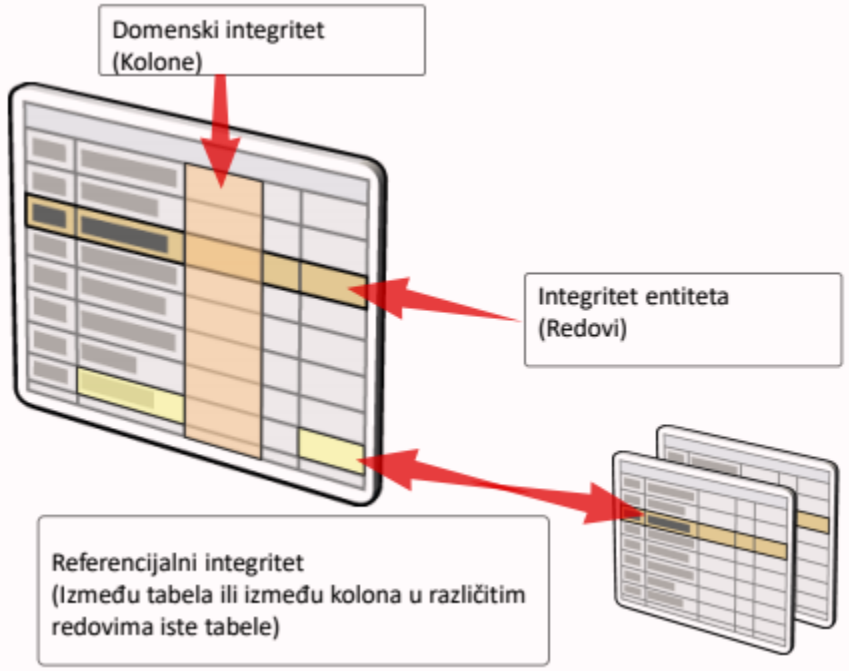
```
SELECT custid, orderid, orderdate
FROM Sales.Orders AS outerorders
WHERE orderdate =
    (SELECT MAX(orderdate)
     FROM Sales.Orders AS innerorders
     WHERE innerorders.custid = outerorders.custid)
ORDER BY custid;
```

EXISTS se koristi kao test za proveru postojanja redova (vraća se TRUE ukoliko se neki redovi vraćaju iz podupita i FALSE ako ne)

```
SELECT custid, companyname
FROM Sales.Customers AS c
WHERE EXISTS (
    SELECT *
    FROM Sales.Orders AS o
    WHERE c.custid=o.custid);
```

```
SELECT custid, companyname
FROM Sales.Customers AS c
WHERE NOT EXISTS (
    SELECT *
    FROM Sales.Orders AS o
    WHERE c.custid=o.custid);
```

