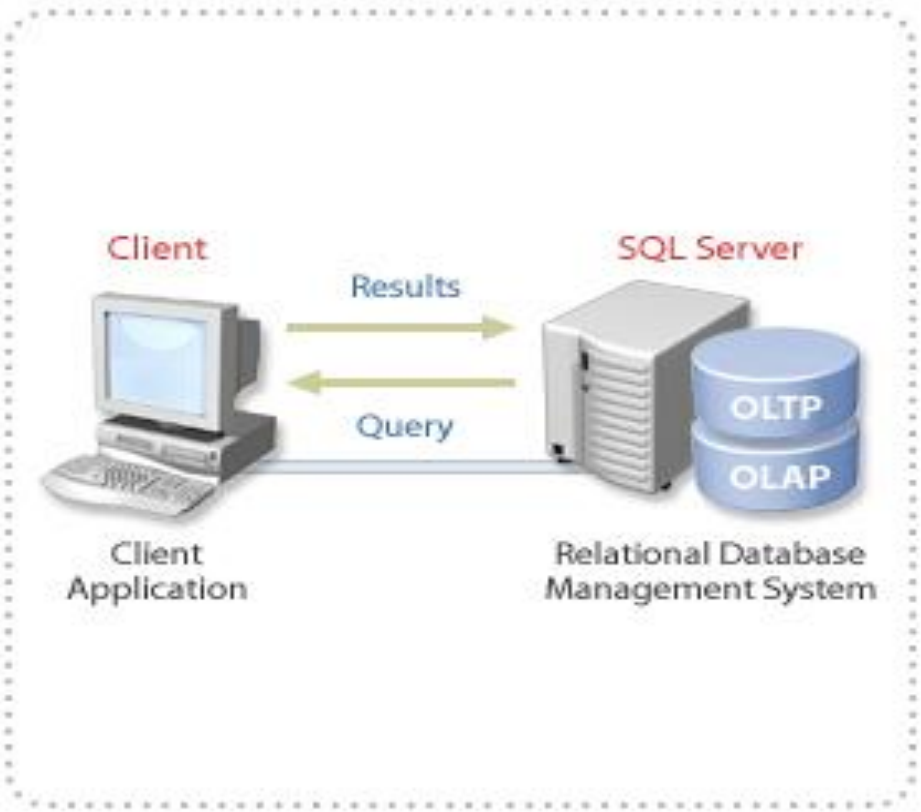
MS SQL Server

по Microsoft

MS SQL SERVER



Семейство от продукти и технологии

Средство за OLTP и OLAP

RDBMS за:

- управлява съхраняването на данни за транзакции и анализ
- отговаря на заявки от клиента
- използва Transact-SQL, Extensible Markup Language (XML), multidimensional expressions (MDX), или SQL Distributed Management Objects (SQL-DMO) за изпращане на заявки между клиента иSQL Server

Data Storage Models

SQL Server управлява OLTP и OLAP бази от данни

OLTP Databases - релационни бази данни и транзакции за много потребители

OLAP Databases - големи организации от данни, с бърз достъп за анализ в реално време

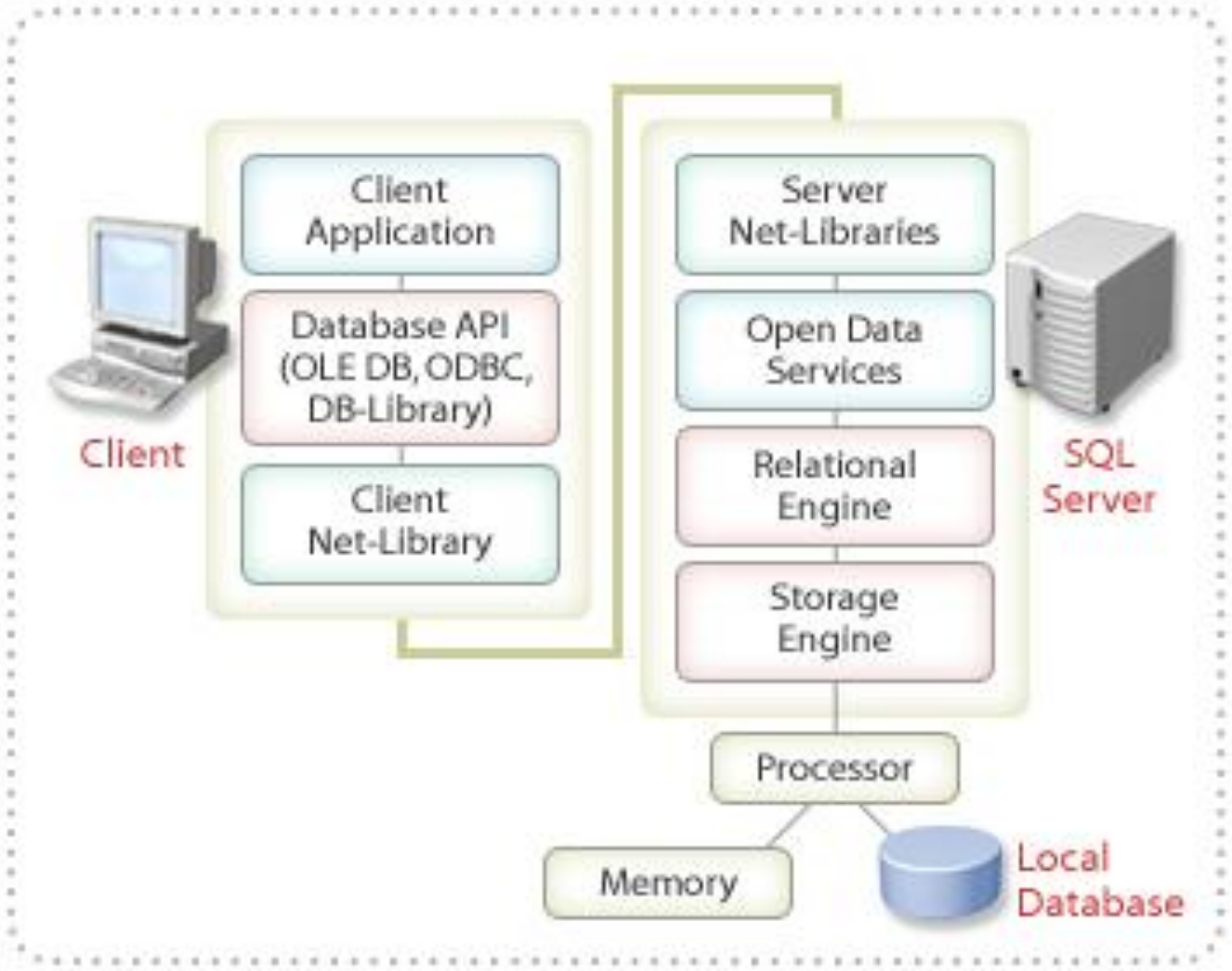
Client Applications

Приложни програми на клиента, посредством които се достига до данните.

Тези програми осъществяват достъп до SQL Server чрез:

- **Transact-SQL**
- **XML**
- **MDX** - за многодименсионни данни в OLAP
- **OLE DB и ODBC APIs**
- **ActiveX Data Objects and ActiveX Data Objects (Multidimensional)** - за използване на OLE DB с Microsoft Visual Basic®, Visual Basic for Applications, Active Server Pages и Microsoft Internet Explorer Visual Basic Scripting.

Client-Server Architecture



- Клиентите са отговорни за
  - бизнес логиката
  - за представяне на информацията на потребителите
- Сървърът управлява
  - базите данни
  - ресурсите на компютъра и мрежата
  - обслужването на заявките

Клиентът и сървърът комуникират през мрежа:

Клиентът изпраща заявка, като се ползва от API, през драйвер или DLL, които капсулират заявката в един или повече пакети Tabular Data Stream (TDS). Пакетите се предават в Net-Library.

