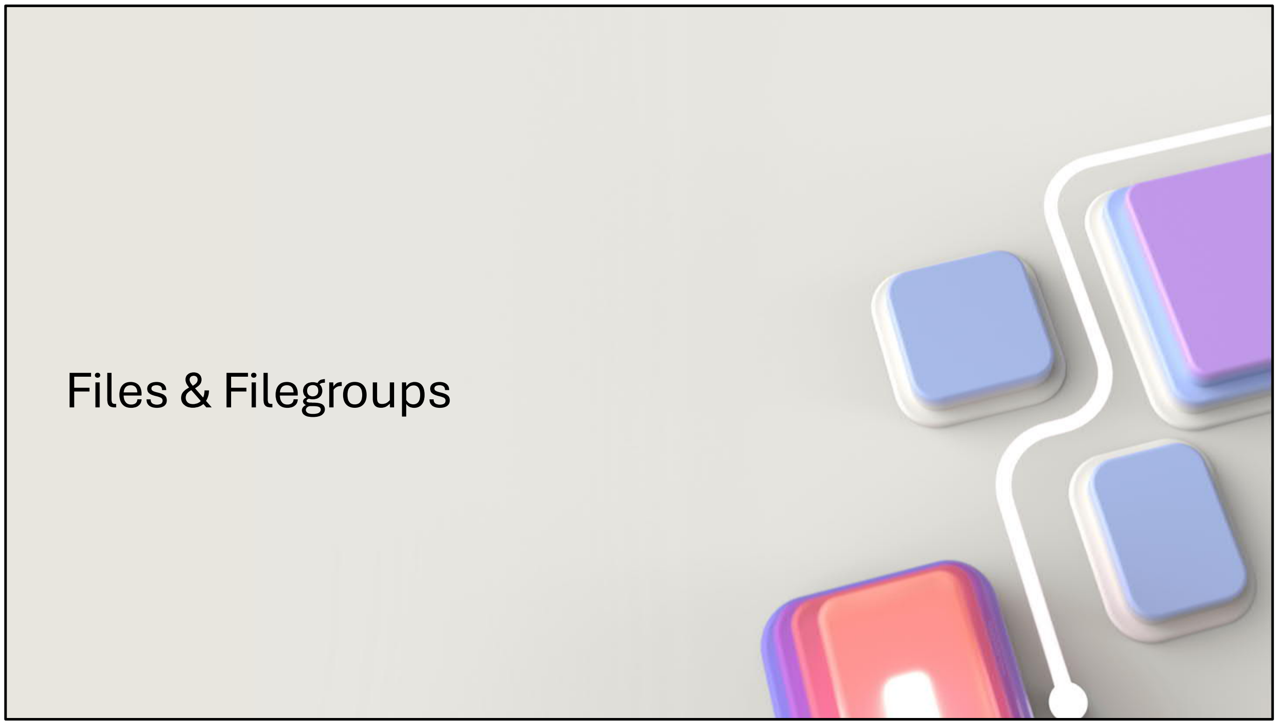
Microsoft

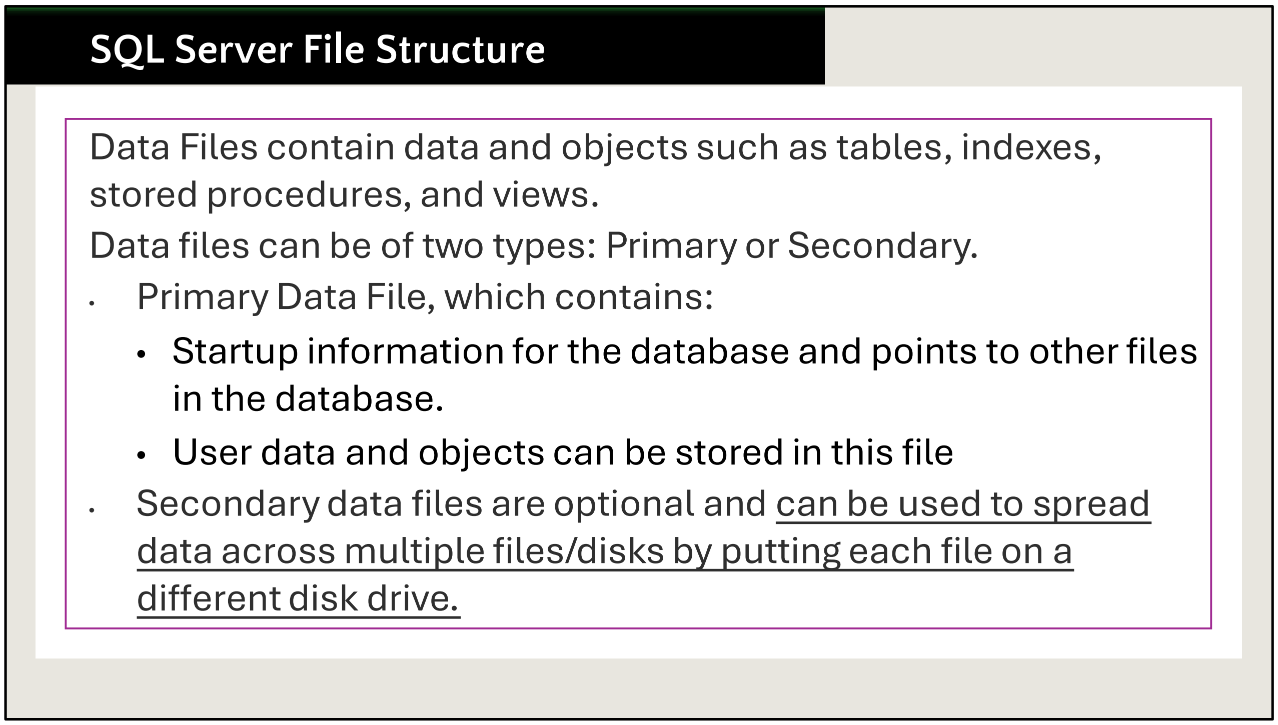
Files & Filegroups

Agenda

- File & Filegroups
- Design Considerations



Files & Filegroups