12 - Obezbeđenje integriteta podataka

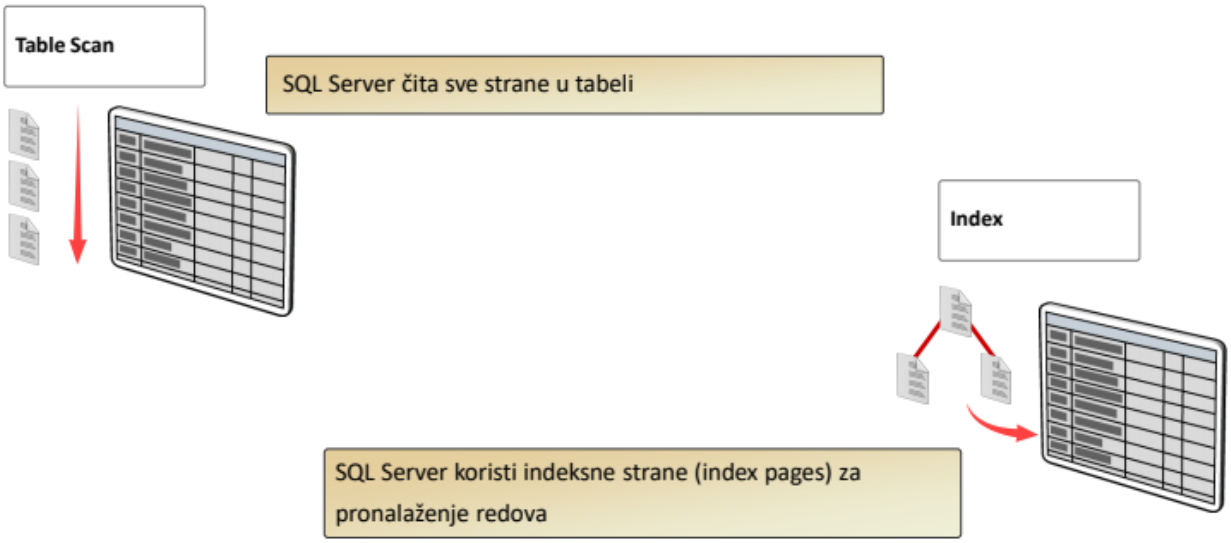


Mehanizam	Opis
Tipovi podataka	Definiše tipove podataka koji se mogu smestiti u kolonu
Nulabilnost	Određuje da li vrednost mora ili ne mora da postoji u koloni
Ograničenje (Constraint)	Definiše pravila koja ograničavaju vrednosti koja se mogu smestiti u kolonu ili kakav međusoban odnos moraju imati vrednosti u različitim kolonama
Okidač (Trigger)	Definiše naredbe koje se izvršavaju automatski kada se tabela menja

Constrainti - DEFAULT,CHECK, PRIMARY KEY, UNIQUE KEY, FOREIGN KEY
Unique key - jedinstven u svakom redu, ali u jednom redu može biti NULL.
SCOPE_IDENTITY() vraća poslednji IDENTITY generisan u toj sesiji.

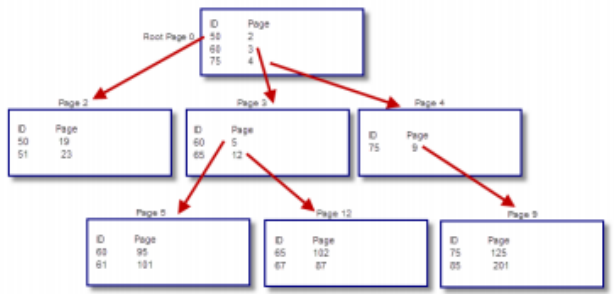
```
CREATE TABLE Sales.Opportunity
(
    OpportunityID int NOT NULL,
    Requirements nvarchar(50) NOT NULL,
    ReceivedDate date
      CONSTRAINT DF_Opportunity_ReceivedDate
      DEFAULT (SYSDATETIME()) NOT NULL,
    LikelyClosingDate date NOT NULL,
    SalespersonID int NOT NULL,
    Rating int NOT NULL
);
```

14 - Indeksi



Svi upiti se mogu izvršiti bez postojanja indeksa. Indeksi poboljšavaju performanse.

- Indeksi se zasnivaju na strukturi drveta
  - Balansno drvo (nije binarno), jer može imati više od dva deteta
- Čvor na vrhu se naziva koreni čvor (root)
- Čvorovi na dnu se nazivaju lišće (leaf nodes)

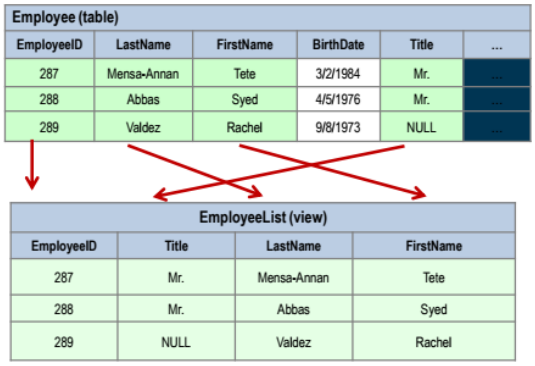


… Ovu lekciju ću preskočiti.

15 - Pogledi

Pogledi su objekti u bazi podataka kojima se pristupa isto kao tabeli. Pogledi su SELECT upit sa imenom. *Možemo da damo selectu naziv i sačuvamo ga kao view, umesto da ga čuvamo u posebnom fajlu. View je zapamćeni select, a ne i zapamćeni podaci.* Imamo korisničke i sistemske poglede.