В Net-Library TDS се трансформират в мрежови и се предават в Net-Library на сървера и към модула Open Data Services.

Open Data Services извлича заявките от TDS и ги изпраща на модула Relationali Engine, който ги компилира, оптимизира и изпълнява, като използва OLE DB interface.

Модулът Storage Engine извлича и предава данните *rowsets* от базата към буфери и тобратно към Relational Engine, който ги комбинира и изпраща в *result set* към Open Data Services.

Open Data Services пакетира и връща резултата, който може да се преобразува в XML формат.

SQL Server Services

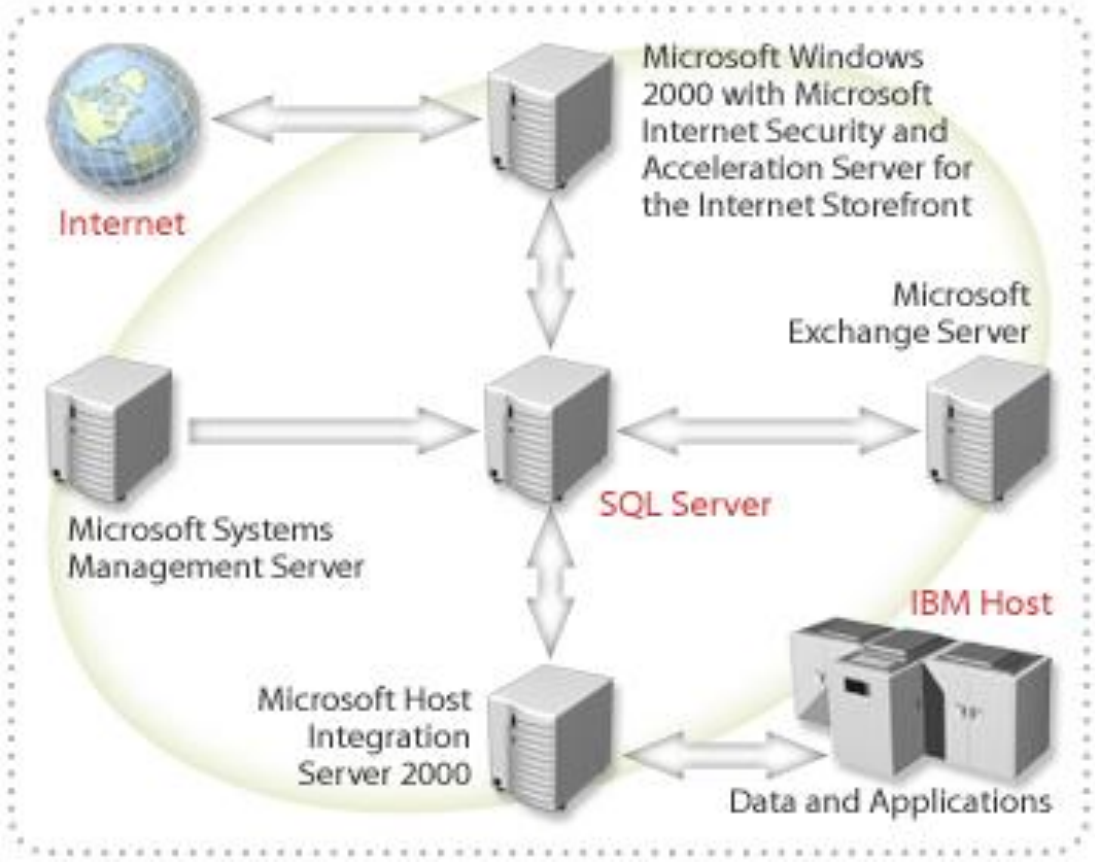
- MSSQLServer - database engine
  - обработва Transact-SQL
  - разпределя ресурсите между клиентите
  - управлява опашките
  - осигурява интегритет на данните
- SQLServerAgent
  - доставя информация за статуса на процесите
  - създава график и автоматизира управлението на задачите
  - при необходимост изпраща e-mail или други съобщения за алармиране
- Microsoft Distributed Transaction Coordinator (MS DTC)
  - позволява включването на различни източници на данни в една заявка
  - координира изпълнението на разпределени транзакции
- Microsoft Search

Екземпляри на SQL Server

- Работят на един и същ компютър със собствени бази данни, като да са на различни машини.
- Един се подразбира, останалите се задават с име:

*computer_name**instance_name*

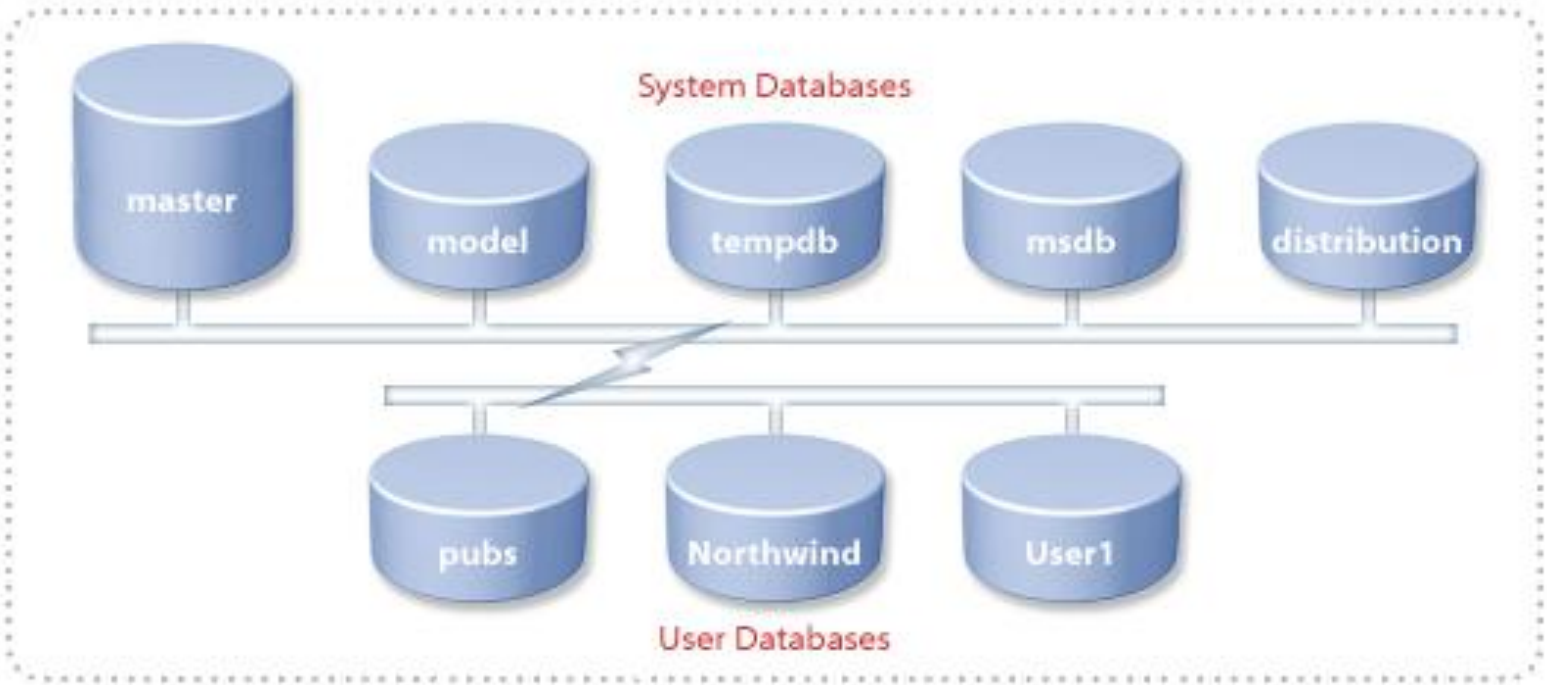
Интегриране с други MS приложения



Типове бази от данни

Два типа:

- system databases - съхраняват данни за SQL Server като система
- user databases - създадени от потребители



Системни бази

master - Метабаза - пази данни за user accounts, configurable environment variables, system error messages

model - Прототип за нови бази

tempdb - Буферна памет за временни обекти

msdb - Буферна памет за история - графици и задачи

distribution - Пази историята и данните в транзакциите, коио се репликират

pubs - Учебно средство

Northwind - Учебно средство

Adventure Works - Учебно средство

Обекти в база от данни

Table - Множество от редове с асоциирани колони

Data type - Множество на допустимите стойности и операции - системни и дефинирани от програмиста

Constraint - Правила за позволените стойности и механизми за поддържане на интегритета

Default - Стойност, попълваща колоната, ако не е зададена друга стойност

Rule - Дефинира валидни стойности, типове и отношения

Index - механизъм за бързо намиране на данни и поддържане на интегритета

View - Начин на визуализация на данни от една или повече таблици

User-defined function - Капсулира често употребявани логически операции. Връща точно един резултат - скалар или таблица. Използва се чрез повикване.

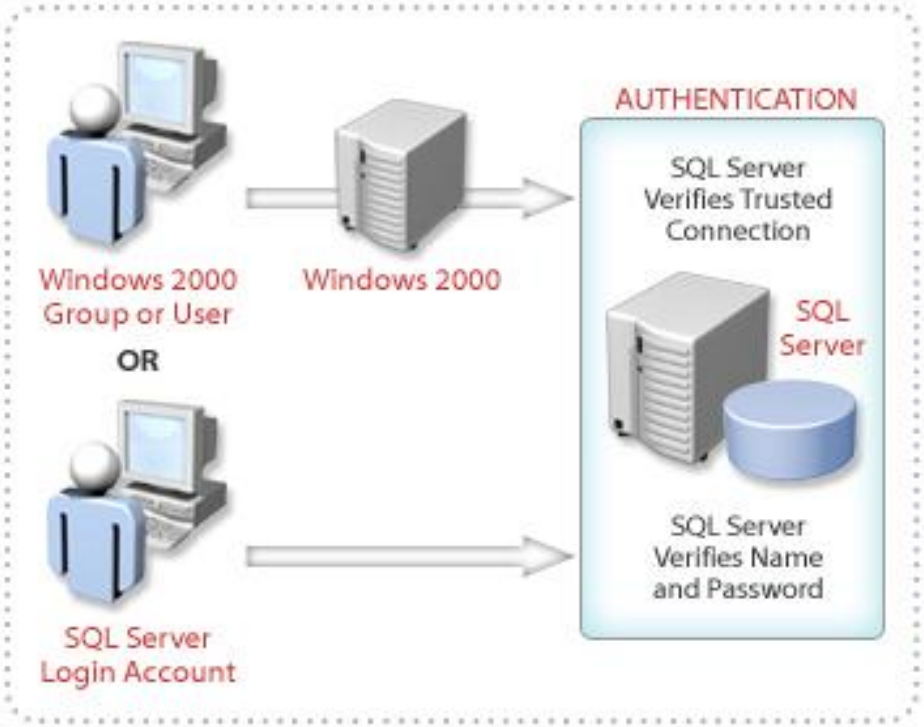
Stored procedure - Наименована колекция от Transact-SQL команди, които се изпълняват заедно.

Trigger - Вид процедура, която се изпълнява автоматично, когато потребителят промени данни в таблица или изглед.

Обръщението към обектите се осъществява посредством уникални имена - **Fully Qualified Names**

server.database.owner.object

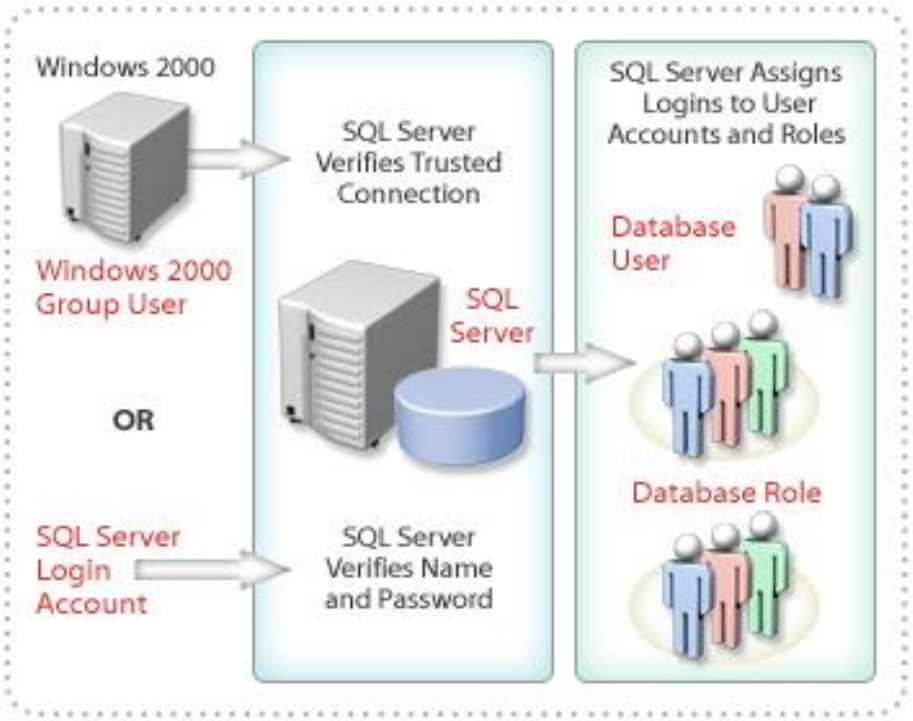
LOGIN



Два вида:

- Windows Authentication
- SQL Server Authentication

Потребителски достъп



Всеки потребител има специфични за определена база от данни права за достъп и роля - **Database User Account.**

Ролята определя групи от потребители с еднакъв достъп.

SQL Server предоставя пре-дефинирани роли:

- fixed server roles
- fixed database roles
- user-defined database roles

Един потребител може да има много роли.

