Paweł Rajba
pawel@cs.uni.wroc.pl
http://itcourses.eu/

Application Security

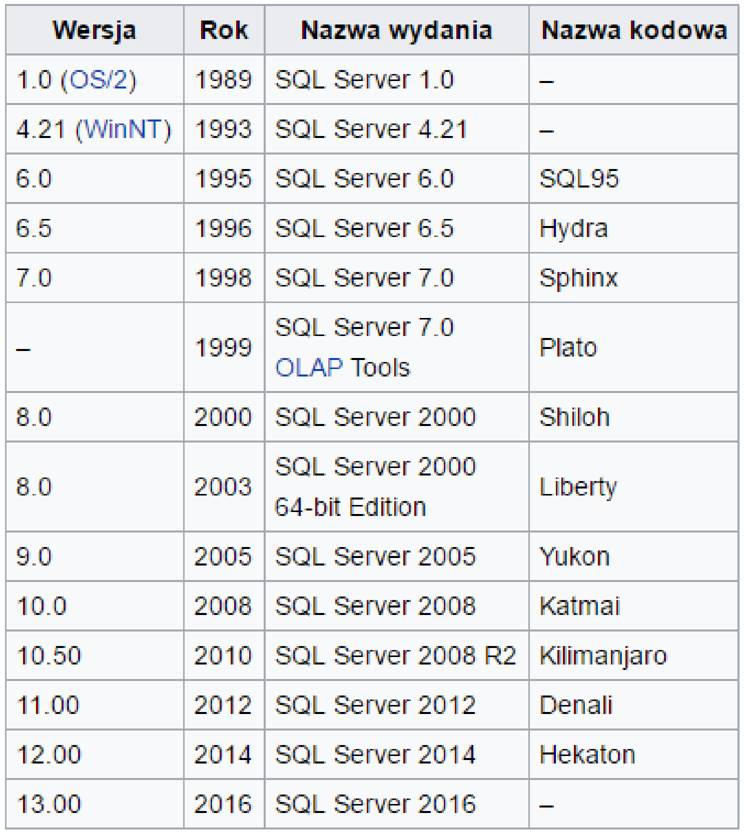
Database security

Agenda

- Introduction
- Data consistency
- Data access control
- Data encryption
  - In storage
  - In transit
- Data availability
  - Backup and restore strategy
  - Transaction log strategy
- Auditing
- Policies

Introduction

- SQL Server is a DBMS
- It evolved for many years and now is a mature product on the market
- One can install many instances on one server
  - Only one is the default
  - Others are named



Wersja	Rok	Nazwa wydania	Nazwa kodowa
1.0 (OS/2)	1989	SQL Server 1.0	–
4.21 (WinNT)	1993	SQL Server 4.21	–
6.0	1995	SQL Server 6.0	SQL95
6.5	1996	SQL Server 6.5	Hydra
7.0	1998	SQL Server 7.0	Sphinx
–	1999	SQL Server 7.0 OLAP Tools	Plato
8.0	2000	SQL Server 2000	Shiloh
8.0	2003	SQL Server 2000 64-bit Edition	Liberty
9.0	2005	SQL Server 2005	Yukon
10.0	2008	SQL Server 2008	Katmai
10.50	2010	SQL Server 2008 R2	Kilimanjaro
11.00	2012	SQL Server 2012	Denali
12.00	2014	SQL Server 2014	Hekaton
13.00	2016	SQL Server 2016	–

Introduction

- Main services for the default instance
  - MSSQLServer
  - SQLServerAgent
- … and for a named instance
  - MSSQLServer$instanceName
  - SQLServerAgent$instanceName
- Main tools
  - Microsoft SQL Server Management Studio
  - SQL Server Profiler

Introduction

- There 2 types of databases
  - System and user
- System databases
  - master
    - Information about databases, its file locations
    - Account information and other like endpoints, configuration, etc.
  - tempdb
    - Temporary workspace, used to processing queries, etc.
    - After restart restored on the basis on model database
  - msdb
    - SQLServerAgent service database
    - Includes information about job schedules, alerts, etc.
  - model
    - Database template

https://docs.microsoft.com/en-us/sql/relational-databases/databases/system-databases

Introduction

- Database files
  - Main files – *.mdf
  - Secondary – *.ndf
  - Transaction log – *.ldf
- Transaction log and recovery model
  - Full
  - Bulk-logged
    - Like full, but excluded bulk operations,
      e.g. bulk, select..into, create index, writetext, updatetext
  - Simple

Introduction

- Authentication modes
  - Windows
  - Mixed mode

Introduction

- We consider security in the following areas
  - Data consistency
  - Data access control
  - Data encryption
    - In storage
    - In transit
  - Data availability
    - Backup and restore strategy
    - Transaction log strategy
  - Auditing
  - Policies

Data consistency

- Secured by ACID property of transactions
- Right choice of isolation level
- Is that enough?
  - What about business rules?
  - Where are they implemented and where is the validation executed?
- Transactional vs. eventual consistency

Basic concepts

- Scopes
  - Server level
  - Database level
  - Schema level
- Principals
  - Entity who wants access to a resource
- Securables
  - Resources that can be requested by principles

Principals

- Server-level principles (logins)
  - SQL Server authentication Login
  - Server role
  - Windows authentication login for a Windows user
  - Windows authentication login for a Windows group
- Every login has a SID
- Can be created…
  - In MGMT studio (Security → Logins)
  - CREATE LOGIN statement
- Some options
  - MUST_CHANGE, DEFAULT_DATABASE = "…", CHECK_EXPIRATION = ON, CHECK_POLICY = ON

https://docs.microsoft.com/pl-pl/sql/relational-databases/security/authentication-access/principals-database-engine

