Seven Awesome SQL Server Features

That You Can Use for Free

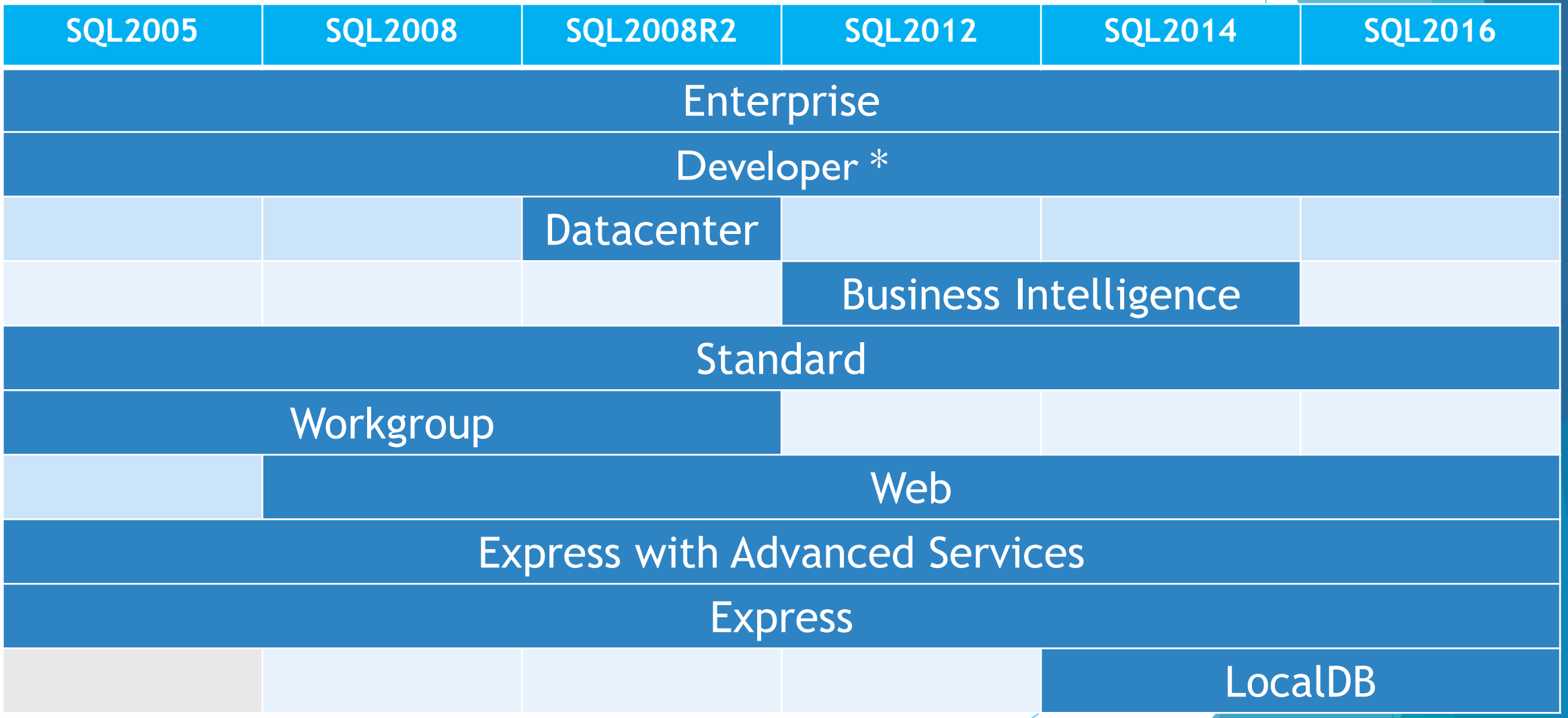
Allison Benneth

Allison@sqltran.org

@SQLTran

www.sqltran.org

SQL Server Editions



* Free starting with SQL Server 2016

2016 Was a Game-Changer!

- A lot changed in 2016
  - March 7 – Microsoft announces SQL Server will be available on Linux in mid-2017
  - June 1 – SQL Server 2016 is released
  - November 16 – Service Pack 1 is released
    - Many formerly Enterprise Edition features are moved into lower SKUs
    - Including Express Edition and LocalDB!
  - Differentiation by scale, not by feature