Fixed Database Roles

public - Maintain all default permissions for users in a database

db_owner - Perform any database role activity

db_accessadmin - Add or remove database users, groups, and roles

db_ddladmin - Add, modify, or drop database objects

db_securityadmin - Assign statement and object permissions

db_backupoperator - Back up databases

db_datareader - Read data from any table

db_datawriter - Add, change, or delete data from all tables

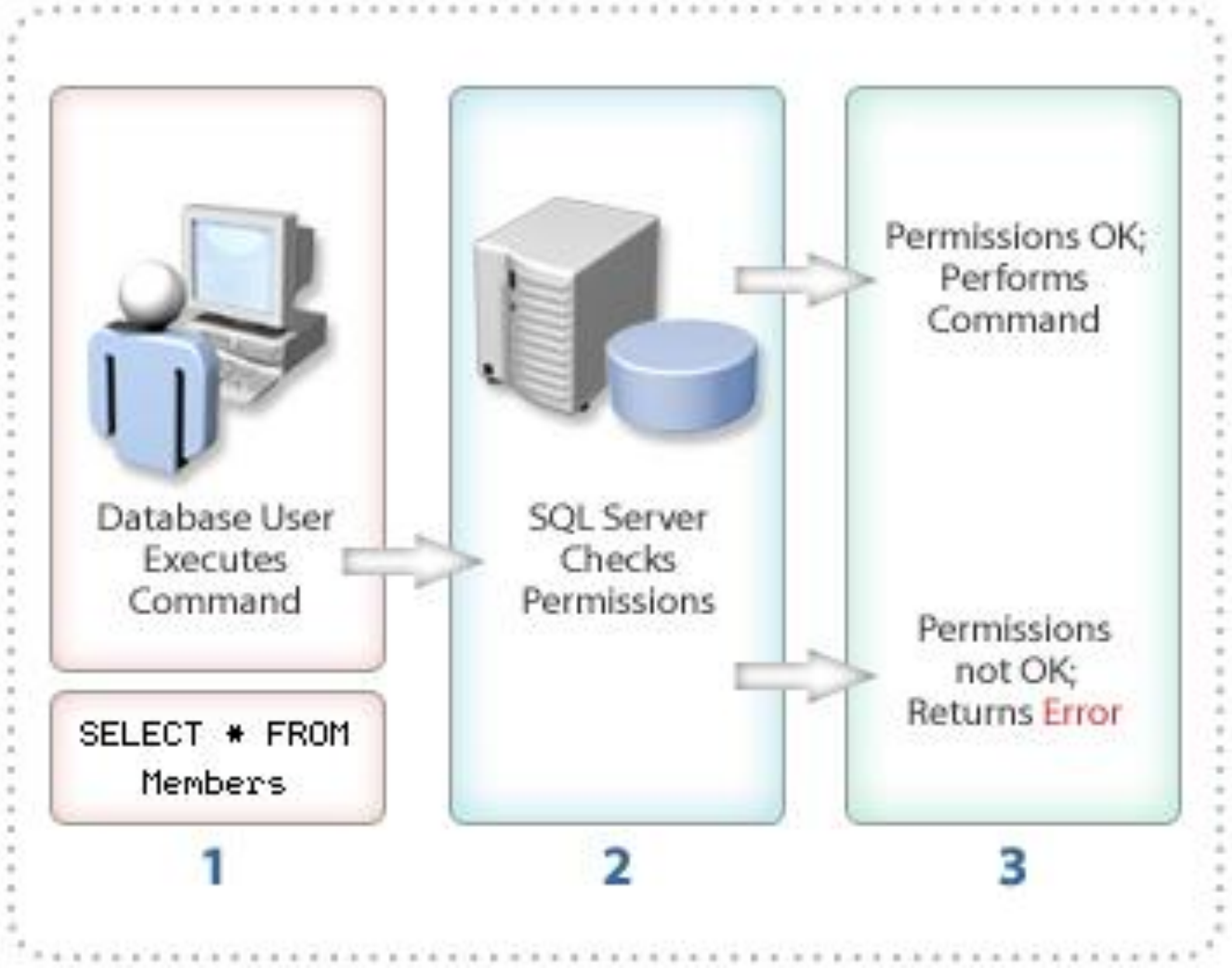
db_denydatareader - Cannot read data from any table

db_denydatawriter - Cannot change data in any table

User-defined Database Roles

Задават се и се променят по програмен път.

Валидиране на достъпа

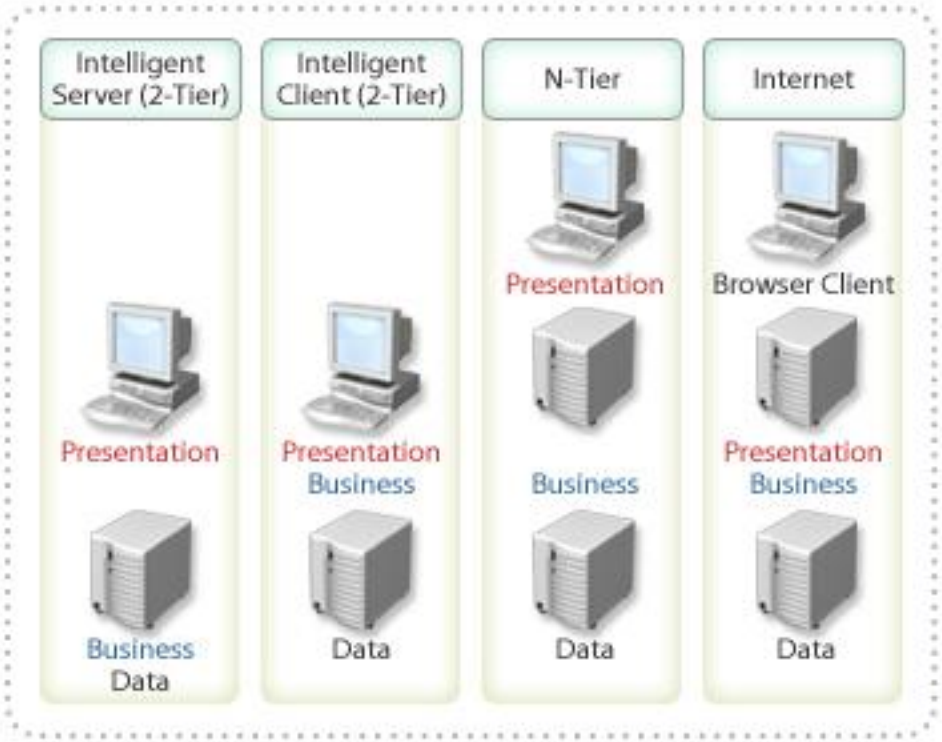


Администриране

Задачи на администрирането

- Инсталиране, конфигуриране и защита на сървера
- Изграждане на база - резервиране на пространство, дефиниране на задачи по поддържането
- Управление на дейности по инициализация на данните

Архитектури на приложения със SQL Server



Представяне

Логиката на представяне на данните и програмите на потребителите

Модулът се намира на клиентския компютър

Бизнес логика

Бизнес правила

Модулът може да бъде и на сървера

Данни

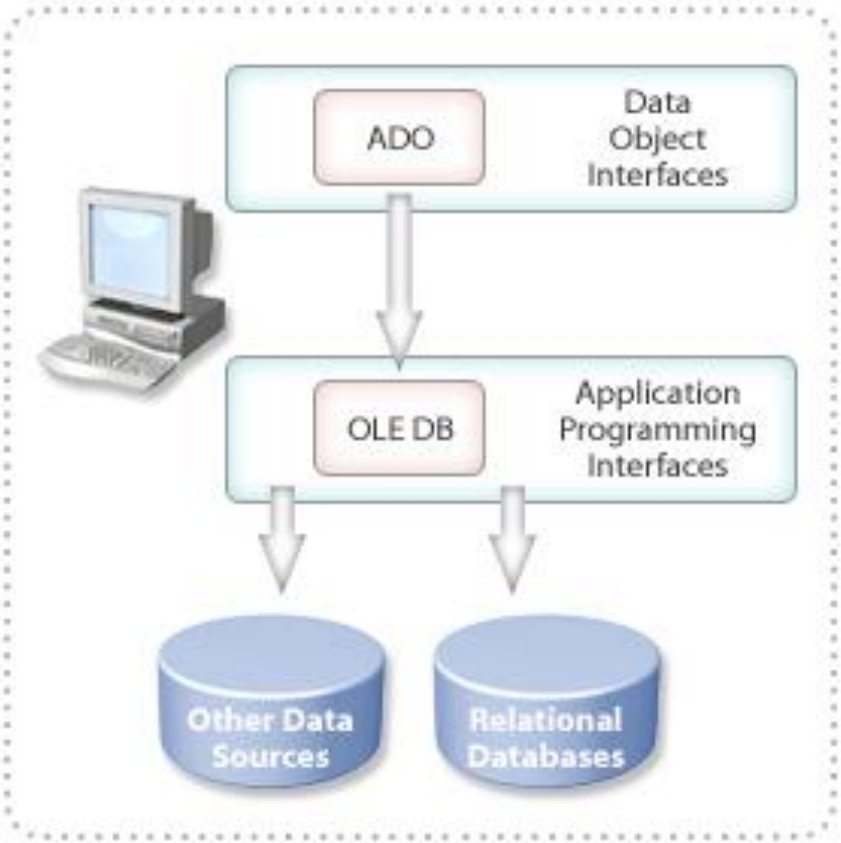
Дефиниция на базата

Правила за интегритет

Операциите с данните от съхранените процедури, тригерите и т.н.

