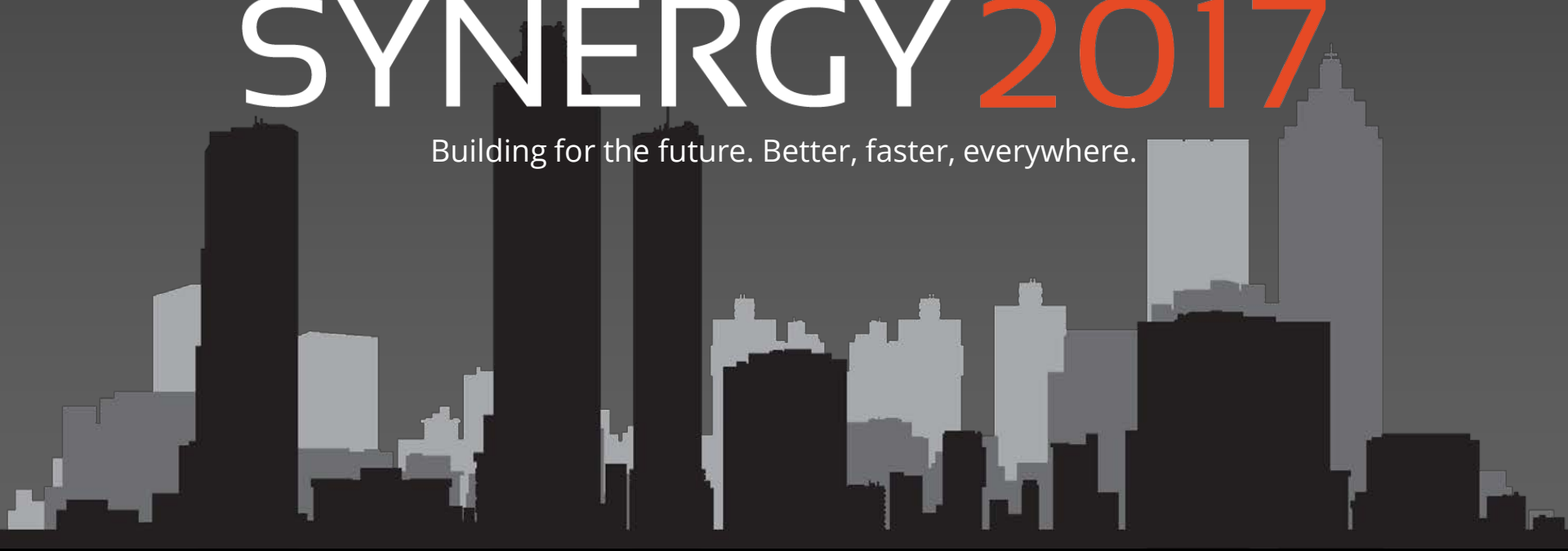
6 – 7 . APRIL ATLANTA, GA, USA

SYNERGY 2017

Building for the future. Better, faster, everywhere.



Let Me Help You Find That; Taking Advantage of SQL Index Support in DataFlex 2017

Stephen W. Meeley

VP, Development – Data Access Worldwide

Interlude: More than “just” Managed Connections

Since we’re here, we might as well take advantage of all the amenities...

What we did...

- DataFlex 18.2 Web application using existing SQL databases (could have been any prior revision)
- Migrated to 19.0 and used Dashboard to easily set to new web application name and virtual directory (on a specific web site)
  - Automatically added table for server properties – at this point we could have started using server properties
- Added code for managed connections (backward compatible)
- Added a managed connection
- Converted server property table using the managed connection – this would be the same step for converting all (or some) of the tables
- “Repaired” the existing SQL connections to used the managed connection
- Added an additional managed connection for data located in another database (could have even been another server) – “Repaired” those connections as well
- Added an alternate managed connection for accessing data through credentialed user
- Created a login program to enable web access via encrypted password