Principals

- Database-level principals users
  - Database User (there are many types)
  - Database Role
  - Application Role
- Basic operations:
  - CREATE USER [TestUser] FOR LOGIN [CustomUser] WITH DEFAULT SCHEMA=[dbo]
    - After user is created there is no permission associated
  - ALTER USER [TestUser] WITH login = [NewLogin]
    - Useful when we attached database

https://docs.microsoft.com/pl-pl/sql/relational-databases/security/authentication-access/principals-database-engine

Principals

- Special principals
  - sa (login)
    - System administrator with full power
    - Member of the sysadmin fixed role
  - dbo (db user)
    - Stands for database owner
    - Alias to the database owner when connected
  - public (login, db user)
    - Assigned to every login (on server) and every user (on db)
    - Cannot be removed, but one can change the permissions
      - It is recommended to not add deny, because it affects all users
  - guest
    - Present in every database
    - Permissions granted to the guest user are inherited by users who have access to the database, but who do not have a user account in the database

https://docs.microsoft.com/pl-pl/sql/relational-databases/security/authentication-access/principals-database-engine

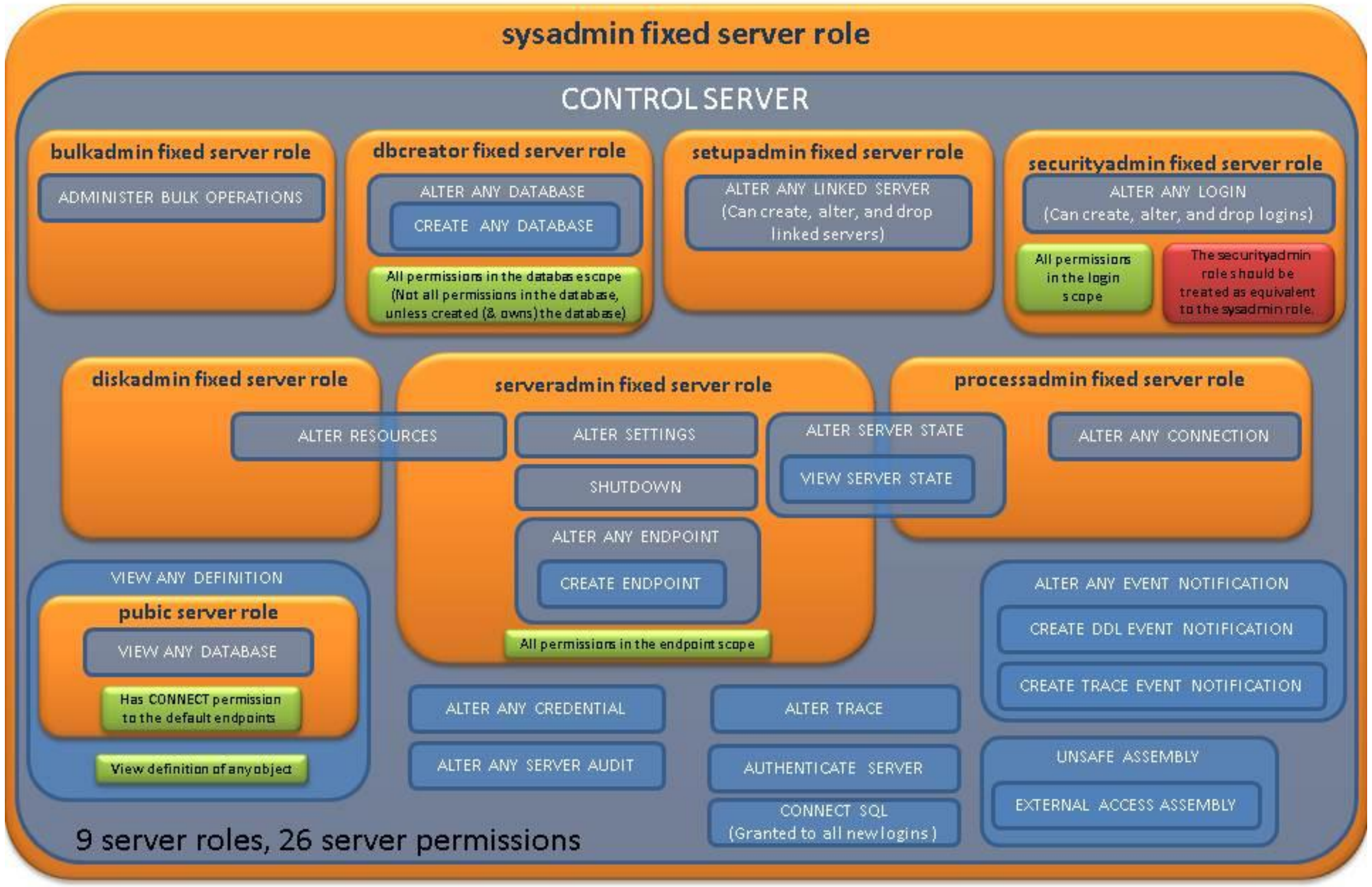
Principals

- Server-Roles
  - There are 2 types
    - 9 fixed server roles (builtin)
      - sysadmin, serveradmin, securityadmin, processadmin, setupadmin, bulkadmin, diskadmin, dbcreator, public
      - The permissions that are granted cannot be changed
    - Custom server roles
      - Basic operations:
        - CREATE SERVER ROLE [SomeRole]
          - It is possibility to create custom server roles
        - ALTER SERVER ROLE [sysadmin] ADD MEMBER [auser]
  - List of permissions for a role
    - sp_srvrolepermission 'securityadmin'

https://docs.microsoft.com/pl-pl/sql/relational-databases/security/authentication-access/server-level-roles

