SQL and NoSQL Database System Development: Tracking Security Breaches by Sector

Nicole Hessner, Khanh Huynh, Alex Jimenez
Team 2, IS 360, MS Computer Science, School of Computing & Technology, City University of Seattle
hessnernicole@cityuniversity.edu
huynhkhanh@cityuniversity.edu
jiminezalex@cityuniversity.edu

Abstract

Cybersecurity incidents affect every sector and every industry. In this project, three databases will be developed: SQL, document, and key-value databases. The SQL database will be the Incident table, with the IncidentID primary key, Date, BreechType, and CompanyID foreign key. The document database will feature the Company Information, including CompanyID primary key, Company Name, and SectorID foreign key. Finally, the key-value database includes the primary key split into the partition key SectorID, and the sort key SectorName. The attributes of the key-value database will be the company names of that sector One view will include a JOIN between Incident and CompanyInfo on CompanyID. Another view will be an aggregate view showing the BreechTypes by Sector. Users will be able to add new Companies to the CompanyInfo table.

Keywords

cybersecurity, breech, SQL, NoSQL, incident response, security

SQL or Structured Query Language is a programming language that is used to manage databases, and is the standard language for storing and retrieving databases. SQL operations take its forms in SQL statements such as: SELECT, ADD, INSERT, UPDATE, and DELETE (Petković, 2020).

The first database to create is the SQL database. As part of the project, we had to create three separate databases. Following the material covered in week 5, working with Database Objects common normalization, we followed the concepts First, Second and Third Form. First Form addresses Using the smallest Form for atomic data, our naming convention for our column's names. Doing so eliminates repeating groups in individual tables. We have identified each set of related data with a primary key. Following the original table, we created separate tables for groups and included the original table's primary key. For these tables to show relation to one another, we used a foreign key to establish the relationship. The second Form, "Functional Dependency," make sure each field depends on the primary key. Eliminate fields that do not rely on the key. Elimination helps prevent duplicate data and INSERT and UPDATE inconsistencies. The third Form, "Transitive Dependency," allows us to look for fields that depend on non-key columns; And removes these fields. Helping to prevent DELETE errors, where deleting a row may delete data that is not available anywhere else.

```
-- Create Database
CREATE DATABASE Cybersecurity
ON
( NAME = Cybersecurity_data,
    FILENAME = '/Users/nikki/Desktop/IS360/TeamProject/Cybersecurity_data.mdf',
    SIZE = 10,
    MAXSIZE = 50,
    FILEGROWTH = 5 )
LOG ON
( NAME = Cybersecurity_log,
    FILENAME = '/Users/nikki/Desktop/IS360/TeamProject/Cybersecurity_data.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB );
GO
```

Data Definition Language (DDL), a subset of SQL statements that change the structure of the database schema in some way, typically by creating, deleting, or modifying schema objects such as databases, tables, and views; was used in creating the following tables. In reference to the example below the SQL statements specify a primary key and a foreign key to establish relational database

```
-- Create Tables for the database*/
CREATE TABLE Sectors.Sectors (
    SectorID int NOT NULL IDENTITY PRIMARY KEY
    , SectorName nvarchar(100) NOT NULL
);
GO

CREATE TABLE Sectors.Companies (
    CompanyID int NOT NULL IDENTITY PRIMARY KEY
    , CompanyName nvarchar(100)
    , SectorID int NOT NULL
        CONSTRAINT FK_SectorID FOREIGN KEY (SectorID)
        REFERENCES Sectors.Sectors (SectorID)
);
GO

CREATE TABLE Incidents.Incidents (
    IncidentID int NOT NULL IDENTITY PRIMARY KEY
    , IncidentDate date
    , BreechType nvarchar(100)
    , CompanyID int NOT NULL
        CONSTRAINT FK_CompanyID FOREIGN KEY (CompanyID)
        REFERENCES Sectors.Companies (CompanyID)
);
GO
```

A login is a security principle or an entity that a secure system can authenticate. For example, users need a login to connect to SQL Server. Users can create a login based on a Windows principle (such as a domain user or a Windows domain group) or create a login not based on a Windows principle (such as an SQL Server login).

```
USE Cybersecurity;
GO

-- Create logins for team members.
CREATE LOGIN HessnerN WITH PASSWORD = 'P@$$w0rd';
GO

CREATE LOGIN JiminezA WITH PASSWORD = 'P@$$w0rd';
GO

CREATE LOGIN HuynhK WITH PASSWORD = 'P@$$w0rd';
GO

-- Create users from logins
CREATE USER HessnerN FROM LOGIN HessnerN;
GO

CREATE USER JiminezA FROM LOGIN JiminezA;
GO

CREATE USER HuynhK FROM LOGIN HuynhK;
GO
```

In the last few decades, the use of the internet and database access has increased exponentially. The creation of SQL allowed users to access information and records with just one single command. These databases contain data definition language and data manipulation language, which allow data retrieval. The ease of

accessibility comes at a cost. System breaches are nothing but a common occurrence, according to Daniel Goldberg, a cybersecurity research expert at Guardicore Labs (No Author, 2019). These vulnerabilities within SQL give hackers an advantage, due to the ease of change or delete databases containing sensitive information. This vulnerability poses a great threat to web-based applications, as they contain information such as: full name, social security number, bank account information, and much more (Aslan & Samet, 2017). The number and seriousness of these attacks and breaches are occurring more rapidly than ever before (Tarake, 2018).

