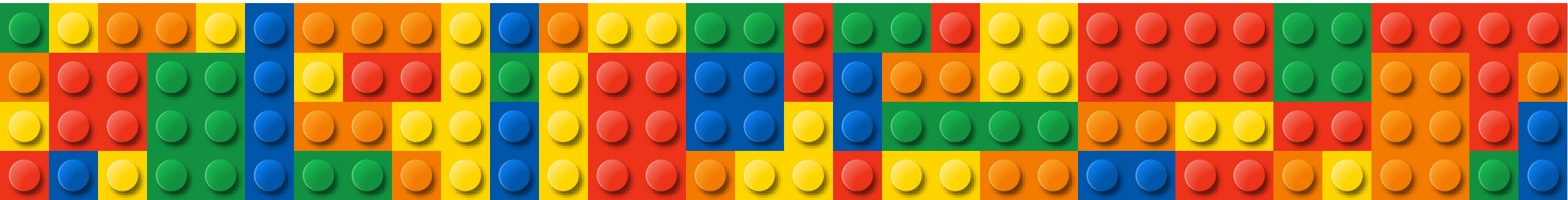




Building Better

SQL Server Databases





Who is this guy?

Eric Cobb

Database Development Manager

MCSE: Data Platform | MCSE: Data Management and Analytics

1999-2013: “Webmaster“, Programmer, Developer

2014+: SQL Server Database Administrator

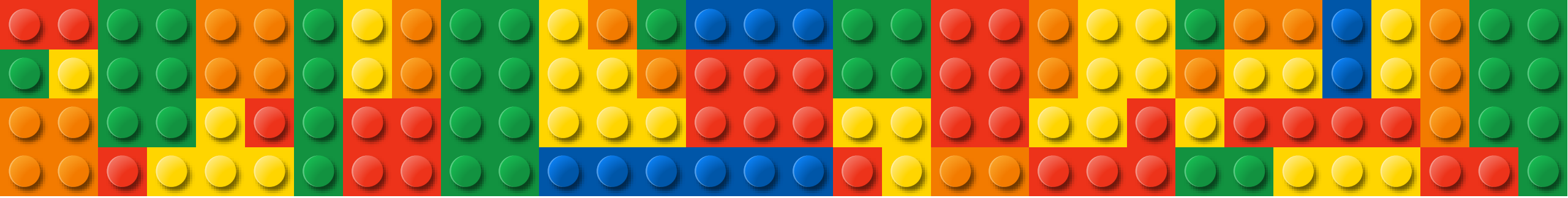
GitHub: <https://github.com/ericcobb>

Blog: <http://www.sqlnuggets.com>

Twitter: @sqlnugg

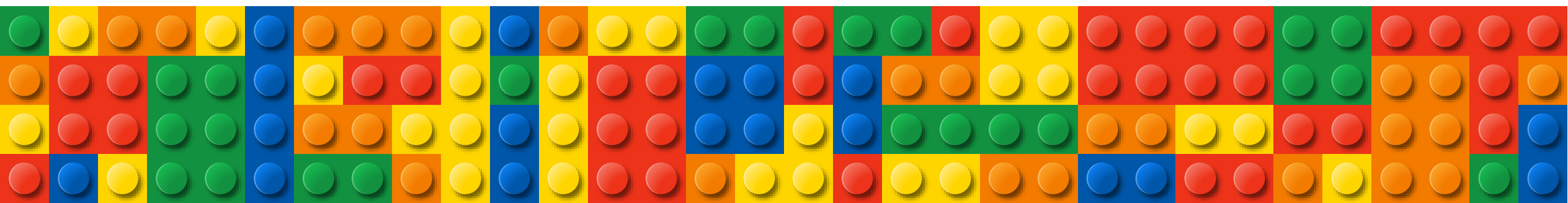
@cfgears





A PEEK UNDER THE HOOD OF SQL SERVER

A BRIEF OVERVIEW OF HOW SQL SERVER STORES AND RETRIEVES DATA





A Peek Under The Hood

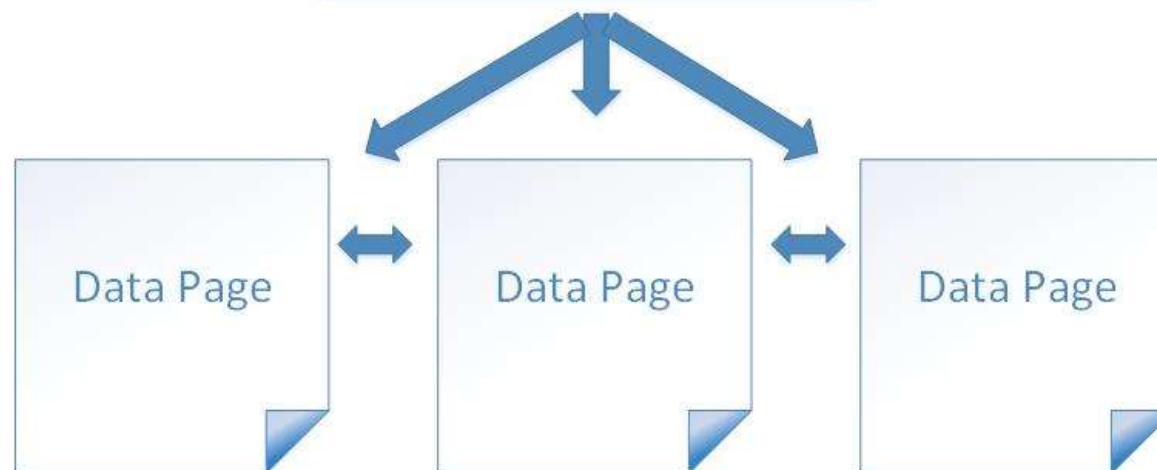
	EmployeeId	LastName	Country	DepartmentId
1	123	Penn	Australia	31
2	124	Fitch	Australia	33
3	145	Jackson	Australia	33
4	201	Shields	United States	34
5	305	Pierre	Germany	34
6	306	Velasquez	Germany	NULL
7	310	Lawal	United States	31
8	311	Bell	China	35
9	312	George	Japan	36