New Features in Express Edition (2016)

- SQL 2016 RTM
  - Stretch DB
  - Query Store
  - JSON support
  - **Temporal tables**
  - T-SQL additions
    - DROP IF EXISTS
    - AT TIME ZONE
    - SESSION_CONTEXT
    - STRING_SPLIT
- SQL 2016 Service Pack 1
  - **In-Memory Tables**
  - **Columnstore**
  - **Snapshots**
  - **Partitioning**
  - Data compression
  - **Row-level security**
  - **Always Encrypted**
  - Dynamic data masking
  - Auditing
  - Polybase (compute node)
  - Additional FILESTREAM support
  - DBCC CLONEDATABASE
- Management Studio – now a separate install … and free to use

All of these features, of course, are in more advanced editions as your application grows!

Limitations on Express Edition

- Performance
  - One CPU / four cores – per instance
  - 1.4 GB RAM (buffer pool) – per instance
  - 350 MB for in-memory tables – per instance, not counted toward buffer pool limit – single-threaded only
  - 350 MB for columnstore data – per database, not counted toward buffer pool limit – single-threaded only
- Functionality
  - 10 GB per database
  - No SQL Agent (service installed, but cannot be started)
    - Schedule backups and other jobs via another SQL Agent or OS scheduler (sqlcmd or PowerShell)

Limitations on Express Edition

- Overcomeable Limitations
  - No TCP/IP by default; be sure to enable it
- Feature Limitations
  - Availability Groups
  - Mirroring
  - Polybase (head node)
  - No SSIS, SSAS, R Server, etc.
  - SSRS with “Express with Advanced Services”
- Beware! Mandatory telemetry

Cumulative Updates

- Bug fixes specific to a SQL Server version and service pack
- Typically issued by Microsoft about every two months
- Are “cumulative,” so only need the most recent update
- Since SP1 contained new functionality, particularly import to apply
- Recent CUs go through more rigorous testing; MS recommends applying them by default
- Current CU for SQL Server 2016 SP1 is CU2 (March 22, 2017)

SQL Server 2017

- Until April 19, simply referred to as vNext
- Current on CTP 2.0 (6th preview version)
  - (SQL Server 2016 had 10 preview versions)
- No release date announced as of yet
- No edition announcements as of yet
- New features: availability on Linux, Python integration, adaptive query plans, graph databases

SQL Server Features (Speed Dating)

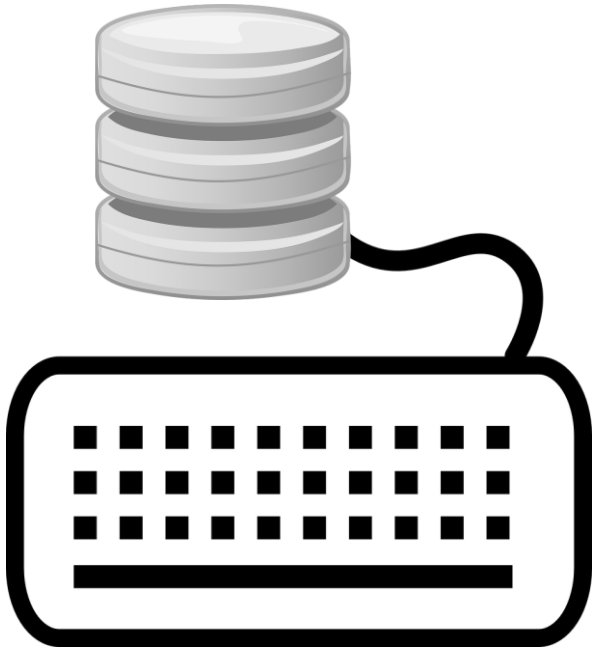
- Security
  - Row-Level Security
  - Always Encrypted
- Utility
  - Snapshots
  - Temporal Tables
- Performance
  - Columnstore Indexes
  - Partitioning
  - In-Memory OLTP (Hekaton)