View - zapamćena select naredba. **CREATE VIEW <ime> AS <SELECT> ; ALTER VIEW … AS … ; DROP VIEW**




```
-- Korak 1: TSQL je aktivna baza
USE TSQL;
GO

-- Korak 2: Jednostavan pogled
CREATE VIEW HR.EmpPhoneList
AS
SELECT empid, lastname, firstname, phone
FROM HR.Employees;
GO

-- Korišćenje prethodno kreiranog pogleda
SELECT empid, lastname, firstname, phone
FROM HR.EmpPhoneList;
GO

-- Korak 3: Složeniji pogled
CREATE VIEW Sales.OrdersByEmployeeYear
AS
    SELECT  emp.empid AS employee ,
            YEAR(ord.orderdate) AS orderyear ,
            SUM(od.qty * od.unitprice) AS totalsales
    FROM    HR.Employees AS emp
            JOIN Sales.Orders AS ord ON emp.empid = ord.empid
            JOIN Sales.OrderDetails AS od ON ord.orderid = od.orderid
    GROUP BY emp.empid, YEAR(ord.orderdate)
GO

-- Korišćenje prethodno kreiranog pogleda
SELECT employee, orderyear, totalsales
FROM Sales.OrdersByEmployeeYear
ORDER BY employee, orderyear;
GO

-- Korak 4: Brisanje pogleda
DROP VIEW Sales.OrdersByEmployeeYear;
DROP VIEW HR.EmpPhoneList;
```

16 - Uskladištene procedure

Prilikom interakcije sa SQL serverom uvek moramo serveru da pošaljemo neki upit a on vraća rezultujući set. Kada šaljemo nešto serveru to mogu biti direktne naredbe ili možemo nekom skupu naredbi da damo ime i da pozivamo po imenu tu grupu naredbi. Taj skup naredbi se pamti na serveru. Izvršavaju se pozivom **EXECUTE** T-SQL naredbe.

```
CREATE PROCEDURE Sales.GetSalesPersonNames
AS
BEGIN
  SELECT sp.SalesPersonID, c.LastName, c.FirstName
  FROM Sales.Salesperson AS sp
  INNER JOIN Person.Contact AS c
  ON sp.SalesPersonID = c.ContactID
  WHERE sp.TerritoryID IS NOT NULL
  ORDER BY sp.SalesPersonID
END;
GO
```

```
EXEC Sales.GetSalespersonNames;
```

```
ALTER PROCEDURE Sales.GetSalespersonNames
AS
BEGIN
 SELECT sp.SalesPersonID, c.LastName, c.FirstName
  FROM Sales.Salesperson AS sp
  INNER JOIN Person.Contact AS c
  ON sp.SalesPersonID = c.ContactID
  WHERE sp.TerritoryID IS NOT NULL
  AND sp.SalesQuota IS NOT NULL
  ORDER BY sp.SalesPersonID
END;
GO
```

```
DROP PROCEDURE Sales.GetSalespersonNames;
```

```
CREATE PROCEDURE Sales.OrdersByDueDateAndStatus
@DueDate datetime, @Status tinyint = 5
AS
SELECT SalesOrderID, OrderDate, CustomerID
FROM Sales.SalesOrderHeader
WHERE soh.DueDate = @DueDate
AND soh.[Status] = @Status;
GO
EXEC Sales.OrdersByDueDateAndStatus '20050613',8;
EXEC Sales.OrdersByDueDateAndStatus '20050713';
EXEC Sales.OrdersByDueDateAndStatus @DueDate = '20050713',
                                    @Status = 5;
```

```
CREATE PROC Sales.GetOrderCountByDueDate
@DueDate datetime, @OrderCount int OUTPUT
AS
   SELECT @OrderCount = COUNT(*)
   FROM Sales.SalesOrderHeader AS soh
   WHERE soh.DueDate = @DueDate;
GO
DECLARE @DueDate datetime = '20050713';
DECLARE @OrderCount int;
EXEC Sales.GetOrderCountByDueDate @DueDate,
                                  @OrderCount OUTPUT;
SELECT @OrderCount;
```

OUTPUT kod deklaracije i exec.