A Peek Under The Hood

	EmployeeId	LastName	Country	DepartmentId
1	123	Penn	Australia	31
2	124	Fitch	Australia	33
3	145	Jackson	Australia	33
4	201	Shields	United States	34
5	305	Pierre	Germany	34
6	306	Velasquez	Germany	NULL
7	310	Lawal	United States	31
8	311	Bell	China	35
9	312	George	Japan	36





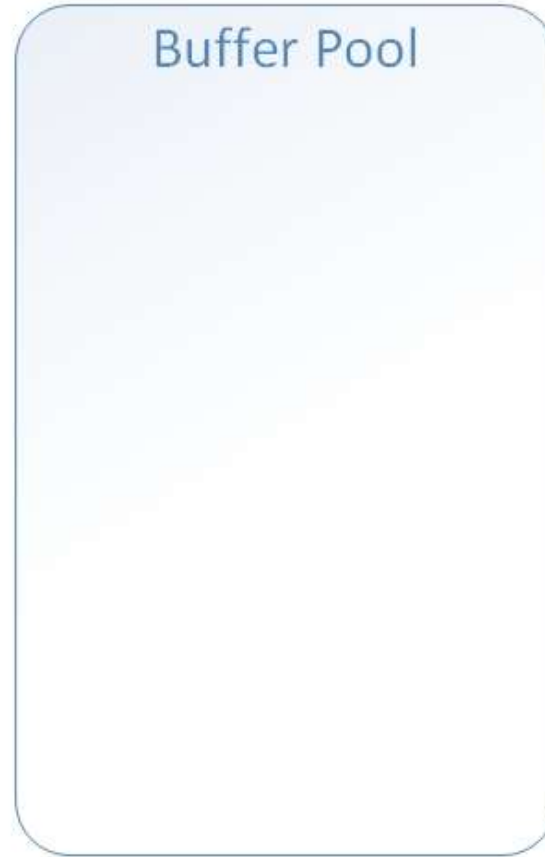
A Peek Under The Hood



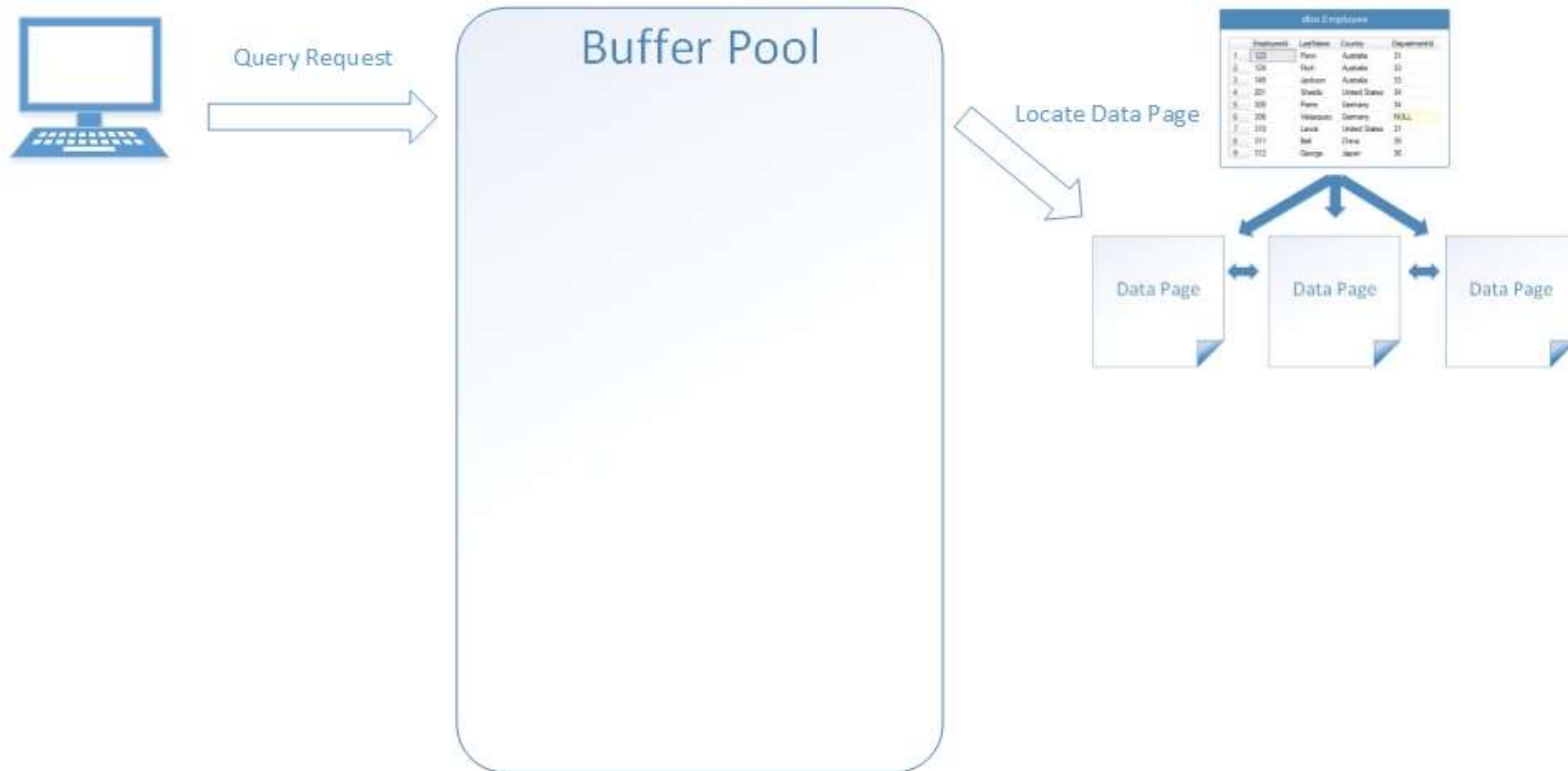
Query Request



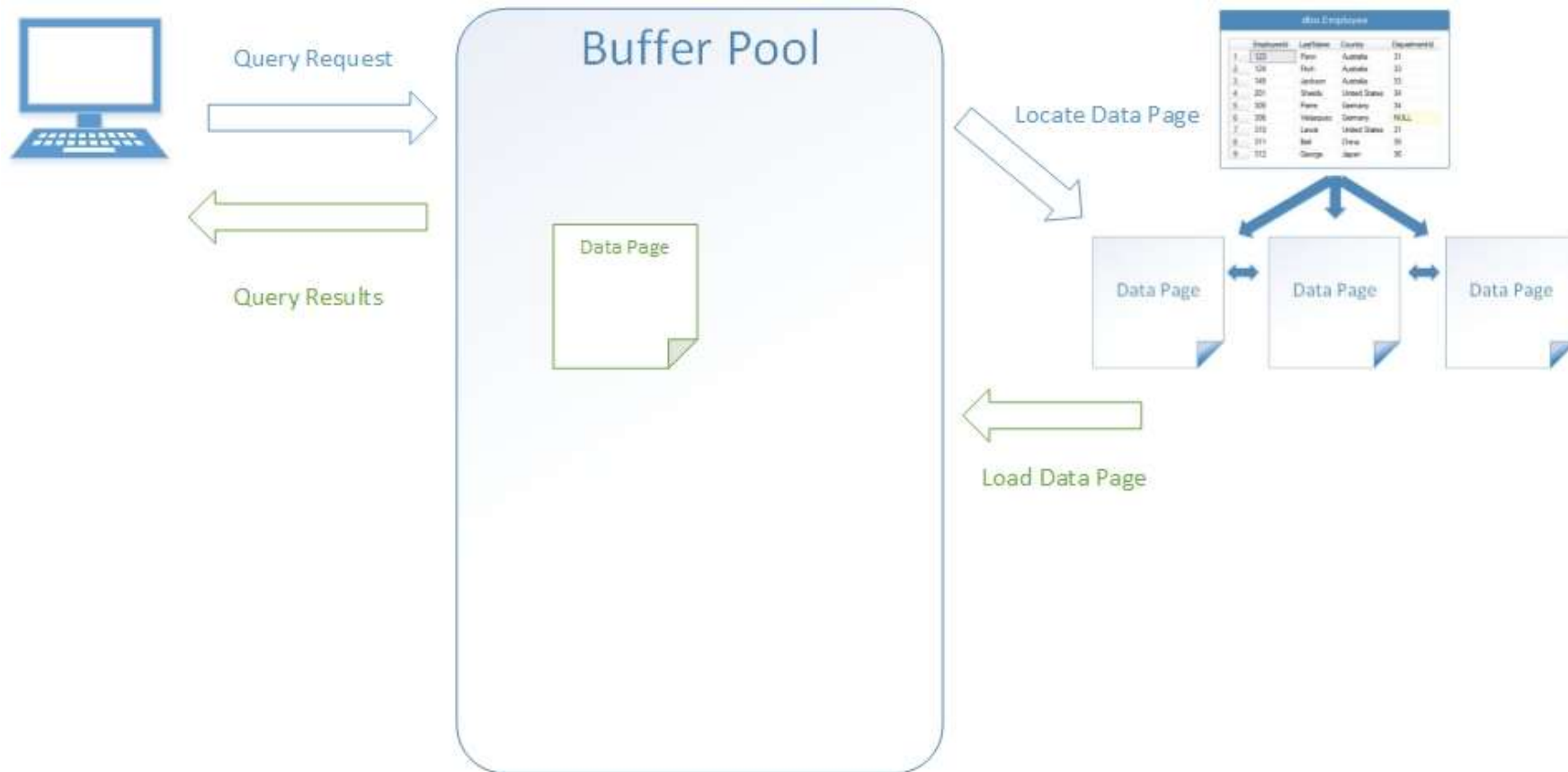
Buffer Pool



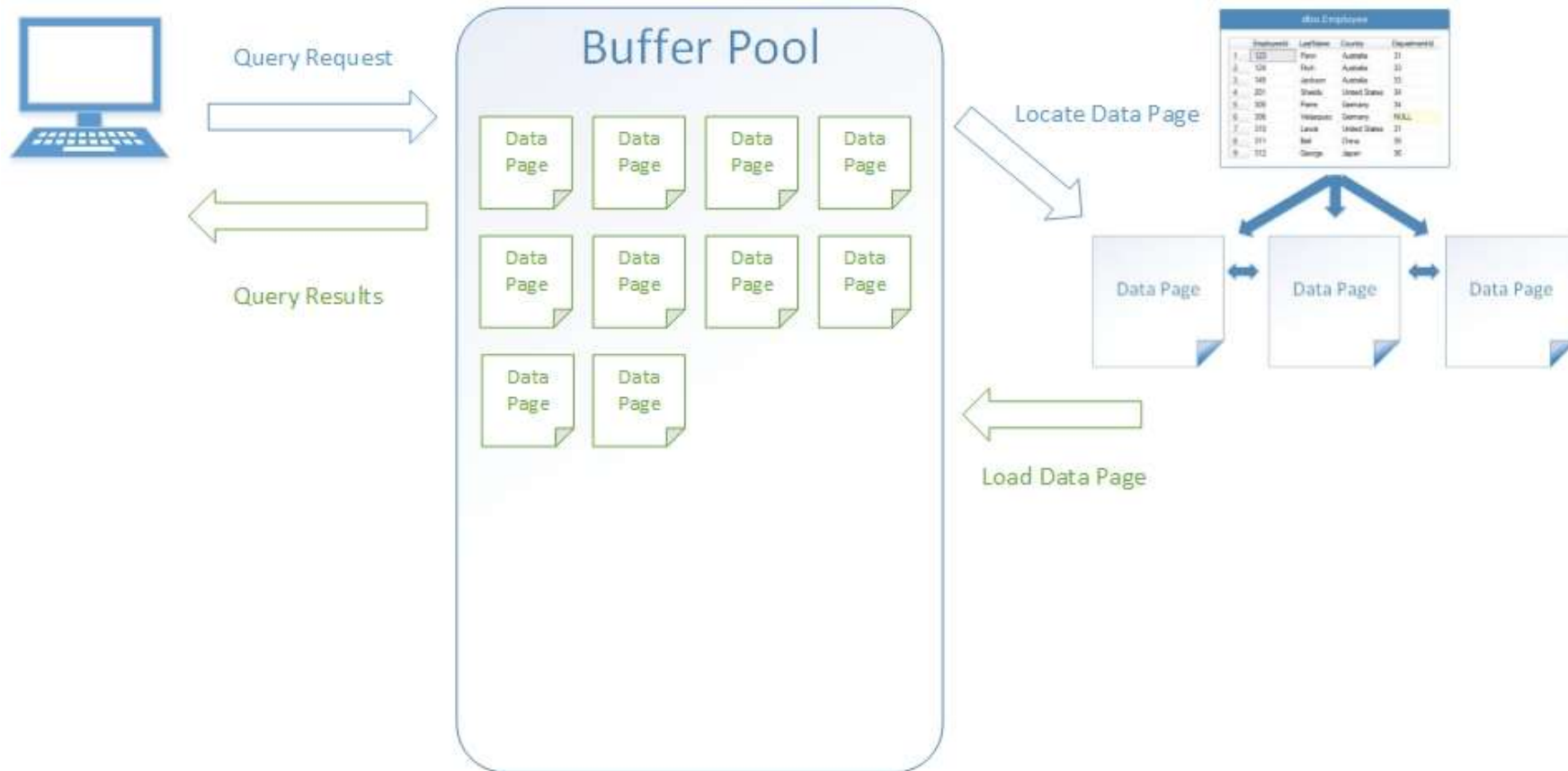
A Peek Under The Hood



A Peek Under The Hood



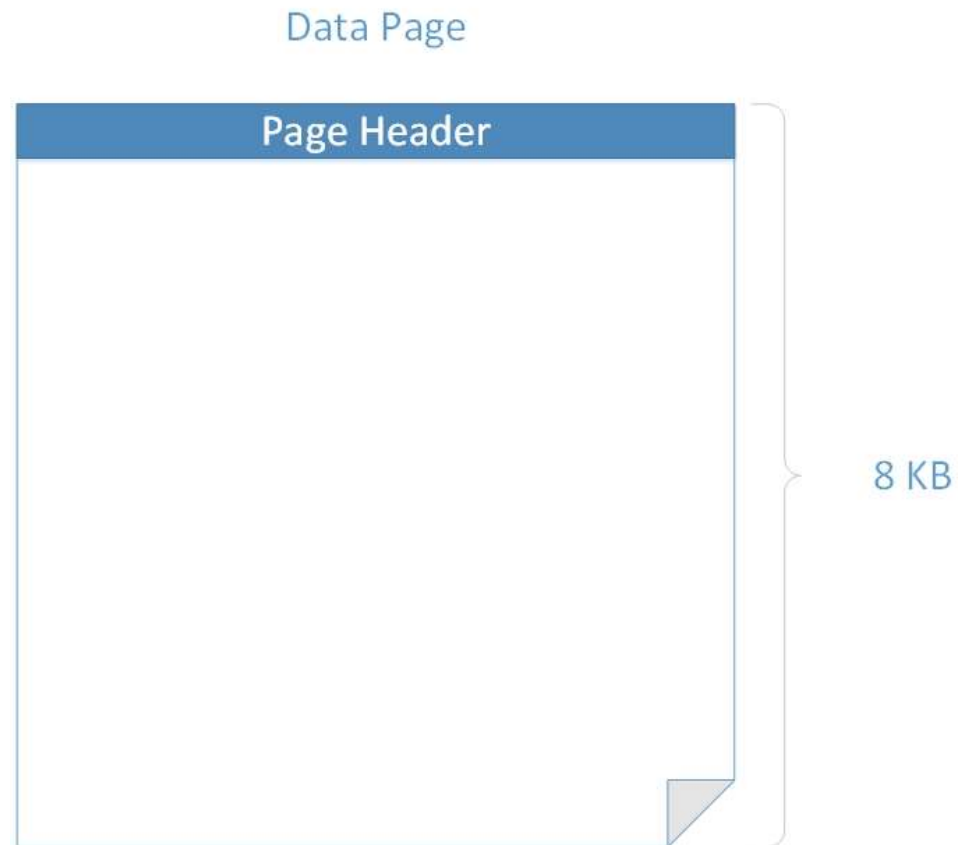
A Peek Under The Hood





A Peek Under The Hood

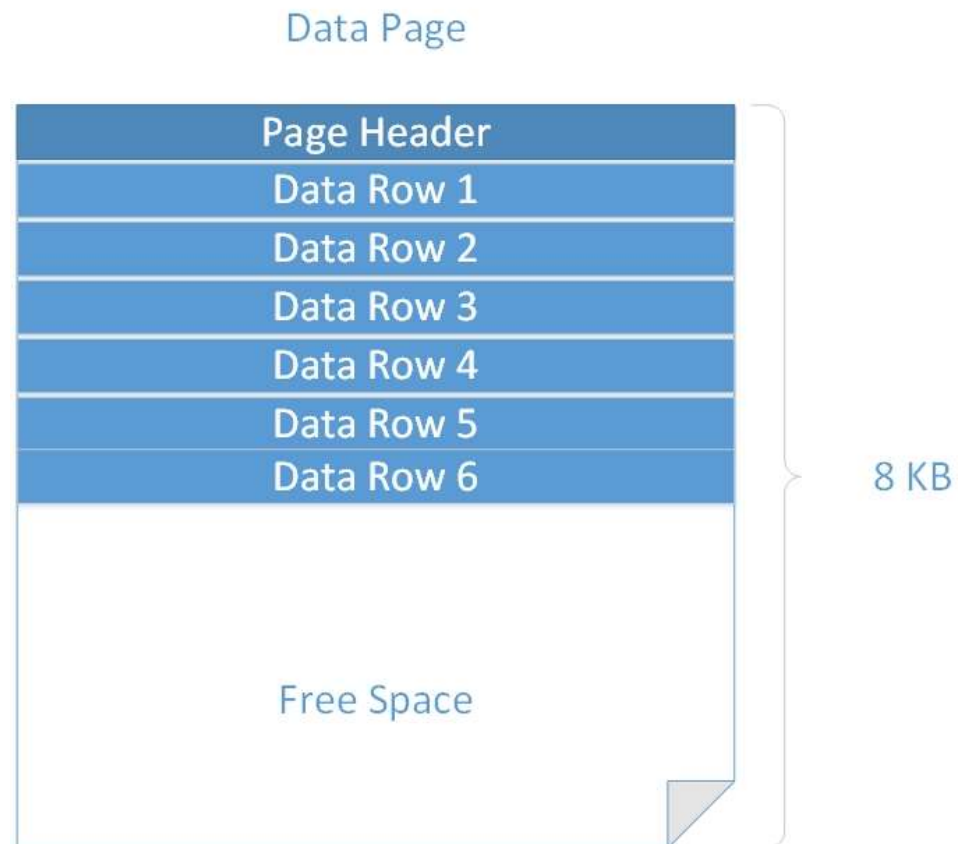
Storing Data in Pages





