ArcGIS Insights: Configuring data connections

Frances Zhu

Shirley Baffoe

Chaka Clarke

2021 ESRI DEVELOPER SUMMIT

Agenda

Configuring data connections to:

- ArcGIS content, local drives, & cloud drives
- Supported databases
- Other databases

Notes:

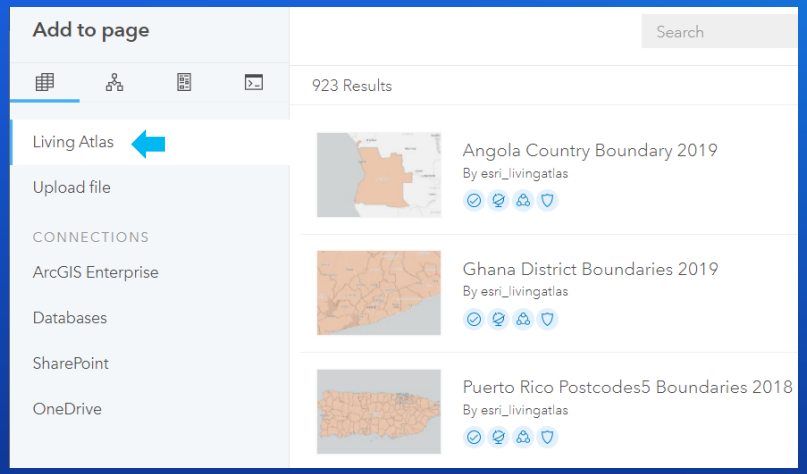
- Data from local drives and cloud drives are snapshots

Connecting to ArcGIS content, local drives, & cloud drives

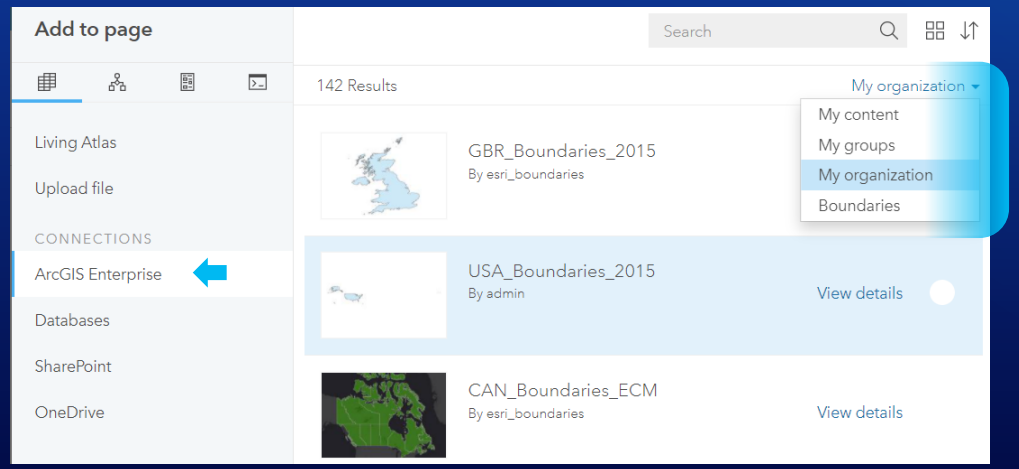
Frances Zhu

Connecting to data in ArcGIS

- **Living Atlas**
  - **Read-to-use global geographic information from Esri**

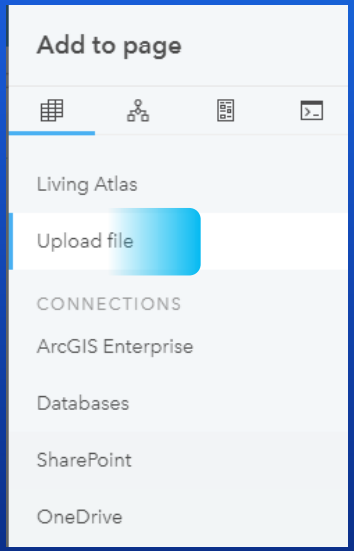


- **ArcGIS Enterprise**
  - **Data from your organization, such as feature layers, boundaries, and remote layers**

