개인정보 보호법과 SQL Server의 보안

㈜엑셈 컨설팅본부/SQL Server팀 이 제춘

개인정보 보호법의 배경

최근 수년 간 정보통신기술은 발전에 발전을 거듭해 왔다. 정보통신기술의 발전은 일상생활을 비롯해 거의 모든 분야에 영향을 끼쳐 편리하고 윤택한 삶을 누릴 수 있는 발판을 마련해 주었다. 오늘 날 우리는 이동 중에도 스마트 기기를 통해 친구들과 실시간으로 소통 하고 온라인 게임이나 쇼핑을 즐길 수 있게 되었다. 집밖에서도 가정의 가전제품을 컨트롤 할 수 있게 되었으며 자택에서 원격 근무도 가능하게 되었다. 앞서 나열한 것들은 극히 일부분이다. 상상으로만 생각 했던 일들이 현실로 다가왔다.

그러나 정보통신기술 발전의 이면에는 여러 가지 문제점들이 나타났다. 해킹을 통해 기업 정보를 훔치거나 서버를 마비시키는 등 사이버 테러가 일어나고 악성 댓 글, 허위사실유포, 인터넷 마녀사냥, 신상 털기 등 사이버 윤리 문제가 발생 하였으며 스팸, 명예 훼손, 피싱 등 과 같은 다양한 범죄가 발생하게 되었다.

이런 가운데 몇몇 기업에서 개인정보 유출사고가 발생했다. 2008 년 인터넷 쇼핑몰 옥션에서 1800 만명의 고객정보가 유출되는 사건이 발생 했고 2011 년 7 월에는 온라인 커뮤니티 네이트와 싸이월드가 해킹 당해 3500 만명의 개인정보가 유출되었다. 온라인 게임 기업 넥슨에서도 1322 만명의 개인정보가 새어나갔다.

농협과 현대캐피털, 삼성카드 등 금융권에서도 개인정보 유출 사고가 발생했다. 개인정보 유출 사고로 인해 기업들은 신뢰라는 이미지에 큰 금이 갔고 개인들 에게도 2 차 피해가 발생 할 우려가 생겼다.

유출 된 개인정보는 스팸의 표적이 되며 계정 도용이나 보이스 피싱, 최근에는 스미싱 등 범죄행위에도 악용되고 있다. 개인정보 침해 및 도용으로 인한 피해는 측정이 어려우며 한 번 유출된 개인 정보는 회수가 사실상 불가능하기 때문에 더욱 심각하다.

아래는 안전행정부와 한국인터넷진흥원에서 발행한『2012 년 개인정보 보호 상담 사례집』출처의 자료로 2007 년부터 2012 년 까지 개인정보 침해신고 및 상담 접수 현황에 대해 나타낸 표이다.

【연도별·유형별 개인정보 침해신고 및 상담 접수 현황】

접수 유형	2007년	2008년	2009년	2010년	2011년	2012년
개인정보 수집 요건	1,166	1,129	1,075	1,267	1,623	3,507
개인정보 수집시 고지·명시 의무	7	6	15	75	53	396
과도한 개인정보 수집	51	87	115	146	379	847
목적 외 이용 또는 제3자 제공	1,001	1,037	1,171	1,202	1,499	2,196
개인정보취급자에 의한 훼손·침해 등	123	125	158	158	278	941
개인정보 처리 위탁	2	6	6	25	36	125
영업 양도·양수	14	9	6	22	64	44
개인정보 보호책임자	10	26	10	21	38	48
개인정보 안전성 확보조치	522	1,321	819	1,551	10,958	3,855
개인정보 미파기	146	294	294	323	488	779
정보주체 권리 (열람, 정정요구 등)	865	949	680	826	662	717
열람·정정을 수집보다 쉽게 해야할 조치	461	503	603	630	800	660
아동 개인정보 수집	14	27	19	35	71	47
주민등록번호 등 타인 정보 훼손·침해·도용	9,086	10,148	6,303	10,137	67,094	139,724
타 법 관련 개인정보 사례	12,497	24,144	23,893	38,414	38,172	12,915
계	**25,965**	**39,811**	**35,167**	**54,832**	**122,215**	**166,801**

시간이 흐를수록 개인정보 침해사례가 늘어나고 있음을 확인 할 수 있으며 표시된 곳을 보면 '주민등록번호 등 타인정보의 훼손, 침해, 도용' 사례가 가장 많이 발생 하고 있음을 알 수 있다. 아직 우리 사회에서 주민등록번호와 같은 타인의 개인정보 도용 사례가 많이 발생하고 있음을 보

여주고 있다. 이처럼 급증하는 개인정보 침해 문제에 대한 예방과 해결을 위해 개인정보 보호법이란 법규를 시행하게 되었다.

개인정보 보호법과 데이터베이스

개인정보 보호법은 다각적 측면에서 기술되어 있는데 데이터베이스 측면에서 관리되고 적용하여야 할 부분은 29 조 안전조치의무를 참조 할 수 있다. 아래는 행정안전부의『개인정보 보호법령 및 지침, 고시 해설』의 일부로 개인정보 보호법 29 조의 법령과 시행령에 대해 나타낸 표이다.

제29조 안전조치의무

법률	제29조(안전조치의무) 개인정보처리자는 개인정보가 분실·도난·유출·변조 또는 훼손되지 아니하도록 내부 관리계획 수립, 접속기록 보관 등 **대통령령**으로 정하는 바에 따라 안전성 확보에 필요한 기술적·관리적 및 물리적 조치를 하여야 한다.
시행령	제30조(개인정보의 안전성 확보 조치) ① 개인정보처리자는 법 제29조에 따라 다음 각 호의 안전성 확보 조치를 하여야 한다. 1. 개인정보의 안전한 처리를 위한 내부 관리계획의 수립·시행 2. 개인정보에 대한 접근 통제 및 접근 권한의 제한 조치 3. 개인정보를 안전하게 저장·전송할 수 있는 암호화 기술의 적용 또는 이에 상응하는 조치 4. 개인정보 침해사고 발생에 대응하기 위한 접속기록의 보관 및 위조·변조 방지를 위한 조치 5. 개인정보에 대한 보안프로그램의 설치 및 갱신 6. 개인정보의 안전한 보관을 위한 보관시설의 마련 또는 잠금장치의 설치 등 물리적 조치 ② 행정안전부장관은 개인정보처리자가 제1항에 따른 안전성 확보 조치를 하도록 시스템을 구축하는 등 필요한 지원을 할 수 있다. ③ 제1항에 따른 안전성 확보 조치에 관한 세부 기준은 행정안전부장관이 정하여 고시한다.

위 시행령에서 블록 표시 된 2, 3, 4 번에 대해 데이터베이스 보안 측면에서 조치가 가능하다. 시행령 2, 3, 4 는 순서대로 접근 제어, 개인 정보 암호화, 접속기록 관리로 구분 할 수 있다. 지금부터 위 시행령에 대한 조치사항을 DBA 관점에서 DBMS 중 하나인 SQL Server 를 통해 정리해 보고자 한다.

SQL Server에서의 개인정보 보호

SQL Server 는 개인정보보호법에서 말하는 개인정보의 안전조치의무에 대해 강력한 보안 기능으로 대응 할 수 있다. 여기에서는 개인정보 안전조치의무와 관련된 방송통신위원회 고시 제 2012-50 호(2012. 8. 23.)『개인정보의 기술적, 관리적 보호조치 기준』을 기반으로 SQL Server 의 보안 기능을 사용해 적용하는 방법을 알아 보도록 하겠다.

본문에 앞서 실습 Database 와 Table 을 생성한다.

