

Big Data Working with Terabytes in SQL Server

Andrew Novick

www.NovickSoftware.com

Agenda

§ Challenges

§ Architecture

§ Solutions

Introduction

§ Andrew Novick – Novick Software, Inc.

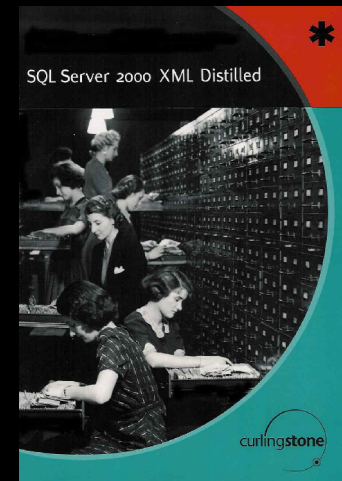
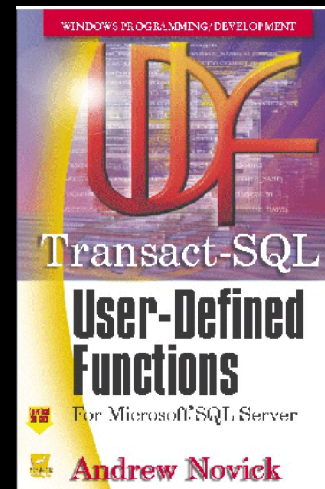
§ Business Application Consulting

- SQL Server
- .Net

§ www.NovickSoftware.com

§ Books:

- Transact-SQL User-Defined Functions
- SQL 2000 XML Distilled



What's Big?

§ 100's of gigabytes and up

to 10's of terabytes

§ 100,000,000 rows and up

to 100's of Billions of rows

Big Scenarios

§ Data Warehouse

§ Very Large OLTP databases
(usually with reporting functions)

Big Hardware

§ Multi-core 8-64

§ RAM 16 GB to 256 GB

§ SAN's or direct attach RAID

§ 64 Bit SQL Server

Challenges

Challenges

§ Load Performance (ETL)

§ Query Performance

§ Data Management Performance

How fast can you load
rows into the database?

Load 1,000,000 rows into
a 400,000,000 million row
table that has 12 indexes?

12 Hours

Load 1,000,000 rows into an
empty table and add 12
indexes?

5 Minutes

How do you speed up
queries on
1,000,000,000
row tables?

Challenge - Backup

§ Let's say you have a 10 TB database.

§ Now back that up.

