

Module 2: Understanding Tables

Designing for Performance
Kimberly L. Tripp

Overview

- **Entities**
- **Attributes**
 - Data types
 - Nullability
- **Relationships**
- **Table Creation and Design**
 - **Internals**
 - The Data Row
 - Rows on a Page
- **Partitioning**
 - Vertical Partitioning
 - Horizontal Partitioning
- **Table Relationships**

Entities

- Basic principle behind a table – the way we describe some specific person, place or thing as a base object within the database
- Focus on a single element - Person, place, thing...
 - Authors – Each author is an Entity described by a row in the authors table.
 - Titles – Each title, separate from the author(s) that wrote the title is described by a row in the titles table.
 - Publishers – Each publisher, separate from any title(s) they may have published is described by a row in the publishers table.

Attributes

- Basic principle behind a column – the way we describe some specific element about the entity
- Each column of a table should describe **ONLY** that entity...
- Core elements of an attribute
 - Attribute Name price
 - Data type money
 - Nullability NULL

Every attribute should describe the primary key, the whole primary key, and nothing but the primary key.

Data Types

System Supplied Data Types

- Character
- Unicode Character
- Numeric
- Date and Time
- Exact Numeric
- Uniqueidentifier
- Binary
- Miscellaneous

Find the right data type for the job:

Use Unicode over ASCII

If the datatype varies:

< 5 chars should be nchar

5-20 chars – questionable

**> 20 char – lean more towards
nvarchar**

For numeric data:

Find the right range

**Standardize on decimal or
numeric**

Use uniqueidentifier sparingly

Understand precision and range

Data Type	Max/Exact Storage	Notes
char(n) varchar(n)	8000 bytes	1 byte per char - 8000 Characters
nchar(n) nvarchar(n)	8000 bytes	2 bytes per char - 4000 Characters
tinyint	1 byte	0-255
smallint	2 bytes	-32768 to 32767
int	4 bytes	-2 ³¹ to 2 ³¹ -1
bigint	8 bytes	-2 ⁶³ to 2 ⁶³ -1
uniqueidentifier	16 bytes	98E94963-F193-4E69-9262-7B692125557F
smalldatetime	4 bytes	Jan 1, 1900 with precision to the minute
datetime	8 bytes	Jan 1, 1753 with precision to a timetick (3.33 ms)
smallmoney	4 bytes	- 214,748.3648 through +214,748.3647
money	8 bytes	-922,337,203,685,477.5808 through +922,337,203,685,477.5807

Character Data

- **Unicode – nchar, nvarchar, ntext**
 - Application (web data) is stored natively
 - Supports wide variety of client environments
 - Make sure to ALWAYS use 'N'
- **Examples:**

```
SELECT * FROM tname WHERE col = N'value'
DECLARE @variable nvarchar(100)
SELECT @variable = N'value'
SELECT @Str = N'string' + N'string' + N'string'
```
- **CLR Extensibility requires unicode!**
- **ASCII – char, varchar, text**
 - Think before using if new data/tables

Nullability

- **No specific value required at INSERT**
- **NULL values DO NOT mean empty space**
(stored separately from the column data)
- **Working with NULLs**
 - Accessing columns which allow NULL values can cause inconsistencies when developers/users are not aware of them
 - Math with NULL values can produce interesting results (? – NULL = NULL)
 - ANSI session settings can affect results sets when accessing columns that allow nulls
- **NULL values v. DEFAULT value**

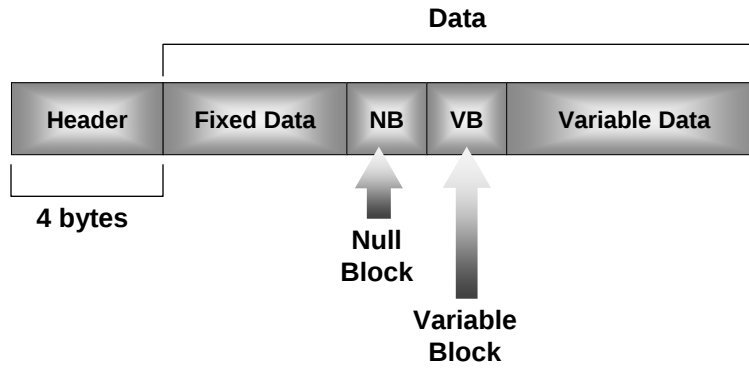
Owner Qualify on Creation

- **CREATE TABLE TName -> dbo.TName**
 - Users connected as a system administrator
 - Creator of a database (if not system admin)
 - Assigned owner of a database – if changed using sp_changedbowner
- **CREATE TABLE TName -> User.TName**
 - Users who are just aliased to dbo (old way)
 - Users who are a member of the db_owner or db_ddladmin role
- **CREATE TABLE dbo.TName**
 - Can be executed by ANY of the above
 - Leads to consistent object creation and ownership (SQL Server 2005 will have schemas in addition to users)

