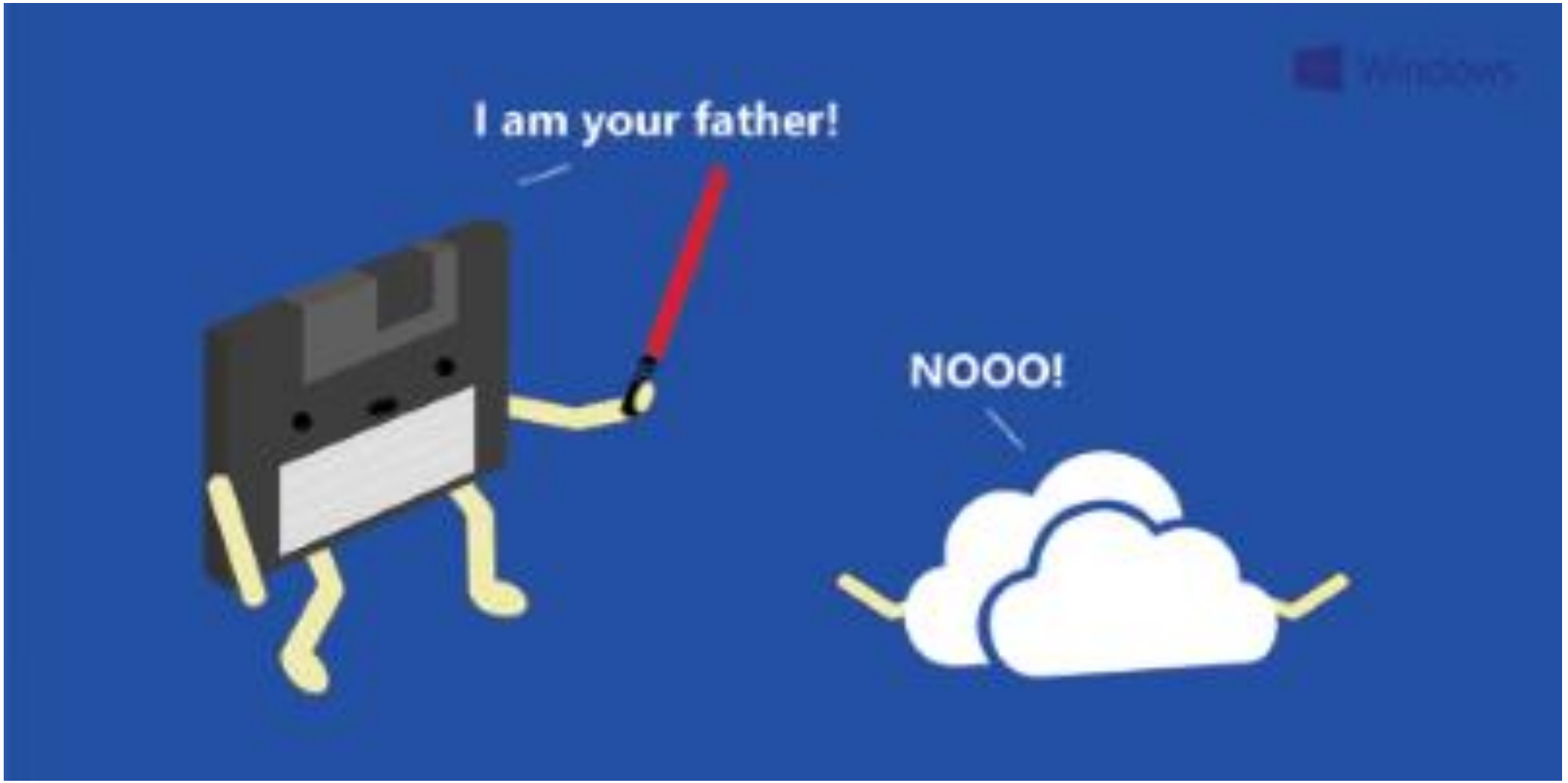
Denver Area **Access** Users Group

Our Sponsors

Real-Life Access and SQL Azure

LIVE DEMO AND LESSONS LEARNED

Jim Pilcher
George Young


I am your father!
NOOO!