Principals

SERVER LEVEL ROLES AND PERMISSIONS


sysadmin fixed server role
CONTROL SERVER
bulkadmin fixed server role
ADMINISTER BULK OPERATIONS
dbcreator fixed server role
ALTER ANY DATABASE
CREATE ANY DATABASE
All permissions in the database scope (Not all permissions in the database, unless created (& owns) the database)
setupadmin fixed server role
ALTER ANY LINKED SERVER (Can create, alter, and drop linked servers)
securityadmin fixed server role
ALTER ANY LOGIN (Can create, alter, and drop logins)
All permissions in the login scope
The securityadmin role should be treated as equivalent to the sysadmin role.
diskadmin fixed server role
ALTER RESOURCES
serveradmin fixed server role
ALTER SETTINGS
SHUTDOWN
ALTER ANY ENDPOINT
CREATE ENDPOINT
All permissions in the endpoint scope
ALTER SERVER STATE
VIEW SERVER STATE
processadmin fixed server role
ALTER ANY CONNECTION
VIEW ANY DEFINITION
pubic server role
VIEW ANY DATABASE
Has CONNECT permission to the default endpoints
View definition of any object
ALTER ANY CREDENTIAL
ALTER ANY SERVER AUDIT
ALTER TRACE
AUTHENTICATE SERVER
CONNECT SQL (Granted to all new logins)
ALTER ANY EVENT NOTIFICATION
CREATE DDL EVENT NOTIFICATION
CREATE TRACE EVENT NOTIFICATION
UNSAFE ASSEMBLY
EXTERNAL ACCESS ASSEMBLY
9 server roles, 26 server permissions

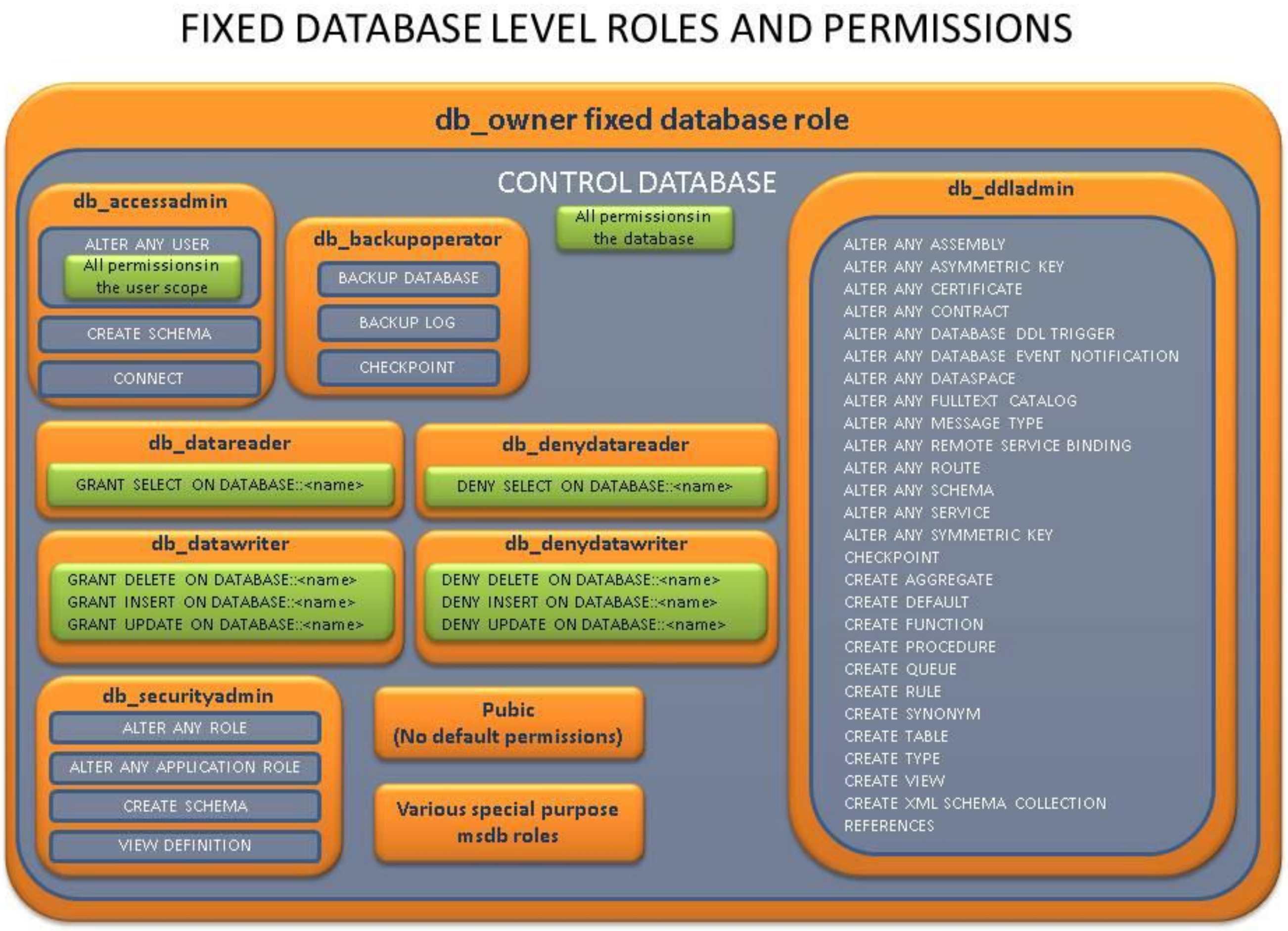
https://docs.microsoft.com/pl-pl/sql/relational-databases/security/authentication-access/server-level-roles

Principals

- Database roles
  - There are 2 types:
    - Fixed (predefined)
      - db_owner, db_securityadmin, db_accessadmin, db_backupoperator, db_ddladmin, db_datawriter, db_datareader, db_denydatawriter, db_denydatareader
      - Permissions that are granted cannot be changed
      - List of permissions for role:
        - sp_dbfixedrolepermission rolename
    - Flexible (defined by user)
      - How to manage roles?
        - From SQL
          - CREATE ROLE rolename
          - ALTER ROLE rolename {ADD|DROP} MEMBER {username|rolename}
        - From MGMT Studio

https://docs.microsoft.com/pl-pl/sql/relational-databases/security/authentication-access/database-level-roles

