MINNESOTA STATE COLLEGE SOUTHEAST
A TECHNICAL & COMMUNITY COLLEGE

Microsoft SQL Server Customized Training

Contact

Ed Callahan

ed@edcallahan.com

http://edcallahan.com/SQLTraining

Database Normalization and Relational Databases

Organize data into tables and columns (attributes) to:

1. Reduce data redundancy (duplicate data)
2. Improve data integrity

We end up with topic-specific tables that are linked together.

Example - denormalized student data

See the DenormalizedData.xlsx

- Difficult to find a student with a given major
- Difficult to rename a major, or move major to new program
- Wasted space for students with only one major
- What if a student has a fourth major?
- Majors change over time …
- Addresses, phone numbers, ...

Structure of a normalized database

- Every table has one row per unique object (person, major, student term, etc) identified by a unique "**primary key**". Usually an integer, often arbitrary.
- No repeating groups of columns
- All columns in the table describe the object represented by the row's primary key
- No transitive dependencies (know value of one field based on another).
- Tables are linked together by their keys

Microsoft SQL Server

Software to implement and manage relational databases. Others include:

- Oracle
- IBM DB2
- MySQL
- PostgresSQL, etc

Based on ANSI standards

Installable Features of SQL Server

- Database Engine (relational database management system)
- Reporting Services
- Analysis Services
- Integration Services
- Management Studio (client-side)

Login types

- Windows (Active Directory)
- SQL logons

SSMS Overview - Server

- Databases (System and User)
- Security (Logins and Roles)
- Server Objects (Linked servers)
- Replication
- Management
  - Extended Events (Profiler)
  - Logs
  - Database Mail
- SQL Server Agent

System Databases

- **master** – system-level data (logon accounts, linked servers, location of other databases)
- **model** – determines default values for newly created databases
- **msdb** – used by SQL Server Agent for scheduling
- **tempdb** – workspace for temporary objects

SSMS Overview -Database

- Diagrams
- Tables
- Views
- Programmability (stored procedures, functions)
- Security (Users, roles, schemas)

SSMS Overview - Table

- Columns
- Keys (primary and foreign)
- Constraints
- Triggers
- Indexes

Data Definition Language (DDL)

- CREATE tables
- ALTER tables
- DROP tables

Good interactive SSMS tools for this, which can generate code

Data Manipulation Language (DML)

- INSERT
- UPDATE
- DELETE
- SELECT

Data Control Language (DCL)

- GRANT
- REVOKE

Often done use SSMS interactive tools

Create table with primary key

```
CREATE TABLE dbo.StudentHistory
(
    StudentHistoryId int IDENTITY(1,1) NOT NULL,
    StudentId int NOT NULL,
    TermID int NOT NULL,
    ClassCode char(2) NULL,
    TermCreditsAttempted int NULL,
    TermCreditsEarned int NULL,
    TermGPA numeric(4, 2) NULL,
  CONSTRAINT StudentHistory_PK PRIMARY KEY CLUSTERED
    (
        StudentHistoryId ASC
    )
)
```

Create table with primary key

- Schema
- Data types
- NULL / NOT NULL
- Auto-increment field
- Primary Key definition

See CreateTable.sql

Schemas

Collection of items in a database, a “namespace”

- Useful for managing permissions
- Default schema is dbo
- Each user can have a different default schema
- sys and information_schema are schemas holding metadata tables and views
- Complete way to reference a table is:
  linked_server.database.schema.table