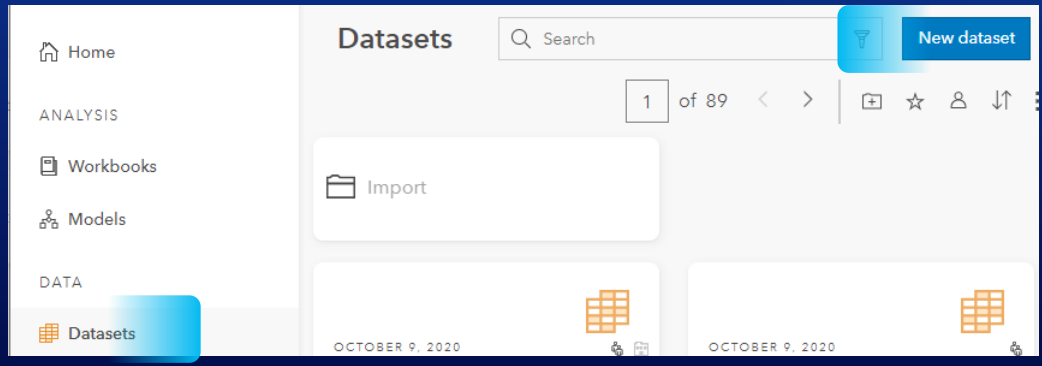


Connecting to data in local drives

- Upload files into a workbook
  - CSV
  - Excel
  - Shapefiles
  - GeoJSON



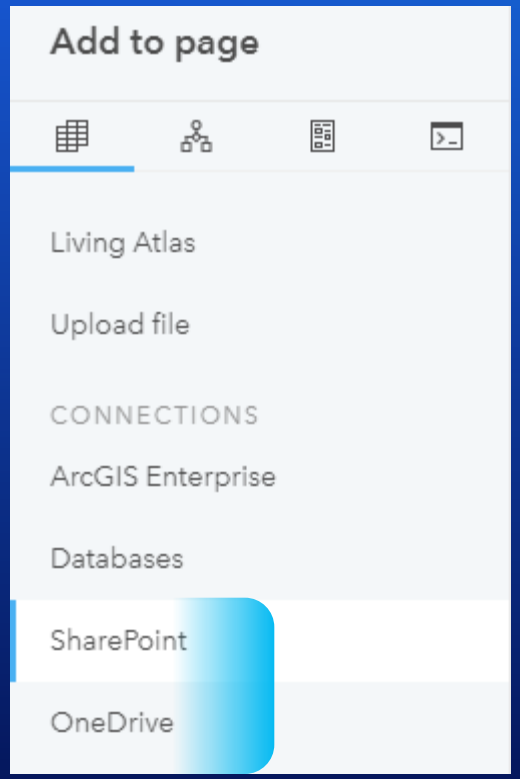
- Import files from the home page
  - CSV, Excel, shapefile, GeoJSON
  - File GDB



Connecting to data in SharePoint and OneDrive

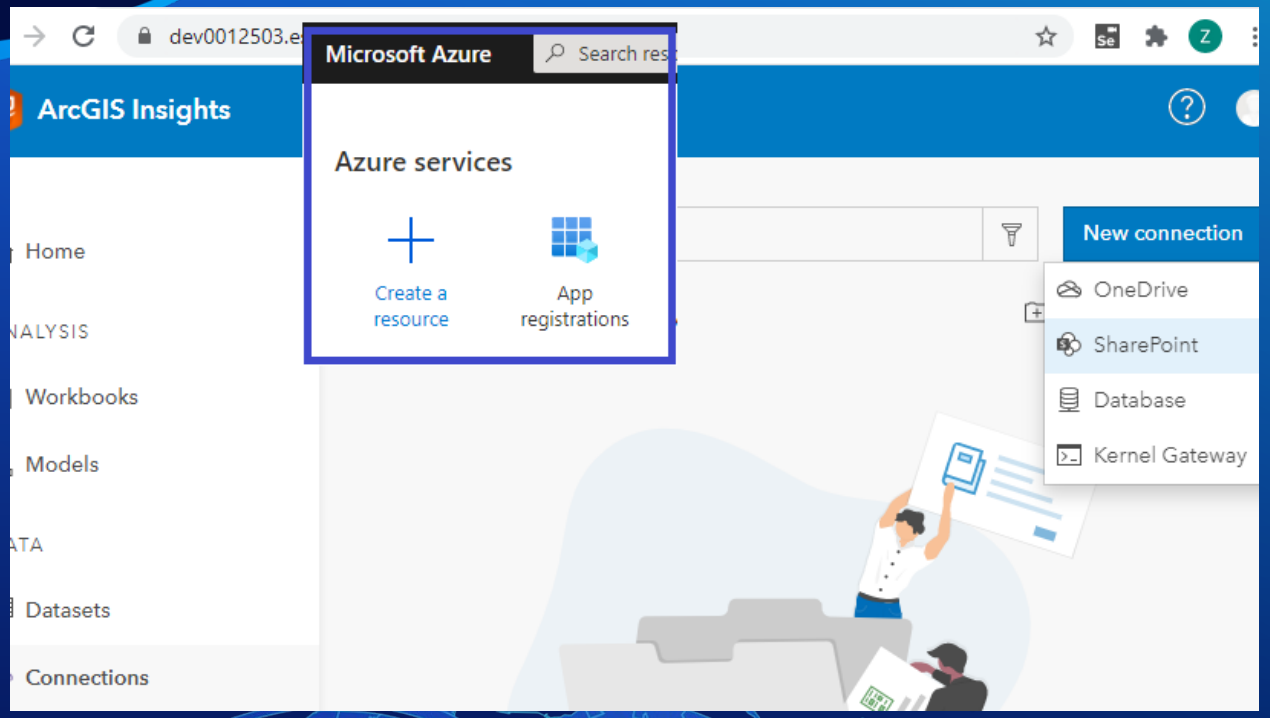
- SharePoint/OneDrive
  - CSV
  - Excel
  - Shapefiles
  - GeoJSON