Модул на сървера

Приложения



Две части на database API:

- Transact-SQL language statements
- Функции за изпращане на Transact-SQL към базата

OLE DB - Component Object Model (COM)-based API – библиотека от COM интерфейси за достъп до разнородни ресурси от данни

ADO – интерфейс на приложно ниво - OLE DB. Този API дефинира как приложението се свързва с базата чрезOLE DB и как се изпращат Transact-SQL команди към нея.

Програмиране

SQL Query Analyzer

- За писане, промяна и съхранение на Transact-SQL код.
- Показва едновременно командите и резултатите.
- Показва план на изпълнението на командите от SQL Server.

Data Control Language (DCL)

- За определяне на права за достъп

GRANT – задава права на потребител за достъп до данни или за изпълнение на операции

DENY – отказва права, забранява

REVOKE – анулира дадени или отказани права

Само членове на **sysadmin**, **dbcreator**, **db_owner** или **db_securityadmin** могат да изпълняват DCL.

Пример

```
USE Northwind
GRANT SELECT ON Products TO public
```

Имена на обекти

Standard Identifiers

- one to 128 characters, including letters, symbols (_, @, or #), and numbers.
- no embedded spaces are allowed in standard identifiers
- правила

- първият символ: a-z или A-Z.
- следващите: letters, numerals, или @, $, #, _
- специални идентификатори
  - (@) – локална променлива или параметър;
  - (#) временна таблица или процедура;
  - (##) глобален временен обект

Delimited Identifiers

- Ако съдържа интервали
- Когато се получават думи със значение на имена
  - Bracketed identifiers are delimited by square brackets ([ ]):

```
SELECT * FROM [Blanks In Table Name]
```

  - Quoted identifiers are delimited by quotation marks ("")

```
SELECT * FROM "Blanks in Table Name"
```

Локални променливи

DECLARE {@local_variable data_type} *[,...n]*

SET *@local_variable_name = expression*

Пример:

```
DECLARE
    @vLastName char(20),
    @vFirstName varchar(11)
SET @vLastName = 'Dodsworth'
SELECT @vFirstName = FirstName
    FROM Northwind..Employees
    WHERE LastName = @vLastName
PRINT @vFirstName + ' ' + @vLastName
GO
```

Оператори

{*constant* | *column_name* | *function* | *(subquery)*}

[{*arithmetic_operator* | *string_operator* |

AND | OR | NOT}

{*constant* | *column_name* | *function* | *(subquery)*}…]

Типове

- **аритметични**
  - **оператори -** (+),(-),(*),(/), (%)
  - **функции -** 3 вида:
    - aggregate - от група стойности се връща една

      SELECT AVG(UnitPrice) FROM Products
    - scalar - от една стойност получава друга

      SELECT DB_NAME() AS 'database'
    - rowset - връща група данни
- **логически**
- **стрингови**

Управление на операциите

BEGIN…END Blocks

IF…ELSE Blocks

WHILE Constructs

Пример:

```
USE Northwind
IF EXISTS (SELECT OrderID FROM OrdersWHERE CustomerID = 'Frank')
   PRINT '*** Customer cannot be deleted ***'
ELSE
   BEGIN
      DELETE Customers WHERE CustomerID = 'Frank'
      PRINT '*** Customer deleted ***'
   END
```

Пример:

```
SELECT ProductID, 'Product Inventory Status' =
  CASE
  WHEN (UnitsInStock < UnitsOnOrder AND Discontinued = 0)
    THEN 'Negative Inventory - Order Now!'
  WHEN ((UnitsInStock-UnitsOnOrder) < ReorderLevel AND
      Discontinued = 0)
   THEN 'Reorder level reached- Place Order'
  WHEN (Discontinued = 1) THEN '***Discontinued***'
  ELSE 'In Stock'
   END
FROM Northwind..Products
```

Изпълнение на команди

- Динамично изпълнени команди
- Динамично съставени команди
- Пакети от команди
- Скриптове
- Транзакции
- XML

Динамични операции

EXECUTE ({*@str_var | 'tsql_string'*} + [{*@str_var | 'tsql_string'*}...])}

- Изпълнява пакети от команди на Transact-SQL
- Използва литерали и локални променливи

Динамично съставени команди

```
DECLARE @dbname varchar(30), @tablename varchar(30)
SET @dbname = 'Northwind'
SET @tablename = 'Products'
EXECUTE     ('USE ' + @dbname +  ' SELECT ProductName FROM ' + @tablename)
```

Пакети - една или повече команди, представени едновременно, изпълняват се с GO

- за разлика от транзакциите, GO не е свързан с възстановяване
- потребителските променливи имат локално значение за пакета и не са валидни след GO

Примери:

```
CREATE DATABASE ...
CREATE TABLE ...
GO

CREATE VIEW1 ...
GO
CREATE VIEW2 ...
GO
```

```
CREATE DATABASE ...
CREATE TABLE ...
GO

CREATE TRIGGER ...
GO

CREATE TRIGGER ...
GO
```

Скрипт

Transact-SQL statements, записани във файл с разширение на името .sql.

Изпълняват се в друга среда.

Транзакции

BEGIN TRANSACTION
COMMIT TRANSACTION или
ROLLBACK TRANSACTION

Пример:

```
BEGIN TRANSACTION

UPDATE savings
  SET balance = (amount - 100)   WHERE custid = 78910
IF @@ERROR <> 0
  BEGIN
    RAISERROR ('Transaction not completed due to savings account problem.', 16, -1)
    ROLLBACK TRANSACTION
  END

UPDATE checking
  SET balance = (amount + 100)  WHERE custid = 78910
IF @@ERROR <> 0
  BEGIN
    RAISERROR ('Transaction not completed due to  checking account problem.', 16, -1)
    ROLLBACK TRANSACTION
  END


COMMIT TRANSACTION
```

XML

Език за програмиране, чрез който се представят на Web страници данни от SQL Server database.

При използване на FOR XML в SELECTt, SQL Server:

- Връща резултата като низ
- Връща атрибутите на данните като tags

Всяка таблица от FROM clause в SELECT clause е представена като XML element: данни и атрибути.