[SQL Script] – 실습 Database 생성

```
CREATE DATABASE DB_Security
GO

USE DB_Security
GO

-- 접근 제어 실습 테이블
IF OBJECT_ID('dbo.ValidIP', 'U') IS NOT NULL
            DROP TABLE dbo.ValidIP
GO

CREATE TABLE dbo.ValidIP (
  IP NVARCHAR(15)
)
GO

-- 암호화 실습 테이블
IF OBJECT_ID('dbo.Personal_Info', 'U') IS NOT NULL
            DROP TABLE Personal_Info
GO

CREATE TABLE Personal_Info(
            Name      VARCHAR(20),
```

```
            IDNum1  VARCHAR(6),
            IDNum2  VARCHAR(7),
            Pass    VARCHAR(20)
)
GO

INSERT INTO Personal_Info(Name, IDNum1, IDNum2, Pass)
            VALUES('김연아', '900905', '2345678', 'rladusdk')
INSERT INTO Personal_Info(Name, IDNum1, IDNum2, Pass)
            VALUES('손연재', '940528', '2987654', 'thsduswo')
INSERT INTO Personal_Info(Name, IDNum1, IDNum2, Pass)
            VALUES('류현진', '870325', '1234567', 'dirndhkd')
INSERT INTO Personal_Info(Name, IDNum1, IDNum2, Pass)
            VALUES('추신수', '820713', '1234567', 'dirndhkd')
GO
```

접근 제어

접근제어는 개인정보처리시스템에 대해 허가 받지 않은 사용자의 접근을 제어하여 개인정보 침해를 방지 하는 것이다. 개인정보관리책임자와 개인정보취급자에 대해 개인정보에 대한 접근권한을 부여하고 그렇지 않은 경우 접근에 대한 제약을 두어야 한다. SQL Server 는 로그인 계정과 데이터베이스 사용자 계정에 대한 인증과 권한 제어를 통해 불법 접근 및 침해에 대한 접근제어가 가능하다. 쉬운 예로 Windows 공유폴더에 대한 사용자 등록과 등록된 사용자에 대해 읽기만 가능하게 할지 읽기/쓰기를 모두 가능하게 할지 제어하는 것도 접근제어라 할 수 있다.

1.1 단계별 접근 제어

SQL Server 접근에는 인스턴스(이하 서버), 데이터베이스, 테이블로 Windows 의 탐색기와 같은 구조로 되어있다. SQL Server 에서는 서버 수준 통제부터 테이블 내 열 수준 까지도 통제가 가능하다. 먼저 가장 상위에 있는 서버부터 알아보도록 하자.

서버 수준 접근 제어

서버 접근에는 기본적으로 두 가지 접속 유형이 존재한다. Windows 계정 인증과 SQL Server 계정 인증이다. Windows 계정 인증은 도메인 계정이나 로컬 서버의 개인 계정 및 그룹 계정을 SQL Server 로그인 계정으로 등록 할 수 있고 등록 된 계정만 SQL Server 에 로그인 할 수 있다. 도메인 또는 로컬 서버에 로그인 할 수 없는 경우 SQL Server 자체 계정을 만들어 접근 할 수 있다. Windows 계정과 다르게 SQL Server 등록 시에는 암호를 지정해 주어야 한다.

[SQL Script] – 서버 로그인 계정 등록

```
USE master
GO

-- Windows 개인 계정 등록하기
CREATE LOGIN [CRAZYCHOON₩SQLAdmin] -- 해당 계정이 존재해야 함
        FROM WINDOWS
GO

-- Windows 그룹 계정 등록하기
CREATE LOGIN [CRAZYCHOON₩SQLGroup] -- 해당 그룹이 존재해야 함
        FROM WINDOWS
GO

-- SQL Server 계정 만들기
CREATE LOGIN SQLLogin
        WITH PASSWORD = 'Pa$$w0rd', CHECK_EXPIRATION=ON,
CHECK_POLICY=ON
GO
```

[개체 탐색기 Path]

Windows 인증과 SQL Server 인증 로그인 등록 순서

1. 개체 탐색기 → 보안 → 로그인(우 클릭) → 새 로그인(N) → Windows 인증(N)
2. 개체 탐색기 → 보안 → 로그인(우 클릭) → 새 로그인(N) → SQL Server 인증(S)

지정된 컴퓨터의 로컬 또는 도메인에 등록된 계정에 대해서만 사용 가능 한 Windows 인증이 SQL Server 인증 보다 보안에 더 강력하다. 하지만 업무상 불편함이 발생 할 수 있다. 다행히도 로그인 유형에 대한 제약사항은 없다. 필요의 경우 SQL Server 인증 로그인을 사용 하되 비밀번호 작성 규칙을 준수하도록 한다.

데이터베이스 수준 접근 제어

SQL Server 는 하나의 서버에 여러 개의 데이터베이스를 운용 할 수 있는 특징이 있다. 각 로그인 계정에 대해 접근을 필요로 하는 데이터베이스에만 접근 권한을 주도록 한다. 해당 데이터베이스에서 접근을 허용 할 로그인을 매핑 해주면 된다.

[SQL Script] – 데이터베이스 계정 매핑

```
USE DB_Security
GO

-- Windows 계정 매핑
CREATE USER [CRAZYCHOON₩SQLAdmin]
            FOR LOGIN [CRAZYCHOON₩SQLAdmin]
GO

-- Windows 그룹 계정 매핑
CREATE USER [CRAZYCHOON₩SQLGroup]
```

```
        FOR LOGIN [CRAZYCHOON₩SQLGroup]
GO

-- SQL Server 계정 매핑
CREATE USER SQLLogin
        FOR LOGIN SQLLogin
GO
```

[개체 탐색기 Path]

데이터베이스 접근 권한 설정 방법 두 가지

1. 개체 탐색기 → 보안 → 로그인 → 대상 아이디 우 클릭 → 속성(R) → 사용자 매핑
2. 개체 탐색기 → 대상 데이터베이스 → 보안 → 사용자 우 클릭 → 새 사용자(N)

첫 번째 방법은 특정 로그인 계정에 대해 여러 데이터베이스에 대한 접근을 제어 할 경우 유용하게 사용되고 두 번째 방법은 특정 데이터베이스에 대해 접근 가능한 계정을 하나씩 설정하는 방식이다. 결과는 동일하다.

데이터베이스 속성의 사용권한에서 데이터베이스에 매핑 된 로그인에 권한 부여 및 제거가 가능하다. 또한 데이터베이스 보안 → 사용자에서 아이디를 우 클릭 하면 나오는 멤버 자격에서도 권한을 줄 수 있는데 데이터베이스 속성의 사용권한은 기능 하나하나에 대한 권한 부여라면 멤버 자격에서는 역할에 맞는 몇 가지 기능이 합쳐진 묶음이라 볼 수 있다.

테이블 수준 접근 제어

데이터베이스 사용자가 테이블과 같은 개체에 대한 접근 권한이 필요한 경우 명시적으로 권한을 부여해 주어야 한다. SQL Server 에서는 사용자 개체에 대해서 권한을 갖기 위해 명시적으로 권한을 부여(GRANT) 받아야 하고 어떤 경로를 통해서든 권한이 거부(DENY) 되어서는 안 된다.

가장 기본적이고 보안상 안전한 방법은 사용자가 접근하고자 하는 개체에 대해 개별적으로 권한을 부여하는 것이다. 테이블 접근 권한을 설정하다 보면 스키마가 나오는데 연관된 테이블들을 스키마로 묶어주어 그룹화 하는 개념으로 알면 된다. 사용자 별 접근 가능한 테이블들을 스키마 수준에서 통제 할 수 있다. 디폴트 스키마를 지정하지 않은 경우 dbo 스키마가 기본 스키마가 된다.