- SharePoint
  - SharePoint list
  - Navigate
    - SharePoint subsite
    - SharePoint document library



Registering ArcGIS Insights with Microsoft Azure

- Key steps
  - Redirect URI
  - Delegated permissions
- Information for creating SharePoint and OneDrive connections
  - Application ID
  - Organization name
- Check the ArcGIS Insights documentation for more information
  - Create a OneDrive connection
  - Create a SharePoint connection



Registering Insights with Azure

Frances Zhu

Configuring database connections to supported databases

Shirley Baffoe

Configuring database connections to supported databases

- Insights 2021.1 supports connections to the following databases:
  - Microsoft SQL Server
  - Microsoft Azure SQL Database
  - Oracle
  - PostgreSQL
  - SAP HANA
  - Snowflake NEW
  - Google BigQuery NEW
  - Other NEW
- See the supported databases documentation for more information

NOTE: Non-Spatial operations are currently supported for Snowflake and Google BigQuery

Configuring database connections to supported databases

STEP 1: Download the required vendor files for your database

STEP 2: Add the connector type

STEP 3: Create a database connection

Configuring database connections to supported databases

STEP 1: Download the required vendor files for your database

- Get the required vendor files for the database
  - Refer to the Insights documentation for information
  - Download the vendor files from the vendor website
    - Microsoft SQL Server
    - Oracle
    - SAP HANA
    - Snowflake
    - Google BigQuery

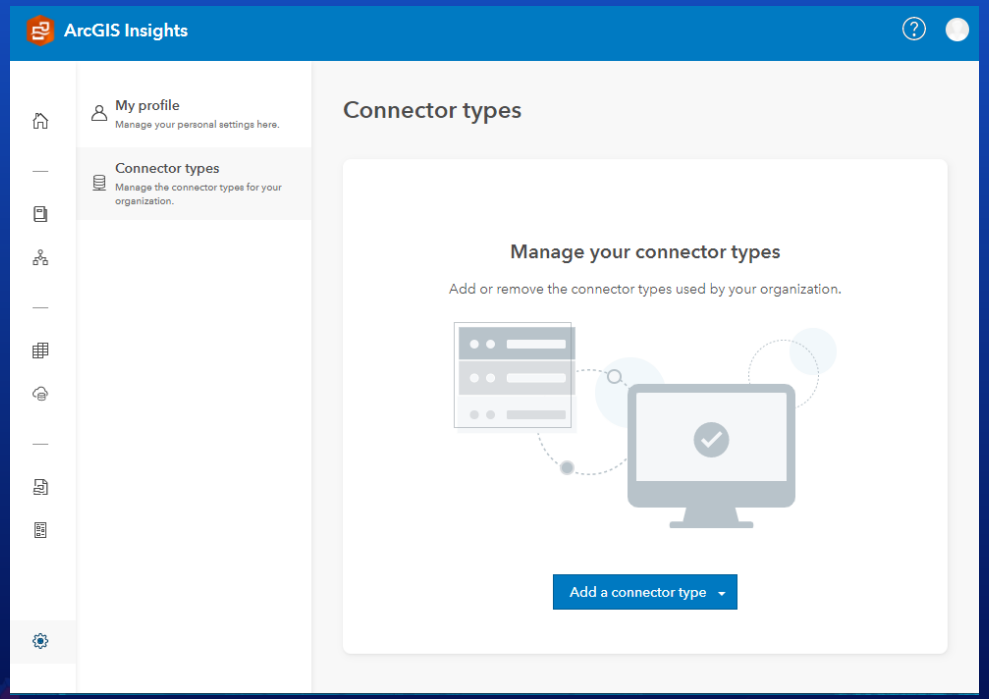
PostgreSQL	PostgreSQL JDBC driver is included in ArcGIS Server.
SAP HANA	SAP HANA JDBC driver: • ngdbc-2.5.50.jar
Snowflake	Snowflake JDBC driver: • snowflake-jdbc-3.12.16.jar

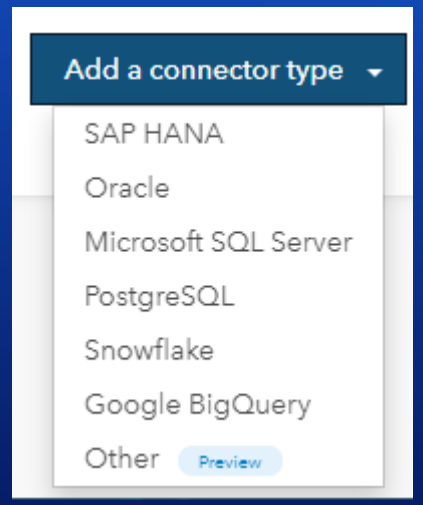
NOTE: PostgreSQL JDBC driver is included in ArcGIS Server