Пример:

три колони от две таблици в един низ

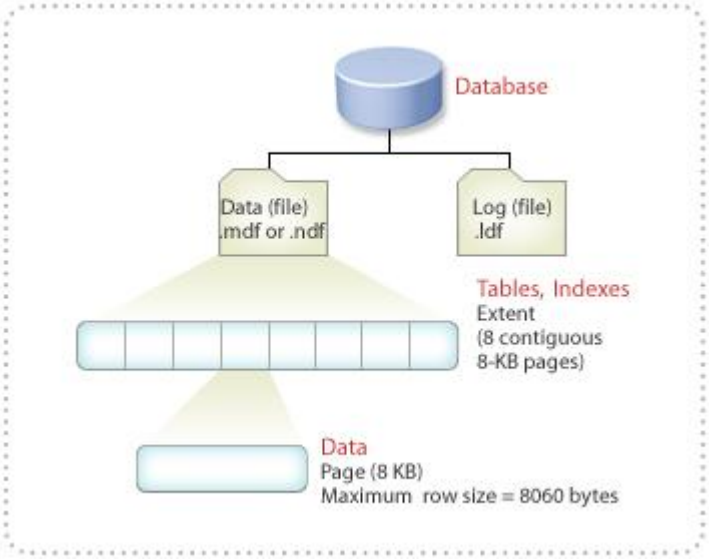
```
SELECT Orders.OrderID, Shippers.CompanyName, Orders.CustomerID
FROM Orders JOIN Shippers
ON Orders.shipvia = Shippers.ShipperID
WHERE OrderID < 10250
FOR XML AUTO
```

Result

XML_F52E2B61-18A1-11d1-B105-00805F49916B

```
<Orders OrderID="10248" CustomerID="VINET">
  <Shippers CompanyName="Federal Shipping"/>
</Orders>
<Orders OrderID="10249" CustomerID="TOMSP">
  <Shippers CompanyName="Speedy Express"/>
</Orders>
```

Представяне на данните на физическо ниво



- Всички бази се записват в един основен файл .mdf и един или повече транзакционни .ldf.
- Всички бази съдържат копие на **model**.
- SQL Server записва, чете, съхранява данни в блокове по 8-KB наречени *pages*. (128 pages per megabyte).
- Max количество данни в един ред - 8060 bytes.
- Всички страници са записани в томове по 8 - *extents* от 64 KB. (16 extents per megabyte).
- Цялата информация, необходима за възстановяване на базата се намира в .ldf. By default, размерът му е 25 % от размера на данните.

Правила за избор на тип данни

- Ако ширината на колоната варира (напр. за списък от имена) - **varchar** вместо **char** (fixed).
- Внимавайте с **tinyint** при разширяващ се бизнес.
- За числа с променлива точност - **decimal**.
- За обекти, по-големи от 8000 B - **text** или **image**; за по-малки - **binary** или **char; когато е възможно -** use **varchar** .
- Използвайте **money** за валути.
- Не използвайте **float** и **real** за първични ключове, защото стойностите им на са точни и не могат да се използват за сравнение.

Потребителски тип данни

- Създаване на чрез системната процедура **sp_addtype**.

  sp_addtype {*type*}, [*system_data_type*] [, ['NULL' | 'NOT NULL']] [, '*owner_name*']

Примери:

```
EXEC sp_addtype city, 'nvarchar(15)', NULL
EXEC sp_addtype region, 'nvarchar(15)', NULL
EXEC sp_addtype country, 'nvarchar(15)', NULL
```

- Изтриване

Чрез **sp_droptype** се изтриват от **systypes** system table, ако не се използват от други обекти.

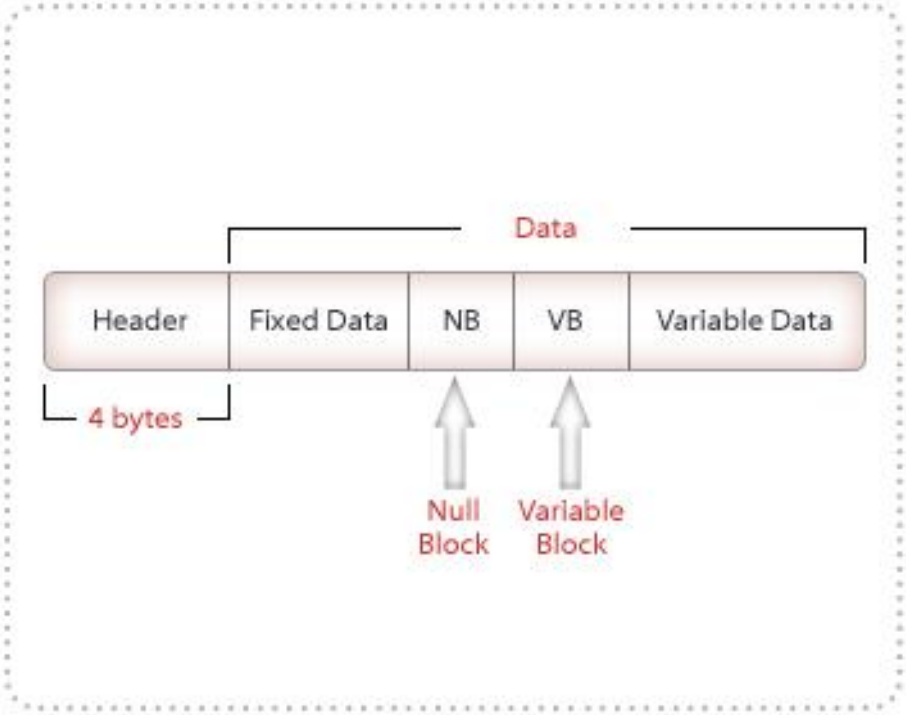
sp_droptype {'*type*'}

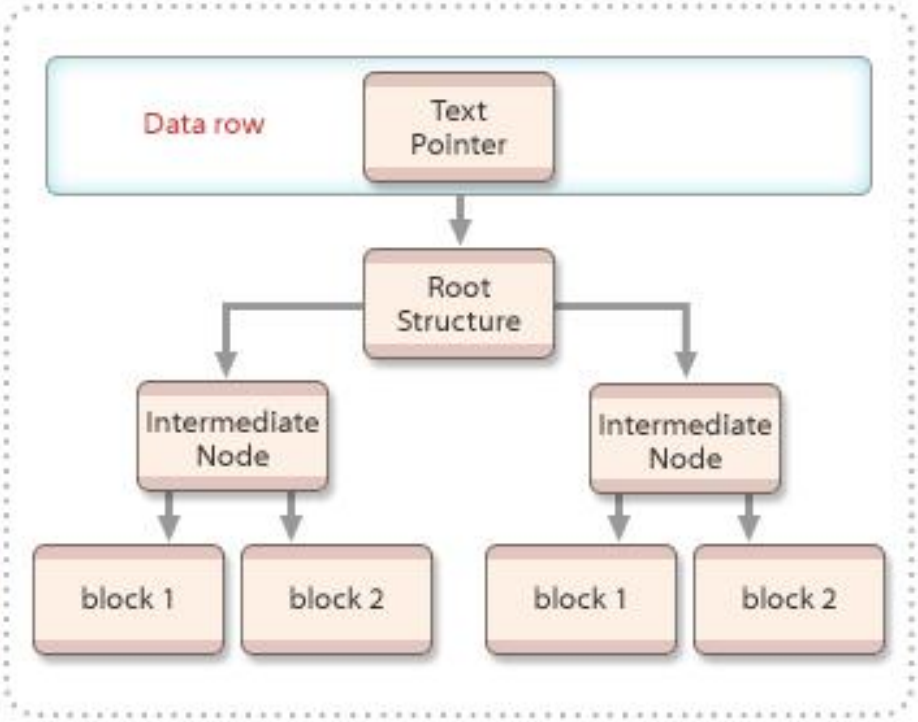
Примери:

```
EXEC sp_droptype city
```

Вътрешна организация на данните

Един запис се състои от row header и data portion.





Row Header

4-byte row header – съдържа информация за колоните в реда данни (напр. указател за край на fixed-data portion of the row, и дали variable-length columns съществуват в реда.

Типовете с променлива дължина (напр. изображения) се съхраняват като колекция от страници.

- **text** data type, which can hold 2,147,483,647 characters. The non-Unicode **text** data type cannot be used for variables or parameters in stored procedures.

- **ntext** data type, which can hold a maximum of $2^{30} - 1$ (1,073,741,823) characters or $2^{31} - 1$ bytes, which is 2,147,483,647 bytes of variable-length Unicode data. The SQL-92 synonym for **ntext** is national text.

Когато **text**, **ntext**, и **image** data types са големи, SQL Server ги запомня извън редовете. 16-byte указател сочи към структурата, в която се помнят.

