Multi-Tenancy in DataFlex Web Applications

Presenter: Vincent Oorsprong

SYNERGY 2019

CRUISING TO NEW HORIZONS

Agenda

- What is it
- Techniques
- Watch out for
- Licenses

What is it

- One application serving multiple sets of data
  - For one customer
  - For more customers
- All users (tenants) use the same software but see different sets of data

Techniques

Techniques

- Shared database
- Separate databases

Techniques

Shared database

Shared database

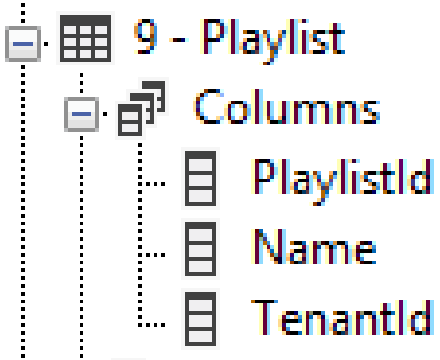
- Per entity (Customers, Orders etc)
  - One table for all tenants
    - E.g. orders with a tenant ID column
  - Table per tenant
    - Schema based

Shared database

Single table per entity

Shared database - table

- Tenant ID column in each table that contain different values per tenant



Shared database - table

- Don’t forget to store the tenant ID
  - Use DD Creating event
- Make use of
  - Global SQL filters
  - And/or Constraints

Shared database – table – SQL filters

- Needs to be set for each table that contains the tenant ID column
- Attributes to set (for each table!)
  - DF_FILE_SQL_FILTER
  - DF_FILE_SQL_FILTER_ACTIVE
  - DF_FILE_SQL_FILTER_EQ

Shared database – table - Constraints

- Advantages
  - You can see the filter directly in the DD class
  - Guarantee you cannot see records out of scope
- Watch out for inherit constraints
  - It might give no results while it should

Shared database – Table - Constraints

- Convert your constraint information to SQL filters automatically
  - Augmented ApplySQLFilter
  - Retrieved DDOConstraints
    - Enumerated through to setup psSQLFilter DDO property

Shared database

- Pro
  - Easier to update table structure after restructure
    - Not true when using SQL Schema to separate the tenant context sensitive data
  - Might perform better

Shared database

- Cons
  - Size of the database
  - Backup and restore of data to/from a backup is more difficult
  - Auto increment ID means “nothing”
  - Danger of seeing / accessing data from another tenant

DataFlex Reports

Filtering data

DataFlex Reports

- Connection string
- Filters

- Both not needed when using RDS

DataFlex Reports – Connection string

- Needs to be set / build when the reports on SQL are deployed on a different machine
  - Isn’t that always true?

Connection string

```
Procedure ChangeODBCDataSource String sReportId
    tConnection ConnectionData
    Handle hoCLIHandler
    Integer iDriver iClientVersion
    String sDriver sDatabaseConnection

    Get ConnectionIdInfo of ghoConnection "ChinookMT" to ConnectionData
    Get Create (RefClass (cMSSQLHandler)) to hoCLIHandler
    Get DriverIndex of hoCLIHandler ConnectionData.sDriver to iDriver
    Get_Attribute DF_DRIVER_SQLSERVER_CLIENT_VERSION of iDriver to ;
        iClientVersion
    Get SqlServerClientDriverName of hoCLIHandler iClientVersion ;
        to sDriver
    Move ("DRIVER=" + sDriver + ";" + ConnectionData.sConnectionString) ;
        to sDatabaseConnection
    Set psDatabaseConnection sReportId to sDatabaseConnection
    Send Destroy of hoCLIHandler
End_Procedure
```

Setting Filter

- Since the database contains the data from multiple tenants don’t forget to set the filter for the tenantID column

```
Procedure SetFilters
    String sReportId

    Get psReportId to sReportId
    Set psFilterValue sReportId 0 to giCurrentTenant
End_Procedure
```