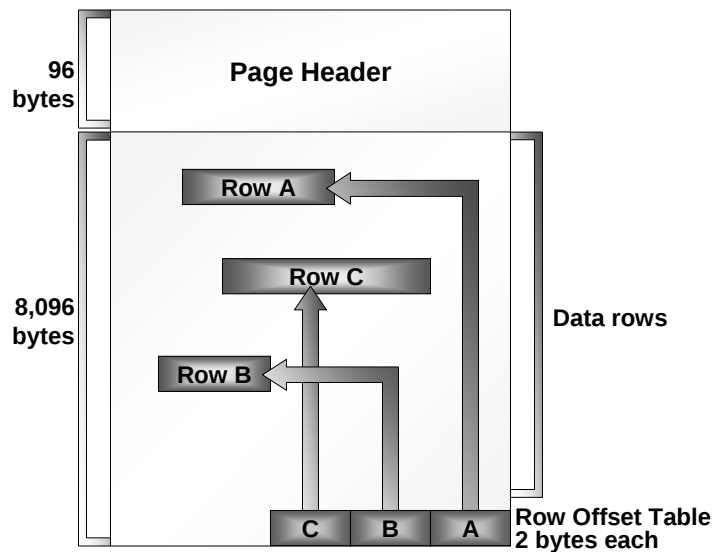
Design Techniques

- **Table Width for Optimal Cache Utilization**
 - The Data Row
 - Rows on a Page
 - Optimal Row Width
 - Vertical Partitioning
- **Constraints**
 - Check Constraints
 - Horizontal Partitioning

The Data Row



Rows on a Page



Optimal Row Width

- Consider Table Usage above all else
- Estimate Average Row Length
 - Overhead
 - Fixed Width Columns
 - Average from realistic sample data
`SELECT avg(datalength(column)) FROM tname`
- Calculate Rows/Page

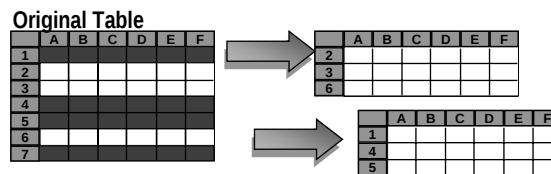
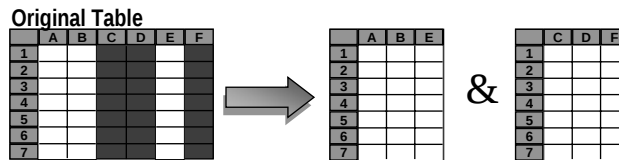
$$\frac{8096 \text{ Bytes/Page}}{??? \text{ Bytes/Row}} = \text{Rows/Page}$$
- Calculate Wasted Bytes/Disk ∴ in Memory

Optimization

- Partitioning
 - Vertical Partitioning to minimize row size
 - Caching
 - Locking
 - Horizontal Partitioning to minimize the impact of maintenance and access – OLTP v. OLAP
 - Table Maintenance
 - Table Access
- Adding Redundant Keys
 - Reduces the number of tables specified in a Join
 - Gives the Optimizer more options for performing Join