[SQL Script] – 테이블 수준 권한 설정

```
USE DB_Security
GO
-- 테이블에 SELECT, INSERT, UPDATE, DELETE 권한 부여
GRANT SELECT ON dbo.Personal_Info TO SQLLogin
GO
GRANT INSERT ON dbo.Personal_Info TO SQLLogin
GO
GRANT UPDATE ON dbo.Personal_Info TO SQLLogin
GO
GRANT DELETE ON dbo.Personal_Info TO SQLLogin
GO

-- 테이블에 UPDATE, DELETE 권한 제거
REBOKE UPDATE ON dbo.Personal_Info TO SQLLogin
GO
REBOKE DELETE ON dbo.Personal_Info TO SQLLogin
GO

* ON [테이블명] 을 제외하면 데이터베이스의 모든 테이블에 대한 권한을 부여하거나 제거 할 수 있다.
```

[개체 탐색기 Path]

테이블 접근 권한 설정 방법

개체 탐색기 → 대상 데이터베이스 → 테이블 → 대상 테이블 우 클릭 → 속성(R) → 사용 권한 페이지

열 수준 접근 제어

SQL Server 는 열 수준 까지 권한을 제어 할 수 있는 기능을 제공한다. 열 수준의 권한을 사용하면 사용자가 필요로 하는 열의 데이터만 접근 가능하게 하고, 나머지 열에 대해서는 접근을 차단할 수 있다. 방법은 테이블 접근 제어와 같고 다만 '테이블(열이름)'으로 접근 대상만 변경된다.

[SQL Script] – 열 수준 권한 설정

```
USE DB_Security
GO

GRANT UPDATE ON dbo.Personal_Info(Name) TO SQLLogin -- Name열 UPDATE 권한 부여
GO
REVOKE UPDATE ON dbo.Personal_Info(NAME) TO SQLLogin -- Name열 UPDATE 권한 해제
GO
```

[개체 탐색기 Path]

열 접근 권한 설정 방법
개체 탐색기 → 대상 데이터베이스 → 테이블 → 대상 테이블 우 클릭 → 속성(R) → 사용 권한 탭 → SELECT 권한 부여 → 열 사용 권한 선택

1.2 트리거를 이용 한 IP 접근제어

트리거를 이용해 IP 를 통제하여 접근제어 하는 방법이 있다. 단계별 접근제어에서는 서버 수준의 접근제어 방법이라 할 수 있다.

[SQL Script] – 트리거를 이용 해 허용된 IP만 접근

```
USE DB_Security
GO

-- 접근 가능 IP INSERT
INSERT INTO dbo.ValidIP (IP) VALUES ('<local machine>')
INSERT INTO dbo.ValidIP (IP) VALUES ('192.168.123.187')
GO

USE master
GO

--DDL Trigger 생성
CREATE TRIGGER tr_logon_CheckIP
ON ALL SERVER
FOR LOGON
AS
BEGIN
  IF IS_SRVROLEMEMBER('sysadmin') = 1
  BEGIN
   DECLARE @IP NVARCHAR(15)
   SET @IP = (SELECT EVENTDATA().value('(/EVENT_INSTANCE/ClientHost)[1]',
'NVARCHAR(15)'))
   IF NOT EXISTS(SELECT IP FROM  DB_Security.dbo.ValidIP WHERE IP = @IP)
     ROLLBACK
  END
END
GO
```

위 코드에서는 접근을 허용 할 IP 정보를 갖고 있는 ValidIP 테이블을 생성하여 허용 된 IP 에서만 접근을 하도록 했다. 반대로 접근을 차단 하는 것도 쉽게 응용할 수 있다. 로컬의 경우 로컬

IP 만 삽입하면 접근이 막힌다. 로컬에서 SSMS 접속을 할 경우 '<local machine>'으로 IP 정보를 받아 온다. 이 점에 유의하도록 한다.

1.3 서버 역할과 데이터베이스 역할

역할은 권한들을 그룹화 한 의미 있는 집합 그룹이라 할 수 있다. 사용자 계정을 이 역할의 멤버로 포함시키면 역할에 포함 된 권한들이 모두 사용자 계정에 부여 된다. 역할은 권한관리를 편리하게 해 주어 관리자의 업무 부담을 덜어준다. 하지만 역할에 따라 여러 중요한 권한을 갖고 있기 때문에 사용자 계정에 역할을 부여 할 때 신중을 기해야 한다.

<고정 서버 역할>

서버 역할	권한
sysadmin	모든 서버 작업을 수행할 수 있다.
serveradmin	서버 차원의 구성 옵션을 변경하고 서버를 종료할 수 있다.
securityadmin	로그인 및 해당 속성을 관리한다. 서버 수준의 사용 권한을 부여(GRANT), 거부(DENY) 및 취소(REVOKE)할 수 있다. 또한 데이터베이스에 대한 액세스 권한이 있는 경우 데이터베이스 수준의 사용 권한을 부여(GRANT), 거부(DENY) 및 취소(REVOKE)할 수도 있으며, SQL Server의 로그인 암호를 다시 설정할 수 있다.
processadmin	SQL Server 인스턴스에서 실행 중인 프로세스를 종료할 수 있다.
setupadmin	연결된 서버를 추가하거나 제거할 수 있다.
bulkadmin	BULK INSERT 문을 실행할 수 있다.
diskadmin	디스크 파일을 관리하는 데 사용된다.

dbcreator	데이터베이스를 생성, 변경, 삭제 및 복원할 수 있다.
public	모든 SQL Server 로그인은 public 서버 역할에 속한다. 다른 역할과는 달리 public에서 사용 권한을 부여, 거부 또는 취소할 수 있다.

<고정 데이터베이스 역할>

데이터베이스 역할	권한
db_owner	데이터베이스에서 모든 구성 및 유지 관리 작업을 수행할 수 있고 데이터베이스를 삭제할 수도 있다.
db_securityadmin	역할 멤버 자격을 수정하고 사용 권한을 관리할 수 있다.
db_accessadmin	Windows 로그인, Windows 그룹 및 SQL Server 로그인의 데이터베이스에 대한 액세스를 추가하거나 제거할 수 있다.
db_backupoperator	데이터베이스를 백업할 수 있다.
db_ddladmin	데이터베이스에서 모든 DDL(데이터 정의 언어) 명령을 실행할 수 있다.
db_datawriter	모든 사용자 테이블에서 데이터를 추가, 삭제 또는 변경할 수 있다.
db_datareader	모든 사용자 테이블의 모든 데이터를 읽을 수 있다.
db_denydatawriter	데이터베이스 내의 사용자 테이블에 있는 데이터를 추가, 수정 또는 삭제할 수 없다.
db_denydatareader	데이터베이스 내에 있는 사용자 테이블의 데이터를 읽을 수 없다.
public	모든 데이터베이스 사용자는 public 데이터베이스 역할에 속한다. 사용자에게 보안 개체에 대한 특정 사용 권한이 부여되지 않았거나 거부된 경우 사용자는 해당 보안 개체에 대해

	public으로 부여된 사용 권한을 상속 받는다.

위에서 정리 한 고정 서버 역할과 고정 데이터베이스 역할은 SQL Server 에 Default 로 존재하는 역할이다. SQL Server 2012 부터는 이 고정 역할 이외에도 사용자가 직접 역할을 생성하여 권한을 줄 수 있다. 예를 든다면 Windows 의 사용자 그룹에서 새 그룹을 생성해주는 것과 동일하다. 이 기능은 서버 혹은 데이터베이스의 보안의 역할 에서 생성할 수 있다.

앞서 데이터베이스 속성의 사용권한에 대해서 알아봤었다. 개인적인 생각으로는 사용자에게 멤버 자격을 부여하면 자동으로 데이터베이스 속성의 사용권한에서도 부여된 멤버 자격에 대한 사용 권한이 체크되어 있을 것으로 생각 했지만 멤버 자격을 부여해도 사용자에 대한 명시적 사용권한은 체크되지 않았다.

개인 정보 암호화

개인 정보 암호화는 고객의 중요한 개인 정보를 안전한 암호 알고리즘을 사용해 암호화하여 개인정보 침해를 방지하도록 하는 기술이다. SQL Server 에서는 열 수준 암호화 방법과 데이터베이스 수준 암호화 방법을 제공한다.

『개인정보의 기술적, 관리적 보호조치 기준』을 참고하면 비밀번호 및 바이오 정보는 복호화 되지아니하도록 일방향 암호화하여 저장하고 주민등록번호, 신용카드번호 및 계좌번호에 대해서는 안전한 암호 알고리즘으로 암호화하여 저장한다. 라고 명기되어 있다.