Principals


FIXED DATABASE LEVEL ROLES AND PERMISSIONS
db_owner fixed database role
CONTROL DATABASE
All permissionsin the database
db_accessadmin
ALTER ANY USER
All permissionsin the user scope
CREATE SCHEMA
CONNECT
db_backupoperator
BACKUP DATABASE
BACKUP LOG
CHECKPOINT
db_datareader
GRANT SELECT ON DATABASE::<name>
db_denydatareader
DENY SELECT ON DATABASE::<name>
db_datawriter
GRANT DELETE ON DATABASE::<name>
GRANT INSERT ON DATABASE::<name>
GRANT UPDATE ON DATABASE::<name>
db_denydatawriter
DENY DELETE ON DATABASE::<name>
DENY INSERT ON DATABASE::<name>
DENY UPDATE ON DATABASE::<name>
db_securityadmin
ALTER ANY ROLE
ALTER ANY APPLICATION ROLE
CREATE SCHEMA
VIEW DEFINITION
Pubic (No default permissions)
Various special purpose msdb roles
db_ddladmin
ALTER ANY ASSEMBLY
ALTER ANY ASYMMETRIC KEY
ALTER ANY CERTIFICATE
ALTER ANY CONTRACT
ALTER ANY DATABASE DDL TRIGGER
ALTER ANY DATABASE EVENT NOTIFICATION
ALTER ANY DATASPACE
ALTER ANY FULLTEXT CATALOG
ALTER ANY MESSAGE TYPE
ALTER ANY REMOTE SERVICE BINDING
ALTER ANY ROUTE
ALTER ANY SCHEMA
ALTER ANY SERVICE
ALTER ANY SYMMETRIC KEY
CHECKPOINT
CREATE AGGREGATE
CREATE DEFAULT
CREATE FUNCTION
CREATE PROCEDURE
CREATE QUEUE
CREATE RULE
CREATE SYNONYM
CREATE TABLE
CREATE TYPE
CREATE VIEW
CREATE XML SCHEMA COLLECTION
REFERENCES

https://docs.microsoft.com/pl-pl/sql/relational-databases/security/authentication-access/database-level-roles

Principals

- Application roles
  - Gives possibility to assign permission to a specific application
  - A scenario
    - A user executes a client application.
    - The client application connects to an instance of SQL Server as the user.
    - The application then executes the sp_setapprole stored procedure with a password known only to the application.
    - If the application role name and password are valid, the application role is enabled.
    - At this point the connection loses the permissions of the user and assumes the permissions of the application role.

https://docs.microsoft.com/pl-pl/sql/relational-databases/security/authentication-access/application-roles

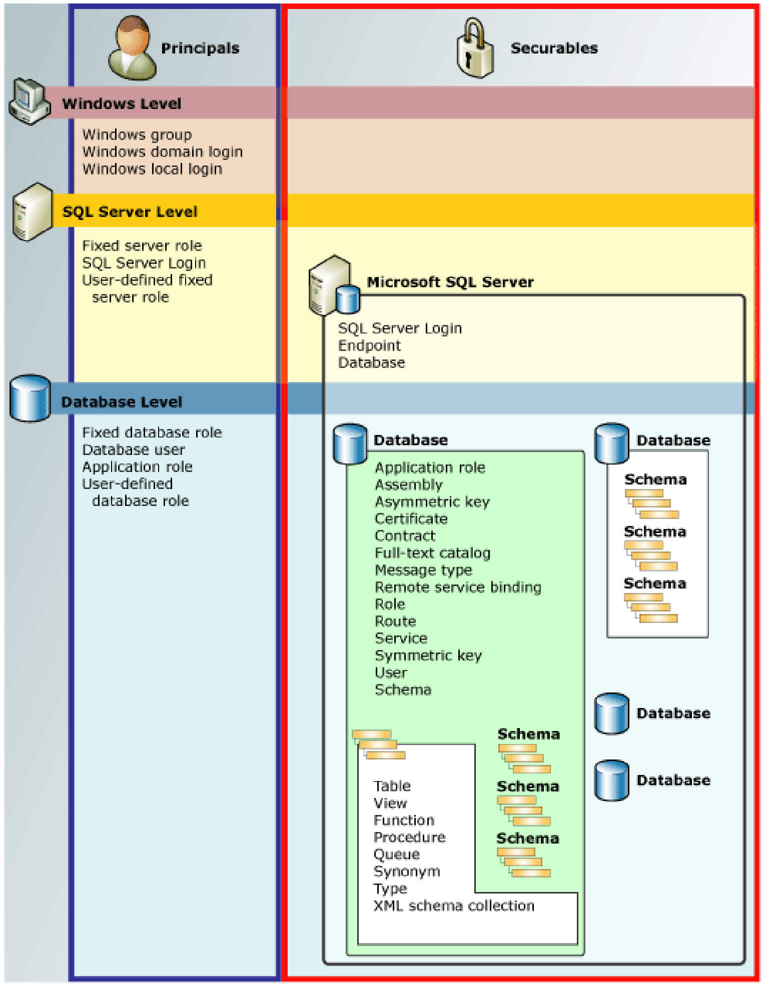
Schema

- In general
  - Complete structure of tables (objects) and relationship
- In SQL Server
  - Collection of objects within a database
  - Database can have many schemas
- Basic operation:
  - CREATE SCHEMA <Warehouse> [AUTHORIZATION <User>]
    - Authorization defines an owner
  - Accesing schemas: [schema].[object]
    - E.g. CREATE TABLE [Warehouse].[Invoice] (…)
  - Default schema: [dbo] (owned by dbo)
  - Changing a schema
    - ALTER SCHEMA NewSchema TRANSFER dbo.Person

Impersonation

- Normally, every statement is executed in the context of the connected user
- Impersonation can be achieved by EXEC AS
    - EXECUTE AS {LOGIN | USER} = 'name'
- Return to the original context
    - REVERT

