Adatbázis titkosítási módszerek összehasonlítása és folyamatuk leírása különböző adatbázis kezelő rendszerekben. Három DBMS-beli (adatbázis kezelő rendszer) megoldás kerül összehasonlításra, implementálásra és bemutatásra, Microsoft SQL Server (MSSQL), MySQL, PostgreSQL. A következő leírás a hivatalos dokumentációk értelmezése majd alkalmazása alapján készült.

MSSQL

TDE: Alkalmazásához a következő lépéseket kell megtenni.

1. Ún. master key, azaz főkulcs létrehozása
2. Ún. certificate, tanúsítány létrehozása vagy beszerzése, amit ez a főkulcs véd.
3. Adatbázis titkosítási kulcs létrehozása, amit a tanúsítvány véd.
4. Adatbázis beállítása a titkosítás használatára.

Implementálás:

```
----Master adatbázis használata
use master;
go
----Főkulcs létrehozása, amit a megadott jelszóval titkosít.
----Ajánlott bonyolultabb jelszó használata, ez csak szemléltetés célját szolgálja.
create master key encryption by password ='testpw'
go
----Tanúsítvány létrehozása. TDE_CERT a neve, a subject pedig X.509 szabványban
----meghatározott tanúsítvány metaadatainak egy mezőjére utal.
create certificate TDE_CERT with subject ='tde_cert'
go


----Titkosítandó adatbázis kiválasztása
use test2;
go
----Adatbázis titkosítási kulcs létrehozása AES-256 titkosítási algoritmussal,
----amit az előbb létrehozott tanúsítvány véd.
create database encryption key
with algorithm = AES_256
encryption by server certificate TDE_CERT;
go

----A test2 adatbázis beállítása a titkosítás használatára.
go
alter database test2 set encryption on;
go
```

Ez a lépés a Properties > Options > State > Encryption Enabled menüpontban is beállítható.

Minden esetben ajánlott a tanúsítvány és a főkulcs biztonsági mentése is:

```
use master
go
backup certificate TDE_CERT
to file='C:\sql_queries\testkey.cer'
with private key (file='C:\sql_queries\testkey.key', encryption by password ='testpw')
go

backup master key
to file = 'C:\sql_queries\masterkey'
encryption by password = 'testpw';
```

Két módon is meggyőződhetünk a titkosítás sikerességéről. Mielőtt beállítanánk az adatbázist a titkosítás használatára így néz ki annak státusza:

```
select DB.name, DB.is_encrypted, DM.encryption_state from sys.databases DB LEFT OUTER JOIN
sys.dm_database_encryption_keys DM ON DB.database_id = DM.database_id;
```

	name	is_encrypted	encryption_state
1	master	0	NULL
2	tempdb	1	3
3	model	0	NULL
4	msdb	0	NULL
5	test2	0	1

Az ’is_encrypted’ értéke 0 vagy 1 lehet. 0 esetében nem igaz, 1 esetében igaz.

Az ’encryption_state’ több értéket is felvehet, valószínúleg az 1, 2 vagy 3-mal fogunk leginkább találkozni. 1 – titkosítva, 2 – titkosítás folyamatban, 3 – titkosítva.

Beállítás után az értékek:

	name	is_encrypted	encryption_state
1	master	0	NULL
2	tempdb	1	3
3	model	0	NULL
4	msdb	0	NULL
5	test2	1	3

, meggyőződhetünk a titkosítás sikerességéről.

Column Level Encryption

Oszlopszintű titkosításhoz szimmetrikus kulcs használható. A végrehajtáshoz szükség van egy adatbázis főkulcsra, illetve hogy ’createés ’alter’ hívásokat tudjunk végezni az adatbázison.