A Peek Under The Hood

Storing Data in Pages





A Peek Under The Hood

Storing Data in Pages

How is the data stored in a Page?

- Unordered (Heap)
 - Query optimizer reads all the rows in the table (table scan), to find the rows that meet the criteria of a query
 - A table scan generates many disk I/O operations and can be resource intensive
 - Heaps should generally be avoided, although can be useful when inserting large amounts of data in ETL/Bulk processes
- Ordered (Clustered Index)
 - Tells SQL Server how to physically sort the records on disk
 - The most important index you can apply to a table
 - Data pages are ordered, for faster data retrieval
 - There is only ever 1 clustered index on a table



A Peek Under The Hood

Storing Data in Pages

How do I create Clustered Indexes?

- Primary Key = Clustered Index (usually)
 - SQL Server automatically creates a clustered index on your Primary Key column if a clustered index does not already exist on the table
 - If you do not want the Primary Key to be your Clustered Index, you can create your Clustered Index on a different column
- Clustered Index (Primary Key) Tips:
 - Use a naturally occurring incremental value
 - Keep as small and narrow as possible (single columns are preferred)
 - Avoid using character data types for a Clustered Index



A Peek Under The Hood

Storing Data in Pages

Page Header						
LastName	...					
Adams						
Allan						
Barnes						
Davis						
Franklin						