SQL Server File Structure

Data Files contain data and objects such as tables, indexes, stored procedures, and views.

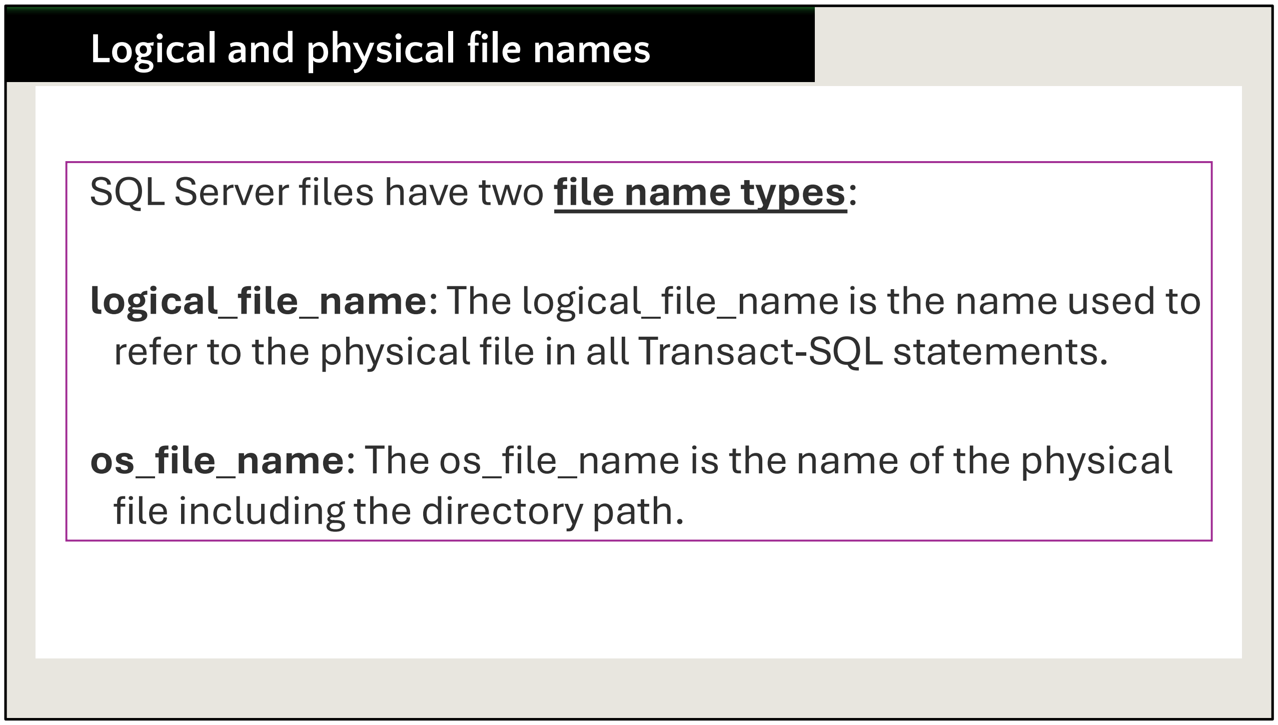
Data files can be of two types: Primary or Secondary.