A folyamat titkosítási része a következőből áll:

```
----Főkulcs létrehozása, amit a megadott jelszóval titkosít.
----Ajánlott bonyolultabb jelszó használata, ez csak szemléltetés célját szolgálja.
create master key encryption by password ='testpw'
go
----Tanúsítvány létrehozása. COLUMN_CERT a neve, a subject pedig X.509 szabványban
----meghatározott tanúsítvány metaadatainak egy mezőjére utal.
create certificate COLUMN_CERT with subject ='Oldal név és tartalom'
go
----Szimmetrikus kulcs lértehozása, AES-256 titkosítási algoritmussal,
----amit az előbb létrehozott tanúsítvány véd.
create symmetric key Sym_key
    with algorithm = AES_256
    encryption by certificate COLUMN_CERT;
go

----Page táblán belül új oszlop létrehozása titkosított adat tárolására.
use test2;
ALTER TABLE dbo.page
    ADD encryptedPageName varbinary(MAX);
GO

----Szimmetrikus kulcs 'megnyitása', amit az adott tanúsítvány dekódol.
open symmetric key Sym_key decryption by certificate COLUMN_CERT;
go

----Titkosítás ezekben a sorokban történik a megadott szimmetrikus kulccsal.
UPDATE test2.dbo.page
set encryptedPageName = EncryptByKey(Key_GUID('Sym_key'), pageName) from test2.dbo.page;
go

----Kulcs bezárása
close symmetric key Sym_key;
go
```

Csak egy oszlop került titkosításra, de ahogy a módszer erősségeinél említettem, lehet több oszlopot is kódolni különböző kulcsok és tanúsítványok használatával. A titkosítás eredménye:

A dekódolás a következő:

```
----Szimmetrikus kulcs 'megnyitása', amit az adott tanúsítvány dekódol.
open symmetric key Sym_key decryption by certificate COLUMN_CERT;
go

----Adatokat kiválasztom és kiiratom, nem mentem le mégegyszer.
----A dekódolás a DecryptByKey függvény használatával történik,
----majd a bináris eredményt átkonvertálom varchar típusúvá.
SELECT pageName, encryptedPageName
    AS 'Titkosítás után',
    CONVERT(varchar, DecryptByKey(encryptedPageName))
    AS 'Dekódolás után'
    FROM test2.dbo.page;
GO

----Kulcs bezárása
close symmetric key Sym_key;
go
```

A titkosítás és dekódolás eredménye

	pageName	Titkosítás után	Dekódolás után
1	New Page 0	0x0048EA68F9CE984894B30434F7A0D8F0020000000C14C285...	New Page 0
2	New Page 1	0x0048EA68F9CE984894B30434F7A0D8F00200000000B9191B0...	New Page 1
3	New Page 2	0x0048EA68F9CE984894B30434F7A0D8F00200000000AFE1A83...	New Page 2
4	New Page 3	0x0048EA68F9CE984894B30434F7A0D8F00200000000F39A122...	New Page 3
5	New Page 4	0x0048EA68F9CE984894B30434F7A0D8F00200000000418206F...	New Page 4
6	New Page 5	0x0048EA68F9CE984894B30434F7A0D8F00200000000ED8A2B...	New Page 5
7	New Page 6	0x0048EA68F9CE984894B30434F7A0D8F00200000000ECB8F8...	New Page 6
8	New Page 7	0x0048EA68F9CE984894B30434F7A0D8F00200000005FB47C5...	New Page 7
9	New Page 8	0x0048EA68F9CE984894B30434F7A0D8F00200000000E9E3CB...	New Page 8
10	New Page 9	0x0048EA68F9CE984894B30434F7A0D8F00200000065C865E...	New Page 9
11	New Page 10	0x0048EA68F9CE984894B30434F7A0D8F00200000003D78CD...	New Page 10
12	New Page 11	0x0048EA68F9CE984894B30434F7A0D8F00200000102C168...	New Page 11
13	New Page 12	0x0048EA68F9CE984894B30434F7A0D8F00200000B2AC448...	New Page 12
14	New Page 13	0x0048EA68F9CE984894B30434F7A0D8F00200000370BBB3...	New Page 13
15	New Page 14	0x0048EA68F9CE984894B30434F7A0D8F00200000005D34FD...	New Page 14
16	New Page 15	0x0048EA68F9CE984894B30434F7A0D8F00200000003636AF7...	New Page 15
17	New Page 16	0x0048EA68F9CE984894B30434F7A0D8F00200000040061DB...	New Page 16

