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.

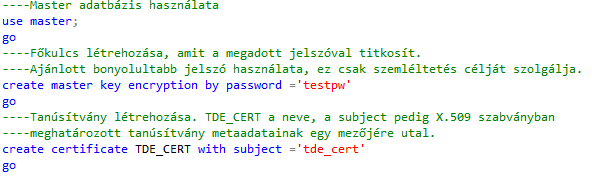
1. Ún. master key, azaz főkulcs létrehozása

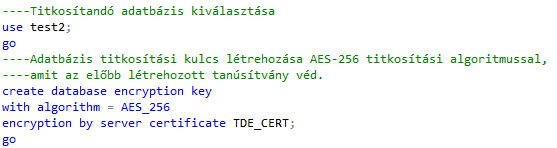
2. Ún. certificate, tanúsítány létrehozása vagy beszerzése, amit ez a főkulcs véd.

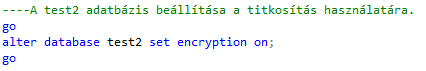
3. Adatbázis titkosítási kulcs létrehozása, amit a tanúsítvány véd.

4. Adatbázis beállítása a titkosítás használatára.

Implementálás:

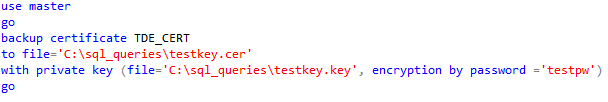






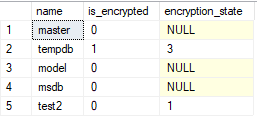
Ez a lépés a Properties > Options > State > Encryption Enabled menüpontban is beállítható.

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



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

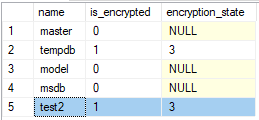


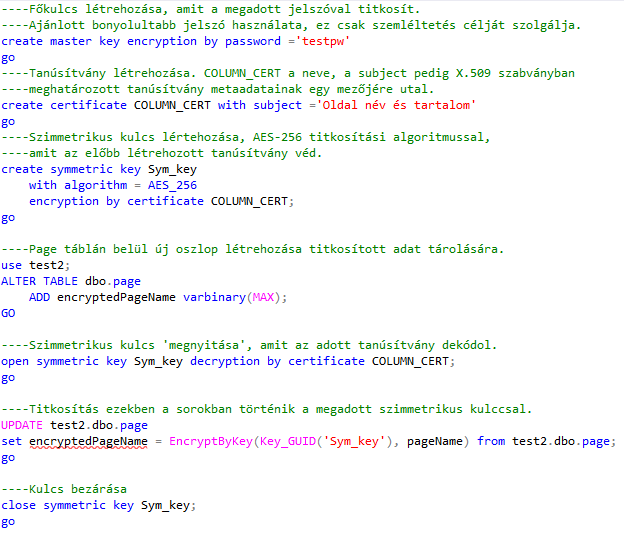


Az ’is\_encrypted’ értéke 0 vagy 1 lehet. 0 esetében nem igaz, 1 esetében igaz.

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:

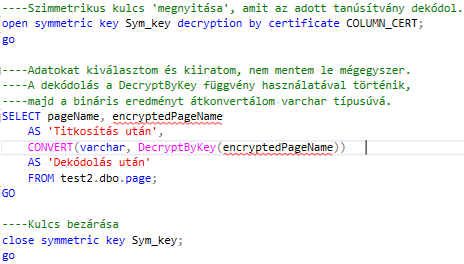
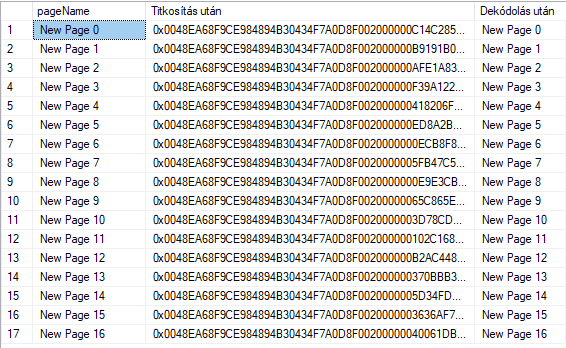
, meggyőződhetünk a titkosítás sikerességéről.

**Column Level Encryption**

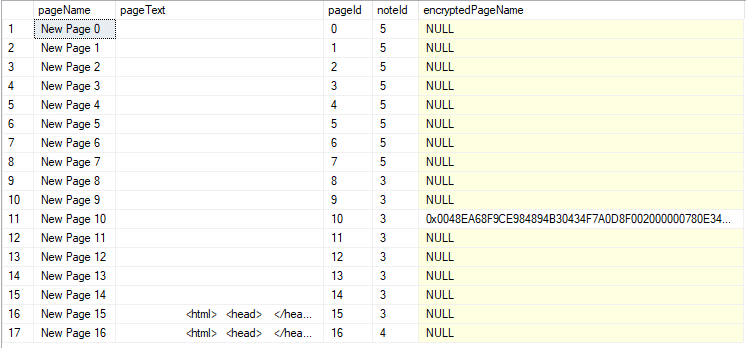
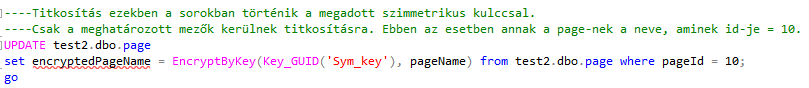
Oszlopszintű titkosításhoz szimmetrikus kulcs használható. A végrehajtáshoz szükség van egy adatbázis főkulcsra, illetve hogy ’createés ’alter’ hívásokat tudjunk végezni az adatbázison.

A folyamat titkosítási része a következőből áll:

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

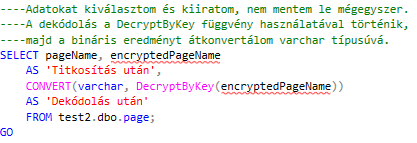
A dekódolás a következő: A titkosítás és dekódolás eredménye összegezve:

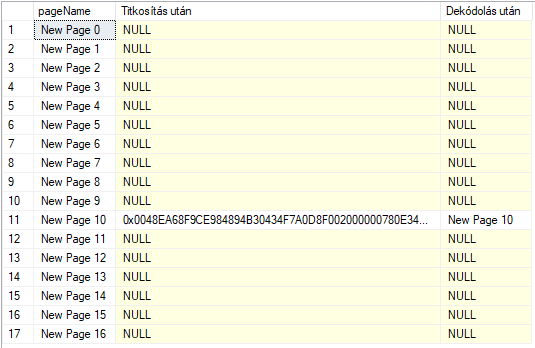
**Cell / Field Level Encryption**

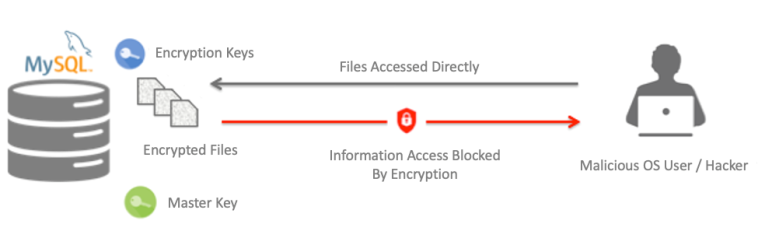
Mező szintű titkosítás folyamata nagyrészt megegyezik az oszlop szintű titkosítással. A titkosítási parancson belül kell specifikációkat megadnunk. Egy ilyen titkosítás például: 

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

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.



**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.

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

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:



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



**Oracle SQL Developer**