Partitioning Data

Vertical Partitioning

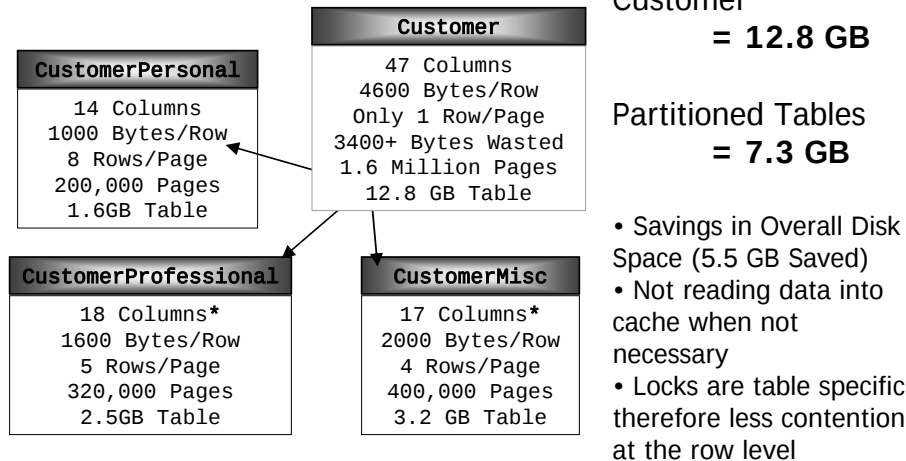


Horizontal Partitioning

Vertical Partitioning

- **Optimizing row size for...**
 - **Caching**
More rows on a page, more rows in memory
 - **Locking**
Only locking the columns that are of interest.
Minimize row based conflicts
- **Usage Defines Vertical Partitions**
 - **Logically Group Columns to minimize joins**
 - **Consider Read Only vs. OLTP Columns**
 - **Consider Columns often used together**

Consider Customer Table with 1,600,000 Rows



* The Primary key column must be made redundant for these two additional tables. 47 Columns in Employee. 49 Columns total between 3 tables.

Check Constraints (optimizer)

- All data will be checked on INSERT/UPDATE
 - Pattern
CHECK (Phone LIKE '(____) ____-____')
 - *Range*
CHECK (Salary < 750000)
The only operators supported for increased performance gains are: BETWEEN, AND, OR, <, <=, >, >=, =. Others are supported for data integrity purposes only.
 - List
CHECK (Country Code IN ('US', 'AU', 'GB', 'FR'))

DO NOT USE WITH NOCHECK

- ALTER TABLE with CHECK – all existing data is checked and verified
- ALTER TABLE with NOCHECK – speeds up the creation BUT there are no performance gains for later use of the constraint

Scenario

Charge Table 1.6 Million Rows

Constraint added with NOCHECK (ChargeAmt < 5000)

```
SELECT * FROM Charge WHERE ChargeAmt >
6000
-- yields 9305 I/O
```

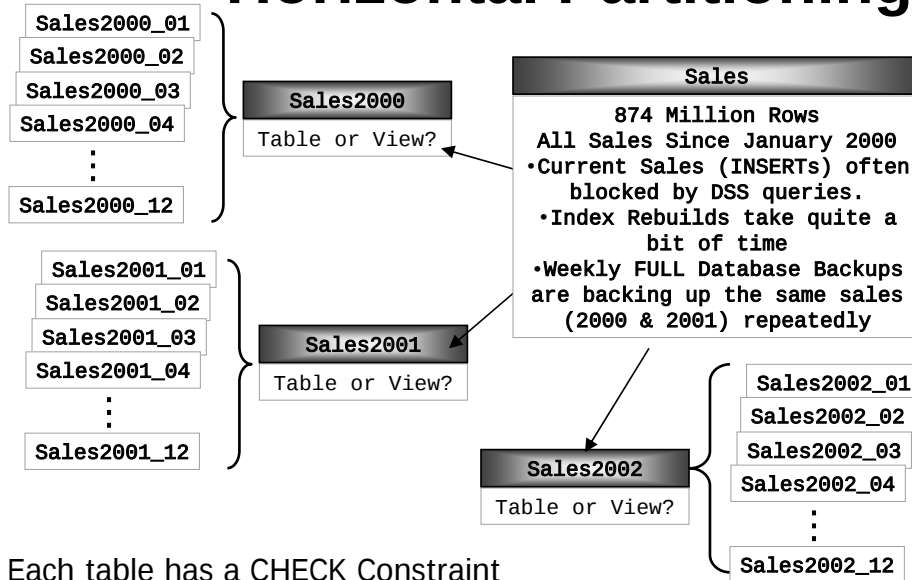
Constraint added with CHECK (ChargeAmt < 5000)

```
SELECT * FROM Charge WHERE ChargeAmt >
6000
```

Horizontal Partitioning

- A Single Large Table...
 - Can be Hard to Manage
 - May have different access patterns
 - Does NOT have to be one table!
- Minimize the impact of maintenance
 - Index Maintenance (Smaller/Less Frequent)
 - Backup/Restore (File/Filegroup strategies to minimize backup frequency of predominantly ReadOnly tables – more options for restore when isolated disk/RAID array corruption)
 - Lock Escalation (Modifications to one partition do not cause escalation against the others – as they likely would if everything were in one table!)

Horizontal Partitioning



How?

- **SQL Server 7.0 – Partitioned Views**
Queryable partitions
- **SQL Server 2000 – Updateable Partitioned Views**
Updateable partitions
- **SQL Server 2005 – Partitioned Tables and Indexes**
More manageable partitioned BASE structures – not many tables with a view

See session DBM405: Partitioning for more details on Partitioning in each release!