이 장에서는 위 두 가지 조건에 부합하는 개인 정보 암호화에 대한 내용을 다루었다.

2.1 데이터 암호화(칼럼 암호화)

데이터를 암호화 할 경우 암호화 된 열의 사이즈가 증가되고 인덱스 생성 및 정렬이 불가능해 지는 등 암호화 전보다 시스템에 부하가 가중 될 수 있다. 관리적 측면에서도 불편함이 따를 수 있다. 그러나 중요 데이터를 보호하기 위해서는 어느 정도 성능문제와 불편함을 감수 해야 한다.

암호화를 할 때 어떤 데이터에 대해 암호화를 해야 하는지를 명확히 해야 한다. 미리 언급한대로 데이터 암호화는 데이터베이스 성능의 저하를 가져 올 수 있기 때문에 무분별한 암호화는 지양해야 한다. 개인 정보 보호법에 따라 필히 암호화 해야 하는 정보는 다음과 같다.

일방향 암호화 데이터	안전한 알고리즘 사용 암호화 데이터
비밀번호 바이오 정보(지문, 홍채 등)	주민등록번호 신용카드번호 계좌번호

여기서 말하는 일방향 암호화와 안전한 알고리즘을 사용한 암호화는 다음과 같이 정의 한다.

일방향 암호화

단방향 암호화(one-way encryption)라고도 불린다. 암호화는 일반적으로 평문을 암호화하여 암호문을 만들 수 있고 이렇게 만들어진 암호문을 복호화를 통해 평문으로 전환할 수 있는데 단방향 암호화는 암호문에서 다시 평문으로 만들 수 없는 암호화를 말한다.

안전한 알고리즘

국내외 전문기관 (KISA, NIST, ECRYPT, CRYPTREC)에서 권고 하고 있는 알고리즘을 말한다.

<안전한 암호 알고리즘(예시) – 행정안전부 발행『개인정보 암호화 조치 안내서』>

구분	알고리즘 명칭
대칭키 암호 알고리즘	SEED ARIA-128/192/256 AES-128/192/256 Blowfish Camelia-128/192/256 MISTY1 KASUMI 등
공개키 암호 알고리즘	RSA KCDSA(전자서명용) RSAES-OAEP RSAES-PKCS1 등
일방향 암호 알고리즘	SHA-224/256/384/512 Whirlpool 등

데이터를 암호화 하는 방법은 대칭키, 비대칭키, 인증서를 사용한 암호화 및 복호화와 일방향 암호화 방식으로 복호화가 불가능 한 HASBYTES 를 사용 한 암호화 방법이 있다. SQL Server 에서는 여러 암호화 함수를 제공하고 있는데 아래 링크를 통해 더 자세한 내용을 확인 할 수 있다.

- http://technet.microsoft.com/ko-kr/library/ms173744(v=sql.110).aspx

『개인정보 암호화 조치 안내서』에서는 아래와 같이 주민등록번호에 대해 일부 정보만 암호화 조치 할 수 있다고 명기하고 있다.

■ 주민등록번호를 시스템 운영을 위한 검색 키로 사용하는 경우, 속도 등 성능을 고려하여 일부 정보만 암호화 조치를 취할 수 있다.

※ 주민등록번호의 경우 뒷자리 6개 번호 이상 암호화 조치 필요(예시: 700101-1#……&)

데이터 암호화 방식 중 대칭키 방법을 이용하여 위와 같이 주민등록번호의 일부만을 암호화 조치하는 방법을 알아보도록 하겠다. 함께 일방향으로 비밀번호를 암호화 하는 방법까지 알아보도록 하겠다.

[SQL Script] – 칼럼 암호화

```
USE DB_Security
GO

SELECT Name, IDNum1, IDNum2, Pass
        FROM dbo.Personal_Info
GO
```

데이터 암호화 연습을 위해 미리 생성해 놓은 테이블을 확인해보자.

	Name	IDNum1	IDNum2	Pass
1	김연아	900905	2345678	rladusdk
2	손연재	940528	2987654	thsduswo
3	류현진	870325	1234567	dirndhkd
4	추신수	820713	1234567	dirndhkd

빈약하지만 연습용 테이블인 Personal_Info 테이블이 가상의 고객정보 테이블이라 생각 하고 다음과 같은 순서로 암호화 연습을 하고 결과를 확인해 보도록 하겠다.

(1) IDNum2열에서 성별 정보를 확인 할 수 있는 첫 자리를 분리한다.

(2) IDNum2열을 대칭키를 사용해 암호화 한다.

(3) Pass열을 HASHBYTES를 사용해 일방향 암호화 한다.

(4) 주민등록번호를 조건으로 복호화 하여 이름을 검색해본다.

(5) 대칭키 암호화 된 IDNum2열과 HASHBYTES암호화 된 Pass열의 결과를 확인해 본다.

[SQL Script] – 암호화 실습

```
-- 1. 주민등록번호 뒤쪽(IDNum2열)의 첫 자리(성별정보) 분리
USE DB_Security
GO

ALTER TABLE dbo.Personal_Info
        ADD SEX CHAR(1)
GO

UPDATE dbo.Personal_Info
        SET SEX = LEFT(IDNum2, 1)
GO
```

	Name	IDNum1	IDNum2	Pass	SEX
1	김연아	900905	2345678	rladusdk	2
2	손연재	940528	2987654	thsduswo	2
3	류현진	870325	1234567	dirndhkd	1
4	추신수	820713	1234567	dirndhkd	1

```
-- 2. 주민등록번호 뒤쪽(IDNum2열) 대칭키 암호화
CREATE SYMMETRIC KEY KEY01
        WITH ALGORITHM = AES_256
        ENCRYPTION BY PASSWORD = 'Pa$$w0rd'
GO

OPEN SYMMETRIC KEY KEY01
        DECRYPTION BY PASSWORD = 'Pa$$w0rd'
GO

ALTER TABLE dbo.Personal_Info
        ADD EncryptedIDNum2 varbinary(256)
GO
UPDATE dbo.Personal_Info
        SET EncryptedIDNum2 = ENCRYPTBYKEY(KEY_GUID('KEY01'),IDNum2)
```

```
GO

ALTER TABLE dbo.Personal_Info
        DROP COLUMN IDNum2
GO

EXEC sp_rename 'dbo.Personal_Info.EncryptedIDNum2', 'IDNum2', 'COLUMN'
GO /* 암호화 칼럼 기존 칼럼의 칼럼명으로 변경 */
```

	Name	IDNum1	IDNum2	Pass	SEX
1	김연아	900905	0x006C664304AED34386D2D3D380F5999201000000DF6...	rladusdk	2
2	손연재	940528	0x006C664304AED34386D2D3D380F5999201000000F3C7...	thsduswo	2
3	류현진	870325	0x006C664304AED34386D2D3D380F5999201000000F7A6...	dirndhkd	1
4	추신수	820713	0x006C664304AED34386D2D3D380F5999201000000EAD...	dirndhkd	1

```
-- 3. 패스워드 열 일방향 암호화(Pass열 HASHBYTES함수로 암호화)
ALTER TABLE dbo.Personal_Info
        ADD EncryptedPass varbinary(256)
GO

UPDATE dbo.Personal_Info
        SET EncryptedPass = HASHBYTES('SHA2_256', Pass)
GO

ALTER TABLE dbo.Personal_Info
        DROP COLUMN Pass
GO

EXEC sp_rename 'dbo.Personal_Info.EncryptedPass', 'Pass', 'COLUMN'
GO
```