- Primary Data File, which contains:
  - Startup information for the database and points to other files in the database.
  - User data and objects can be stored in this file
- Secondary data files are optional and can be used to spread data across multiple files/disks by putting each file on a different disk drive.

https://learn.microsoft.com/en-us/sql/relational-databases/databases/database-files-and-filegroups?view=sql-server-ver17

Database Files

File	Description
Primary (mdf)	Contains startup information for the database and points to the other files in the database. Every database has one primary data file.
Secondary (ndf)	Optional user-defined data files. Data can be spread across multiple disks by putting each file on a different disk drive.
Transaction Log (ldf)	The log holds information used to recover the database. There must be at least one log file for each database.



Logical and physical file names

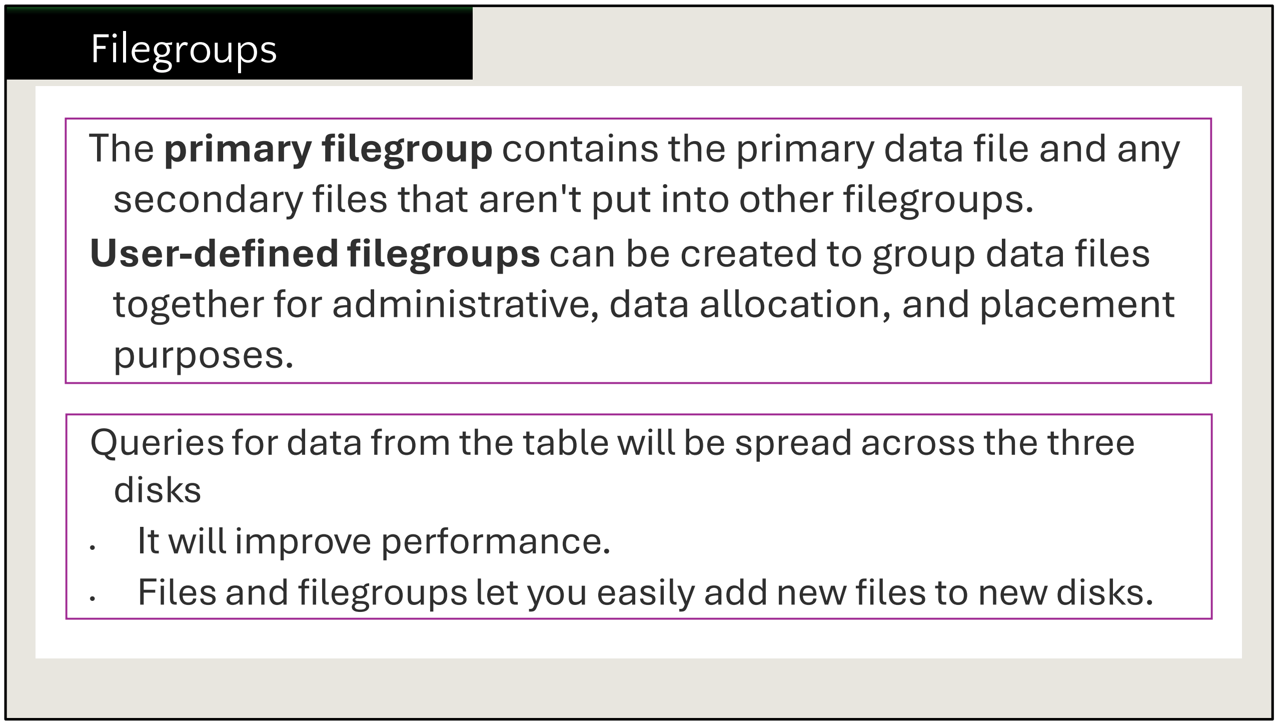
SQL Server files have two **file name types**:

logical_file_name: The logical_file_name is the name used to refer to the physical file in all Transact-SQL statements.

os_file_name: The os_file_name is the name of the physical file including the directory path.

The logical file name must comply with the rules for SQL Server identifiers and must be unique among logical file names in the database.

Os_file_name must follow the rules for the operating system file names.



Filegroups

The **primary filegroup** contains the primary data file and any secondary files that aren't put into other filegroups.

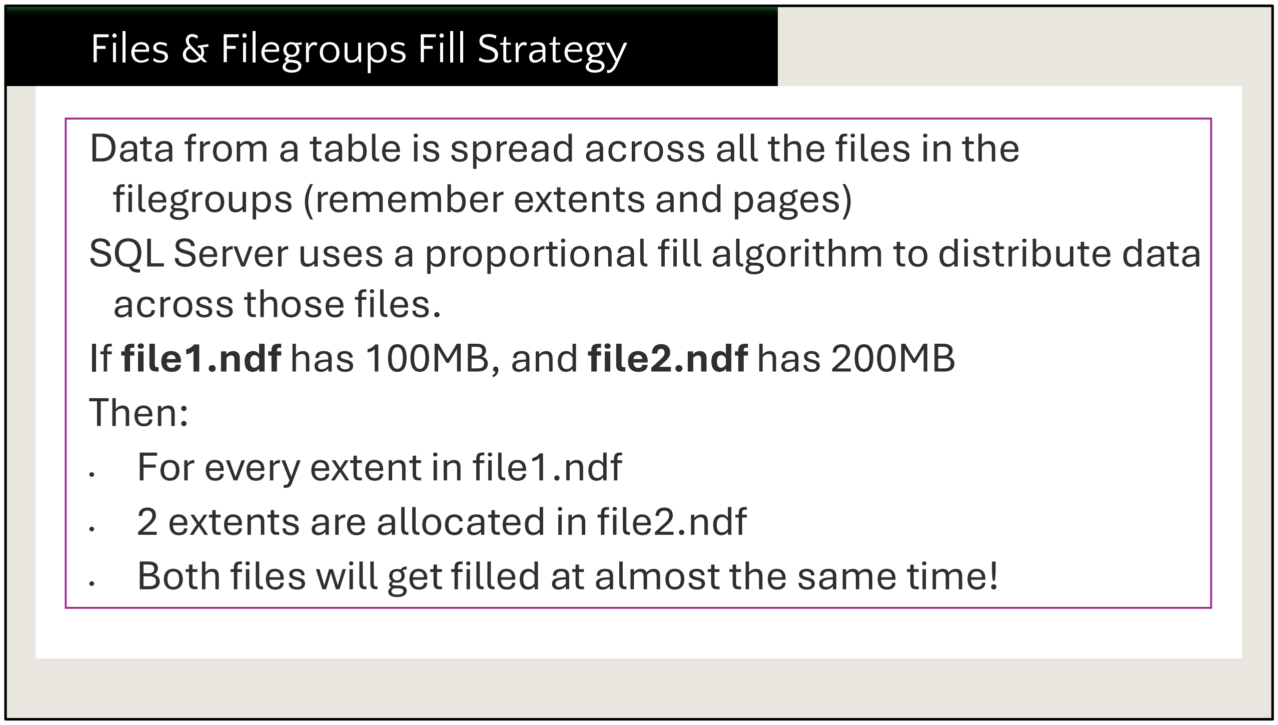
User-defined filegroups can be created to group data files together for administrative, data allocation, and placement purposes.