összegezve:

Cell / Field Level Encryption

Mező szintű titkosítás folyamata nagyrészt megegyezik az oszlop szintű titkosítással. A titkosítási parancson belül kell specifikációkat megadnunk. Egy ilyen titkosítás például:

```
----Titkosítás ezekben a sorokban történik a megadott szimmetrikus kulccsal.
----Csak a meghatározott mezők kerülnek titkosításra. Ebben az esetben annak a page-nek a neve, aminek id-je = 10.
UPDATE test2.dbo.page
set encryptedPageName = EncryptByKey(Key_GUID('Sym_key'), pageName) from test2.dbo.page where pageId = 10;
go
```

Ezt megelőző kód (főkulcs, szimmetrikus kulcs létrehozása, stb…) megegyezik az oszlop szintű titkosításnál bemutatott kóddal.

	pageName	pageText	pageId	noteId	encryptedPageName
1	New Page 0		0	5	NULL
2	New Page 1		1	5	NULL
3	New Page 2		2	5	NULL
4	New Page 3		3	5	NULL
5	New Page 4		4	5	NULL
6	New Page 5		5	5	NULL
7	New Page 6		6	5	NULL
8	New Page 7		7	5	NULL
9	New Page 8		8	3	NULL
10	New Page 9		9	3	NULL
11	New Page 10		10	3	0x0048EA68F9CE984894B30434F7A0D8F002000000780E34...
12	New Page 11		11	3	NULL
13	New Page 12		12	3	NULL
14	New Page 13		13	3	NULL
15	New Page 14		14	3	NULL
16	New Page 15	<html> <head> </hea...	15	3	NULL
17	New Page 16	<html> <head> </hea...	16	4	NULL

Nyilván, ha több adat felel meg a feltételnek, akkor nem csak egy cella lesz titkosítva. Eredménye ennek a három sornak:

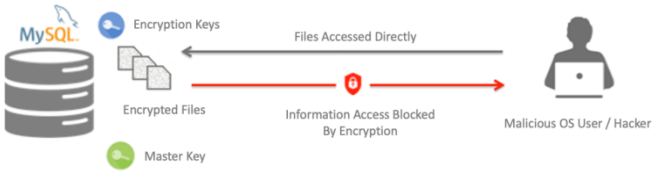
A dekódoláshoz használható az oszlop szintű titkosításnál szemléltetett parancs, nem szükséges a specifikáció. A NULL értékekkel nem fog semmi történni, viszont az id=10-es page neve dekódolásra kerül.

```
----Adatokat kiválasztom és kiiratom, nem mentem le mégegyszer.
----A dekódolás a DecryptByKey függvény használatával történik,
----majd a bináris eredményt átkonvertálom varchar típusúvá.
SELECT pageName, encryptedPageName
    AS 'Titkosítás után',
    CONVERT(varchar, DecryptByKey(encryptedPageName))
    AS 'Dekódolás után'
    FROM test2.dbo.page;
GO
```

	pageName	Titkosítás után	Dekódolás után
1	New Page 0	NULL	NULL
2	New Page 1	NULL	NULL
3	New Page 2	NULL	NULL
4	New Page 3	NULL	NULL
5	New Page 4	NULL	NULL
6	New Page 5	NULL	NULL
7	New Page 6	NULL	NULL
8	New Page 7	NULL	NULL
9	New Page 8	NULL	NULL
10	New Page 9	NULL	NULL
11	New Page 10	0x0048EA68F9CE984894B30434F7A0D8F002000000780E34...	New Page 10
12	New Page 11	NULL	NULL
13	New Page 12	NULL	NULL
14	New Page 13	NULL	NULL
15	New Page 14	NULL	NULL
16	New Page 15	NULL	NULL
17	New Page 16	NULL	NULL

MySQL

MySQL esetében nem sikerült egyik módszert sem implementálnom, mert folyamatosan hibaüzenetkebe ütköztem a fejlesztőkörnyezet használata során.