Create Schema

```
use ReportBuilderTraining
go

create schema evc
```

Data Types

- char, nchar
- varchar, nvarchar, varchar(max)
- int, tinyint, bigint
- float, double
- numeric, decimal
- datetime
- text, image

https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql

Primary Keys

- Simple vs. Composite
- Natural vs. Arbitrary

Primary keys should be stable, not changed once set

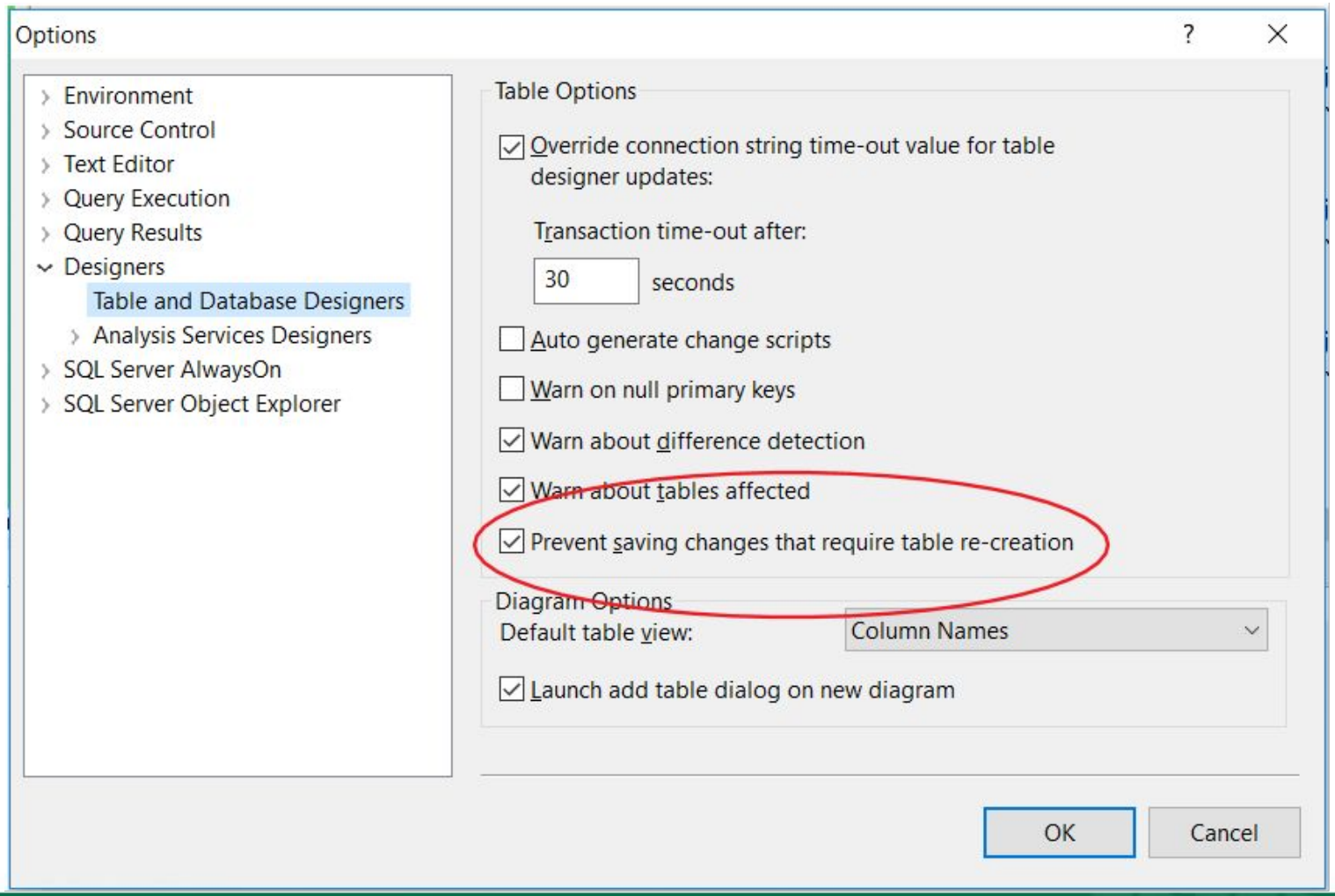
Designer

Set identity primary key

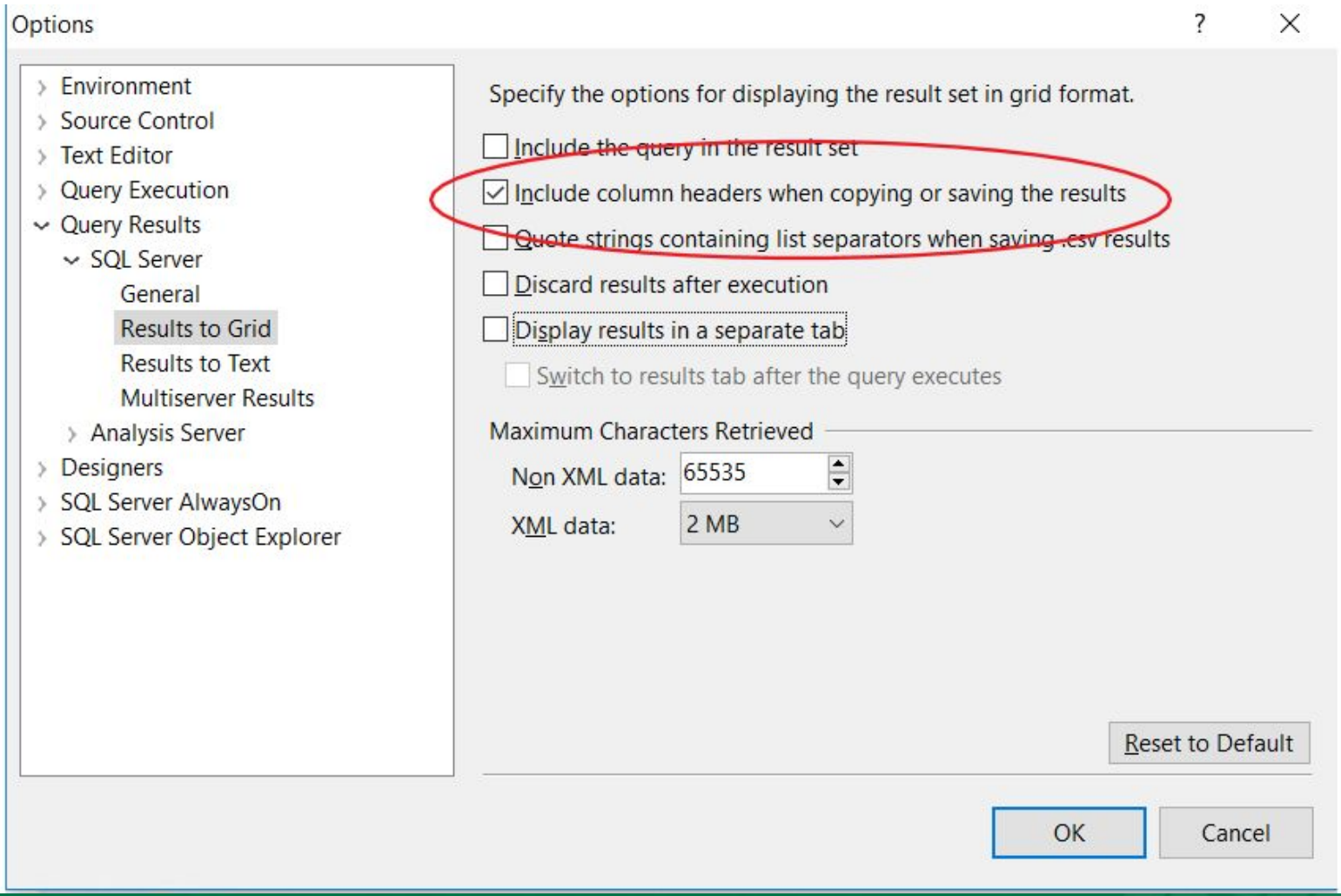
Set default value

Script CREATE TABLE statement

Enable Table Designer



Column Headers in Query Results



More DDL commands

- See DDL.sql

DROP ***table_name***

ALTER TABLE ***table_name***
ADD ***column_name datatype***

ALTER TABLE ***table_name***
DROP ***column_name***

More DDL commands

ALTER TABLE ***table_name***
ALTER ***column_name datatype***

ALTER TABLE Persons
ADD CONSTRAINT PK_Person
PRIMARY KEY (ID)

Indexes

Clustered vs. Unclustered

Unique vs. Not Unique

Ordered, so faster to search. Typically better than a table scan.