A Peek Under The Hood

Page Splits

Page Header			
LastName		
Adams			
Allan			
Barnes			
Cobb			

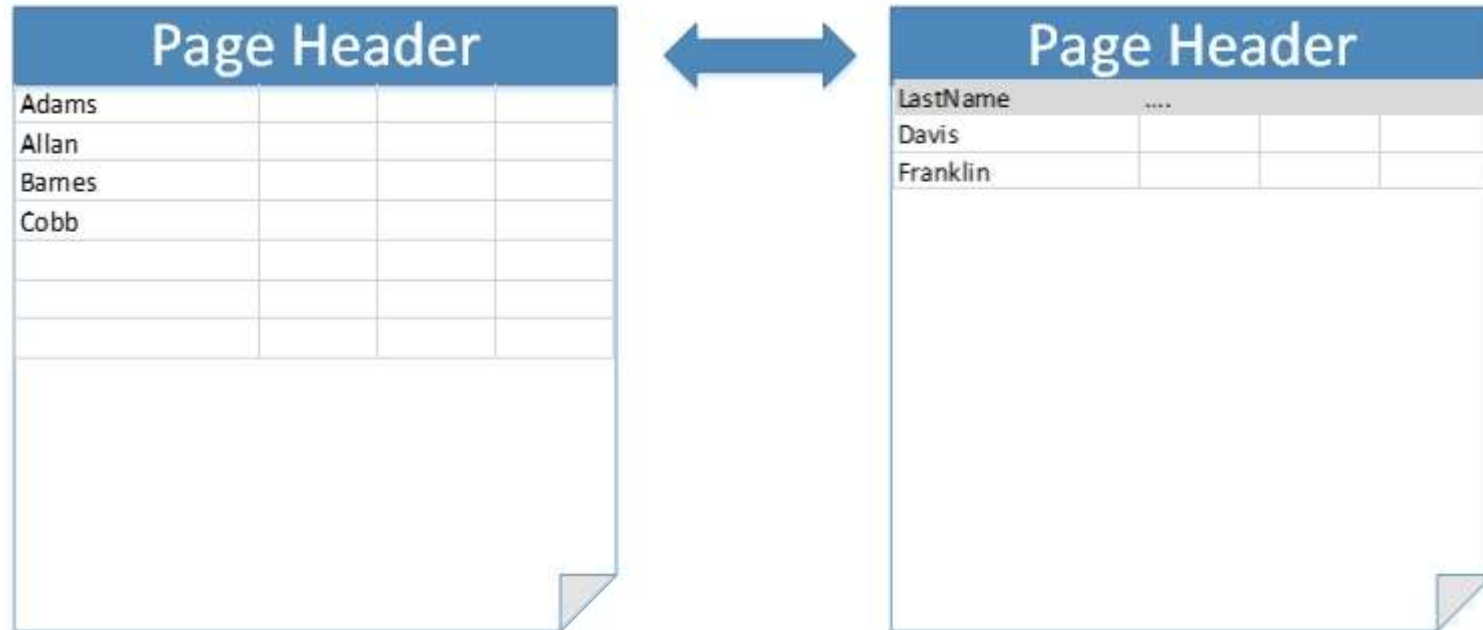


Page Header			
LastName		
Davis			
Franklin			



A Peek Under The Hood

Page Splits





A Peek Under The Hood

Page Splits





A Peek Under The Hood

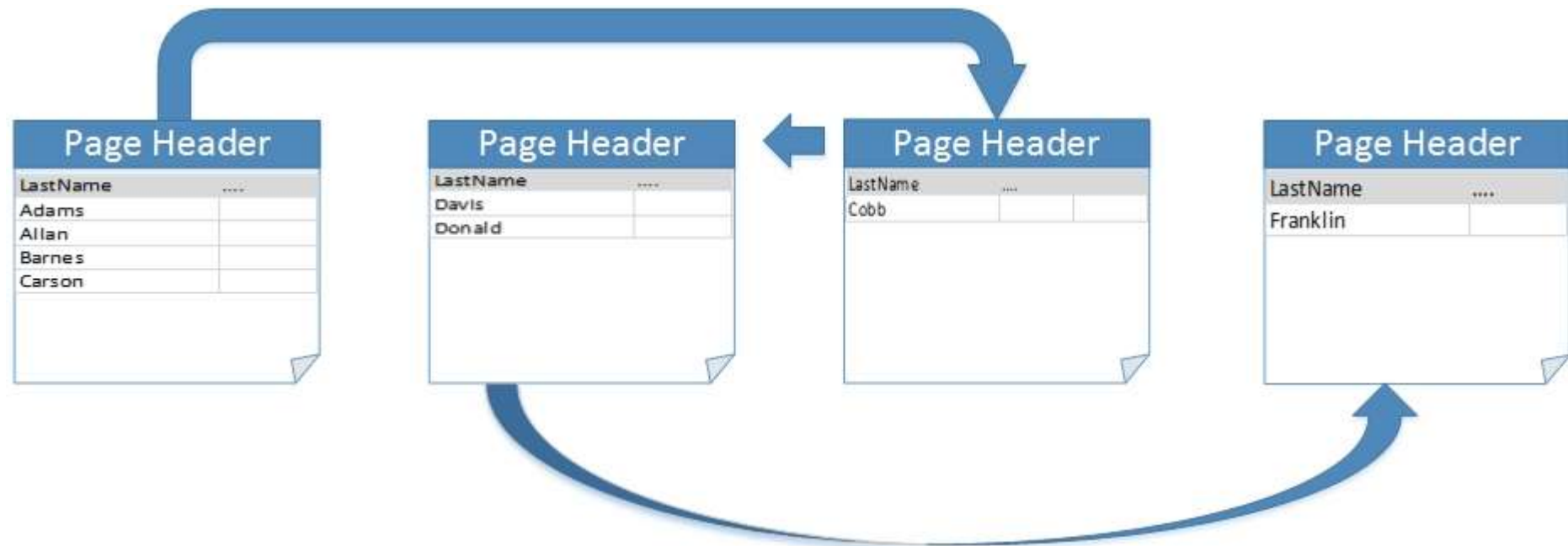
Page Splits





A Peek Under The Hood

Page Splits





A Peek Under The Hood

Page Splits

Clustered Index Tips

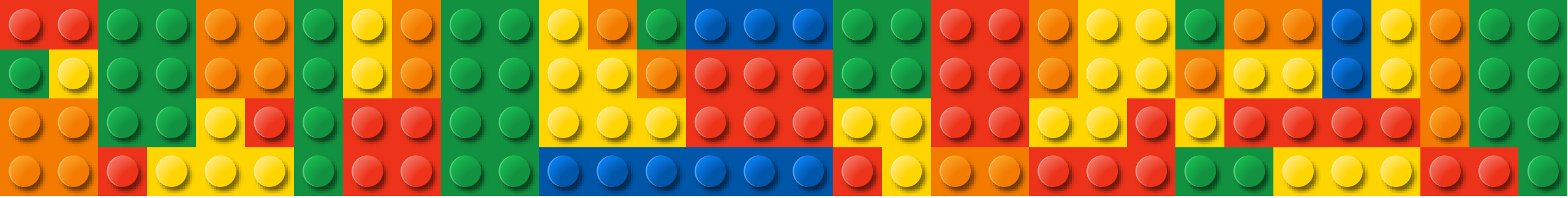
- Should be unique, narrow, static, and incremental
- Good Clustered Index examples:
 - A numeric identity column (smallint, int, bigint)
 - A pseudo sequential GUID (using the NEWSEQUENTIALID() function)
 - Not recommended, but the best you can do if you absolutely have to use a GUID
- Clustered Indexes to avoid:
 - Unique Identifier (GUID/UUID)
 - Character columns (CHAR, VARCHAR, NVARCHAR, etc...)
 - Columns that undergo frequent changes



A Peek Under The Hood

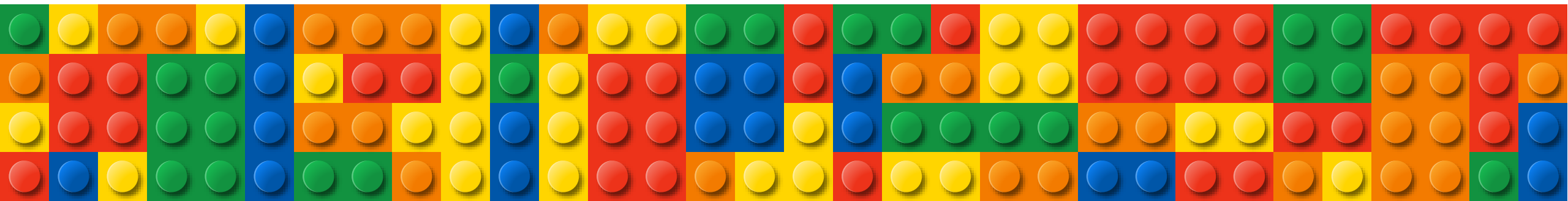
Page Splits

Demo



BUILDING BETTER TABLES

DESIGNING TABLES WITH EFFECIENCY IN MIND





Building Better Tables

Using The Right Data Types

Data Types Are Important!