Row-Level Security

- Powerful and flexible way to control who can view or modify data at the row-level grain
- Access is controlled by a user-defined function that is applied to the table's security
- Non-qualifying rows are silently blocked
  - Select predicate – controls read access to the row
  - Block predicate – controls modification to the row (either before or after the modification)

Row-Level Security

DEMO



SQL Server Features (Speed Dating)

- Security
  - Row-Level Security
  - Always Encrypted
- Utility
  - Snapshots
  - Temporal Tables
- Performance
  - Columnstore Indexes
  - Partitioning
  - In-Memory OLTP (Hekaton)

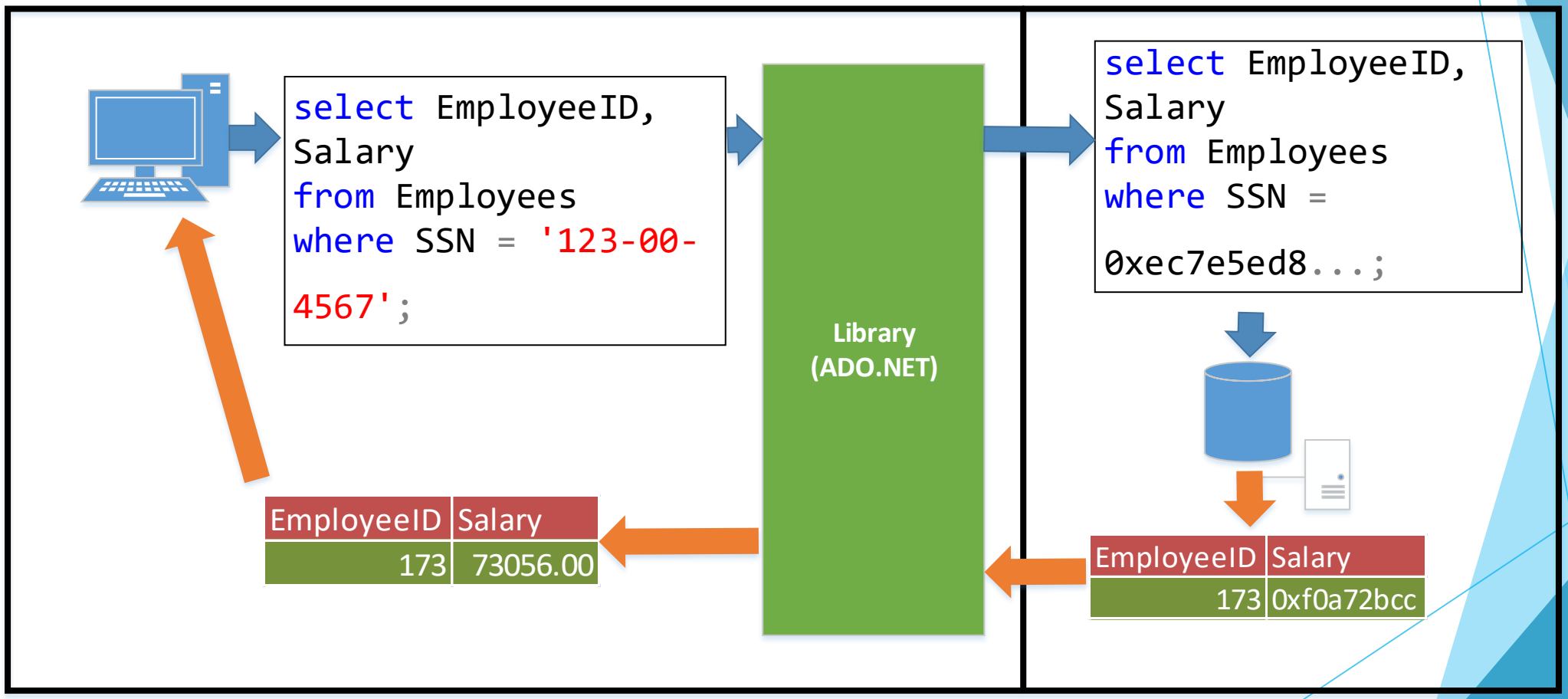
Always Encrypted

- Applies at the column level
- SQL Server box never sees data in unencrypted form (both at-rest and in-transit)
  - Encrypted columns are stored (and transmitted) as varbinary behind the scenes
- Certificate is generated on client machine and shared with other clients

Always Encrypted

- Encryption can be random or deterministic (required if column is indexed or used in a join)
- Requires a change to the connection string in the application