Backup Calculation

§ 10 TB = 10000 GB

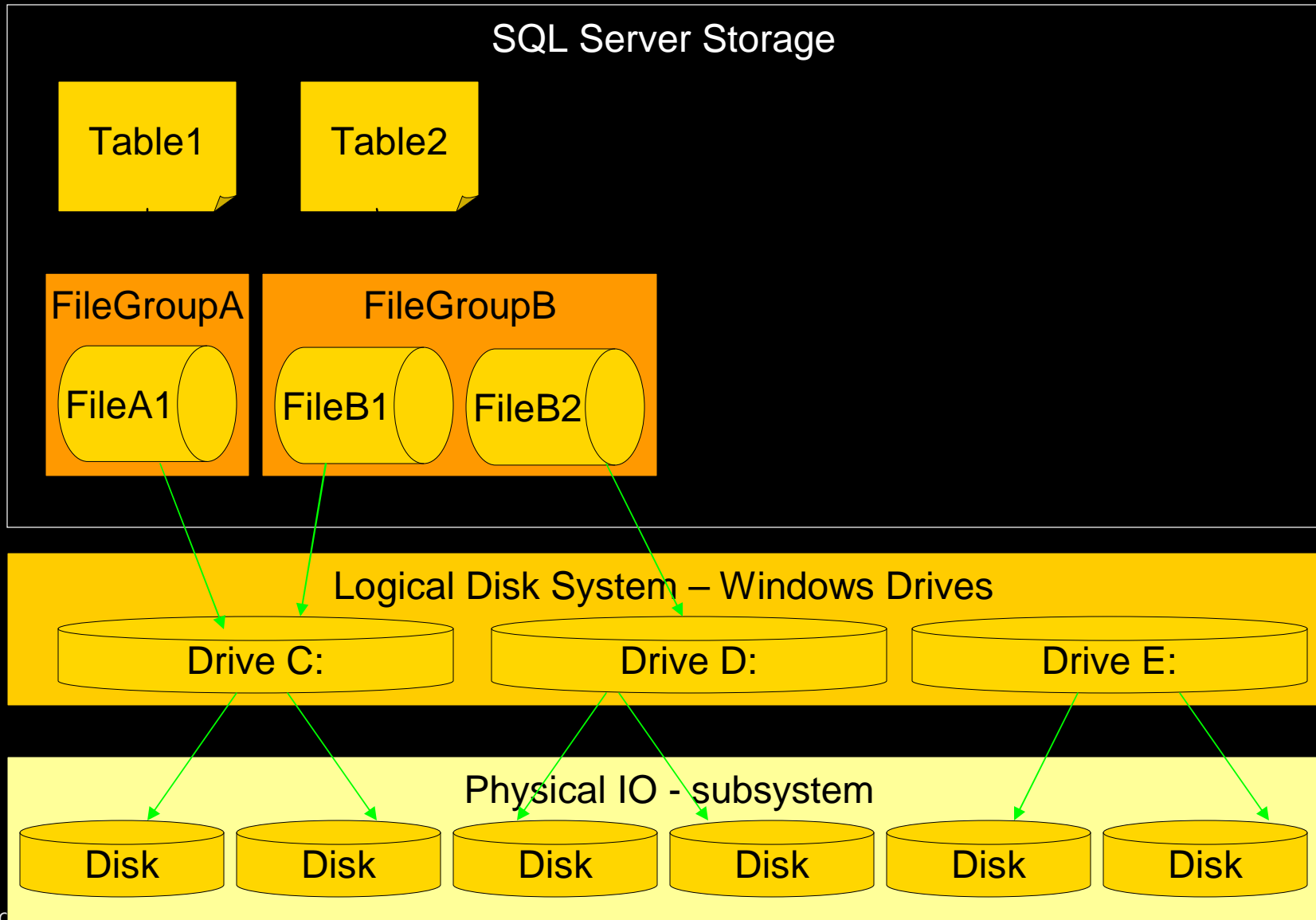
§ Typical Backup speed - 1 to 20 GB / Min

§ At 10 GB/Minute Who as 16
hours to spare?

Architecture

What do we have to work with?

SQL Server Storage Architecture



Solutions

Solutions

§ Use Multiple FileGroups/Files

§ INSERT into empty unindexed tables

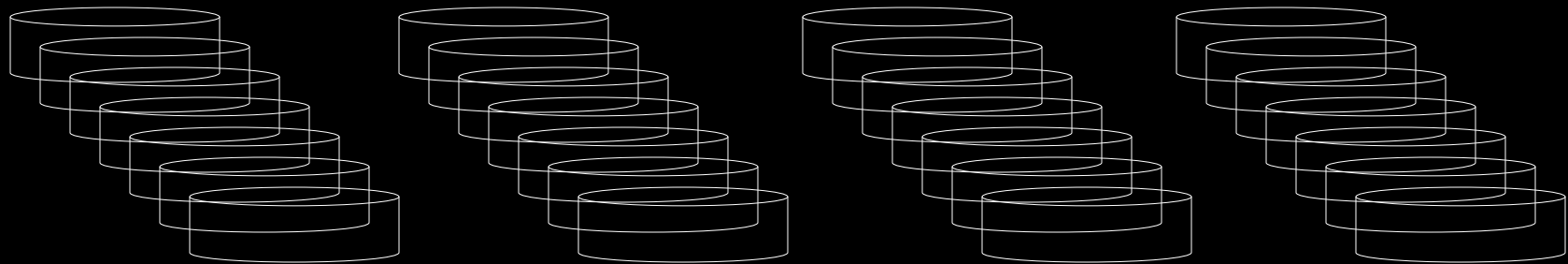
§ Partitioned Tables and/or Views

§ Use READ_ONLY FileGroups

At 3 PM on the 1st of the month:

**Where do you want your
data to be?**

Spread to as many disks as possible



I/O Performance

§ Little has changed in 50 years

**I/O throughput is a function of
the number of disk drives.**

§ Watch out for bottlenecks in the I/O Path

§ Memory reduces the need for I/O

Solution: Load Performance

§ Insert into empty tables

§ Index and add foreign keys after the insert

§ Add the Slices to

- Partitioned Views
- Partitioned Tables

Partitioning

Partitioned Views

Created like any view

```
CREATE VIEW Fact AS
    SELECT * FROM Fact_20080405
UNION ALL SELECT * FROM Fact_20080406
```

Partitioned Views: Check Constraints

§ Check constraints tell SQL Server which data is in which table

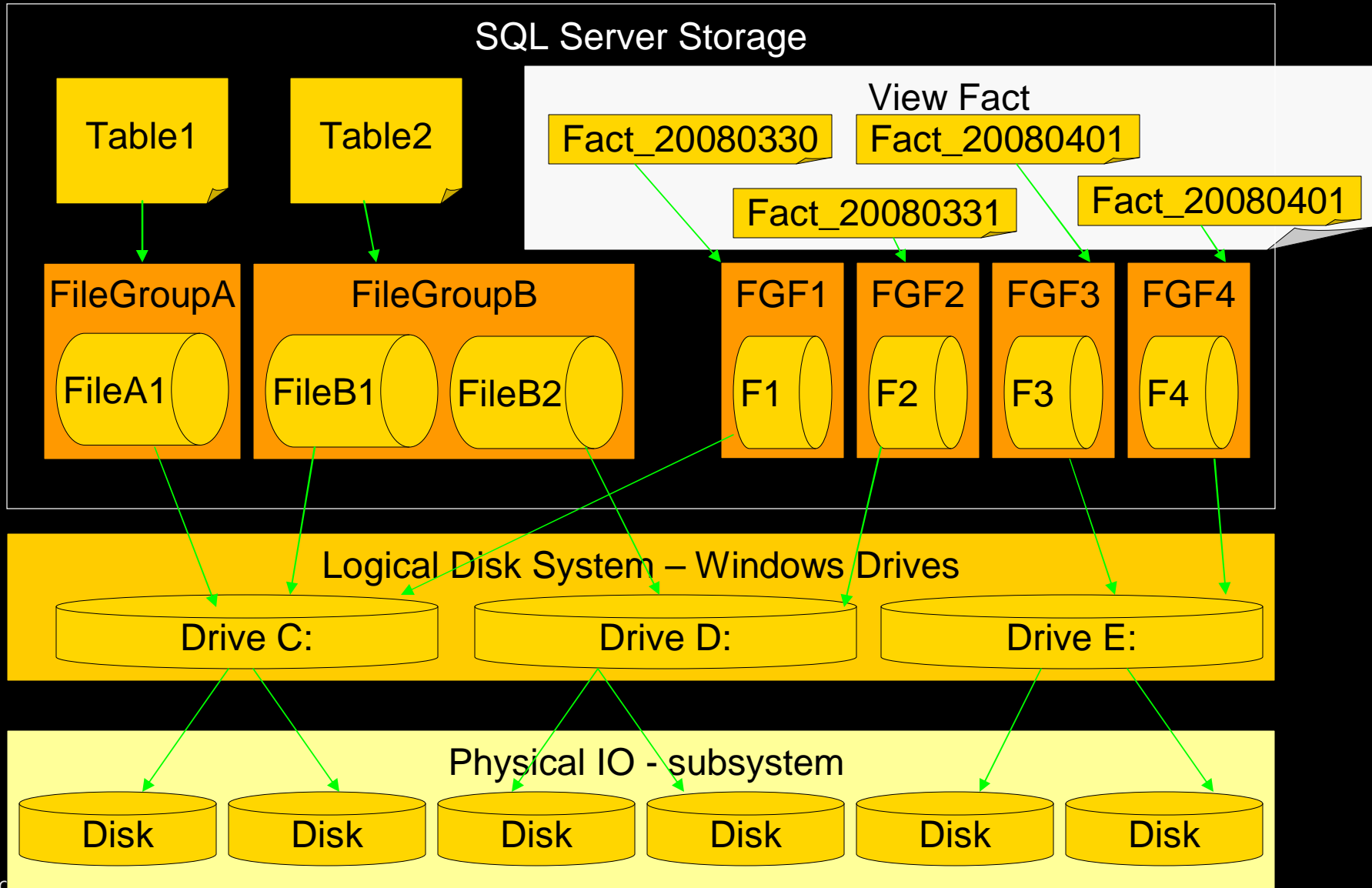
```
ALTER TABLE Fact_20080405
  ADD CONSTRAINT CK_FACT_20080405_Date
  CHECK (FactDate >= '2008-04-05'
        and FactDate < '2008-04-06' )
```