Agenda

An Overview of the Access application

Why Move to a SQL Azure Backend – Client Objectives and Requirements

The Migration Process

A Demo of the Integrated Application – Access, SQL, Web and Power BI

Optimizing the Client Application

Lessons Learned

An Overview of the Access Application…

Database Size and Usage

- 94 tables
- 907 fields
- 438 queries
- 93 forms
- 48 reports
- 41 users, most in the database once or twice a week for 1-2 hours

Main Features

- Civil Engineering project tracking
- Project cost estimating
- Contractor bid notification and tracking
- Gantt Chart timelines and workload visuals
- Extensive complex reporting for upper management and external stakeholders

An Overview of the Access Application

Architecture is traditional Microsoft Access

- Database is split, front end and back end
- Each user has own copy of the front end residing on the local PC
- Each front end has an associated “local” database for temp tables
- Back end resides on network server

- Demo of Access-only application

Why Move to a SQL Azure Backend – Client Objectives and Requirements

So, the client has a working Access database application. Why do they want to move to a cloud-based database?

- Simpler user interface for non-admin users
- Access to database in the field
- Access from non-PC devices, e.g. iPad and Android tablets
- Access from remote PCs, e.g. from home
- Mobile support

The Migration Process

Generate an initial export to SQL Server of the Access database

- SQL Server Migration Assistant for Microsoft Access

Create Azure SQL Database account for database and website

Develop initial web prototype

- Develop locally against local SQL Server database
- Periodically deploy to Azure

Copy the Access client and connect it to the SQL Azure database

As work progresses optimizing the SQL database against the Access client, deploy the updated database to Azure, and update the website as needed

A Demo of the Integrated Application

Access + SQL Azure

Web + SQL Azure

Power BI + SQL Azure

Optimizing the Client Application…

Always attempt to minimize trips to the database

Microsoft Access has lots of hidden database calls built-in

- Bound forms and subforms
- Data-hungry controls
  - Combo boxes
  - List boxes
- Nested queries, including derived tables, subqueries, and unions

Un-optimized cloud-connected performance *will* slow to a crawl

Optimizing the Client Application…

Consider converting to SQL views all queries that include more than one table

- Test each query when in doubt
- What about a hybrid join of an Azure table/view with a local table?

Run the application with the VBA debugger looking for performance issues

Use temporary pass-through queries while in VBA *(more on this later)*

Use ADO to invoke complex data procedures with many parameters

Optimizing the Client Application…

Cache scalar values that trigger database hits

- Use the TempVars collection for best reliability and flexibility
- Use global VBA variables cautiously

Cache static lookup tables in a local ACCDB

- At startup
- or upon first access

Avoid domain aggregate VBA functions (DSum, DLookup, etc)

Consider unbound forms

- Candidates: simple data entry forms
- Difficult to do when subforms are involved

Optimizing the Client Application…

Convert your custom VBA functions called in queries to SQL UDFs (User-Defined Functions)

- You have to become good at T-SQL
- Do this before converting your Access queries to SQL views
- Not all VBA intrinsic functions have a direct counterpart in T-SQL
  - Date() = GetDate()
  - IIF() is available in the most recent versions of SQL Server
  - Otherwise, IIF() = use the Case construct
  - Switch() and Choose() = use Case construct
  - Data type conversions = Cast() or Convert()

Optimizing the Client Application

Convert action queries

- Temporary pass-through queries for simple one-table actions
  - Build the add, update, delete SQL in VBA code
  - Must use SQL Server T-SQL syntax, not Access SQL
- Permanent pass-through queries
  - Stores the connection string
  - Just replace the SQL when you need to run it
- Use SQL stored procedures
  - Complex multi-join action queries
  - Complex multi-table updates that your VBA does with multiple action queries
  - Best run by setting up ADODB command object to reference the stored proc
  - Can also be run using pass-through queries. Just specify stored procedure name and provide parameters

Lessons Learned

Two environment requirements:

- Microsoft ODBC Driver for SQL Server (current version is 13.1)
- Port 1433 must be open on network to access database

Internet latency plays an important role

It’s the number of database calls on a form that will kill performance. Reduce toward one call per form, and cache what’s not changing.

Turn table joins into SQL Views

Some default Access features are not good in a remote SQL environment