What didn’t we do... Optimizing for SQL

- First, determine what your universe is for any particular application
  - Supporting multiple backends (especially continuing to support embedded) may reduce the level of change you want to make
- Hit the low-hanging fruit
  - **Identity columns to replace auto-increment columns**
  - Add client-only indexes (or even server indexes) that you may have been hesitant to add for fear of slowing down the system
  - Use Runtime indexes
- SQL Filters in Data Dictionaries to move constraints to the server for faster performance (and reduce traffic “over the wire”)
- Take in the View
  - Though many aspects of living in the SQL world are life-changing, SQL views make powerful connections between the two worlds

SQL Views

- Join data from different tables / databases
- Server side execution
- Employ complex filters
- Limit the number of columns returned
- Table structure independence
- Simplify the data structure, especially for reports
  - Team members can do more without needing to be experts in complex, many-layered data structures

Indexes

And now back to our regularly scheduled program...

What SQL Has vs. What DataFlex Needs

- SQL
  - Primary and Foreign Keys
  - Unique indexes? We don’t need no stinkin’ unique indexes!
  - We don’t need no stinkin’ indexes at all...
- DataFlex
  - Relationships (hard or soft)
  - Unique indexes
  - Indexes for everything you want to find or sort by

- Notice a pattern? So much revolves around indexes (in other words, finding stuff)

Indexes

- Unlike the embedded database, SQL can (and will) order data with as much (or as little) of an index as exists (even none)
- DataFlex 19.0 provides support for server, client-only and server-only indexes
    - Server = exits on server, defined in .int file, requires restructure
    - Client Only = defined in .int only, requires .int update
    - Server Only = exists on server, not defined in .int, changing them would switch to server and require restructure
- Easy to manage indexes in the Studio (let's go play...)
- Provides the basis for runtime temporary indexes

Runtime Temporary Indexes

- Index creation is supported through database API and required restructure on server
- Client-side indexes removed the need for a restructure (.int only)
- We used the same capability to support indexes that are not stored in the .int file and only exist at runtime
- Uses the same interface, but not within a structure_start / structure_end

  - Create_Index
  - Set_Attribute DF_INDEX_NUMBER_SEGMENTS
  - Set_Attribute DF_INDEX_SEGMENT_FIELD
  - Set_Attribute DF_INDEX_SEGMENT_DIRECTION
  - Delete_Index

How to Create a Runtime Index

```
// create a temporary index for State x City x Customer and return its Index handle
Function StateCityOrdering Returns Integer

    Handle hTable
    Integer iIndex
    Move Customer.File_Number to htable

    Create_Index hTable at iIndex

    Set_Attribute DF_INDEX_NUMBER_SEGMENTS of hTable iIndex to 3
    Set_Attribute DF_INDEX_SEGMENT_FIELD of hTable iIndex 1 to (RefTable(Customer.State))
    Set_Attribute DF_INDEX_SEGMENT_FIELD of hTable iIndex 2 to (RefTable(Customer.City))
    Set_Attribute DF_INDEX_SEGMENT_FIELD of hTable iIndex 3 to (RefTable(Customer.Customer_Number))

  Function_Return iIndex

End_Procedure
```

How to Use a Runtime Index

```
Procedure StartReport

    Integer iOrder

    // use the temporary index
    Get StateCityOrdering to iOrder

    Set Ordering of oReport to iOrder

    Send Run_Report of oReport

    // delete the temporary index
    Delete_Index Customer.File_Number iOrder

End
```

The “even none” index reference was a just joke, right?

- Au contraire...
  - Converted Order Entry Tab example to SQL (OrderEntry182 database)
  - Duplicated the database (OrderEntryNOIndex database) using a script
  - Using SSMS, deleted every index – there are none left, not even for Primary Keys
  - Edited the Access Miles customer name so we could tell which database we are attached to
- Create two managed connections (Order)
- Use the new DF_DATABASE_DEFAULT_DATABASE attribute to switch databases


Thank You for Your Time