- Choose your table column data types wisely
 - They can affect the performance of your database as it grows
- Know your data, use the appropriate data type for the data you are storing
 - The more accurate your data type is, the more efficiently SQL Server can handle your data
- Use the smallest data type possible (within reason)
 - The smaller the column, the less data you have to store and retrieve
 - The longest city name in the U.S. is *Rancho Santa Margarita* in California (22 chars)
 - You don't need VARCHAR(MAX)
 - The true name of Bangkok, Thailand is: *Krungthepmahanakhon Amonrattanakosin Mahinthalayutthaya Mahadilokphop Noppharatchathaniburirom Udomratchaniwetmahasathan Amonphimanawatansathit Sakkathattiyawitsanukamprasit.* (176 chars)
 - Still don't need VARCHAR(MAX)



Building Better Tables

Using The Right Data Types

CHAR vs VARCHAR

- CHAR(n): Fixed-length string data, and the storage size is n bytes.
- VARCHAR(n): Variable-length string data, the storage size is the actual length of the data entered + 2 bytes.
- If you know the length of the string will always be the same, use CHAR to avoid the additional 2 bytes added to every VARCHAR record

NCHAR vs NVARCHAR

- If you have databases that support multiple languages, consider using the Unicode NCHAR or NVARCHAR data types to minimize character conversion issues
- Carefully evaluate whether you really need NCHAR or NVARCHAR
- NCHAR(n): Fixed-length Unicode string data, and the storage size is two times n bytes
- NVARCHAR(n): Variable-length Unicode string data, and the storage size, in bytes, is two times the actual length of data entered + 2 bytes



Building Better Tables

Using The Right Data Types

Demo



Building Better Tables

Using The Right Data Types

Numeric Data Types

Data Type	Range	Storage
BIGINT	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (Quintillion)	8 Bytes
INT	-2,147,483,648 to 2,147,483,647 (Billion)	4 Bytes
SMALLINT	-32,768 to 32,767	2 Bytes
TINYINT	0 to 255	1 Byte

- Choose the appropriate Data Type for the range of numbers you will be storing



Building Better Tables

Using The Right Data Types

Date and Time Data Types

Data Type	Range	Storage
TIME	00:00:00.0000000 through 23:59:59.9999999	3 - 5 Bytes
DATE	0001-01-01 through 9999-12-31	3 Bytes
SMALLDATETIME	1900-01-01 through 2079-06-06	4 Bytes
DATETIME	1753-01-01 through 9999-12-31	8 Bytes

- Choose the appropriate Data Type for the range of dates you will be storing.



Building Better Tables

Using The Right Data Types

Why does this matter?

Performance

- Smaller data sets = faster queries
- Optimized data pages = optimized resource usage (Remember the Buffer Pool?)
 - Saving 32 bytes in 1 table saved 30.5GB when the table reached 1 Billion rows*

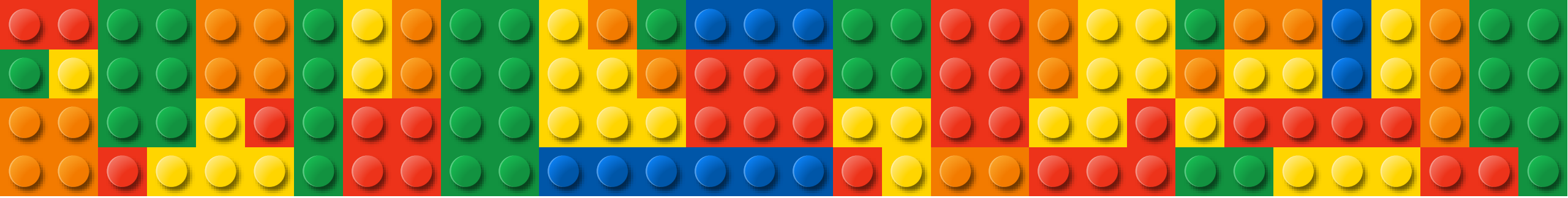
** Taken from Kimberly Tripp's Pluralsight Course: [SQL Server: Why Physical Database Design Matters](#)*

Scalability

- Helps you build better, more scalable applications
- Scalable applications do not happen by accident

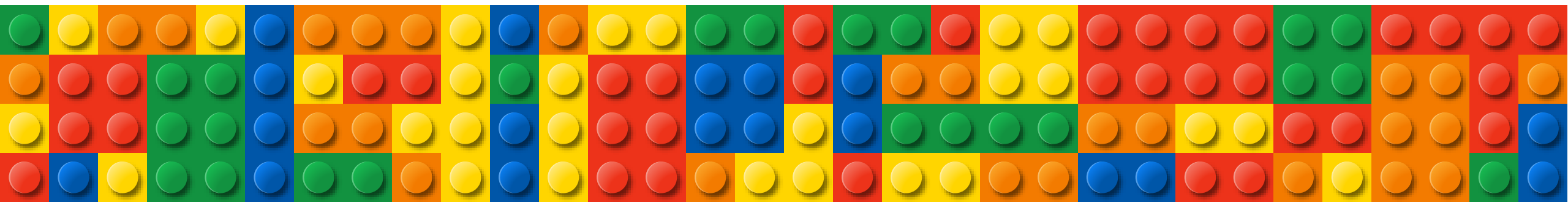
Time spent on proper database design is well worth it

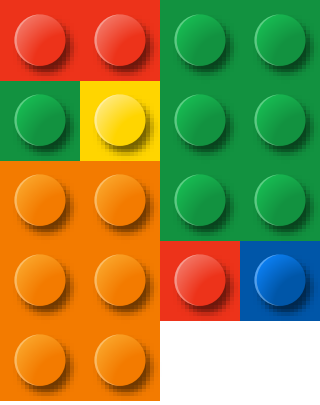
- Minor changes can have a major impact
 - It takes more effort to rebuild an application than it does to originally design one correctly
 - When testing, don't think in terms of 1 row of data, think about millions



Understanding Indexes

A LOOK AT MORE EFFICIENT DATA RETRIEVAL





Understanding Indexes

What Are Indexes?