Queries for data from the table will be spread across the three disks

- It will improve performance.
- Files and filegroups let you easily add new files to new disks.

For example: Data1.ndf, Data2.ndf, and Data3.ndf, can be created on three disk drives, respectively, and assigned to the filegroup fgroup1. A table can then be created specifically on the filegroup fgroup1.

The same/similar performance improvement can be accomplished by using a single file created on a RAID (redundant array of independent disks) stripe set (To be discussed later in the course).



Files & Filegroups Fill Strategy

Data from a table is spread across all the files in the filegroups (remember extents and pages)

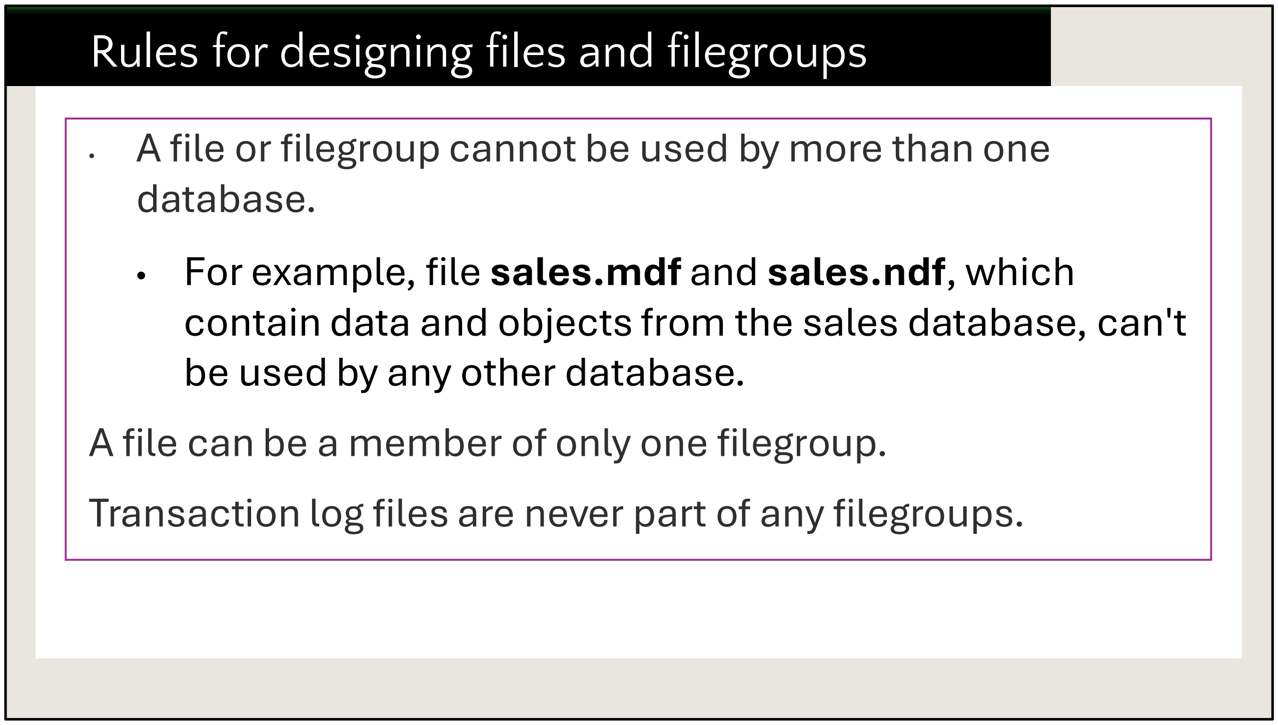
SQL Server uses a proportional fill algorithm to distribute data across those files.