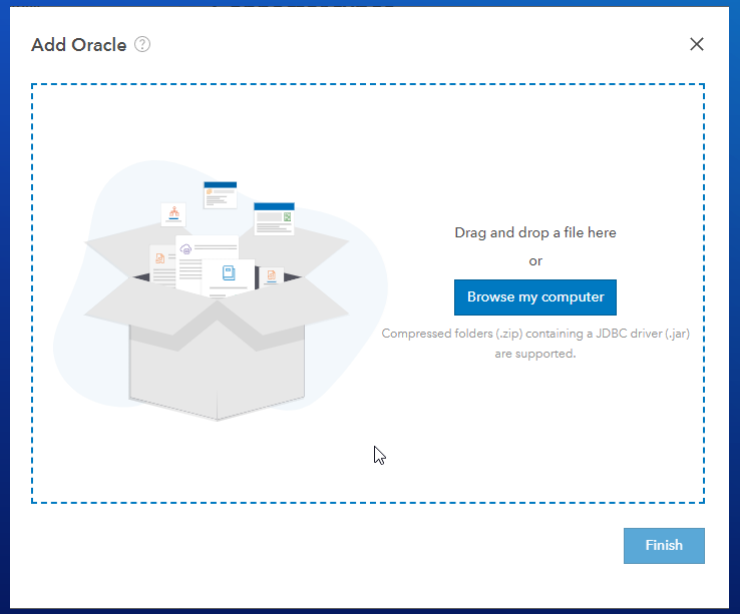
Configuring database connections to supported databases

STEP 2: Add the Connector type

- Add the connector type in ArcGIS Insights NEW





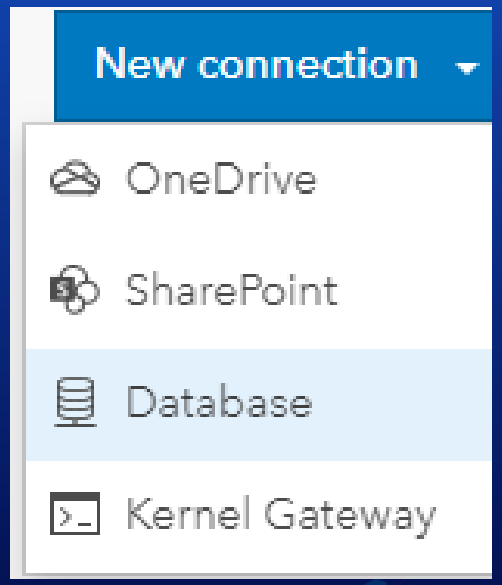


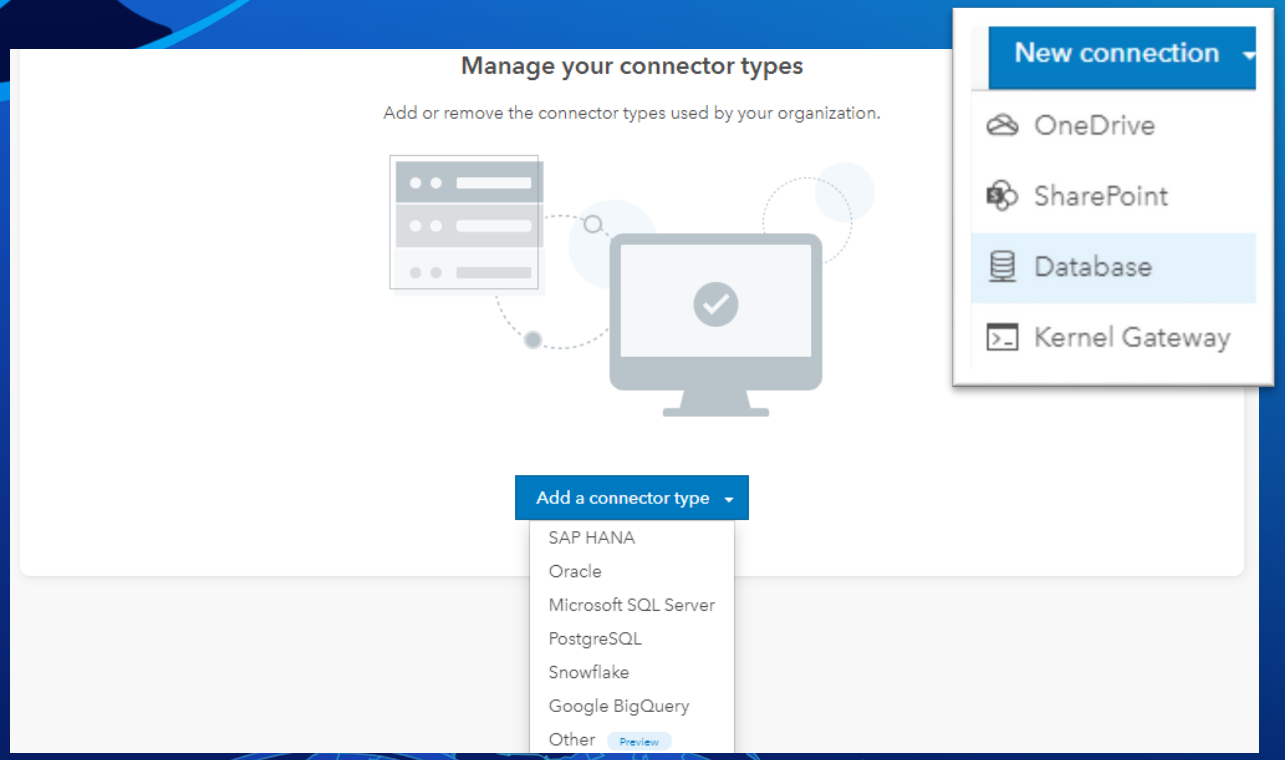
NOTE: Available for admins in ArcGIS Insights in ArcGIS Enterprise and all ArcGIS Insights desktop users