Kimondottan a **TDE** alkalmazásáról MySQL-ben csak egy rövid, pár soros leírást találtam, ami a következőket írja le:

„A MySQL TDE lehetővé teszi a nyugvó adatok titksoítását az adatbázis fizikai fájljainak titkosításával. Az adatok titkosítása automatikusan, valós időben, a tárolóba való írás előtt történik, és a tárolóból való olvasáskor dekódolásra kerül. Ennek eredményeképpen a hackerek és a rosszindulatú felhasználók nem tudják az érzékeny adatokat közvetlenül az adatbázisfájlokból kiolvasni. A MySQL TDE a szabványos AES algoritmusokat használja.”

Mivel komplexebb leírást nem találtam, ezért a nyugvó adatokra vonatkozó titkosítási lehetőségeket kellett megnéznem, mivel a TDE is nyugvó adatokat titkosít (data-at-rest). Ilyen adatok titkosítását az InnoDB végzi (MySQL tárolási motorja (storage engine)) a következő képpen:

Az InnoDB (MySQL tárolási motorja) kétszintű titkosítási kulcsarchitektúrát alkalmaz, amely egy főkulcsból és táblakulcsokból áll. Egy asztaltér titkosításakor egy táblatér kulcsot titkosítanak és tárolnak az táblatér fejlécében. Ha egy alkalmazás vagy hitelesített felhasználó szeretne hozzáférni az adatokhoz, az InnoDB a főkulcsot használja a táblatér kulcs dekódolásához.

A nyugalmi adatok titkosítási funkciója egy kulcstartó komponensre vagy bővítményre támaszkodik a titkosítási kulcsok kezeléséhez.

A ’default_table_encryption’ rendszer változó meghatározza az alapértelmezett titkosítási beállítást sémákra és táblaterekre anélkül, akkor is, ha azok az ’ENCRYPTION’ megadása nélkül lettek volna definiálva.

```
mysql> SET GLOBAL default_table_encryption=ON;
```

Alapértelmezett titkosítási beállítás a séma létrehozásakor vagy módosításakor a ’DEFAULT ENCRYPTION’ megadásával is meghatározható.

```
mysql> CREATE SCHEMA test DEFAULT ENCRYPTION = 'Y';
```

További titkosítási funkciók is használhatóak oszlop és cella adatok titkosítására.

Fontos kikötés titkosított adatok tárolásához, hogy az oszlop adattípusa VARBINARY vagy BLOB típusú legyen, így valószínűleg nem lesznek tárolási problémák. Gyakori titkosítási algoritmusok alkalmazására ad lehetőséget, mint az AES és MD5. Egy pár ilyen funkció megnevezése:

AES_ENCRYPT(), AES_DECRYPT(), MD5(), SHA1(), SHA2(), stb… és ezekhez kapcsolódó tömörítő és dekompressziós funkciók.

Tegyük fel, hogy egy alkalmazás MD5() string értékeket tárol egy CHAR(32) oszlopban:

```
CREATE TABLE md5_tbl (md5_val CHAR(32), ...);
INSERT INTO md5_tbl (md5_val, ...) VALUES(MD5('abcdef'), ...);
```

Egy hex string tömörebb formába történő átalakításához úgy módosítja az alkalmazást, hogy helyette az UNHEX() és a BINARY(16) parancsot használja az alábbiak szerint:

```
CREATE TABLE md5_tbl (md5_val BINARY(16), ...);
INSERT INTO md5_tbl (md5_val, ...) VALUES(UNHEX(MD5('abcdef')), ...);
```

Oracle SQL Developer