17 - Obrada grešaka

Grupa	Opis
0 - 9	Informativne poruke
10	Informativne poruke koje vraćaju status
11 - 16	Greške koje može korisnik da ispravi
17 - 19	Greške koje može korisnik ne može da ispravi
20 - 24	Ozbiljne sistemske greške

RAISERROR - stari način

```
DECLARE @DatabaseID int = DB_ID();
DECLARE @DatabaseName sysname = DB_NAME();
RAISERROR
(N'Current database ID:%d, database name: %s.',
    10, -- Severity.
    1, -- State.
    @DatabaseID, -- First substitution argument.
    @DatabaseName); -- Second substitution argument.
```

THROW - jednostavniji način, nivo ozbiljnosti uvek 16.

```
THROW 51000, 'The record does not exist.', 1;

Rezultat:
Msg 51000, Level 16, State 1, Line 1

The record does not exist.
```

@@ERROR- vraća 0 ukoliko je naredba ispravno izvršena, tj. broj ukoliko je naredba generisala grešku. Menja vrednost nakon izvršavanja svake naredbe.

```
DECLARE @ErrorValue int;
RAISERROR(N'Message', 16, 1);
SET @ErrorValue = @@ERROR;
IF @ErrorValue <> 0
  PRINT 'Error=' + CAST(@ErrorValue AS VARCHAR(8));
```

TRY-CATCH blok - *BEGIN TRY <naredbe> END TRY BEGIN CATCH <naredbe> END CATCH;*

```
CREATE PROCEDURE Error.GeneralHandler
AS
    SELECT
        ERROR_NUMBER() AS ErrorNumber,
        ERROR_SEVERITY() AS ErrorSeverity,
        ERROR_STATE() as ErrorState,
        ERROR_PROCEDURE() as ErrorProcedure,
        ERROR_LINE() as ErrorLine,
        ERROR_MESSAGE() as ErrorMessage;
GO
```

```
BEGIN TRY
  -- Generate divide-by-zero error.
  SELECT 1/0;
END TRY
BEGIN CATCH
  -- Execute the error retrieval routine.
  EXECUTE Error.GeneralHandler;
END CATCH;
```

```
BEGIN TRY
    SELECT CAST(N'Some text' AS int);
END TRY
BEGIN CATCH
    PRINT 'Greska';
    THROW;
END CATCH
GO
```

18 - Funkcije

Postoje **sistemske** i **korisnički definisane funkcije** (**skalarne** vraćaju jednu vrednost, **table-valued** funkcije (TVF) vraćaju tabelu).

```
-- kreiranje skalarne funkcije
CREATE FUNCTION dbo.Diagonal(@x int, @y int)
RETURNS float
AS BEGIN
  RETURN (SQRT(@x*@x + @y*@y));
END;
GO
-- pozivanje skalarne funkcije
SELECT dbo.Diagonal(2,4) as Dijagonala;
GO
-- rezultat
4.47213595499958
```

Determinističke funkcije - Uvek vraćaju isti rezultat za određeni dati ulaz i dato stanje baze
Nedeterminističke funkcije - Mogu vratiti različite rezultata za određeni ulaz čak iako je stanje baze isto

```
-- primer determinističke funkcije
CREATE FUNCTION dbo.AddInteger(@FirstValue int, @SecondValue int)
RETURNS int
AS BEGIN
  RETURN @FirstValue + @SecondValue;
END;
GO
-- primer nedeterminističke funkcije
SELECT SYSDATETIME();
-- provera da li je funkcija deterministička ili ne
SELECT OBJECTPROPERTY(OBJECT_ID('dbo.AddInteger'),'IsDeterministic');
```

Table-Valued funkcije - vraćaju TABLE tip podataka, telo f-je se nalazi između BEGIN i END. Obezbeđuje se definicija tabele koja se vraća kao rezultat koja se popunjava u okviru tela f-je i potom vraća.
Inline - u okviru tela funkcije se nalazi samo jedna SELECT naredba. *Parametrizovani pogledi*. Vraćaju rezultujući set i nemaju telo (BEGIN i END), vraćaju tabelu na osnovu jedne SELECT naredbe.
Multi-statement - kreira, popunjava i vraća tabelu

```
-- kreiranje funkcije
CREATE FUNCTION Sales.GetLastOrdersForCustomer
(@CustomerID int, @NumberOfOrders int)
RETURNS TABLE
AS
RETURN (SELECT TOP(@NumberOfOrders)
               soh.SalesOrderID,
               soh.OrderDate,
               soh.PurchaseOrderNumber
        FROM Sales.SalesOrderHeader AS soh
        WHERE soh.CustomerID = @CustomerID
        ORDER BY soh.OrderDate DESC, soh.SalesOrderID DESC);
GO
-- korišćenje funkcije
SELECT * FROM Sales.GetLastOrdersForCustomer(17288,2);
```

```
CREATE FUNCTION dbo.GetDateRange
(@StartDate date, @NumberOfDays int)
RETURNS @DateList TABLE (Position int, DateValue date)
AS BEGIN
    DECLARE @Counter int = 0;
    WHILE (@Counter < @NumberOfDays) BEGIN
        INSERT INTO @DateList
            VALUES(@Counter + 1,
                   DATEADD(day,@Counter,@StartDate));
        SET @Counter += 1;
    END;
    RETURN;
END;
GO
-- korišćenje funkcije
SELECT * FROM dbo.GetDateRange('20091231', 14);
```