Configuring database connections to supported databases

STEP 3: Create a database connection

- Create a database connection in ArcGIS Insights
- Verify database account has the required database privileges





Configuring connections to Snowflake database

Shirley Baffoe

Configuring database connections to supported databases

Recap

STEP 1: Download the required vendor files for your database

STEP 2: Add the connector type

STEP 3: Create a database connection

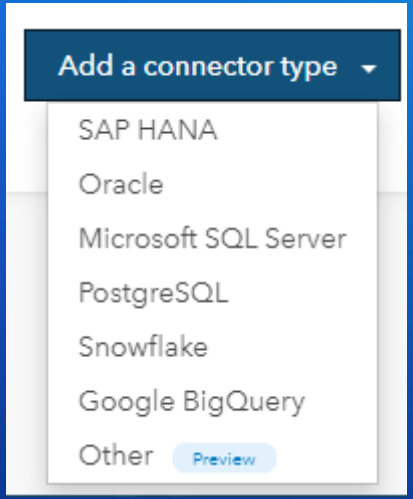
Configuring database connections to other databases

NEW

Chaka Clarke

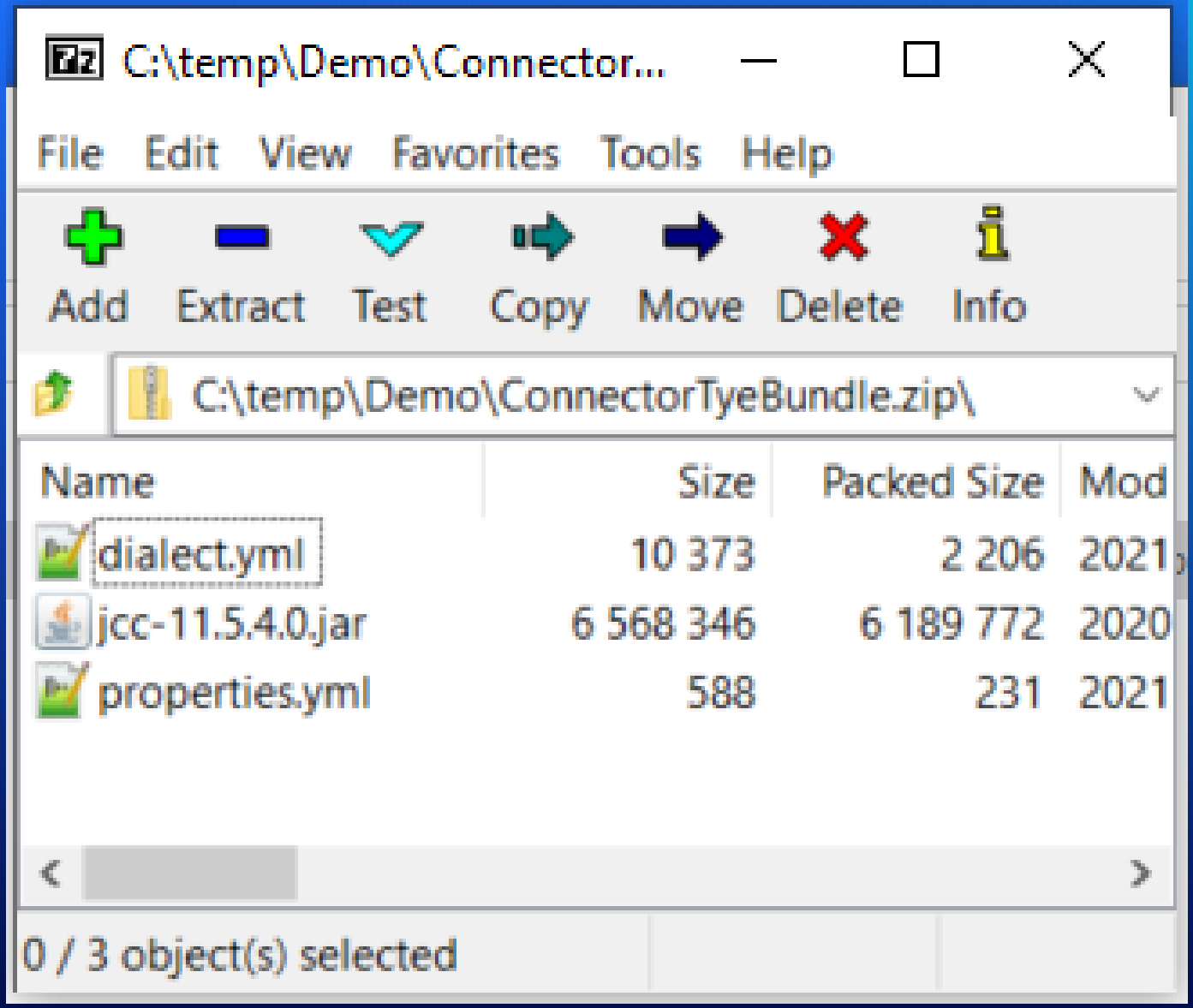
Configuring database connections to other databases