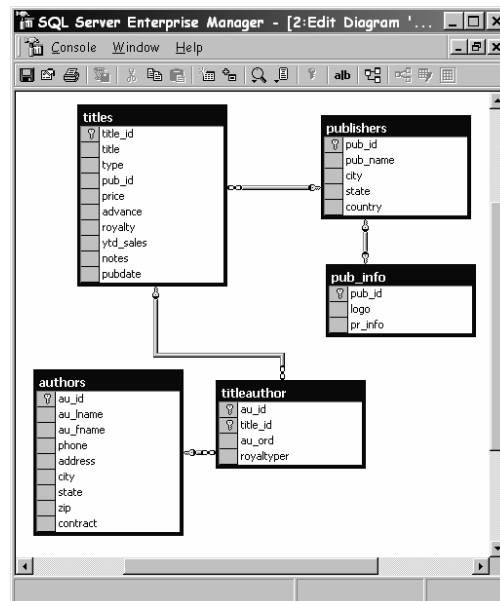
Relationships

- Basic principle behind a relational database – the way we describe the entities and their associations
- Relationships
 - One to One
 - For any Publisher there can be at most one Pub_info row (a FK which is also a PK)
 - One to Many (mathematically)
 - For any Publisher there can be many Titles
 - For any Title there can be only one Publisher
 - Many to Many
 - For any Author there can be many Titles, for any Title there can be many Authors

ERD – Entity Relationship Diagram

Relationships

- Titles have a Publisher
- Titles have potentially many Authors
- Authors may have written more than one Title
- For a given title an Author is given an order and a royalty

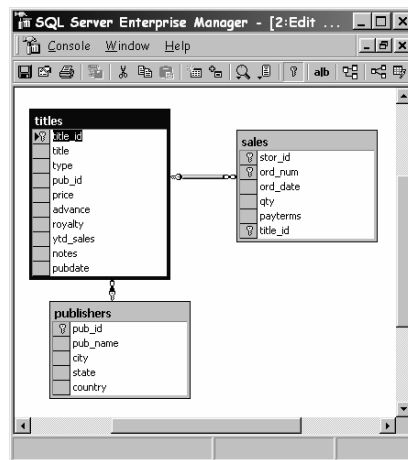


Describing Relationships

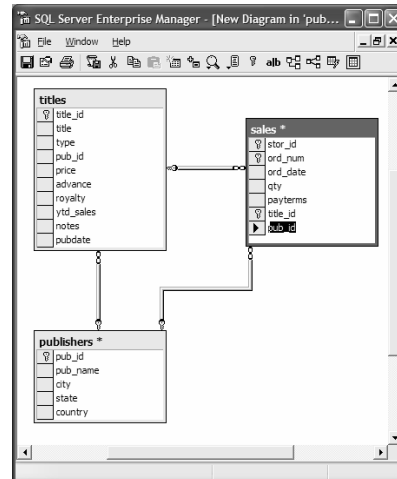
- **Mathematically – One to Many (1-n) Relationship**
 - For Every Sale there is Only one Title
 - For Each Title there can be Many Sales
The Relationship between Sales and Titles is One to Many
 - For Every Title there is Only one publisher
The Relationship between Titles and Publishers is One to Many
 - For Each Publisher there can be Many Titles
- **Directionally**
 - Sales to Titles = One Sale to One Title
 - Titles to Publishers = One Title to One Publisher
- **Can we describe Sales to Publishers?**

Duplicating Keys

BEFORE adding redundant pub_id



AFTER adding redundant pub_id



Design Techniques Summary

- Rows cannot span pages therefore record size can impact disk space, cache utilization and concurrency
- The lowest granularity of locking holds an entire row – partitioning a table vertically or horizontally can improve concurrency and cache utilization
- Optimizer is aware of constraints – efficient horizontal partitioning requires constraints
- Duplicating Keys which define relationships can improve join performance by giving the optimizer more options for performing a join

Key Points

- Minimize data redundancy by using attributes specific only to the entity table definition
- Specify Appropriate Data Types and Data Type Sizes – try to keep row sizes compact
- Always Specify Column Characteristics in CREATE TABLE – especially column Nullability
- Take into account the usage of the data when defining tables, consider partitioning tables based on usage
- Define Constraints on your tables – for performance as well as centralization

Review

- **Entities**
- **Attributes**
 - Data types
 - Nullability
- **Relationships**
- **Table Creation and Design**
 - **Internals**
 - The Data Row
 - Rows on a Page
- **Partitioning**
 - Vertical Partitioning
 - Horizontal Partitioning
- **Table Relationships**