Partitioned View - 2

§ Looks to a query like any table or view

```
SELECT FactDate, .....  
FROM Fact  
WHERE CustID=334343  
AND FactDate = '2008-04-05'
```

Partitioned View



Partition Elimination

- § The query compiler can eliminate partitions from consideration in the plan
- § Partition elimination happens at query compile time.
- § It is often necessary to make partition values string constants.

Demo 1 – Partitioned Views

Partitioned Tables

§ SQL Server Enterprise/2005

§ Require a non-null partitioning column

§ Check constraints tell SQL Server what data is in each partition

§ All tables are partitioned!

Partitioned Function

§ Defines how to split data

```
CREATE PARTITION FUNCTION
```

```
    Fact_PF (small datetime)
```

```
    RANGE RIGHT FOR VALUES
```

```
        (' 2001-07-01' , ' 2001-07-02' )
```

Partition Scheme

§ Defines where to store each range of data

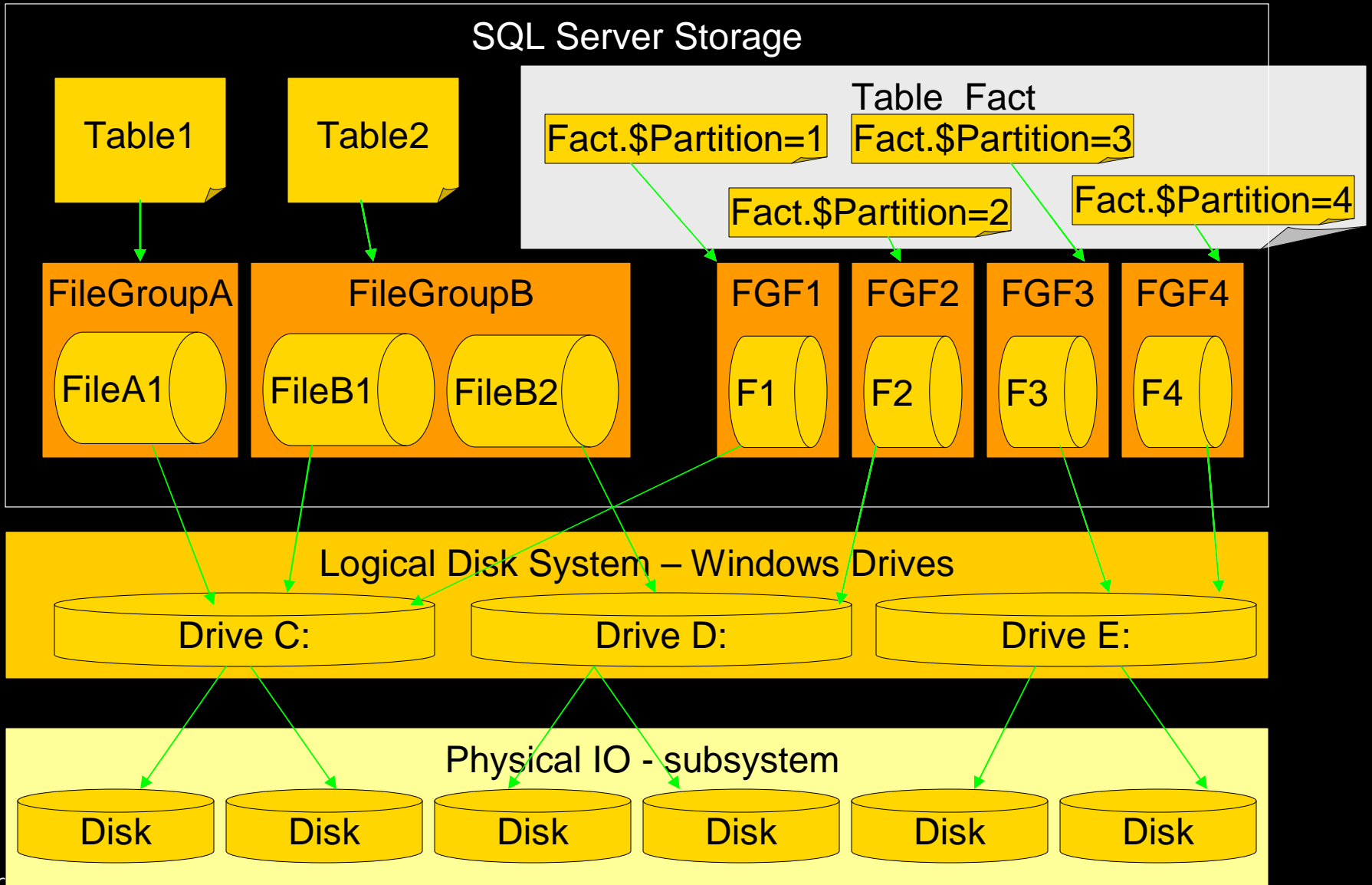
```
CREATE PARTITION SCHEME Fact_PS  
  
    AS PARTITION Fact_pf  
  
    TO (PRIMARY, FG_20010701, FG_20010702)
```