- **Connect to any relational database** NEW
- **Available now our new Insights Connector Guide**
  https://links.esri.com/insights-connector-guide NEW
- **Creating a connector type bundle**



Connector type bundle

- Vendor provided JDBC Drivers
- Properties YML
- Dialect YML



Other databases - Properties YML File

- Creates a JDBC connection to a new database connection.

esri.base.properties.yml

```yml
id: esri.DBName
name: Database Name
driver: com.myVendor.jdbc.Driver
url: jdbc:myVendor://${server}:${port}/${databaseName}:user=${username};password=${password};
properties:
  - name: server
    type: string
    alias: Server Name
    required: true
  - name: port
    type: int
    alias: Port Number
    required: false
    default: 1043
  - name: databaseName
    type: string
    alias: Database Name
    required: true
  - name: username
    type: string
    alias: User Name
    required: true
  - name: password
    type: string
    alias: Password
    required: true
  - name: SSL
    type: boolean
    alias: Enable HTTPS
    required: false
    default: false
excluded schemas:
  - SYS
```

Connector type bundle – Properties YML File

- Creates a JDBC connection to a new database connection.

- ID - unique identifier for the new connector type. This same ID must be used in dialect.Yml file
- Name – name of connector type.
- Driver - class name of the JDBC driver
- Url - JDBC connection string

- .

esri.base.properties.yml

```yaml
id: esri.DBName
name: Database Name
driver: com.myVendor.jdbc.Driver
url: jdbc:myVendor://${server}:${port}/${databaseName}:user=${username};password=${password};
properties:
  - name: server
    type: string
    alias: Server Name
    required: true
  - name: port
    type: int
    alias: Port Number
    required: false
    default: 1043
  - name: databaseName
    type: string
    alias: Database Name
    required: true
  - name: username
    type: string
    alias: User Name
    required: true
  - name: password
    type: string
    alias: Password
    required: true
  - name: SSL
    type: boolean
    alias: Enable HTTPS
    required: false
    default: false
excluded schemas:
  - SYS
```

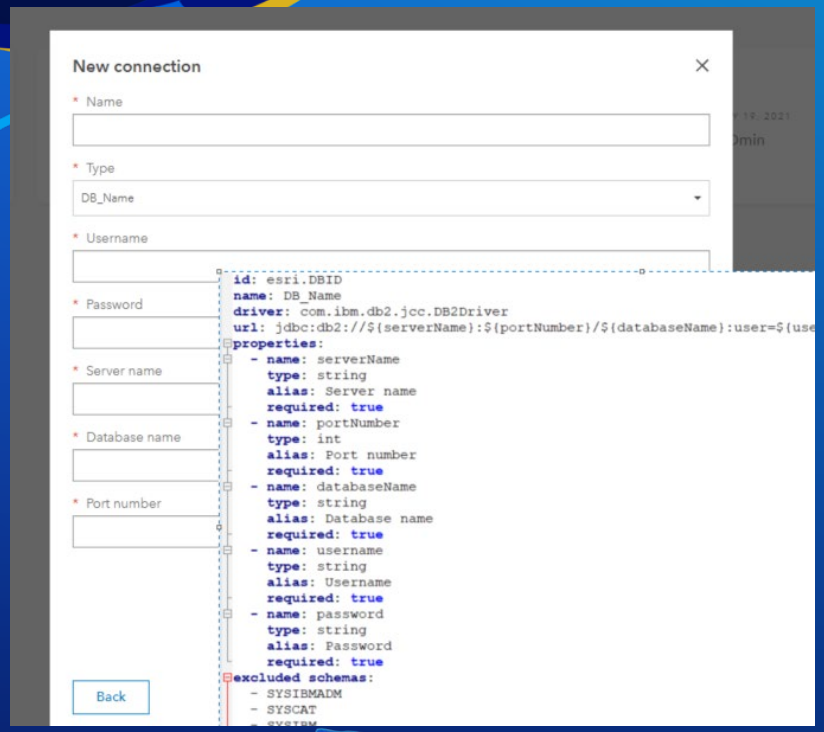
Connector type bundle – Properties YML File