Creating an Index

```
CREATE NONCLUSTERED INDEX [NC_Name] ON
[dbo].[Test]
(
    [FirstName] ASC,
    [LastName] ASC
)
```

- Also CREATE UNIQUE NONCLUSTERED …
- We will discuss creating indices when we talk about optimizing queries

Foreign Keys

A form of constraint

```
ALTER TABLE Orders
ADD FOREIGN KEY (PersonID)
REFERENCES Persons(PersonID)
```

- **ON DELETE CASCADE, ON UPDATE**

We will discuss when looking at many-to-many joins

Data Manipulation Language (DML)

- INSERT
- UPDATE
- DELETE
- SELECT

SELECT

```
SELECT column1, column2, column3
    FROM table_name
    WHERE condition
    ORDER BY column1, column2, column3
```

Functions

- CAST and CONVERT and Implicit Conversions
- Datetime functions
- String functions
- Numeric Functions
- CASE statement

See Functions.sql

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

WHERE Statement

- >, <, =, !=
- BETWEEN
- IN, NOT IN
- LIKE, NOT LIKE
- AND, OR, NOT

See Where.sql

NULL

- Tri-state logic
- IS NULL, IS NOT NULL

See NULLS.sql

JOINS

- INNER
- LEFT JOIN
- RIGHT JOIN
- FULL OUTER JOIN

See Joins.sql

Order of Execution

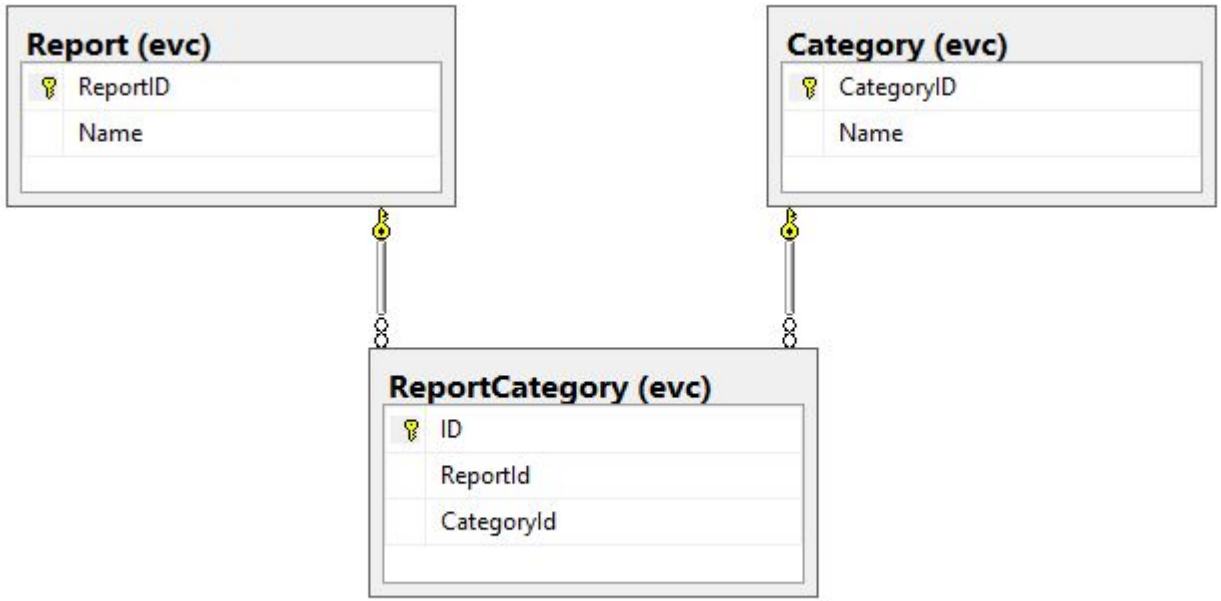
1. FROM
2. JOINS (top to bottom)
3. WHERE
4. SELECT
5. ORDER BY