19 - Transakcije

Transakcija je skup SQL naredbi definisan kao jedinica posla. Ili se sve naredbe izvršavaju, ili nijedna.
BEGIN/COMMIT/ROLLBACK TRANSACTION
ACID - Atomicity, Consistency, Isolation, Durability - skup osobina SUBP čije postojanje garantuje pravilno izvršavanje transakcija. Nedeljivost (sve ili ništa), konzistentnost, izolacija, trajnost.
SQL baze zasnovane su na ACID konceptu, **NoSQL baze** zasnovane su na CAP teoremi.

```
33  -- Korak 3: Kreiranje transakcije za 2 INSERT naredbe
34  BEGIN TRY
35      BEGIN TRANSACTION
36          INSERT INTO dbo.SimpleOrders (custid, empid, orderdate) VALUES (68, 9, '2006-07-12');
37          INSERT INTO dbo.SimpleOrderDetails (orderid, productid, unitprice, qty) VALUES (1, 2, 15.20, 20);
38      COMMIT TRANSACTION
39  END TRY
40  BEGIN CATCH
41      SELECT ERROR_NUMBER() AS [Error Number], ERROR_MESSAGE() AS [Error Message];
42      ROLLBACK TRANSACTION
43  END CATCH;
```

Metode kontrole konkurentnosti

Pesimističko - Podaci se zaključavaju prilikom čitanja, drugi korisnici su blokirani

Optimističko - Podaci se ne zaključavaju posle čitanja, već kada se izvršava njihova promena

Zaključavanje je mehanizam koji sinhronizuje pristup više korisnika istim podacima u isto vreme
Read locks - dozvoljava drugima da čitaju, ali ne i da menjaju podatke
Write locks - zabranjuje drugima da čitaju ili menjaju podatke
Zaključuavanje stavlja i drži ključ nad resursom, blokiranje je kada jedan proces čeka drugi da oslobodi zaključane resurse.
Lost updates (izgubljena ažuriranja) **-** Kada 2 ili više transakcija čitaju isti red i ažuriraju ga, posl. ažu. pobeđuje.
Uncommitted dependency (dirty read) - Kada druga tran. bira red koji se tren. ažurira od strane prve tran.
Inconsistent analysis (non-repeatable read) - Kada druga tran. pristupa istom redu više puta i ima razl. rezultate.
Phantom reads - Dešava se kada se ubacuju ili brišu redove koje čita transakcija

Problem / Izolacioni nivo	Dirty Read	Nonrepeatable Read	Phantom Read
Read uncommitted	+	+	+
Read committed (default)	-	+	+
Repeatable read	-	-	+
Snapshot	-	-	-
Serializable	-	-	-

Tip	Opis
Shared (S)	Koristi se za operacije čitanja.
Update (U)	Koristi se za zaključavanje resursa koji mogu biti promenjeni.
Exclusive (X)	Koristi se za operacije modifikovanja, kao što su INSERT, UPDATE, DELETE.
Intent	Koristi se za uspostavljanje hijerarhije zaključavanja.
Schema	Koristi se kada se izvršava operacija zavisna od šeme tabele.
Bulk Update (BU)	Koristi se kod masovnog kopiranja podataka u tabelu i kada je TABLOCK hint definisan.
Key-range	Štiti skup redova koje čita upit, kada se koristi SERIALIZABLE izolacioni nivo.