Permissions Hierarchy



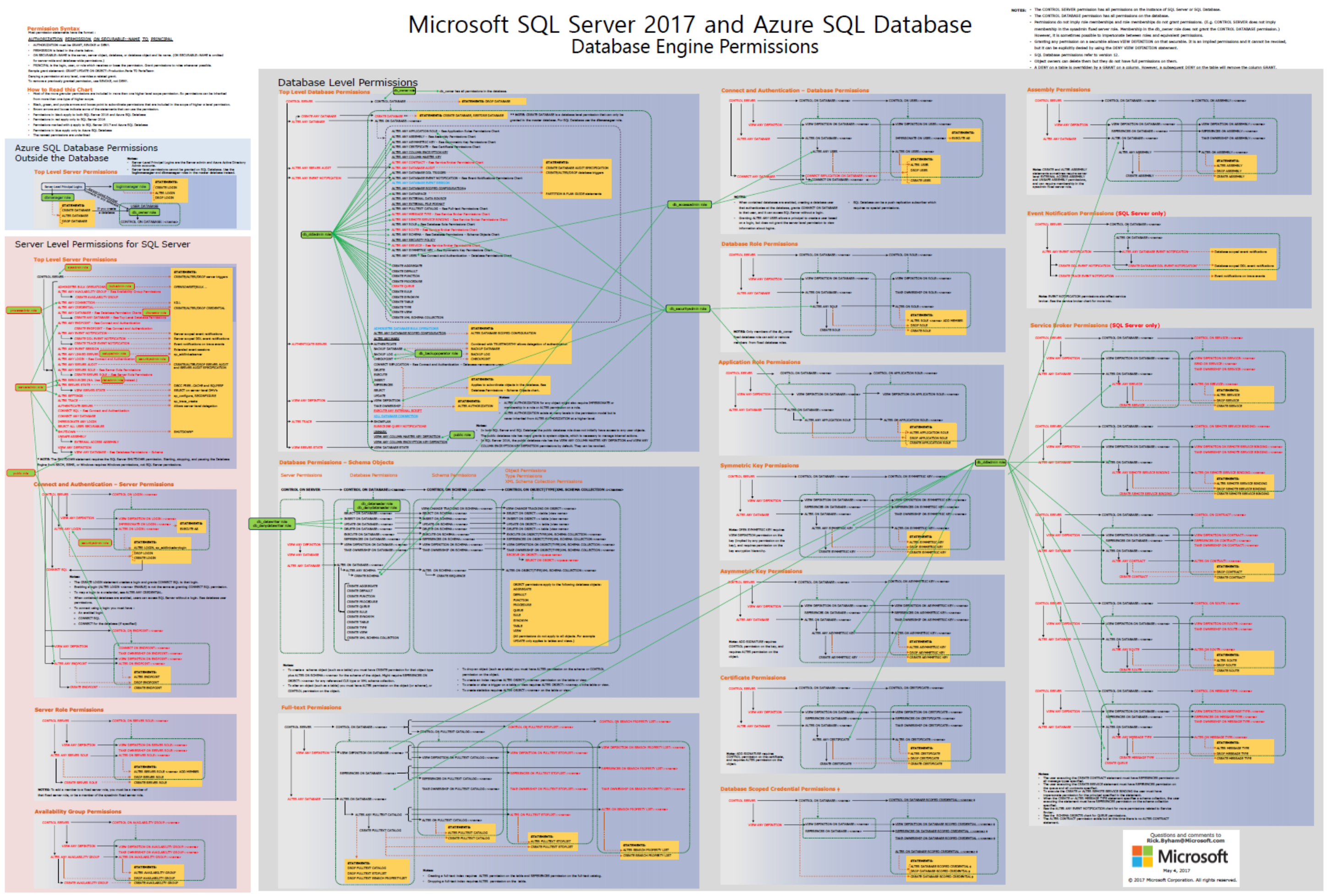
https://docs.microsoft.com/pl-pl/sql/relational-databases/security/permissions-hierarchy-database-engine

Permissions

- Managing permissions
  - From MGMT Studio
    - Open Database properties
    - Change tab to permissions
  - From SQL (there are much more syntax)
    - GRANT { ALL [PRIVILEGES] }
      ```
      | permission [ ( column [ ,...n ] ) ] [ ,...n ]
      [ ON [ class :: ] securable ] TO principal [ ,...n ]
      [ WITH GRANT OPTION ] [ AS principal ]
      ```
    - REVOKE
      ```
      <permission> [ ,...n ]
      [ ON [ <class_type> :: ] securable ]
      [ FROM | TO ] principal [ ,...n ]
      [ CASCADE ]
      ```
    - DENY { ALL [PRIVILEGES] }
      ```
      | permission [ ( column [ ,...n ] ) ] [ ,...n ]
      [ ON [ class :: ] securable ] TO principal [ ,...n ]
      [ CASCADE] [ AS principal ]
      ```

https://docs.microsoft.com/en-us/sql/t-sql/statements/statements

Permissions


Microsoft SQL Server 2017 and Azure SQL Database
Database Engine Permissions
Permission Syntax
How to Read this Chart
Azure SQL Database Permissions Outside the Database
Top Level Server Permissions
Server Level Permissions for SQL Server
Top Level Server Permissions
Connect and Authentication – Server Permissions
Server Role Permissions
Availability Group Permissions
Database Level Permissions
Top Level Database Permissions
Database Permissions – Schema Objects
Full-text Permissions
Connect and Authentication – Database Permissions
Database Role Permissions
Application Role Permissions
Symmetric Key Permissions
Asymmetric Key Permissions
Certificate Permissions
Database Scoped Credential Permissions
Assembly Permissions
Event Notification Permissions (SQL Server only)
Service Broker Permissions (SQL Server only)
Questions and comments to Rick.Byham@Microsoft.com
Microsoft
May 4, 2017
© 2017 Microsoft Corporation. All rights reserved.