Пример:

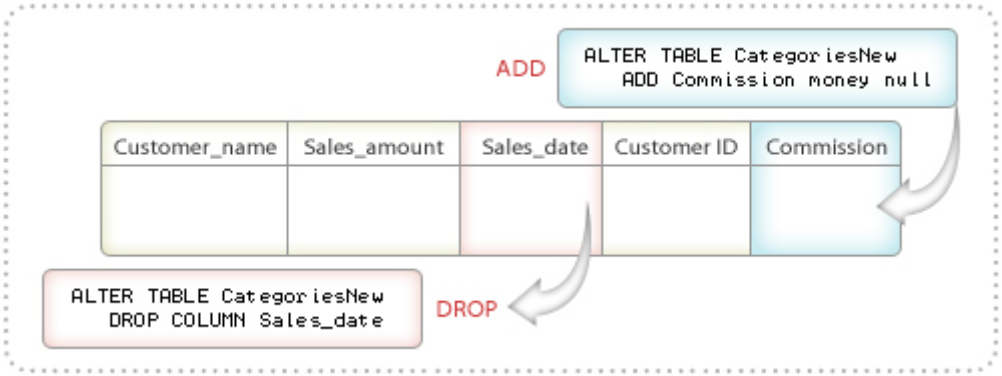
```
EXEC sp_tableoption N'Employees', 'text in row', '1000'
```

Creating a Table

- Ограничения:
  - Two billion tables per database.
  - 1,024 columns per table.
  - 8060 bytes per row (this approximate maximum length does not apply to **image**, **text**, and **ntext** data types).
- Изтриване на таблица - ако не е свързана с други обекти (което може да се провери чрез , **sp_depends.**

  DROP TABLE *table_name* [,…*n*]
- Изменения в колните на таблица – триене или добавяне

  ALTER TABLE table
  {| [ALTER COLUMN *column_name*]
  |{ ADD
  { <*column_definition*> ::= *column_name data_type*
  { [NULL | NOT NULL]
  | DROP column *column_name*} [,…*n*]



- Създаване на колона **Identity**

  CREATE TABLE *table*
  (*column_name data_type* [IDENTITY [(*seed*, *increment*)]] NOT NULL)

ограничения:

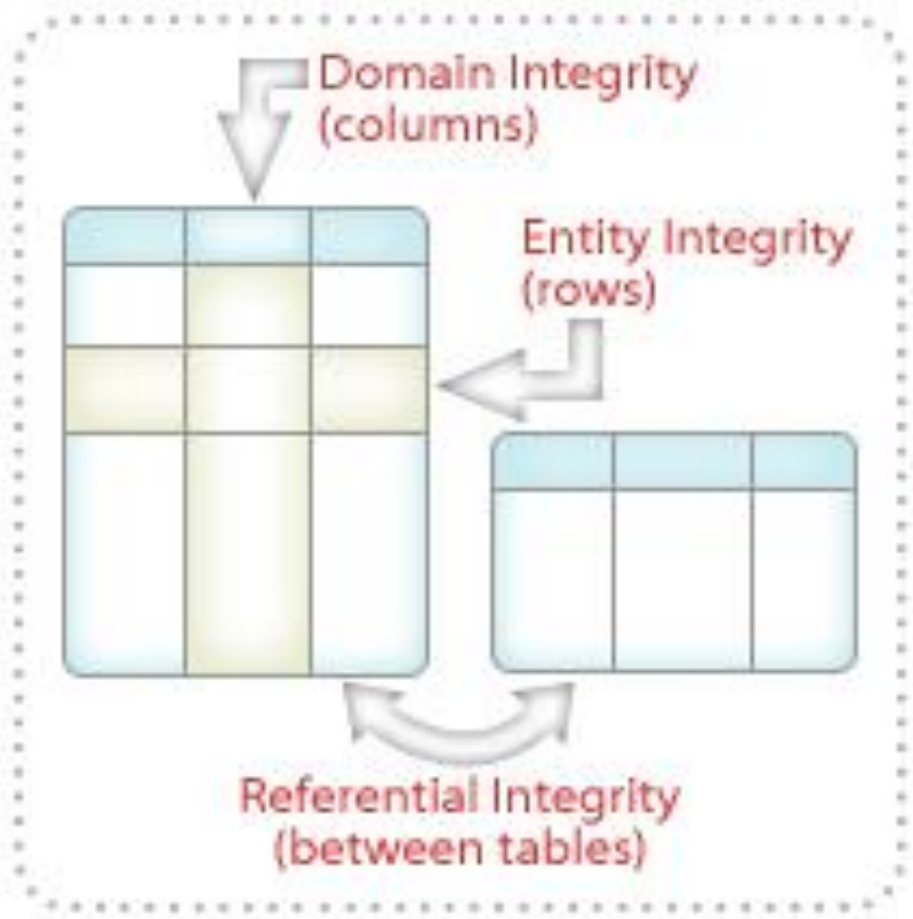
- само една колона за таблица;
- само с **integer** (**int**, **bigint**, **smallint**, или **tinyint**), **numeric**, или **decimal**;
- не може да се актуализира;
- може да се използва под името IDENTITYCOL ;
- не позволява нулеви стойности.

Пример:

CREATE TABLE Class
(StudentID int IDENTITY(100, 5) NOT NULL, Name varchar(16))

Интегритет

Адекватност и точност на данните в базата.



Domain Integrity

Domain (or column) integrity – дефинира валидните, за колона, стойности. Осигурява се чрез контрол за валидност по отношение на тип, формат, множество на допустимите стойности.

Entity Integrity

Entity (or table) integrity – изисква уникалност на редовете - *primary key value*.

Referential Integrity

Referential integrity – осигурява съответствие между свързаните таблици посредством отношението на primary keys (in the referenced table) и *foreign keys* (in the referencing tables).

Поддържане на интегритета

- **Declarative Data Integrity**
  - декларира се като част от дефиницията на базата директно върху таблиците или колоните
  - чрез constraints, defaults, и rules
- **Procedural Data Integrity**
  - прилага се върху клиента или сървера
  - чрез програмен код - triggers istored procedures.

Ограничения върху една или повече колони се задават чрез CREATE TABLE или ALTER TABLE

- constraint върху отделна колона - **column-level** constraint;
- върху повече колони - **table-level** constraint, дори да не е върху всички колони

```
CREATE TABLE table_name
  ( { < column_definition >
  | < table_constraint > } [ ,...n ])

< column_definition > ::= { column_name data_type }
  [ [ DEFAULT constant_expression ]
  [ < column_constraint > ] [ ,..n]

< column_constraint > ::=
 [ CONSTRAINT constraint_name ]
  | [ { PRIMARY KEY | UNIQUE }
    [ CLUSTERED | NONCLUSTERED ] ]

| [ [ FOREIGN KEY ]
    REFERENCES ref_table [ ( ref_column ) ]
    [ ON DELETE { CASCADE | NO ACTION } ]
    [ ON UPDATE { CASCADE | NO ACTION } ]]
  | CHECK ( logical_expression ) }
```

```
< table_constraint > ::=
  [ CONSTRAINT constraint_name ]
    { [ { PRIMARY KEY | UNIQUE }
      [ CLUSTERED | NONCLUSTERED ]
      { ( column [ ASC | DESC ] [ ,...n ] ) } ]
    | FOREIGN KEY
      [ ( column [ ,...n ] ) ]
      REFERENCES ref_table [ ( ref_column [ ,...n ] ) ]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
    | CHECK ( search_conditions ) }
```