	Name	IDNum1	IDNum2	Pass	SEX
1	김연아	900905	0x006C664304...	0xB4E9C5CEBBAD3246DC1774C0D04EB6BBC4...	2
2	손연재	940528	0x006C664304...	0x27D2183398F26952DADA0F124F01892DF5A8B...	2
3	류현진	870325	0x006C664304...	0x8BBC4D9A9F7A33F71805526404973C566DB9...	1
4	추신수	820713	0x006C664304...	0x8BBC4D9A9F7A33F71805526404973C566DB9...	1

```
-- 4. 주민등록번호를 조건으로 복호화 하여 검색
SELECT Name, IDNum1, IDNum2, Pass, SEX
        FROM dbo.Personal_Info
        WHERE IDNum1 = 900905 AND CONVERT(varchar, DecryptByKey(IDNum2))
= 2345678
GO -- 김연아의 주민등록번호를 조건으로 SELECT
```

	Name	IDNum1	IDNum2	Pass	SEX
1	김연아	900905	0x006C664304...	0xB4E9C5CEBBAD3246DC1774C0D04EB6BBC4...	2

```
-- 5. 암호화 후 테이블 확인
SELECT Name, IDNum1, CONVERT(varchar, DecryptByKey(IDNum2)) AS IDNum2, Pass,
SEX
        FROM dbo.Personal_Info
GO -- 대칭키 복호화 하여 SELECT (HASHBYTES는 복호화 못 함)
```

	Name	IDNum1	IDNum2	Pass	SEX
1	김연아	900905	2345678	0xB4E9C5CEBBAD3246DC1774C0D04EB6BBC4E4D4B1A...	2
2	손연재	940528	2987654	0x27D2183398F26952DADA0F124F01892DF5A8B4F21B564...	2
3	류현진	870325	1234567	0x8BBC4D9A9F7A33F71805526404973C566DB9D599254A2...	1
4	추신수	820713	1234567	0x8BBC4D9A9F7A33F71805526404973C566DB9D599254A2...	1

HASHBYTES 의 경우 같은 평문을 암호화 하면 결과도 같아지는 반면 대칭키는 암호화 결과 값이 다르게 나타난다. Password 의 경우 평문 Password 를 입력 받아 HASHBYTES 로 암호화 하여 Database 에 암호화 되어 있는 Password 와 비교 할 수 있다. 실습해본 테이블을 처음부터 주의 깊게 봤다면 3 번째 행 류현진과 4 번째 행 추신수의 주민등록번호 뒷자리와 패스워드가 같은 것을 눈치 챘을 것이다. 아래 암호화 된 데이터를 보면 대칭키를 사용해 암호화 한 주민등록 번호 뒷자리는 서로 다르고 HASHBYTES 를 사용해 암호화 한 패스워드는 서로 같음을 알 수 있다.

대칭키를 사용 한 암호화 (류현진과 추신수의 IDNum2 는 같은 값이었지만 암호화 결과가 다르다)

	Name	IDNum2
1	김연아	0x006C664304AED34386D2D3D380F5999201000000DF6B7689A75B4C46F6225926C4...
2	손연재	0x006C664304AED34386D2D3D380F5999201000000F3C7A6482E52CCFBE1C0DBFA9...
3	류현진	0x006C664304AED34386D2D3D380F5999201000000F7A6C9E2484A085C337A5EBB0D...
4	추신수	0x006C664304AED34386D2D3D380F5999201000000EADE3E66936D904BCAE5F4F608...

HASHBYTES 를 사용 한 암호화 (류현진과 추신수의 Pass 가 암호화 전과 후 모두 같다)

	Name	Pass
1	김연아	0xB4E9C5CEBBAD3246DC1774C0D04EB6BBC4E4D4B1A6FBC893B6657B00F6534710
2	손연재	0x27D2183398F26952DADA0F124F01892DF5A8B4F21B564387EDC00C33440CED7D
3	류현진	0x8BBC4D9A9F7A33F71805526404973C566DB9D599254A2271DDC2C96FA92A6091
4	추신수	0x8BBC4D9A9F7A33F71805526404973C566DB9D599254A2271DDC2C96FA92A6091

간단한 실습이었지만 차근차근 이해하며 따라 한다면 암호화에 대한 감은 조금 잡을 수 있지 않을까 생각한다. 비밀번호는 왜 HASHBYTES 를 이용하여 일방향 암호화를 하는 것인지, 암호화를 통해 어느 정도의 부하가 가중 될 지, 관리자 입장에서 얼만큼의 수고로움이 될지 짐작이라도 할 수 있게 된다면 좋겠다.

2.2 데이터베이스 암호화(TDE)

위에서 알아본 칼럼 암호화를 모든 칼럼에 적용하기는 쉽지 않다. 일일이 암호화 하는 것도 힘들 뿐더러 설령 모두 암호화 하더라도 운영에 많은 불편함이 따른다. 칼럼에 접근 할 때 마다 복호화 해야 하는 번거로움이 발생 하기 때문이다. 필요한 칼럼에만 암호화를 하면 문제 없다고 생각할 수 있겠지만 이 같은 경우 데이터 파일이나 DB 백업 파일이 유출되면 암호화된 칼럼 이외 모든 정보가 유출되는 일이 발생한다. 이런 문제를 해결하고자 나온 기술이 바로 데이터베이스 암호화 기술인 TDE 다.

TDE 를 이용해 데이터베이스 수준의 암호화를 하게 되면 데이터베이스 백업 파일이나 원본파일이 외부로 유출된다 하더라도 복원이나 연결이 차단되어 정보유출 문제를 방지 할 수 있다. 암호화 및 복호화 함수를 사용하는 것이 아니라 데이터베이스 엔진이 암호화 및 복호화를 자동으로 수행해주기 때문에 응용프로그램이나 관련 쿼리를 수정할 필요가 없다. 때문에 TDE(Transparent Data Encryption), 투명한 데이터 암호화 라고 한다.

『개인정보 암호화 조치 안내서』를 참고하면 일반기업, 공공기관, 금융기관에 대해 TDE 방식을 사용한 암호화가 개인정보보호법에 위반되지 않다고 안내하고 있다.

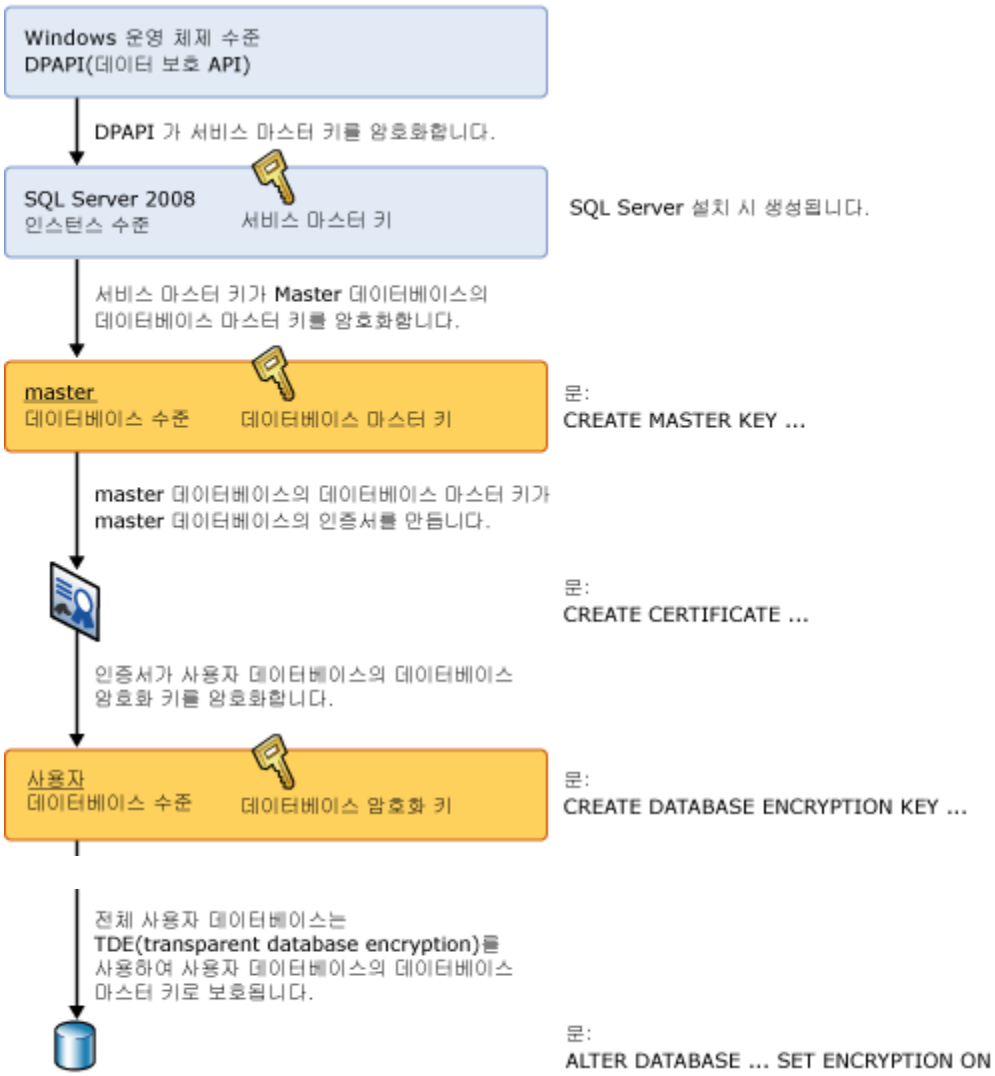
TDE 사용시 데이터베이스 엔진에 의해 자동으로 암호화, 복호화 된다. TDE 암호화는 페이지 수준에서 수행되고 메모리로 읽어 들일 때 암호가 해독된다. TDE 암호화는 자동으로 암호화 복호화 되는 장점과 페이지 단위 암호화로 데이터베이스 크기가 커지지 않는 장점이 있지만, CPU 사용량이 증가하는 단점이 있다. 이런 점을 숙지하고 암호화로 인한 문제가 발생하지 않도록 정확한 성능 측정이 필요하다. SQL Server 에서 TDE 는 2008 버전 이상, Enterprise Edition 이상부터 지원한다.