  - `Column Encryption Setting=enabled`
- Queries must be parameterized

Always Encrypted in Action



Always Encrypted - Cons

- Data size bloat, especially for smaller data types
- Adds considerable difficulty troubleshooting in tools like SSMS
- String columns must have a BIN collation – they won't sort by traditional SQL rules
- Extra round trips to determine metadata, retrieve keys

SQL Server Features (Speed Dating)

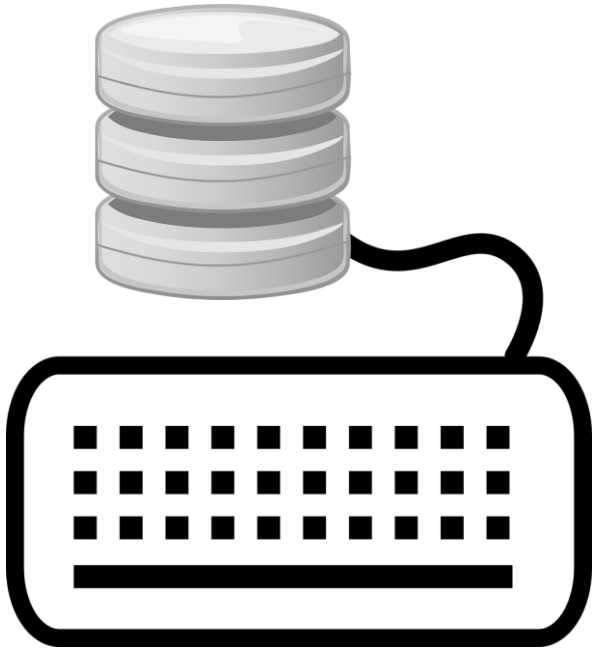
- Security
  - Row-Level Security
  - Always Encrypted
- Utility
  - Snapshots
  - Temporal Tables
- Performance
  - Columnstore Indexes
  - Partitioning
  - In-Memory OLTP (Hekaton)

Snapshots

- Provides a transactionally consistent, read-only point-in-time view of a database
- Can take multiple snapshots at different points on the same database
- Useful for stable reporting against a transactional system
- Can be used to revert to a previous database state
  - Failed upgrade / administrative tasks
  - QA cycles
- Resources required dependent mostly on how much underlying database is changed
- Absolutely, positively not a substitute for proper backups!

Snapshots

DEMO



SQL Server Features (Speed Dating)