Пример:

```
USE Northwind
CREATE TABLE dbo.Products
(
  ProductID int IDENTITY (1,1) NOT NULL,
  ProductName nvarchar (40) NOT NULL,
  SupplierID int NULL,
  CategoryID int NULL,
  QuantityPerUnit nvarchar (20) NULL,
  UnitPrice money NULL CONSTRAINT DF_Products_UnitPrice DEFAULT(0),
  UnitsInStock smallint NULL CONSTRAINT DF_Products_UnitsInStock DEFAULT(0),
  UnitsOnOrder smallint NULL CONSTRAINT DF_Products_UnitsOnOrder DEFAULT(0),
  ReorderLevel smallint NULL CONSTRAINT DF_Products_ReorderLevel DEFAULT(0),
  Discontinued bit NOT NULL CONSTRAINT DF_Products_Discontinued DEFAULT(0),

  CONSTRAINT PK_Products PRIMARY KEY CLUSTERED (ProductID),

  CONSTRAINT FK_Products_Categories FOREIGN KEY (CategoryID)
    REFERENCES dbo.Categories (CategoryID) ON UPDATE CASCADE,
  CONSTRAINT FK_Products_Suppliers FOREIGN KEY (SupplierID)
    REFERENCES dbo.Suppliers (SupplierID) ON DELETE CASCADE,

  CONSTRAINT CK_Products_UnitPrice CHECK (UnitPrice >= 0),
  CONSTRAINT CK_ReorderLevel CHECK (ReorderLevel >= 0),
  CONSTRAINT CK_UnitsInStock CHECK (UnitsInStock >= 0),
  CONSTRAINT CK_UnitsOnOrder CHECK (UnitsOnOrder >= 0)
)
GO
```

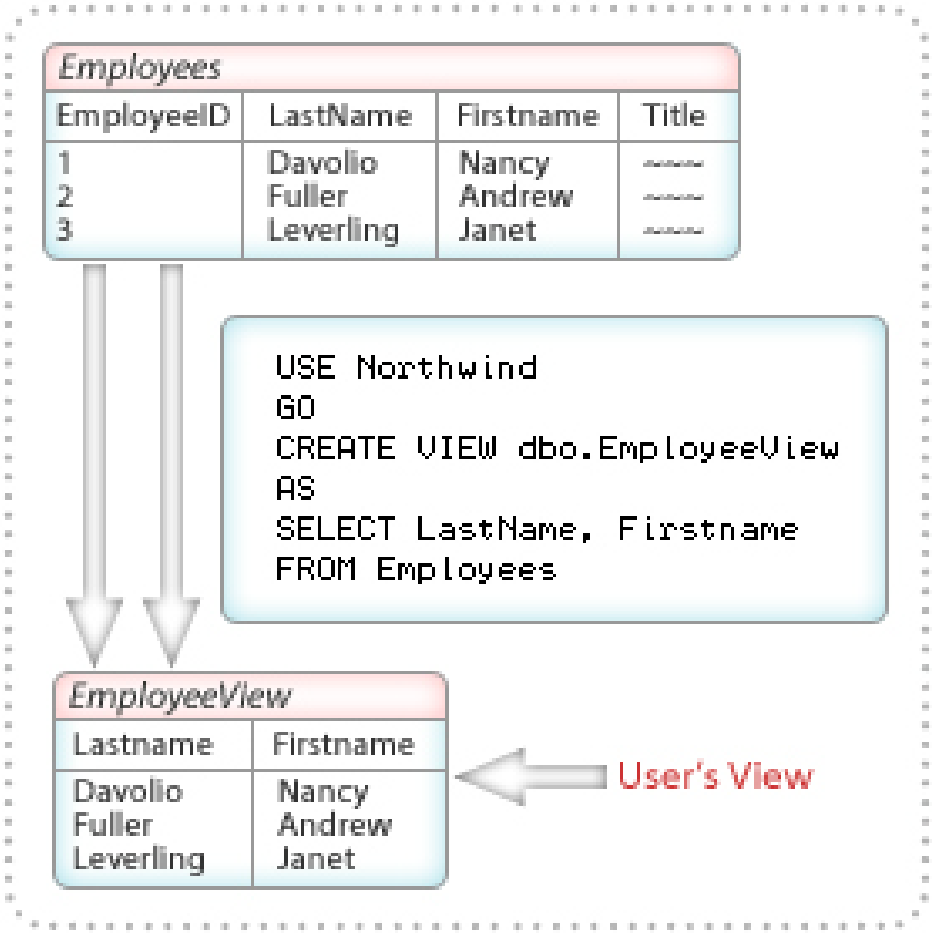
Видове ограничения

- DEFAULT constraints
- CHECK constraintss
- PRIMARY KEY constraints
- UNIQUE constraints
- FOREIGN KEY constraints
- Cascading referential integrity

Изгледи View

Съхраняване на предварително дефинирани обекти за бъдещо ползване. Почти всеки SELECT statement може да бъде запомнен като изглед.

Примери:



- A subset of rows or columns of a base table.
- A union of two or more base tables.
- A join of two or more base tables.
- A statistical summary of a base table.
- A subset of another view, or some combination of views and base tables.

```
USE Northwind
GO
CREATE VIEW dbo.EmployeeView
AS
SELECT LastName, Firstname
FROM Employees

SELECT * from EmployeeView
```

Предимства в използването на изгледи

- **Focus the Data for Users -** Контролирана среда до специфични данни
- **Mask Database Complexity –** Опростяване на модела на данните за потребителите
- **Simplify Management of User Permissions** – Контрол и ограничаване на достъпа на клиенти до изгледите, вместо до таблиците
- **Improve Performance** – Многократно използваеми заявки за информация
- **Organize Data for Export to Other Applications** – улесняване на обмена на данни между приложения

Пример:

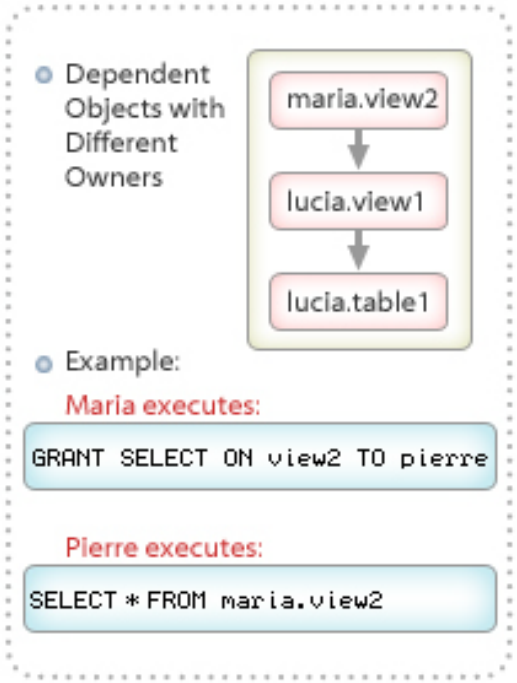
```
USE Northwind
GO
CREATE VIEW dbo.ShipStatusView
AS
SELECT OrderID, ShippedDate, ContactName
FROM Customers c INNER JOIN Orders o  ON c.CustomerID = O.CustomerID
WHERE RequiredDate < ShippedDate

SELECT * FROM ShipStatusView
```

Пример:

Maria creates **view2**. With the following statement, she grants permission to Pierre to query it.

GRANT select ON view2 TO pierre



maria.view2 зависи от (**view1**), притежавано от друг (Lucia). Проверяват се правата за достъп до всеки обект. Pierre се обръща към изгледа чрез:

SELECT * FROM maria.view2

Тъй като **maria.view2** зависи от **lucia.view1**, SQL Server проверява правата на **maria.view2** и **lucia.view1**. Ако Lucia е дала предварително права на Pierre за **view1**, Pierre има достъп. Ако не, достъпът е отказан, Lucia запазва контрола върху оторизираните потребители.