- The properties - defines a list of properties of the connection.
- A property consists of 4 key/values:
  - Name – name of the variable
    type – type of variable [string, integer, boolean]
  - Alias – name used in the new connection form
  - Required – is a required field(true) or an optional field(false)

- Exclude schema
  - list of schemas that will be excluded

esri.base.properties.yml

```yml
id: esri.DBName
name: Database Name
driver: com.myVendor.jdbc.Driver
url: jdbc:myVendor://${server}:${port}/${databaseName}:user=${username};password=${password};
properties:
  - name: server
    type: string
    alias: Server Name
    required: true
  - name: port
    type: int
    alias: Port Number
    required: false
    default: 1043
  - name: databaseName
    type: string
    alias: Database Name
    required: true
  - name: username
    type: string
    alias: User Name
    required: true
  - name: password
    type: string
    alias: Password
    required: true
  - name: SSL
    type: boolean
    alias: Enable HTTPS
    required: false
    default: false
excluded schemas:
  - SYS
```



Creating a properties yaml file for connector to DB2 database

Chaka Clarke

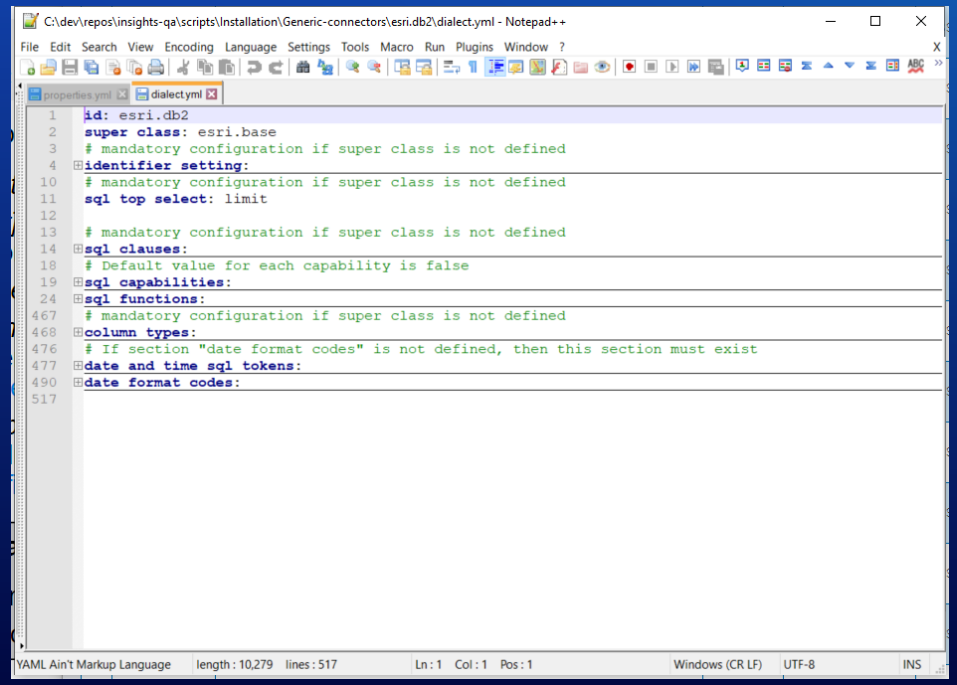
Creating a dialect YAML file

Chaka Clarke

Connector type bundle – Dialect YML File

- Allows administrator to setup relevant database settings. Mostly consisting of definition of capabilities and a list of supported SQL functions

- The sqldialect YML file is made up of 10 key sections
  - Id
  - Super class
  - Identifier setting:
  - Sql top select
  - Sql capabilities
  - Sql clauses
  - Sql functions
  - Column types
  - Date and time sql tokens
  - Date format codes