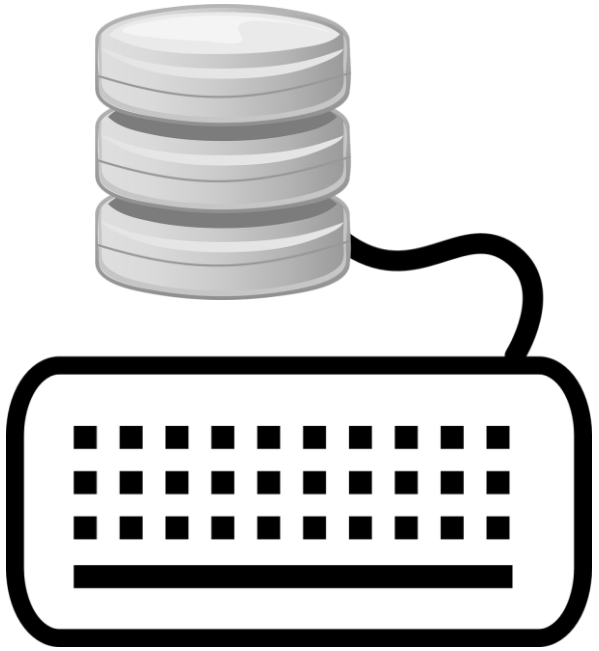
- Security
  - Row-Level Security
  - Always Encrypted
- Utility
  - Snapshots
  - Temporal Tables
- Performance
  - Columnstore Indexes
  - Partitioning
  - In-Memory OLTP (Hekaton)

Temporal Tables

- Most applications / databases inherently contain a temporal element
- If temporal components are tracked, traditionally done with triggers or change detection
- Temporal tables handle tracking automatically
- Allows greatly simplified point-in-time querying
- Requires additional columns on source table and requires history table
- Schema changes in source table are reflected in the history table

Temporal Tables

DEMO



Temporal Tables

Temporal querying: `FROM TableName FOR SYSTEM_TIME _____`	
Point in time	`AS OF '2017-02-06 11:30:00'`
Full history	`ALL`
Between (‘start’ < EndTime AND ‘end’ >= StartTime)	`BETWEEN '2017-01-11 18:55:04' AND '2017-05-06 11:30:00'`
From (‘start’ < EndTime AND ‘end’ > StartTime)	`FROM '2017-01-11 18:55:04' TO '2017-05-06 11:30:00'`
Contained in (‘start’ <= EndTime AND ‘end’ >= StartTime)	`CONTAINED IN ('2017-01-11 18:55:04', '2017-05-06 11:30:00')`

Temporal Tables

- Performance
  - Insert operations – no difference than non-temporal tables
  - Update operations – overhead due to writes to both source and history tables
  - Read operations – Default clustered index on history table usually not helpful – consider changing it

Temporal Tables

- Beware of v1 limitations!
  - Dropping a column in the source table will drop the column in the history table – all history is lost!
  - Cannot add a non-nullable column to the source table
  - Pruning history is an offline operation

SQL Server Features (Speed Dating)

- Security
  - Row-Level Security
  - Always Encrypted
- Utility
  - Snapshots
  - Temporal Tables
- Performance
  - Columnstore Indexes
  - Partitioning
  - In-Memory OLTP (Hekaton)

Columnstore Indexes

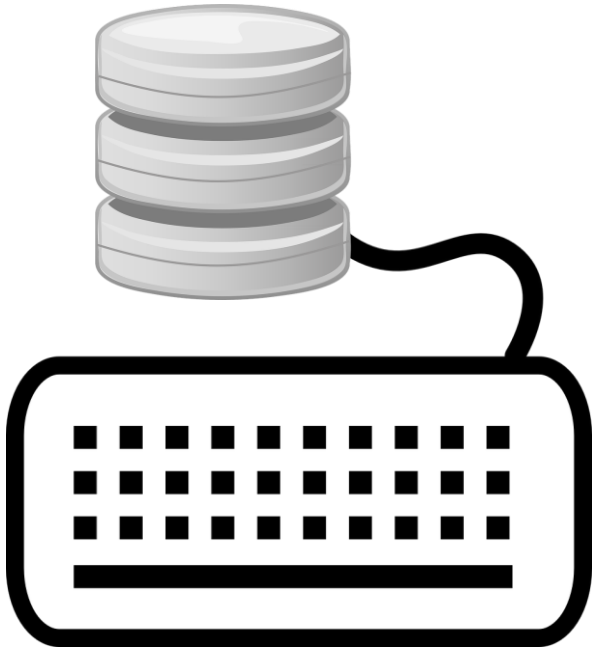
- Traditional indexes are row-based copies of selected columns in table
- Columnstore turns this around and orders the index by column
- Can be the entire table (clustered index) or a subset of columns (nonclustered index)
- Can be combined with row-based indexes