Deadlock - dva ili više procesa permanentno blokiraju jedan drugog jer su zaključali resurse koji drugi proces pokušava da zaključa.

20 - Okidači

Okidači su specijalne uskladištene procedure koje se izvršavaju kada se dese određeni događaji.
DML okidači se ispaljuju na INSERT, UPDATE, DELETE naredbe.

- **AFTER (FOR) okidači** - ispaljuju se posle izvršenja naredbi na koje se odnose, čine deo trans.
- **INSTEAD OF okidači** - dozvoljavaju izvršavanje alternativnog koda.

DDL okidači se ispaljuju na CREATE, ALTER, DROP naredbe.
INSERTED i **DELETED** virtuelne tabele omogućavaju pristup podacima u stanju kakvi su bili pre ili posle promene.

Naredba	inserted	deleted
INSERT	Redovi koji su upravo ubačeni	
DELETE		Redovi koji su upravo obrisani
UPDATE	Sadržaj vrednosti u promenjenim redovima	Sadrži originalne vrednosti iz promenjenih redova

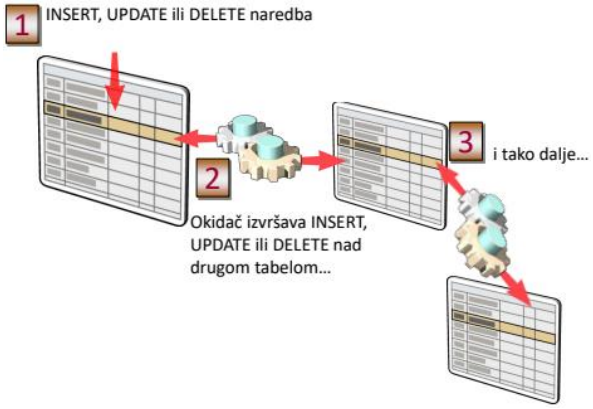
SET NOCOUNT ON - Okidači ne treba da vraćaju redove podataka.

```
CREATE TRIGGER TR_Opportunity_Insert
ON Sales.Opportunity
AFTER INSERT AS
BEGIN
  SET NOCOUNT ON;
  INSERT INTO Sales.OpportunityAudit
      (OpportunityID, ActionPerformed, ActionOccurredAt)
      SELECT i.OpportunityID,
             'I',
             SYSDATETIME()
      FROM inserted AS i;
END;
```

```
CREATE TRIGGER TR_Category_Delete
ON Product.Category
AFTER DELETE AS
BEGIN
    SET NOCOUNT ON;
    UPDATE p SET p.Discontinued = 1
    FROM Product.Product AS p
    WHERE EXISTS (SELECT 1 FROM deleted AS d
          WHERE p.CategoryID = d.CategoryID);
END;
GO
```

```
CREATE TRIGGER TR_ProductReview_Update
ON Product.ProductReview
AFTER UPDATE AS
BEGIN
    SET NOCOUNT ON;
    UPDATE pr
    SET pr.ModifiedDate = SYSDATETIME()
    FROM Product.ProductReview AS pr
    INNER JOIN inserted AS i
    ON i.ProductReviewID = pr.ProductReviewID;
END;
```

```
CREATE TRIGGER TR_ProductReview_Delete
ON Product.ProductReview
INSTEAD OF DELETE AS
BEGIN
  SET NOCOUNT ON;
  UPDATE pr SET pr.Discontinued = 1
  FROM Product.ProductReview AS pr
  INNER JOIN deleted as d
  ON pr.ProductReviewID = d.ProductReviewID;
END;
```



21 - Sigurnost

Principali - osnovne jedinice provere autentičnosti
Securables - resursi na SQL serveru kojima može biti dozvoljen ili zabranjen pristup
Autentifikacija - proces provere identiteta
Autorizacija - davanje dozvola/zabrana autentifikovanim korisnicima da pristupaju ili menjaju objekte.
GRANT, DENY i REVOKE.

```
USE AdventureWorks;
GO
GRANT CREATE TABLE TO HRManager;
GO
GRANT VIEW DEFINITION TO James;
GO
```

```
USE MarketDev;
GO

GRANT SELECT ON OBJECT::Marketing.Salesperson
  TO HRApp;
GO

GRANT SELECT ON Marketing.Salesperson
  TO HRApp;
GO
```