https://docs.microsoft.com/pl-pl/sql/relational-databases/security/permissions-database-engine

Permissions monitoring

- Basic information
  - master.sys.syslogins
  - db.sys.sysusers
  - db.sys.database_principals
  - db.sys.database_permissions
  - db.sys.database_role_members
- Look at permissions:
  - fn_my_permissions

Ownership

- Every securable has an owner
  - Owner can do everything with an object
  - Anyone can revoke owner’s privileges
  - User can’t be dropped if it owns something
- By default an owner of an object is a database owner
- Changing ownership
  - ALTER AUTHORIZATION ON <Object> TO <User>
  - More:
    - http://msdn.microsoft.com/en-us/library/ms187359.aspx

Ownership

- Access in chain (referenced objects) is verified differently than in separated objects
  - If a referenced object has the same owner as the source object, permissions are not checked
  - If a procedure references a table and owners are the same, table permissions are not checked
  - Ownership chaining doesn’t apply to dynamic SQL (in such case all permissions must be explicitly granted)

Ownership

- In other words:
  - Let's assume that there is a chain of calls
    O1→O2→O3→…→On
    and all Oi has the same owner
  - Then permissions are checked only on access to O1
- Let's see the consequences

DEMO

- Practical example: roles usage
  - There is default good way to give an EXECUTE permission to a user
  - The solution

    -
      ```
      CREATE ROLE db_executor
      GRANT EXECUTE TO db_executor
      EXEC sp_addrolemember 'db_executor', 'username'
      ```

DEMO

- Practical example: the ownership chain consequences

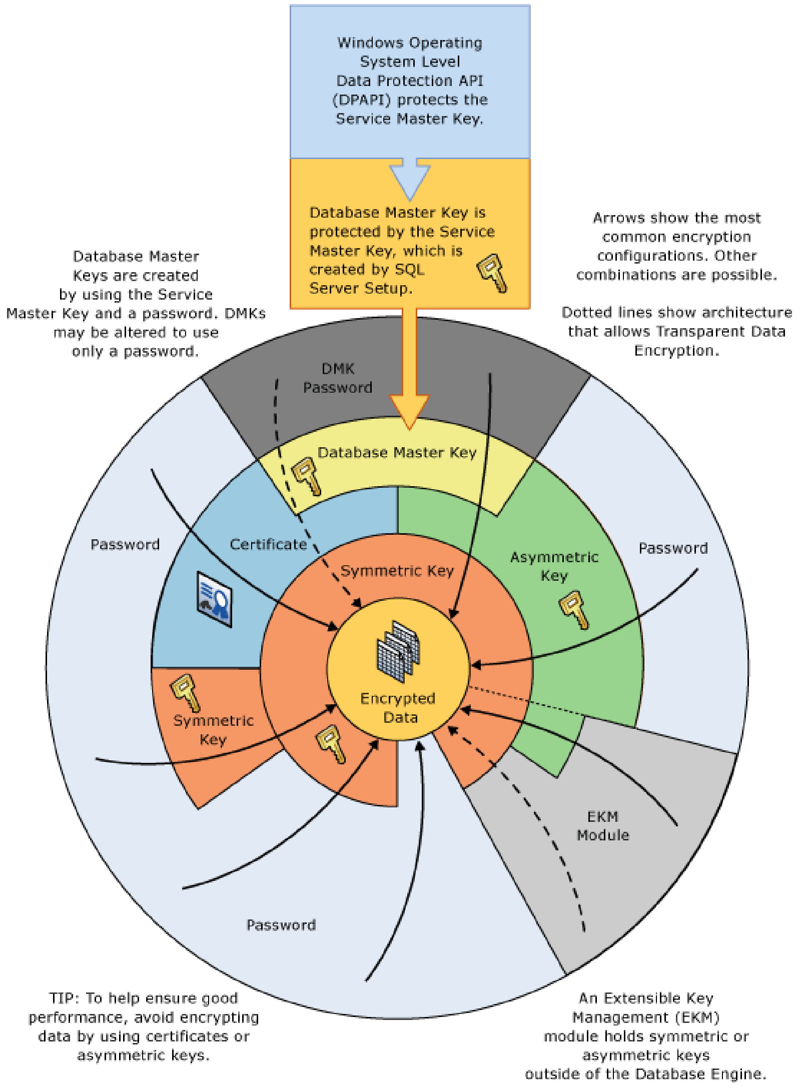
```
CREATE TABLE SomeData (Number INT)
GO

CREATE PROCEDURE ShowSomeData AS SELECT * FROM SomeData
GO

--ALTER AUTHORIZATION ON SomeData TO dbo --SCHEMA OWNER
--ALTER AUTHORIZATION ON ShowSomeData TO dbo --SCHEMA OWNER
--GO

SELECT * FROM sys.all_objects WHERE name LIKE '%SomeData'
GO

GRANT EXECUTE ON ShowSomeData TO Test
DENY SELECT ON SomeData TO Test
GO

EXECUTE AS USER = 'Test'
GO
SELECT * FROM SomeData
GO
EXEC ShowSomeData
GO
REVERT
GO
```

Encryption in a database

- There are situations in which protecting access to a database is not enough
  - Someone breach this access level protection
  - Access rights are assigned in a wrong way
  - Backup files are stolen
  - Protection of filesystem is compromised
  - And many others...
- If we have a very sensitive data, encryption in a database is a one more layer of defense

Encryption Hierarchy

- Encryption can be achieved through different ways
- Every way is implied by a different chain of keys
- Every way has pros and cons, so should be evaluated according to the requirements


Windows Operating System Level Data Protection API (DPAPI) protects the Service Master Key.
Database Master Key is protected by the Service Master Key, which is created by SQL Server Setup.
Database Master Keys are created by using the Service Master Key and a password. DMKs may be altered to use only a password.
Arrows show the most common encryption configurations. Other combinations are possible.
Dotted lines show architecture that allows Transparent Data Encryption.
DMK Password
Database Master Key
Password
Certificate
Symmetric Key
Asymmetric Key
Password
Encrypted Data
Symmetric Key
EKM Module
Password
TIP: To help ensure good performance, avoid encrypting data by using certificates or asymmetric keys.
An Extensible Key Management (EKM) module holds symmetric or asymmetric keys outside of the Database Engine.