Columnstore Indexes

- Previous versions of SQL Server imposed limitations, but SQL Server 2016 removes many of these limits
- Particularly useful for warehouse / analytic queries
  - However performance usually degrades for OLTP workloads
- Much of performance benefit derives from high compression of columnstore (typically 20x or more)

Columnstore Indexes

DEMO



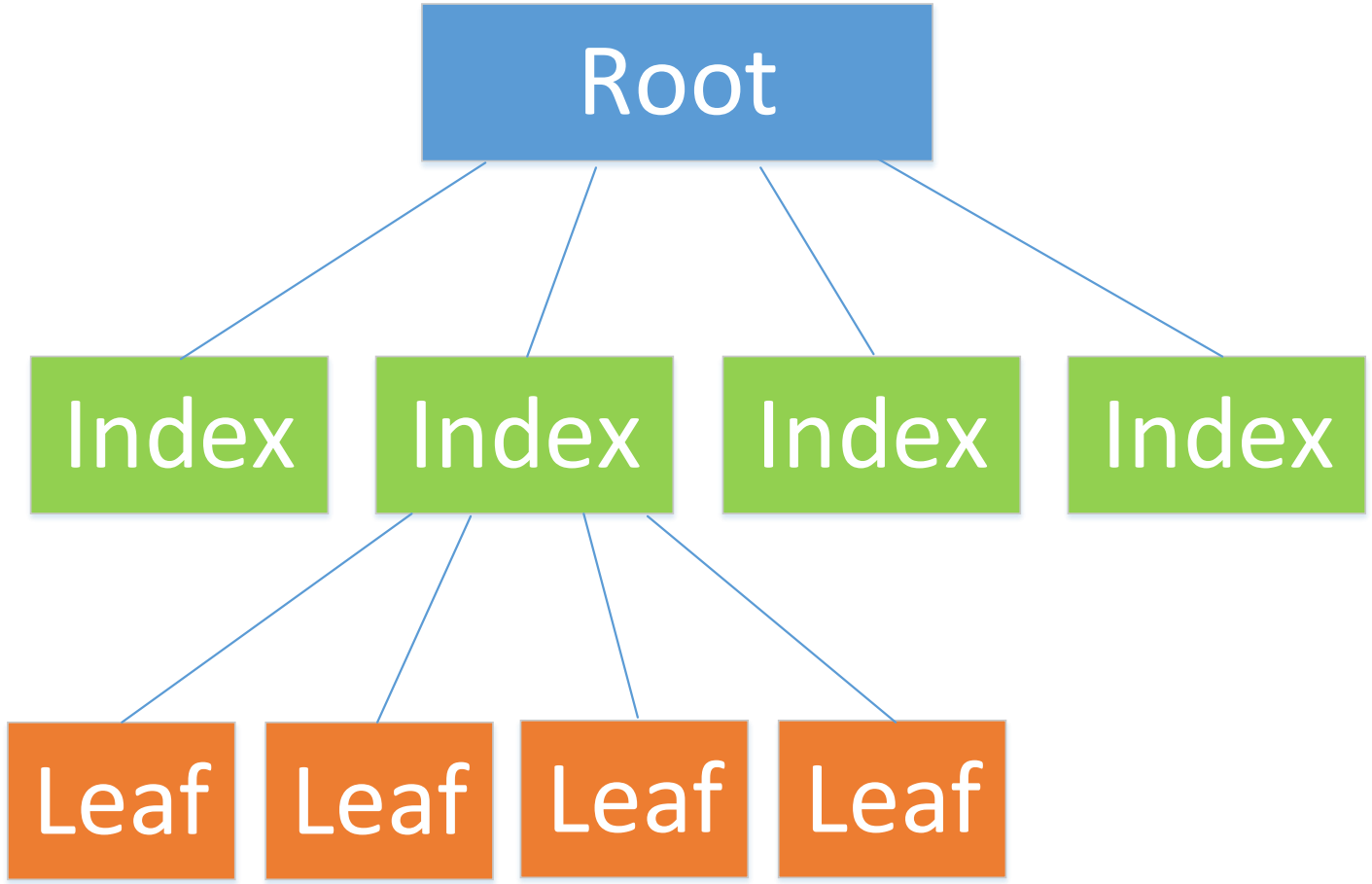
SQL Server Features (Speed Dating)