If **file1.ndf** has 100MB, and **file2.ndf** has 200MB

Then:

- For every extent in file1.ndf
- 2 extents are allocated in file2.ndf
- Both files will get filled at almost the same time!

File and filegroup fill strategy
Filegroups use a proportional fill strategy across all the files within each filegroup. As data is written to the filegroup, the SQL Server Database Engine writes an amount proportional to the free space in the file to each file within the filegroup, instead of writing all the data to the first file until full. It then writes to the next file. For example, if file f1 has 100 MB free and file f2 has 200 MB free, one extent is given from file f1, two extents from file f2, and so on. In this way, both files become full at about the same time, and simple striping is achieved.
For example, a filegroup is made up of three files, all set to automatically grow. When space in all the files in the filegroup is exhausted, only the first file is expanded. When the first file is full and no more data can be written to the filegroup, the second file is expanded. When the second file is full and no more data can be written to the filegroup, the third file is expanded. If the third file becomes full and no more data can be written to the filegroup, the first file is expanded again, and so on.



Rules for designing files and filegroups

- A file or filegroup cannot be used by more than one database.
  - For example, file **sales.mdf** and **sales.ndf**, which contain data and objects from the sales database, can't be used by any other database.

A file can be a member of only one filegroup.

Transaction log files are never part of any filegroups.

Database Creation with Files & Filegroups

```
USE master;
GO
CREATE DATABASE MyDB
ON PRIMARY
  ( NAME='MyDB_Primary',
    FILENAME=
       ‘var/opt/mssql/data/MyDB_PRI.mdf',
    SIZE=4MB,
    MAXSIZE=10MB,
    FILEGROWTH=1MB),
```

Database Creation with Files & Filegroups

```
FILEGROUP MyDB_FG1
  ( NAME = 'MyDB_FG1_Dat1',
    FILENAME = ‘var/opt/mssql/data/MyDB_FG1_1.ndf',
    SIZE = 1MB,
    MAXSIZE=10MB,
    FILEGROWTH=1MB),
  ( NAME = 'MyDB_FG1_Dat2',
    FILENAME = ‘var/opt/mssql/data/MyDB_FG1_2.ndf',
    SIZE = 1MB,
    MAXSIZE=10MB,
    FILEGROWTH=1MB),
```