Speed retrieval of data from a table

- Improves performance of SELECT statements
- Also used in UPDATE and DELETE statements

Without an index, SQL Server has to check every row in the table

- Known as a table scan; should be avoided

Proper indexing is one of the best performance enhancements you can make to your database



Understanding Indexes

Types Of Indexes

- Clustered
 - Nonclustered
 - Hash
 - Spatial
 - Columnstore
 - XML
 - Full-text
 - Computed column
- Generic "Index" →



Understanding Indexes

Nonclustered Indexes

- Used for supporting queries
- Copies the values from the specified columns
- Points to the actual data rows (via Clustered Index or Heap Row ID)
- Can have multiple Nonclustered Indexes on a table
 - SQL Server 2005 supports up to 249 per table
 - SQL Server 2008+ supports up to 999 per table
- Foreign Key \neq Index
 - SQL Server does NOT automatically create indexes on foreign key columns
 - Indexing foreign keys can provide performance benefits



Understanding Indexes

Nonclustered Indexing Tips

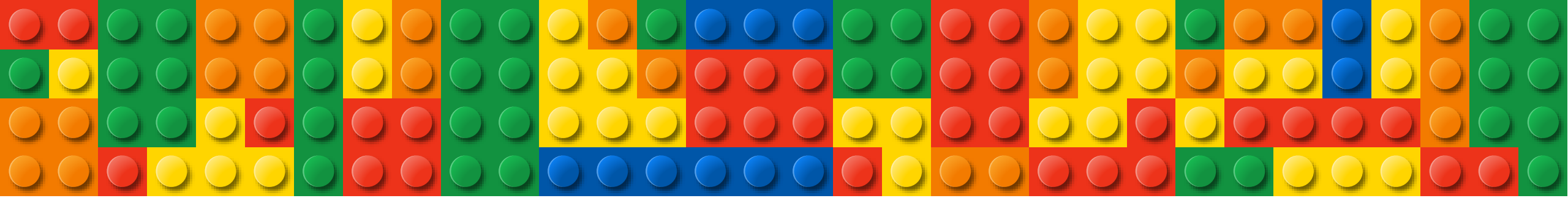
- Create on columns used in your WHERE and JOIN conditions, and columns referenced by IN predicates
- A column with few unique values is seldom a good candidate to be indexed
- Create across multiple columns; Nonclustered indexes on single columns are rarely useful
- Nonclustered Indexes can be filtered for smaller, more targeted result sets
- It is better to have fewer indexes that can serve many queries than it is to have indexes created specifically for each query



Understanding Indexes

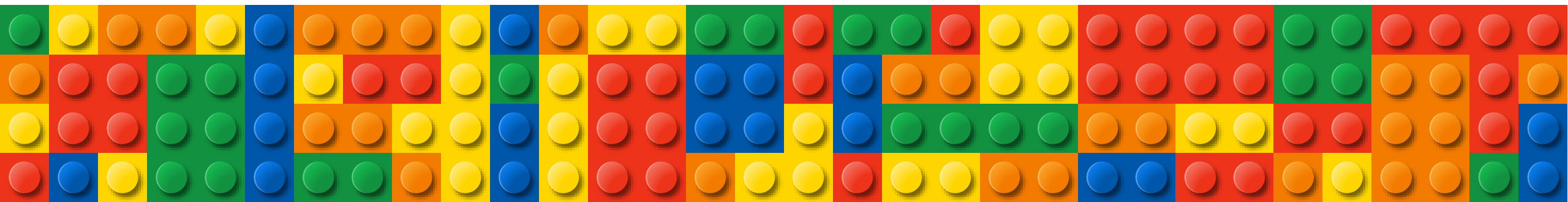
Nonclustered Indexing Tips

- Indexes can both help and hinder performance
- Indexes are written to disk, every index will take up space in your database
 - Clustered Index is always included with Nonclustered Indexes
- Indexes are automatically updated when Inserts, Updates, and Deletes are performed on the table
 - More indexes = more processing
- Avoid over-indexing heavily updated tables
- Remove any duplicate or unused indexes



T-SQL TIPS

A LOOK AT SOME T-SQL HABITS THAT CAN HURT QUERY PERFORMANCE





T-SQL Tips

Why Is My Query Slow?

- Using ORDER BY, GROUP BY, or DISTINCT
 - Could be forcing SQL Server to write your results to TempDB
 - Try to sort/filter the data in your application instead
- Using Scalar Functions in SELECT statements, WHERE clauses, or JOINS
 - Forces row-by-row operations; Forces single-threaded execution plan
- Cursors and Loops in your T-SQL statements
 - Forces row-by-row operations
- Data Type Mismatches (aka Implicit Conversions)
 - Variables and table columns used in WHERE clauses should have matching data types
 - Columns used in JOIN conditions should have matching data types



T-SQL Tips

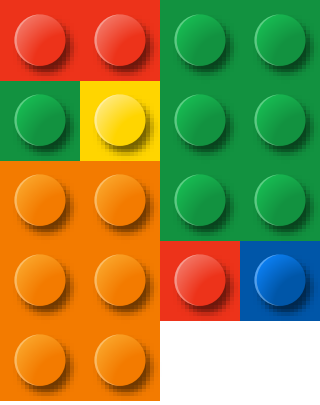
NOLOCK

Allows a Dirty Read

- Does not issue locks to prevent other transactions from modifying data being read
- Allows other transactions to modify the data while you're trying to read it
- Data returned to the SELECT statement may or may not actually exist in the database, and in some cases it may cause a query to return the same row multiple times or even skip rows

When should I use NOLOCK?

- If your query doesn't necessarily need to return precise figures, and can tolerate some inconsistencies
- If you are querying data that does not get modified often



T-SQL Tips

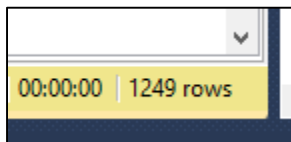
Stored Procedures

Do not name your stored procedures with the “sp_” prefix!

- This is reserved for system stored procedures
- SQL Server first checks the Master database for these procedures

Use SET NOCOUNT

- Can improve stored procedure performance
- Turns off the messages that SQL Server sends back to the client after each T-SQL statement is executed



```
1 CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
2
3 AS
4 BEGIN
5     -- SET NOCOUNT ON added to prevent extra result sets from
6     -- interfering with SELECT statements.
7     SET NOCOUNT ON;
8
9
10 END
11 GO
12
```



Questions?





Thank You!

Eric Cobb

<http://www.sqlnuggets.com>