- Security
  - Row-Level Security
  - Always Encrypted
- Utility
  - Snapshots
  - Temporal Tables
- Performance
  - Columnstore Indexes
  - Partitioning
  - In-Memory OLTP (Hekaton)

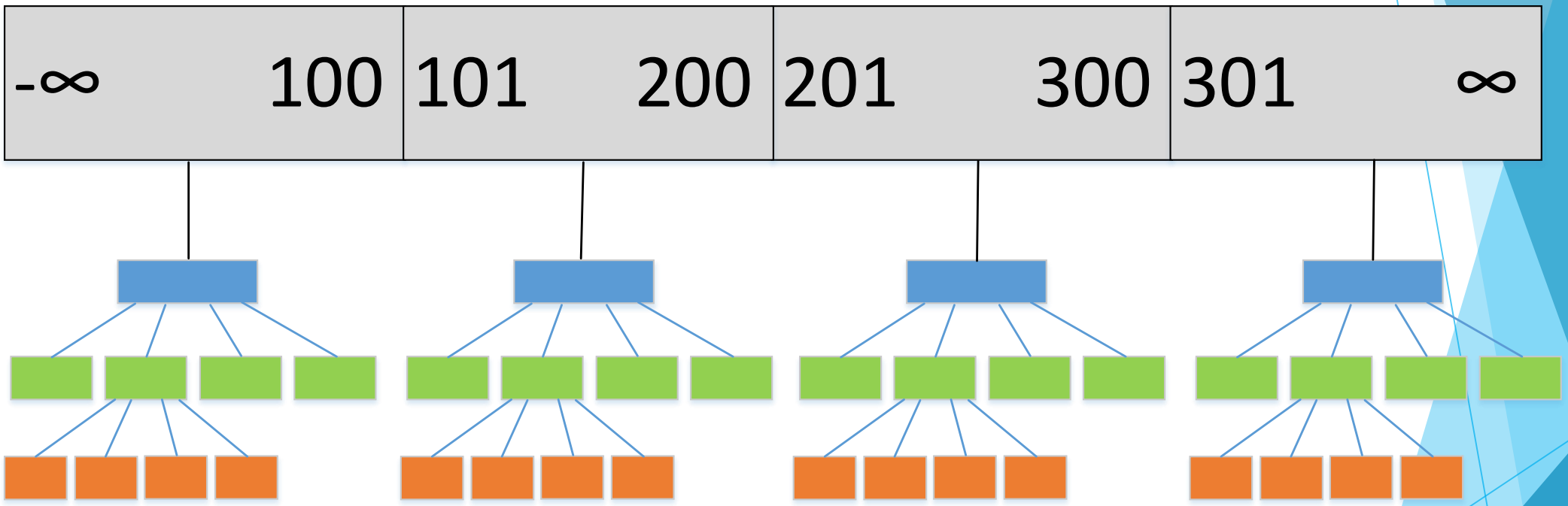
Partitioning

- Spread table data across multiple B-trees
- For example, place older data on slower, cheaper storage
- Usually for very large data sets, but has other purposes
- Separation defined by a “partitioning function” and a “partitioning scheme”
  - Range LEFT (think of as >=)
  - Range RIGHT (think of as <)
- Another use: combine with temporal tables to enable quick archival capability