Database Creation with Files & Filegroups

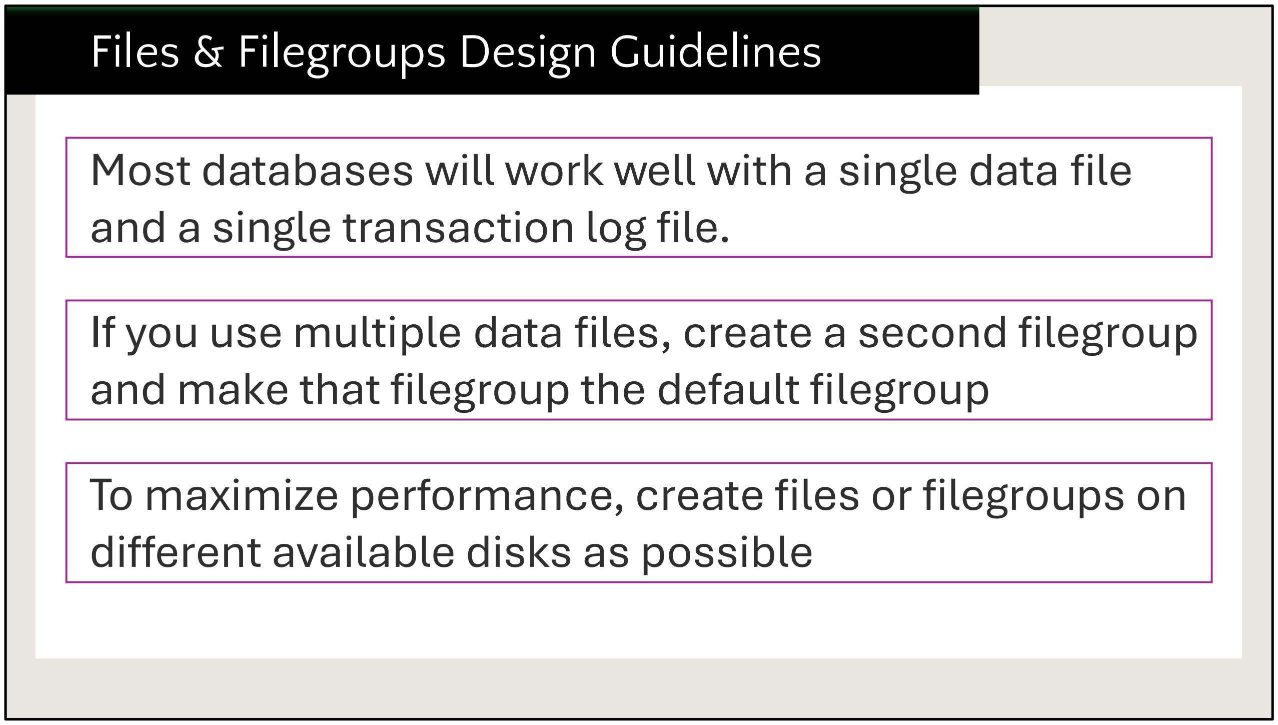
```
LOG ON
  ( NAME='MyDB_log',
    FILENAME = ‘/var/opt/mssql/data/MyDB.ldf',
    SIZE=1MB,
    MAXSIZE=10MB,
    FILEGROWTH=1MB);
GO
```

Database Creation with Files & Filegroups

```
ALTER DATABASE MyDB
  MODIFY FILEGROUP MyDB_FG1 DEFAULT;
GO

USE MyDB;
CREATE TABLE MyTable
  ( col_a int PRIMARY KEY,
    col_b char(8) )
ON MyDB_FG1;
GO
```

Files & Filegroups Design Guidelines



Files & Filegroups Design Guidelines

Most databases will work well with a single data file and a single transaction log file.