Creating a Partitioned Table

§ The table is created ON the Partitioned Scheme

```
CREATE TABLE Fact (Fact_Date smalldatetime  
                    , all my other columns)  
  
ON Fact_PS (Fact_Date)
```

Partitioned Table



Demo 2 – Partitioned Tables

Partitioning Goals

§ Adequate Import Speed

§ Maximize Query Performance

– Make use of all available resources

§ Data Management

– Migrate data to cheaper resources

– Delete old data easily

Achieving Query Speed

§ Eliminate partitions during query compile

§ All disk resources should be used

- Spread Data to use all drives
- Parallelize by querying multiple partitions

§ All available memory should be used

§ All available CPUs should be used

Issues with Partitioning

- § No foreign keys can reference the Partitioned Table
- § Identity columns must be more closely managed.
- § UPDATES on partitioned tables with part of the table in READ_ONLY filegroups must have partition elimination that restricts the updates to READ_WRITE filegroups.
- § INSERTs into partitioned views require all columns and face additional restrictions.

Solution: Backup Performance

§ Backup less!

§ Maintain data in a READ_ONLY state

§ Compress Backups

Read_Only FileGroups

```
ALTER DATABASE <database>  
    MODIFY FILEGROUP <filegroup> SET READ_ONLY
```

- § Requires only one Backup after becoming read_only
- § Don't require page or row locks
- § Don't require maintenance
- § The ALTER requires exclusive access to the database before SQL 2008

Partial Backup

§ Partial Base

- Backs up read_write filegroups

```
BACKUP DATABASE <db name>  
    READ_WRITE_FILEGROUPS .....
```

§ Partial Differential

- Differential backup of read_write filegroups

```
BACKUP DATABASE <db name>  
    READ_WRITE_FILEGROUPS  
    WITH DIFFERENTIAL .....
```