데이터베이스 암호화(TDE) 하기

데이터베이스 암호화는 아래와 같은 순서로 수행된다.

데이터베이스 암호화(TDE) 순서
마스터 키 만들기 → 인증서 만들기 → 데이터베이스 암호화 키 만들기 → 데이터베이스 암호화 속성 설정(ON)

아래는 MSDN 에서 인용한 그림으로 TDE 암호화에 대한 아키텍처이다.



위 아키텍처 그림에서 우측에 보이는 4 단계가 TSQL 코드가 TDE 암호화 순서이다. 앞의 두 단계는 master 데이터베이스에서 수행해야 하고 나머지 두 단계는 암호화 대상 데이터베이스에서 수행해야 한다.

[SQL Script] – 암호화(TDE)

```
-- 1. 마스터 키 만들기
USE master
GO

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Pa$$w0rd'
GO

-- 2. 인증서 만들기
CREATE CERTIFICATE DB_Cert WITH SUBJECT = 'DB_Security Certificate'
GO

-- 3. 데이터베이스 암호화 키 만들기

USE DB_Security
GO

CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128 /* AES_128, AES_192, AES_256, TRIPLE_DES_3KEY
알고리즘 사용 가능 */
ENCRYPTION BY SERVER CERTIFICATE DB_Cert
GO

-- 4. 데이터베이스 암호화 속성 변경(활성화)
ALTER DATABASE DB_Security
SET ENCRYPTION ON
GO

--※ 데이터베이스 암호화 과정 확인 스크립트
SELECT DB_NAME(database_id) AS 'db_name', encryption_state
FROM sys.dm_database_encryption_keys
GO
```

DMV sys.dm_database_encryption_keys의 encryption_state열(암호화 과정) 상태 표

값	설 명
0	데이터베이스 암호화 키가 없고 암호화되지 않음
1	암호화되지 않음
2	암호화 진행 중
3	암호화 됨
4	키 변경 진행 중
6	해동 진행 중
7	보호 변경 진행 중. 데이터베이스 암호화 키를 암호화하는 인증서 또는 비대칭 키를 변경하고 있습니다.

<3. 데이터베이스 암호화 키 만들기>를 실행시키면 인증서와 개인키가 백업되지 않았다는 경고 메시지가 출력된다. TDE 를 통한 데이터베이스 암호화 시 관련된 인증서를 필히 백업 받아 철저히 관리해야 한다.

[SQL Script] – 인증서와 개인 키 백업

```
USE master
GO

BACKUP CERTIFICATE DB_Cert TO FILE = 'D:₩Backup₩DB_Cert'
WITH PRIVATE KEY (FILE = 'D:₩Backup₩key', ENCRYPTION BY PASSWORD = 'Pa$$w0rd')
GO
```

위와 같이 인증서와 개인 키를 백업 할 수 있다. 인증서와 키 파일은 별도의 확장자 없이도 저장 가능하고 임의의 확장자를 지정해도 무관하다. 단, 일반 백업과 마찬가지로 파일이 저장되는 경로에는 스크립트에서 지정해 준 폴더가 생성되어 있어야 한다.

[SQL Script] - TDE암호화 된 DB의 백업과 복원

```
-- 1. 백업
BACKUP DATABASE DB_Security
            TO DISK = 'D:₩Backup₩DB_Security_Full.bak'
            WITH INIT
GO

-- ※ 기존과 다른 인스턴스에서 아래 작업 수행
-- 2. 복원
USE master – 기존과 다른 인스턴스의 마스터에서 실행
GO

RESTORE DATABASE DB_Security
            FROM DISK = 'D:₩Backup₩DB_Security_Full.bak'
            WITH MOVE 'DB_Security' TO 'D:₩Restore₩DB_Security.mdf',
                        MOVE 'DB_Security_log' TO 'D:₩Restore₩DB_Security_log.ldf'
GO

/* 수행 시 아래와 같은 메시지 출력 */
메시지 33111, 수준 16, 상태 3, 줄 1
지문이 '0x43FD48F1CCF011B626C63FC07159B1E364D7E319'인 서버 인증서을(를) 찾을 수 없습니다.
메시지 3013, 수준 16, 상태 1, 줄 1
RESTORE DATABASE이(가) 비정상적으로 종료됩니다.
-- 3. 마스터 키 생성 및 인증서 생성
USE master
GO

CREATE MASTER KEY ENCRYPTION
            BY PASSWORD = 'Pa$$w0rd'
```

```
GO

CREATE CERTIFICATE DB_Cert
            FROM FILE = 'C:₩Backup₩DB_Cert'
            WITH PRIVATE KEY (FILE = 'D:₩Backup₩key', DECRYPTION BY PASSWORD =
'Pa$$w0rd')
GO

/* 백업으로부터 마스터 키와 인증서를 생성하고 위의 <2. 복원> 스크립트를 다시 실행
*/
```

위 예제 스크립트는 1 → 2 → 3 → 2 순으로 실행한다. 마스터 키와 인증서를 생성하지 않고는 복원되지 않음을 보여주기 위함이다. 백업으로부터 마스터 키와 인증서를 생성하면 정상적으로 복원 된다. 칼럼 암호화에서 생성해주었던 Personal_Info 테이블에 대해 SELECT 작업이 가능할 것이다. 여기서 대칭키를 OPEN 해주어야 대칭키로 암호화 된 칼럼에 접근 가능함을 잊지 말자.

TDE 암호화 해제

TDE 암호화를 해제하기 위해서는 설정할 때와 반대의 순서로 해제가 가능하다. 추가로 기존의 암호화 되어 기록 된 트랜잭션 로그를 제거하는 과정이 포함되어 있다.