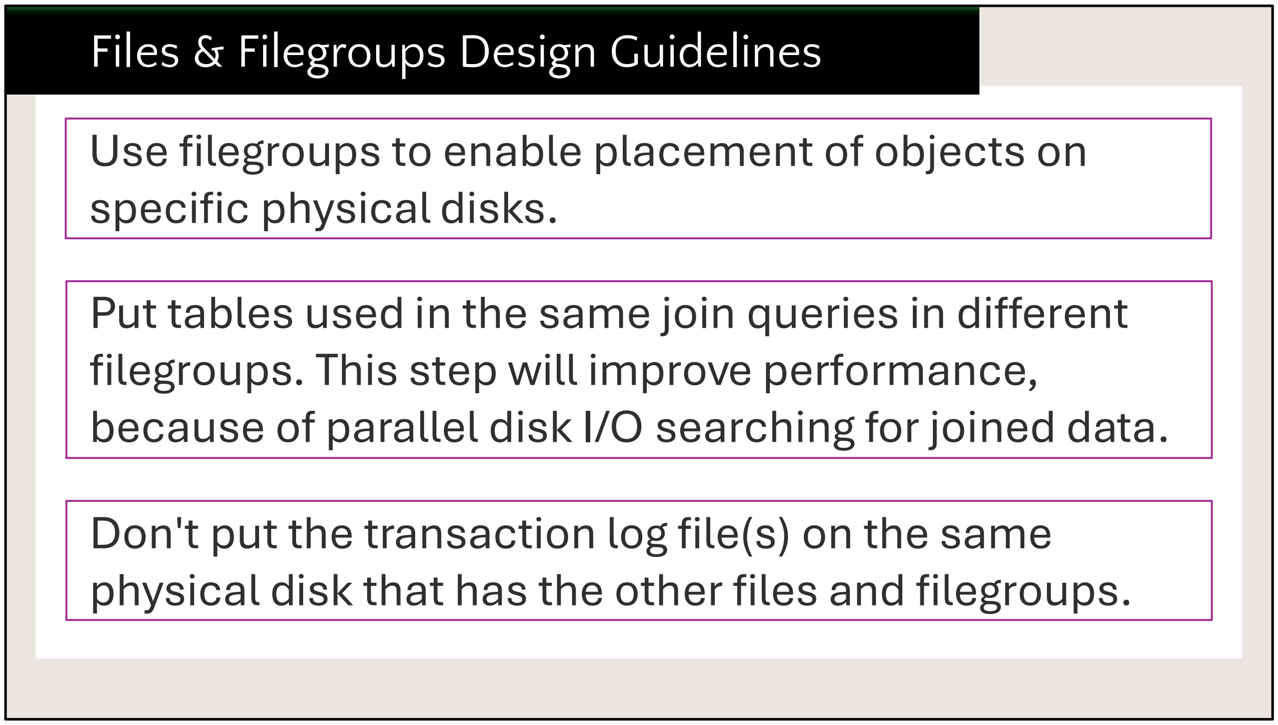
If you use multiple data files, create a second filegroup and make that filegroup the default filegroup

To maximize performance, create files or filegroups on different available disks as possible

- Most databases will work well with a single data file and a single transaction log file.
- If you use multiple data files, create a second filegroup for the additional file and make that filegroup the default filegroup. In this way, the primary file will contain only system tables and objects.
- To maximize performance, create files or filegroups on different available disks as possible. Put objects that compete heavily for space in different filegroups.
- Use filegroups to enable placement of objects on specific physical disks.
- Put different tables used in the same join queries in different filegroups. This step will improve performance, because of parallel disk I/O searching for joined data.
- Put heavily accessed tables and the nonclustered indexes that belong to those tables on different filegroups. Using different filegroups will improve performance, because of parallel I/O if the files are located on

different physical disks.

- Don't put the transaction log file(s) on the same physical disk that has the other files and filegroups.



Files & Filegroups Design Guidelines

Use filegroups to enable placement of objects on specific physical disks.

Put tables used in the same join queries in different filegroups. This step will improve performance, because of parallel disk I/O searching for joined data.

Don't put the transaction log file(s) on the same physical disk that has the other files and filegroups.

- Most databases will work well with a single data file and a single transaction log file.
- If you use multiple data files, create a second filegroup for the additional file and make that filegroup the default filegroup. In this way, the primary file will contain only system tables and objects.
- To maximize performance, create files or filegroups on different available disks as possible. Put objects that compete heavily for space in different filegroups.
- Use filegroups to enable placement of objects on specific physical disks.
- Put different tables used in the same join queries in different filegroups. This step will improve performance, because of parallel disk I/O searching for joined data.
- Put heavily accessed tables and the non-clustered indexes that belong to those tables on different filegroups. Using different filegroups will improve performance, because of parallel I/O if the files are located on

different physical disks.

- Don't put the transaction log file(s) on the same physical disk that has the other files and filegroups.