This database has two views based on its tables. The first, Incidents.CompanyInfo, presents cybersecurity incident information with human-readable company information.

```
CREATE VIEW Incidents.CompanyInfo
AS
SELECT I.IncidentID, I.IncidentDate, I.BreechType, C.CompanyName, C.EstablishedDate
FROM Incidents.Incidents AS I
FULL OUTER JOIN Sectors.Companies AS C
    ON (I.CompanyID = C.CompanyID);
GO
```

The next view is the Incidents.Count view, which counts how many incidents have occurred within each company.

```
CREATE VIEW Incidents.Count(CompanyName, IncidentCount)
AS
SELECT C.CompanyName, COUNT(I.IncidentID)
FROM Incidents.Incidents as I
JOIN Sectors.Companies as C
    ON (C.CompanyID = I.CompanyID)
GROUP BY C.CompanyName;
GO
```

The database utilizes a stored procedure to find all incidents of a certain type in the database. This utilizes the LIKE keyword to match an input variable approximately to words within the IncidentType column.

```
-- Create stored procedure
CREATE PROCEDURE Incidents.findIncidentType
    @incidentType nvarchar(100)
AS
SELECT *
FROM Incidents.Incidents
WHERE BreechType LIKE @incidentType
ORDER BY IncidentID;

EXEC Incidents.findIncidentType @incidentType = 'Malware';
GO
```

This database contains a combination of real and fictional data. Sectors.Sectors utilizes real US Economic Sectors, while Sectors.Companies uses fictionalized companies inspired by real companies. All incidents are fictitious.

```
INSERT INTO Sectors.Sectors (SectorName)
VALUES ('Mining'),
       ('Utilities'),
       ('Construction'),
       ('Manufacturing'),
       ('Wholesale'),
       ('Retail'),
       ('Trasnportation & Warehousing'),
       ('Information'),
       ('Finance & Insurance'),
       ('Real Estate'),
       ('Professional Services'),
       ('Management'),
       ('Education'),
       ('Health Care & Social Assistance'),
       ('Waste Management'),
       ('Arts, Entertainment, & Recreation'),
       ('Accommodation & Food Services'),
       ('Other Services');
GO
```

```
INSERT INTO Incidents.Incidents (IncidentDate, BreechType, CompanyID)
VALUES  ('2020-10-10', 'Malware', 1),
        ('2017-09-19', 'Phishing', 1),
        ('2018-02-04', 'SQL Injection', 1),
        ('2018-03-17', 'Man-in-the-Middle', 2),
        ('2017-05-04', 'Password Attack', 2),
        ('2015-06-15', 'Denial of Service', 2),
        ('2016-08-28', 'Cross-Site Scripting', 3);
GO
```

```
INSERT INTO Sectors.Companies (CompanyName, SectorID)
VALUES  ('Goeing', 4),
        ('Halmart', 6),
        ('Jefferson Province College', 13);
GO
```

After adding initial datasets, additional columns and constraints were added to the tables.

```
--Alter tables with fewer than three rows.
ALTER TABLE Sectors.Sectors
ADD SectorValue MONEY DEFAULT NULL;

ALTER TABLE Sectors.Sectors
ADD UNIQUE (SectorName);

ALTER TABLE Sectors.Companies
ADD UNIQUE (CompanyName);

ALTER TABLE Sectors.Companies
ADD EstablishedDate date DEFAULT '1900-01-01';

ALTER TABLE Incidents.Incidents
ADD CONSTRAINT CHK_incidentDate CHECK (IncidentDate >= '1834-01-01');
```

REFERENCES

Aslan, Ö., & Samet, R. (2017, September). Mitigating cyber security attacks by being aware of vulnerabilities and bugs. In *2017 International*

Conference on Cyberworlds (CW) (pp. 222-225). IEEE.

Kareem, F. Q., Ameen, S. Y., Salih, A. A., Ahmed, D. M., Kak, S. F., Yasin, H. M., Ibrahim, I. M., Ahmed, A. M., Rashid, Z. N., & Omar, N. (2021). SQL Injection Attacks Prevention System Technology: Review. *Asian Journal of Research in Computer Science*, *10*(3), 13-32. https://doi.org/10.9734/ajrcos/2021/v10i330242

No Author. (2019, September 24). *SQL attacks are a piece of cake for hackers and the risk to firms is high.* Tech Monitor. Retrieved November, from

https://techmonitor.ai/techonology/cybersecurity/sql-attacks.

Petković, D. (2020). *Microsoft Sql Server 2019: A beginner's guide*. McGraw-Hill.

S. Vyamajala, T. K. Mohd and A. Javaid, "A Real-World Implementation of SQL Injection Attack Using Open Source Tools for Enhanced Cybersecurity Learning," 2018 IEEE International Conference on Electro/Information Technology (EIT), 2018, pp. 0198-0202, doi: 10.1109/EIT.2018.8500136.

Tareke, T. A., & Datta, S. (2018, February). Automated and Cloud Enabling Cyber Security Improvement in Selected Institutions/Organizations. In *2018 Second International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 533-538). IEEE.