Evaluating Performance and Optimizing Queries

- Typically use Execution Plans to evaluation queries and compare performance
- Address issues with query structure and indices
- Do not rely on run time as it depends on caching and statistics

See Performance.sql

Many-to-Many Joins



See ManyToMany.sql, ManyToMany2.sql

Sub-queries

See Subqueries.sql

Correlated subqueries

See CorrelatedSubqueries.sql

Window Functions and CTEs

- CTEs help organizationally, but often at a performance cost. They are not a compiler hint.

see WindowFunction.sql

Set Functions

- UNION and UNION ALL
- INTERSECT
- EXCEPT

See SetFunctions.sql

Aggregate Queries

```
SELECT column1, column2, AGG_FUNC(column3)
    FROM table1
    WHERE condition
    GROUP BY column1, column2
    HAVING condition
    ORDER BY column1, column2
```

See AggregateQuery.sql

Order of Execution

1. FROM
2. JOINS (top to bottom)
3. WHERE
4. GROUP BY
5. HAVING
6. SELECT
7. ORDER BY

INSERT

```
INSERT
INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Insert/Select

```
INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;
```

- Select statement can be complex

SELECT INTO and Temp tables

```
SELECT *
INTO newtable
FROM oldtable
WHERE condition;
```

- #temp
- ##temp

Other methods to populate files

- GUI
- Import
- Applications

UPDATE

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

- Can also be combined with a SELECT-like statements

See UpdateDelete.sql

DELETE

DELETE FROM ***table_name***
WHERE ***condition***;

- Can also be combined with a SELECT-like statements

MERGE

- Combines insert/update/delete into one operation
- See: https://www.simple-talk.com/sql/learn-sql-server/the-merge-statement-in-sql-server-2008/
- I've encountered significant performance issues caused by the delete portion of the merge

Transactions

- BEGIN TRANSACTION
- ROLLBACK TRANSACTION
- COMMIT TRANSACTION

- Locks
- Dirty Reads

Deadlock

See Deadlocks 01.sql and Deadlocks 02.sql

1. Process A locks Address table
2. Process B locks Phone table
3. Process A now waiting on Process B to commit so Phone table is unlocked
4. Process B now waiting on Process A to commit so Address table is unlocked

A waiting on B and B waiting on A -- Deadlock

Event Monitor

See Deadlock Event Monitor.sql to create Event Monitor

Replaces Profiler system

Security

- **Logins** - Windows users, Windows groups or SQL users. Defined at the server level and granted server-level permissions
- **Users** - principals scoped to the database level, usually mapped to a login

Often a one-to-one relationship, with the same name

Relationship can break with database moved to a new server

sa Login and dbo User

- and NT AUTHORITY\SYSTEM
- Special built-in principals with elevated rights

Built-in Roles

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

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

Security Precedence

GRANT, DENY and REVOKE

- Explicit Deny overrides explicit Grants
- Except … *"A table-level DENY does not take precedence over a column-level GRANT. This inconsistency in the permissions hierarchy has been preserved for backward compatibility."*

https://docs.microsoft.com/en-us/sql/t-sql/statements/grant-object-permissions-transact-sql

Granting/Denying Permissions

See Security.sql

https://docs.microsoft.com/en-us/sql/t-sql/statements/grant-object-permissions-transact-sql

https://www.simple-talk.com/sql/database-administration/sql-server-security-cribsheet/

Using Roles to control access

See Roles.sql

AD Domain Groups is an alternative approach

Programmability

- Views
- Triggers
- Scripts
- Stored Procedures
- Scalar Functions
- Table Functions
- CLR functions

Programmability

- Variables
- IF THEN ELSE
- WHILE
- Cursors
- Dynamic SQL

See Variables_LogicFlow.sql
See Cursors_DynamicSQL.sql

Procedures vs Functions

Stored Procedures can:

- May optionally take arguments
- May optionally return results
- May make database and other system changes (insert, update, etc)
- Cannot be used in a SQL statements