Techniques

Separate databases

Separate databases

- On one SQL server
- On multiple SQL servers
  - Increases scalability

Separate databases

- Use DF_DATABASE_DEFAULT_DATABASE
  - Compare with using the USE statement in an SQL query
  - Can only switch between databases at the same database server
  - Credentials and Schema must be the same
  - Table structure must be identical

Separate databases

- Use RedirectConnectionId
  - Method of the cConnection class
  - For better performance keep the connection to the previous used database open
  - Need to be used when using multiple SQL servers
    - DF_DATABASE_DEFAULT_DATABASE won't support multiple SQL servers

Separate databases

- Pro
  - Physical separation of the data
  - Simple backup & restore per tenant
  - Simple import & export of data
  - Better scalability

Separate databases

- Cons
  - Licensing costs could be higher
  - Database structure updates more complicated
    - You need to apply changes to all databases and servers
    - Longer downtime

DataFlex Reports

Using multiple database servers

Change connection string

- Integration wizard can write stub to pickup current connection string and you can alter it to get the report using the right data

- No need for this when using RDS (of course)

Change connection string

- On top of the code shown with a single database you need to set the psDatabaseName property as this changes

Windows locale

- As explained yesterday in the DataFlex Reports update the locale that can be set from v7.0 may be very welcome when switching tenants
  - A tenant could be a different (regional) office in a large company and different products / prices may apply

Watch out for

Watch out for

- User account
- Process pooling
- Use of debugger

User account

User account

- The user account used by WebApp may not be able to access the SQL server
  - Especially watch out for trusted connection

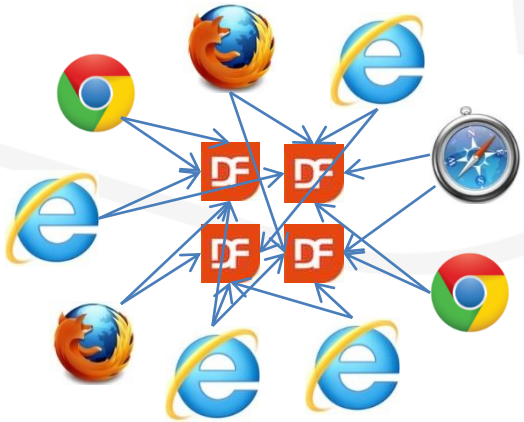
Process pooling

What is process pooling

- Invented in early 2000's
- Pool of processes waiting to handle traffic
- Started when webapp server starts
- Extended when busy
- HTTP requests are assigned to processes

Persistency

- There is no persistency between requests
  - Requests from a single session are handled by different processes
  - Processes handle requests from different sessions



Issues – Symptoms

- Users see data from other sessions
- Users report data gets lost
  - While they were not deleting it!

Issues - Problems

- Storing data in memory
  - And assuming it to still be there the next request
    - E.g. RegisterDownloadFolder on one process only
  - Properties (unless they are web properties)
    - E.g. Setting Read_Only_State of a DDO
    - E.g. Field_Changed_Value of a DDO
  - Using global variables

Issues - Problems

- Table buffers (unless synchronized by WAF)
  - Can bring in data from a previous request
  - Users report from time to time they see / have processed data not belonging to them

Issues - Difficulties

- You usually only test with a single user
- Application behaves differently when using outside the debugger

Issues – Solutions

- Never use regular properties
  - Unless you are really, really sure that it is a single request
  - Use web properties instead
- Make sure all your DDO's are synchronized
  - Use AddDDOStructure
- Make sure all your constraints are properly built

Debugger

Debugger

- WebApp behaves different outside the debugger
- The debugger emulates a pool of one single process
- WebApp uses a different Windows account
  - Trusted connection might not work

Licenses

Licenses

- When using WebApp Server for a SaaS solution you need a different license
- SQL Server used for a web application requires a different (more expensive) license
  - Core license (own company only)
  - SPLA license otherwise
  - Not for SQLExpress

Training

DataFlex Learning Center

Video

- Multi tenancy training
  - https://learning.dataaccess.com/courses/multi-tenancy/

Thank you for listening

Make use of SQL!