More:
http://technet.microsoft.com/en-us/library/ms189586(v=sql.110).aspx

Encryption hierarchy components

- Asymmetric Keys
- Symmetric Keys
- Certificates
- Extensible Key Management (EKM)
  - Since SQL Server 2008
  - Gives a possibility to manage keys by an external source such as Hardware Security Module (HSM)

Column Encryption vs. Transparent Data Encryption

- Column Enryption: data is encrypted explicitly
  - Applications and users are impacted
  - One can choose what exactly should be encrypted – no overhead for encryption less sensitive data
- TDE: the whole database is encrypted
  - Encryption is hidden and transparent, so if one can connect, one can see the data
  - Everything is encrypted, also less sensitive data
- The choice depends on business needs

Column Encryption

- This is supported by set of built-in functions and procedures together with key hierarchy
- Operations are performed manually
- Encrypted data needs to be stored in a varbinary column type
- Main steps
  - Create database master key for every database
    - Notice: service master key has been created when the instance has been created
  - Create a certificate to protect keys
  - Create a symmetric key which is protected by the certificate created in the previous step
  - Enjoy encrypting data: open the symmetric key, encrypt the data, close the key
- Decryption is similar to encryption, but a function for decryption should be used

Example

```
USE Test

CREATE TABLE Person
(
    ID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    CreditCard VARBINARY(200)
)
GO

INSERT INTO Person (ID, FirstName, LastName) VALUES(1, 'J1', 'K1');
INSERT INTO Person (ID, FirstName, LastName) VALUES(2, 'J1', 'K1');
INSERT INTO Person (ID, FirstName, LastName) VALUES(3, 'J1', 'K1');
GO

CREATE MASTER KEY ENCRYPTION BY PASSWORD='SomePassword'
GO

CREATE CERTIFICATE CertForTest WITH SUBJECT='Test'
GO

CREATE SYMMETRIC KEY CreditCardKey WITH ALGORITHM=TRIPLE_DES ENCRYPTION BY CERTIFICATE CertForTest
GO

OPEN SYMMETRIC KEY CreditCardKey DECRYPTION BY CERTIFICATE CertForTest
UPDATE Person SET CreditCard = ENCRYPTBYKEY(KEY_GUID('CreditCardKey'), '11111') WHERE ID=1;
UPDATE Person SET CreditCard = ENCRYPTBYKEY(KEY_GUID('CreditCardKey'), '22222') WHERE ID=2;
UPDATE Person SET CreditCard = ENCRYPTBYKEY(KEY_GUID('CreditCardKey'), '33333') WHERE ID=3;
CLOSE SYMMETRIC KEY CreditCardKey
GO

SELECT * FROM Person
GO

OPEN SYMMETRIC KEY CreditCardKey DECRYPTION BY CERTIFICATE CertForTest
SELECT ID, FirstName, LastName, CONVERT(VARCHAR, DECRYPTBYKEY(CreditCard)) [Credit Card] FROM Person
CLOSE SYMMETRIC KEY CreditCardKey
GO
```

Transparent Data Encryption

- TDE is one of usages of encryption by symmetric keys
- There is whole database encrypted by a symmetric key called database encryption key
- Database encryption key is protected by certificate which is protected by database master key or asymmetric key from EKM
- Available only on Enterprise Edition or Developer Edition
- Provides query optimization
- Main steps
  - Create master key encryption password
  - Create a certificate
  - Backup the certificate
  - Create a database encryption symmetric key
  - Alter the database to set encryption on
  - Optionally monitor the encryption process
- More: http://msdn.microsoft.com/en-us/library/bb934049.aspx

Example

```
USE master

CREATE MASTER KEY ENCRYPTION BY PASSWORD='SomePassword'
GO

CREATE CERTIFICATE TestDatabaseServerCertificate WITH SUBJECT='Test Certificate'
GO

BACKUP CERTIFICATE TestDatabaseServerCertificate
TO FILE ='C:\Temp\TestDatabaseServerCertificate'
WITH PRIVATE KEY(
    FILE = 'C:\Temp\TestDatabaseServerCertificate.private',
    ENCRYPTION BY PASSWORD = 'AnotherPassword')

USE Test

CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE TestDatabaseServerCertificate
GO

ALTER DATABASE Test SET ENCRYPTION ON
GO

SELECT DB_NAME(database_id), encryption_state, key_algorithm, key_length
FROM sys.dm_database_encryption_keys
GO
```

Encryption algorithms

- There are many available algorithms:
  - DES, Triple DES, TRIPLE_DES_3KEY, RC2, RC4, 128-bit RC4, DESX, 128-bit AES, 192-bit AES, 256-bit AES and more
- However, other than AES_128, AES_192, and AES_256 are considered as deprecated

Encryption in transit

- When it comes to communication we consider two challenges
  - Storing credentials to a database server in a secure way
    - This was covered in OWASP Top 10 topic
  - Encrypting communication channel
    - SQL Server supports encrypting connection using TLS
      - A valid certificate is required
    - DEMO
      - Open: Configuration Tools → SQL Server Configuration Manager
      - Open: Properties for SQL Server Network Configuration
    - More
      - https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/enable-encrypted-connections-to-the-database-engine
      - http://technet.microsoft.com/en-us/library/ms189067(v=sql.105).aspx