Traditional SQL Server Index

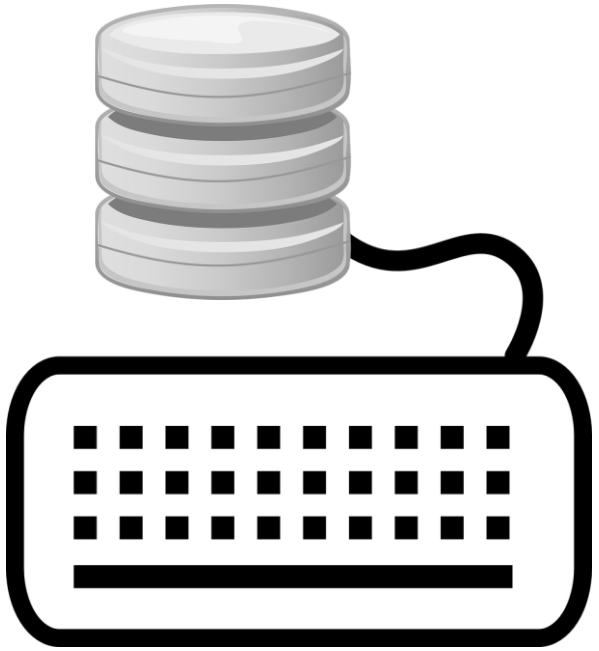


Partitioned SQL Server Index



Partitioning

DEMO



SQL Server Features (Speed Dating)

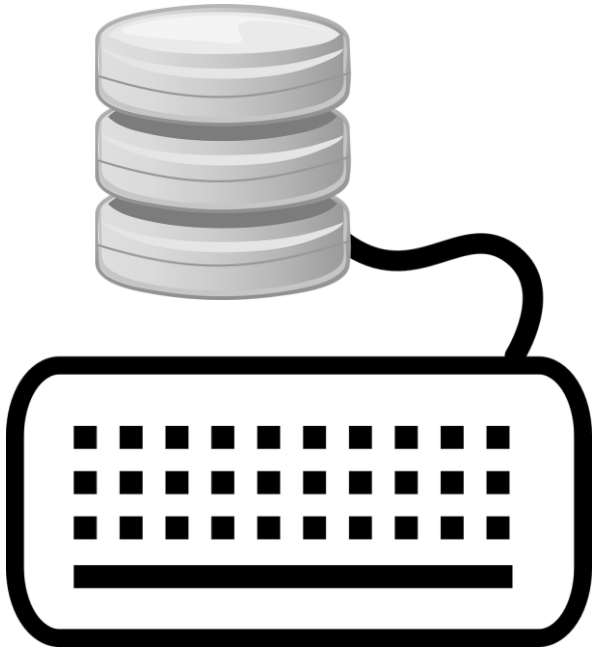
- Security
  - Row-Level Security
  - Always Encrypted
- Utility
  - Snapshots
  - Temporal Tables
- Performance
  - Columnstore Indexes
  - Partitioning
  - In-Memory OLTP (Hekaton)

In-Memory OLTP

- First introduced in SQL Server 2014
- Stores data in memory
- Lock-free structures
  - Multi-version concurrency control (optimistic)
- Fully ACID compliant (durability optional)
- Designed for OLTP workloads
- Can yield 10-20x performance boost
- Native compilation of stored procedures

In-Memory OLTP

DEMO



In-Memory OLTP

- Need to give a table hint such as **`with (snapshot)`** when used inside an explicit transaction
  - Or, set database option **`memory_optimized_elevate_to_snapshot`**
- Error handling considerations
  - Entire transaction will roll back if validation phase fails (optimistic concurrency assumptions failure)

Resources

- SQL Server 2016 Express Edition download
  www.microsoft.com/en-us/sql-server/sql-server-editions-express

- Companion blog page to this session
  www.sqltran.org/7features

Allison Benneth

Allison@sqltran.org

@SQLTran

www.sqltran.org