데이터베이스 암호화(TDE) 해제 순서
데이터베이스 암호화 속성 설정(OFF) → 데이터베이스 암호화 키 제거 → 인증서 제거 →마스터 키 제거 → 토랜잭션 로그 제거

[SQL Script] – TDE 암호화 해제

```
-- 1. 데이터베이스 암호화 속성 설정(OFF)
ALTER DATABASE DB_Security
SET ENCRYPTION OFF
GO

-- 2. 데이터베이스 암호화 키 제거
USE DB_Security
GO

DROP DATABASE ENCRYPTION KEY
GO

-- 3. 인증서 제거 (다른 곳에서 사용 하지 않아 불필요 할 경우)
USE master
GO

DROP CERTIFICATE DB_Cert
GO

-- 4. 마스터 키 제거 (다른 곳에서 사용 하지 않아 불필요 할 경우)
USE master
GO

DROP MASTER KEY
GO

-- 5. 트랜잭션 로그 제거
ALTER DATABASE DB_Security
SET RECOVERY Simple
GO

USE DB_Security
```

```
GO

CHECKPOINT
GO

ALTER DATABASE DB_Security
SET RECOVERY Full
GO
```

지금까지 암호화에 관련된 내용을 다뤄 보았다. 암호화는 장단점이 존재한다. 성능과 보안에 대해신중히 검토한 후 암호화를 진행 할 것이며 인증서 및 패스워드 관리에 유의해야 할 것이다. 데이터 유출을 막기 위해 보안 조치를 해놓고 정작 필요로 할 때 인증서나 패스워드가 없어 데이터에 대한 접근을 못하게 되는 우를 범하지 않도록 하자.

또한 TDE 를 데이터베이스에 적용하더라도 패스워드 칼럼에 대해서는 추가로 일방향 암호화인 HASHBYTES 암호화를 해야 한다는 점도 상기 하자.

접속기록 관리

접속기록은 개인정보의 입·출력 및 수정사항이나 파일 별 또는 담당자 별 데이터접근내역 등을 기록하여 불법적인 접근이나 행동을 확인할 수 있는 자료로 사용 된다. 개인정보 보호법에서는 개인정보 DB 에 대해서 개인정보의 열람·수정·삭제와 같은 작업 접속기록 남기도록 하고 있다.

조항의 해설을 보면 개인정보취급자가 개인정보처리시스템에 접속한 기록을 월 1 회 이상 정기적으로 확인·감독하여야 하며, 최소 6 개월 이상 접속기록을 보존·관리하도록 하고 있다. 기간통신사업자의 경우는 사업자 특성상 최소 2 년 이상 접속기록을 보존·관리하도록 한다.

접속기록의 항목에 대해서도 지정하고 있다. 정보주체 식별번호, 개인정보취급자 식별번호, 접속 일시, 접속지 정보, 부여된 권한 유형에 따른 수행업무 등을 포함하여야 한다. 해당 항목을 포

함하며 사업자 필요의 경우 다른 항목을 추가하여 기록 할 수 있다. 저장된 접속기록은 별도의 물리적인 장치에 보관하여야 하며 정기적인 백업을 수행하여야 한다.

3.1 SQL Server의 접속기록 관리

SQL Server 에는 접속기록을 관리하는데 있어 몇 가지 방법이 있다. 프로파일러 또는 Trace 기능을 사용하는 방법, 트리거를 이용한 방법, 그리고 SQL Server 2008 부터 사용 할 수 있는 감사 기능이 있다. 프로파일러 또는 Trace 기능을 사용 하는 경우 서버에 가해지는 부하가 커 튜닝과 같은 특수한 상황에서도 신중하게 사용되며, 트리거의 경우 스크립트를 작성해야 하는 불편함과 SELECT 문을 추적 할 수 없는 점, 트랜잭션 처리에 성능 저하를 가져 올 수 있는 단점이 있다. 여기에서는 감사 기능에 대해서만 설명 하고자 한다. SQL Server 감사는 프로파일러 또는 Trace 기능, 트리거의 단점을 보완 한 감사 전용 기능이다.

감사 기능 사용 전에 주의 할 것은 감사 범위 설정 이다. 과다한 범위의 정보를 감사 하게 되면 성능저하를 초래 할 수 있다. 앞서도 언급했지만 접속기록의 필수 항목은 정보주체 식별번호, 개인정보취급자 식별번호, 접속 일시, 접속지 정보, 부여된 권한 유형에 따른 수행업무로 5 개이다. 최소한 위 5 개 항목에 대해서는 접속기록을 남겨야 한다.

감사 로그의 기록은 3 가지 방법으로 기록이 가능하다. 파일, Windows 응용 프로그램 로그, Windows 보안 로그에 기록이 가능한데 여기서 파일 기록이 가장 일반적이고 효과적이다. 백업 관리도 수월 하며 다른 이벤트가 함께 쌓이지 않기 때문에 관리에도 편리하다. Windows 응용 프로그램 로그와 Windows 보안 로그는 다른 이벤트가 함께 저장되고 보안 로그의 경우 SQL Server 서비스 계정에 대한 별도의 권한을 부여해야 한다.

3.2 서버 감사 만들기

SQL Server 감사는 사용자가 모니터링 하려고 하는 서버 또는 데이터베이스의 이벤트 로그를 어디에 기록할지 정의한 것이며 위에서 말한 것처럼 Windows 응용 프로그램 로그, Windows 보안 로그, 파일 중 선택 하여 만들 수 있다.

[SQL Script] – 서버 감사 생성

```
-- 감사 생성
USE master
GO

CREATE SERVER AUDIT Personal_Audit /* 테이블 액세스 감사 */
TO FILE (FILEPATH = 'D:₩Audit')
GO

CREATE SERVER AUDIT Login_Audit /* 로그인 감사 */
TO FILE (FILEPATH = 'D:₩Audit')
GO

-- 감사 활성화
ALTER SERVER AUDIT Personal_Audit
WITH (STATE = ON)
GO

ALTER SERVER AUDIT Login_Audit
WITH (STATE = ON)
GO
```

감사를 생성 할 때에는 여러 가지 옵션을 함께 입력 할 수 있다. 개체탐색기를 이용하여 새 감사를 만들어 주는 경우에도 옵션 지정이 가능하다.

<감사 생성 옵션>

옵션	설명
TO { FILE | APPLICATION_LOG | SECURITY }	감사 대상의 위치, 왼쪽부터 파일, 응용프로그램 로그, 보안 로그
FILEPATH ='os_file_path'	감사 로그의 경로
MAXSIZE = { max_size }	감사 파일의 최대 크기, 기본값은 UNLIMITED
MAX_ROLLOVER_FILES ={ integer | UNLIMITED }	파일 시스템에 보관할 최대 파일 수, 기본값은 UNLIMITED
RESERVE_DISK_SPACE = { ON | OFF }	디스크의 파일을 MAXSIZE값으로 할당, 기본값은 OFF
QUEUE_DELAY = integer	감사 강제 처리 전까지 허용되는 시간, 1000 = 1초
ON_FAILURE = { CONTINUE | SHUTDOWN }	감사 대상에 쓰기 실패 시 인스턴스 강제종료, 기본값은 CONTINUE
AUDIT_GUID = uniqueidentifier	DB 미러링과 같은 시나리오 지원 시 미러 DB와 GUDI가 같은지 확인

3.3 데이터베이스 감사 사양 만들기

위에서 파일 형식으로 감사 이벤트 로그가 기록되도록 서버 감사를 만들어 주었다. 이제 만들어 둔 해당 감사 파일에 어떤 이벤트가 기록되도록 할 지 지정해 보도록 하자. 암호화 예제에서 사용 했던 DB_Security Database의 Personal_Info 테이블에 대한 감사 사양을 만들어 보도록 하겠다.

[SQL Script] – 데이터베이스 감사 사양 생성

```
USE DB_Security
GO

CREATE DATABASE AUDIT SPECIFICATION Personal_Audit_Spec
FOR SERVER AUDIT Personal_Audit
ADD (SELECT ON OBJECT :: dbo.Personal_Info BY Public)
WITH (STATE = ON)
GO
```

Personal_Audit_Spec 이란 감사 사양에 해당하는 이벤트를 먼저 생성 했던 서버 감사에 저장 하도록 지정 했다. 감사 사양으로는 Personal_Info 테이블에서 조회(SELECT)하는 모든(Public) 사용자에 대해 지정 하고 감사 사양을 즉시 활성화 해 주었다.