- Good error handling
- Can create and use temp tables within

Procedures vs Functions

Functions can:

- Optionally take arguments
- Must return a value
- May not make system changes to database
- Can be used in a SQL statement
- No temp tables or try/catch blocks

Stored Procedure Example

Simple table update with error handling

See StoredProcedure.sql

Error Handling and Nested Transactions

Technically, transactions can be nested and the SQL compiler will not complain. However, a ROLLBACK rolls back to the very first BEGIN, so nesting doesn't act as expected.

See NestedTransactions.sql

Stored Procedure Example II

Taking parameters and returning data

See StoredProcedure2.sql

Elevated permissions through stored procedures and views

A user with SELECT permissions on a view, or EXECUTE permissions on a stored procedure, can see the data returned by those objects even if they do not have permission to the underlying tables.

Handling SSRS parameters in a stored procedure

Table value function example

See:

- fnSplitFunction.sql
- StoredProcedure3.sql

Triggers

See Triggers.sql

See:

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

SQL Injection Attacks

See SQLInjectionAttack.sql

See:

- https://www.owasp.org/index.php/SQL_Injection
- https://www.w3schools.com/sql/sql_injection.asp

Backup and Restore

see BackupRestore 01.sql

Remember, the directory locations of the server's, not your workstation's. You are typically backing up to a server's drive, which you may not have direct permissions to.

Recovery Models

In SSMS, Database Properties > Options

Determines how long to keep data in the transaction log file

- Simple
- Full

https://www.mssqltips.com/sqlservertutorial/2/sql-server-recovery-models/

Simple recovery mode

- Transactions immediately removed from transaction log
- Can do complete and differential backups
- Allows for smaller transaction log files, but increases risk of data loss

Full recovery mode

- All transactions kept in transaction log until it is backed up or truncated
- Can recover to a point-in-time
- Can lead to very large transaction files, especially if there is no scheduled transaction log backup
- Can do complete and partial backups as well as transaction log backups

See BackupRestore 02.sql

Restore Order

1. Last complete backup
2. Last differential backup
3. All transaction backups since last differential, in order

Backup Strategy

One strategy:

1. Nightly full backup
2. Hourly differentials
3. Transaction logs every … ten minutes?
4. Retain backups
    a. last week's
    b. Last 4 Fridays
    c. Last 4 first Friday of month
    d. Last 4 first Friday of year

Restore to point of failure

For Full recovery model only

See RestoreBackup 03.sql

https://docs.microsoft.com/en-us/sql
/relational-databases/backup-restore
/restore-database-to-point-of-failure-full-recovery

Controlling size of log file

Log files automatically increase in size, and will claim more size that it needs each time it expands.

Normally you do not need to shrink log files manually. Except … every DBA at some point has had databases in Full recovery mode that he hasn't been backing up :-(

See LogFileSize.sql

Install Concerns

- Splitting installation across drives
  - Log and Data files on different drives
  - tempdb on a different drive
- handling sa account
- handling default port

See http://sqlmag.com/storage/sql-server-storage-best-practices

Maintenance Plans and SQL Server Agent

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

See IndexFragmentation.sql

See https://docs.microsoft.com/en-us/sql/t-sql/database-console-commands/dbcc-checkdb-transact-sql

Enabling SQL Agent and Maintenance Plans

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'Agent XPs', 1;
GO
RECONFIGURE
GO
```

See EnableAgent.sql

Monitoring SQL Performance

- Extended Events
- Spotlight - https://www.spotlightessentials.com/
- Redgate SQL Monitor - http://www.red-gate.com/products/dba/sql-monitor/

Important to monitor trends