Always encrypted

- Available since SQL Server 2016
- Combines encryption both in storage and transit
- Encryption/decryption executed on client
  - Requires .NET 4.6 SQL Client Driver
- Column level encryption
  - Spefic columns need to be selected
- Types of encryption
  - Deterministic: the same ciphertext for the same values
    - One can benefit from equality joins, grouping, indexing, etc.
    - … but it is less secure, e.g. columns is limited set of values like True/False, North/South/East/West values can be discovered
  - Randomized: different ciphertexts for different values
    - More secure, but one can't benefit from making operation on a database
- Transparent for the applications
  - Driver will handle the traffic „on the fly"
  - A section: Column Encryption Setting=Enabled needs to be added to the connection string

https://docs.microsoft.com/pl-pl/sql/relational-databases/security/encryption/always-encrypted-database-engine

SQL Server Audit

- It is a mechanism which allows to monitor who is doing what on which objects
- There a lot of possibilities what can be audited
- It is based on Extended Events, new feature since SQL Server 2008
  - Audit is specialized usage of Extended Events

SQL Server Audit

- DEMO: Let’s create Server-Level audit
  - MGMT → Security → Audits
    - Create an audit DatabaseRoleMemberChange
  - MGMT → Security → Server Audit Specifications
    - Create a specification DatabaseRoleMemberChange related to DatabaseRoleMemberChange event
  - Add any user to any role
    - USE Test; ALTER ROLE db_owner ADD MEMBER test
  - MGMT → Security → Audits
    - Pick the audit
    - Choose View Audits Logs option

SQL Server Audit

- DEMO: Let’s create Database-Level audit
  - MGMT → Security → Audits
    - Create TestDatabaseSelect audit
  - MGMT → Test database → Security → Server Audit Specification
    - Create TestDatabaseSelect specification on
      - SELECT event
      - Osoba table
      - [public] role
  - Perform a select on the Osoba table in Test DB
  - View TestDatabaseSelect audit

SQL Server Audit

- There is another way to see audit entries which is based on review files
- DEMO

  -
    ```
    SELECT * INTO Test.dbo.SQLAudits
    FROM sys.fn_get_audit_file(
        'C:\Temp\TestDatabase*.sqlaudit',Default, Default);
    ```

  -
    ```
    SELECT * FROM Test.dbo.SQLAudits
    ```

Policy based management

- Allows to apply and force policies and rules
- Three main components
  - Policy management. Policy administrators create policies.
  - Explicit administration. Administrators select one or more managed targets and explicitly check that the targets comply with a specific policy, or explicitly make the targets comply with a policy.
  - Evaluation modes. There are four evaluation modes:
    - On demand. This mode evaluates the policy when directly specified by the user.
    - On change: prevent. This automated mode uses DDL triggers to prevent policy violations.
    - On change: log only. This automated mode uses event notification to evaluate a policy when a relevant change is made.
    - On schedule. This automated mode uses a SQL Server Agent job to periodically evaluate a policy.

https://docs.microsoft.com/en-us/sql/relational-databases/policy-based-management/administer-servers-by-using-policy-based-management

Policy based management

- Let’s see some examples
  - MGMT → Management → Policy Management
  - Review Facets
  - Create a policy RecoveryModelFull for ensuring that every database has a full recovery model
    - Create a condition using Database Options facet
    - Create a policy based on that condition and evaluate it
  - Create a policy for ensuring that no table is created in dbo schema (do the same for procedure)
    - Create a condition using Table facet (analogously Stored Procedure)
    - Create a policy based on that condition and evaluate it
    - Try to enable that policy and try to create an object in that schema
      - E.g. `CREATE PROCEDURE dbo.GetServerName AS SELECT @@SERVERNAME`

Data Availability

- Data can be lost
  - By accident (someone forget WHERE clause when DELETE, click a wrong button)
  - By WANNACRY (or any other malicious crap)
  - Natural disasters
  - Theft, robbery
  - … many others
- Making backups is one of ways to mitigate the data lost

Data Availability

- Questions need to be answer for backups:
  - Which databases to backup?
  - How often to do that? How often data are modified?
  - What is acceptable period of data loss?
  - How fast we need data back after failure?
    - Disk vs. Tape
  - Where to store backups?
    - It should be different location?
  - How backups are protected?

Data Availability

- Main types of backups
  - Full
  - Differential backups
  - Transaction log backups
- There are also
  - File backups
  - Filegroup backups
  - Partial backups
- To make differential and transaction log backup you need
  - Full backup
  - Correct sequence of differential or transaction log backups

Data Availability

- Very important thing:

Test your backup by regular restoration

Data Availability

- High Availability
  - Failover clustering
  - Database mirroring
  - Log shipping
  - Replication

References

- Documentation from Microsoft
  - https://docs.microsoft.com/pl-pl/sql/relational-databases/security/security-center-for-sql-server-database-engine-and-azure-sql-database
- Who is the Default Owner of your Database and Server Objects?
  - http://sqlity.net/en/2180/default-owner/
- Schema-Based Access Control for SQL Server Databases
  - https://www.simple-talk.com/sql/sql-training/schema-based-access-control-for-sql-server-databases/
- Understanding SQL Server fixed database roles
  - https://www.mssqltips.com/sqlservertip/1900/understanding-sql-server-fixed-database-roles/
- System Compatibility Views (Transact-SQL)
  - https://docs.microsoft.com/en-us/sql/relational-databases/system-compatibility-views/system-compatibility-views-transact-sql
- Transact-SQL statements
  - https://docs.microsoft.com/en-us/sql/t-sql/statements/statements
- System Stored Procedures (Transact-SQL)
  - https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/system-stored-procedures-transact-sql
- Catalog Stored Procedures (Transact-SQL)
  - https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/catalog-stored-procedures-transact-sql
- sys.objects (Transact-SQL)
  - https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sys-objects-transact-sql
- Security Through Ownership Chains
  - http://sqlmag.com/sql-server/security-through-ownership-chains