그럼 감사 사양에 조회되도록 조회 작업을 해보고 감사 파일을 조회하여 정상적으로 작동 하는 지 확인해 보자.

[SQL Script] – 감사 테스트

```
USE DB_Security
GO

SELECT * FROM dbo.Personal_Info
GO

UPDATE dbo.Personal_Info
SET NAME = '박찬호' WHERE NAME = '류현진'
GO

SELECT * FROM dbo.Personal_Info WHERE NAME = '류현진'
GO

UPDATE dbo.Personal_Info
```

```
SET NAME = '류현진' WHERE NAME = '박찬호'
GO

SELECT * FROM sys.fn_get_audit_file ('D:₩Audit₩Personal*', DEFAULT, DEFAULT)
GO
```

server_instance_name	database_name	schema_name	object_name	statement
CRAZYCHOON				
CRAZYCHOON	DB_Encryption	dbo	Persnal_Info	SELECT * FROM dbo.Persnal_Info
CRAZYCHOON	DB_Encryption	dbo	Persnal_Info	UPDATE dbo.Persnal_Info SET NAM
CRAZYCHOON	DB_Encryption	dbo	Persnal_Info	SELECT * FROM dbo.Persnal_Info W
CRAZYCHOON	DB_Encryption	dbo	Persnal_Info	UPDATE dbo.Persnal_Info SET NAM

감사 결과(sys.fn_get_audit_file 함수의 반환 값)의 일부 이다. 누가 언제 어떤 데이터베이스의 개체에 대해 접근 했는지 확인 할 수 있다. 여기에는 감사 사양에 필터 되는 이벤트만 기록되고 결과가 존재하지 않는 경우에도 기록 된다. 앞서 감사 사양을 작성할 때 SELECT 에 대해서만 감사 사양을 정의 했는데 statement 열을 확인하면 UPDATE 에 대해서도 기록하고 있다. DELETE 작업이 있다면 DELETE 도 기록되는데, UPDATE 나 DELETE 작업의 경우 SELECT 가 선행되기 때문이다. SELECT 에 대한 감사 사양이 없다면 UPDATE, DELETE 는 각 각 따로 지정해 줄 수 있다. SELECT 와 중복으로 감사 사양을 지정하면 UPDATE 나 DELETE 시 두 개씩 중복으로 기록된다. INSERT 의 경우는 SELECT 와 별개로 동작한다. 추가로 해당 테이블을 직접 이용하지 않더라도 관련 뷰나 프로시저를 사용 할 경우 감사 대상이 된다.

결과에서 첫 행은 감사에 대한 변경사항으로 action_id 의 값이 AUSC 이다. sys.dm_audit_actions 뷰에서 action_id 로 필터 하면 full name 을 알 수 있고 sys.dm_audit_class_type_map 뷰를 통해서는 클래스 타입을 확인 할 수 있다. 위 두 개의 뷰들을 sys.fn_get_audit_file 함수와 함께 사용하면 감사 결과를 효과적으로 확인 할 수 있다.

예제에서 감사 결과를 조회 할 때 파일 위치를 'D:\Audit\Personal*' 라고 적었다. 감사 파일이 생성될 때 이름 뒤에 임의의 문자열이 따라 붙기 때문이다.

3.4 서버 감사 사양 만들기

데이터베이스 감사 사양은 해당 데이터베이스에 대한 이벤트만을 기록 한다. 서버 수준에서의 이벤트를 감사하고자 한다면 서버 감사 사양을 만들어야 한다.

[SQL Script] – 서버 감사 사양 생성

```
USE master
GO

CREATE SERVER AUDIT SPECIFICATION Login_Audit_Spec
FOR SERVER AUDIT Login_Audit
ADD (SUCCESSFUL_LOGIN_GROUP)
WITH (STATE = ON)
GO
```

앞서 만들었던 서버 감사 Login_Audit 에 로그인(SUCCESSFUL_LOGIN_GROUP) 동작에 대해 기록 하고 감사 사양의 이름은 Login_Audit_Spec 로 했다. 감사 조건 동작은 로그인 이외에도 여러가지가 있는데 개체 탐색기를 이용해 보면 알 수 있다. 개체 탐색기의 새 서버 감사 사양에서 감사 동작 유형들을 확인 해 볼 수 있다.

[SQL Script] – 감사 확인

```
WITH XMLNAMESPACES(DEFAULT
'http://schemas.microsoft.com/sqlserver/2008/sqlaudit_data')
SELECT event_time, session_id, server_principal_name,
        CAST(additional_information AS
xml).value( '(action_info/address/text())[1]','varchar(255)') AS 'IP'
FROM sys.fn_get_audit_file ('C:₩Audit₩Login*', DEFAULT, DEFAULT) WHERE action_id
<> 'AUSC'
GO
```

데이터베이스 감사와 마찬가지로 sys.dm_get_audit_file 함수를 통해 기록된 내용을 확인 할 수 있다. 로그인 기록에는 additional_infomation 열에 xml 형태로 포함되어 있는데 XQuery 를 통해 IP 주소만을 추출 할 수 있다.

	event_time	session_id	server_principal_name	ClientIPAddress
1	2013-09-11 04:07:57.4433333	71	ignite_m	192.168.123.187
2	2013-09-11 04:16:19.6333333	71	ignite	192.168.123.187
3	2013-09-11 04:16:19.6333333	65	ignite	192.168.123.187
4	2013-09-11 04:16:19.6400000	63	ignite	192.168.123.187
5	2013-09-11 04:17:57.4466666	68	ignite_m	192.168.123.187
6	2013-09-11 04:27:57.4366666	72	ignite_m	192.168.123.187

로그인 트리거를 이용해서도 이와 같이 접속자의 IP 를 기록해두는 접속기록 관리 테이블을 만들 수 있다. 앞서 설명했던 접근 제어에서 <1.2 트리거를 이용 한 IP 접근 제어>를 응용하여 만들 수 있다. SQL Server 의 버전이 2008 하위 버전 이거나 감사 기능을 사용 할 수 없는 경우에는 불편함이 따르지만 트리거를 이용하여 감사를 대신 할 수 있다.

개인정보의 기술적 관리적 보호조치 기준 해설서에 의하면 접속기록에 포함 되어야 하는 항목으로는 정보주체 식별번호, 개인정보취급자 식별번호, 접속일시, 접속지 정보, 부여된 권한 유형에 따른 수행업무 등을 포함해야 한다. 아래는 해설서에서 예시로 든 접속기록 항목 표이다.

[접속기록 항목(예시)]

정보주체 식별정보	취급자 식별정보	접속일시	접속지	수행업무
123456789	홍길동(HGD)	2012.06.03, 15:00:00	172.168.168.11	조회(고객응대)

감사는 편리한 접속기록 관리 기능이다. 하지만 감사를 만들어 놓을 뿐 주기적인 관리와 모니터가 없다면 의미가 무색해진다. 기준에 맞도록 감사를 만들고 또 관리 하도록 하자.

마치며

지금까지 개인정보보호법에 대한 SQL Server 의 보안에 대해 알아보았다. 길다면 긴 내용이지만 위 내용들은 기본에 불과하고 개인정보보호법의 일부분에 지나지 않는다. 관리자들은 개인정보보호법에 관련 된 법령 등을 충분히 숙지하여 대비해야 할 것이다.

본 문서에서는 데이터베이스 관리자 측면에서 개인정보보호법에 대한 대응 방법의 기본을 담아보려 노력했다. 기본이니만큼 최대한 이해하기 쉽게 쓰고자 했으며 스스로에게도 매뉴얼이 될 수 있게 하고자 했다. 본 문서를 바탕으로 보안의 기본을 숙지하고 그 기본을 배경으로 SQL Server 의 보안에 활용 되기를 바란다.