Maintenance Operations

§ Maintain only READ_WRITE data

- DBCC CHECKFILEGROUP

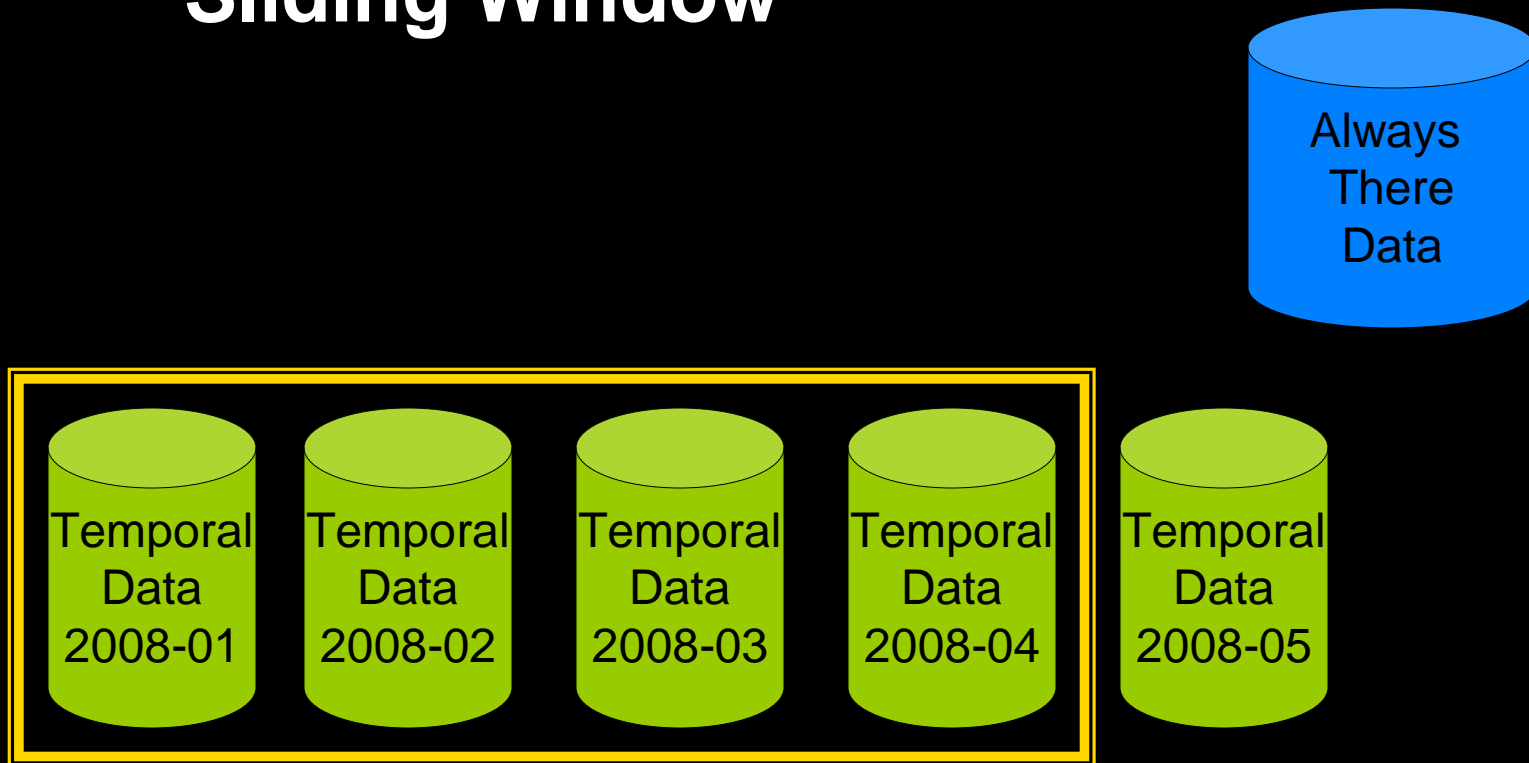
- ALTER INDEX

 - § REBUILD PARTITION =

 - § REORGANIZE PARTITION =

§ Avoid SHRINK

Sliding Window



SQL Server 2008 – What's New

- § Row, page, and backup compression
 - § Filtered Indexes
 - § Optimization for star joins
 - § MERGE T-SQL DML
 - § Partitioned Indexed Views
-
- § Fewer operations require exclusive access to the database

Thanks for Coming

Andrew Novick

§

anovick@NovickSoftware.com

§

www.NovickSoftware.com