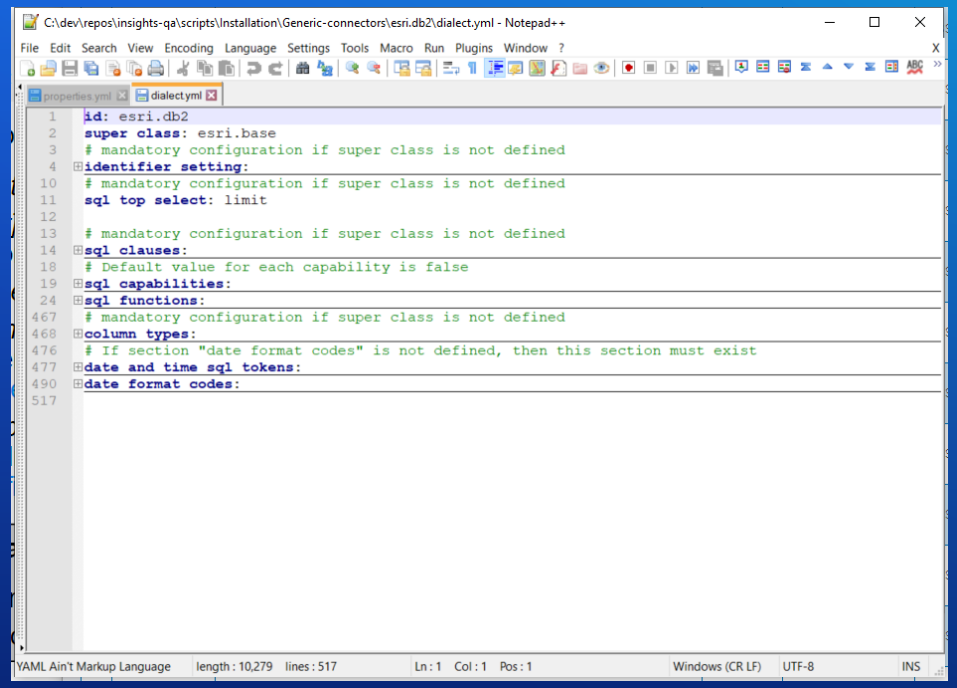


Connector type bundle – Dialect YML File

- ID - unique identifier for the new connector type. This same ID must be used in properties.Yml file

- Super class - name a base class to inherit default values.
  - When this key exists and is correctly configured the remaining part of the dialect file is optional, any additional settings will be used to overwrite default settings for the dialect. See repo for more information on setting this

- .



Dialect YML - “identifier setting”

- These settings describe how this database identifies strings and special characters and the maximum expected length of database identifiers such as table name and column names

properties.yml | dialect.yml

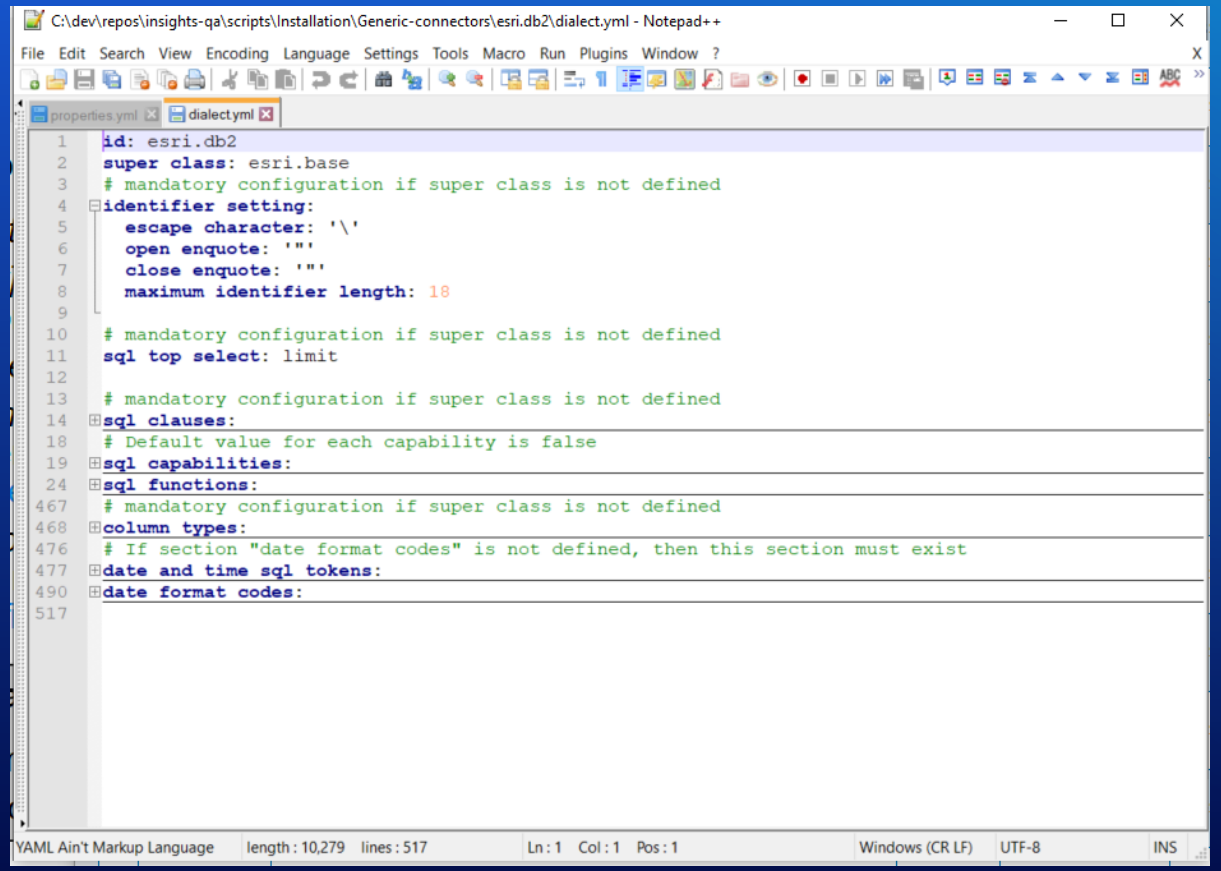
```yml
id: esri.db2
super class: esri.base
# mandatory configuration if super class is not defined
identifier setting:
  escape character: '\'
  open enquote: '"'
  close enquote: '"'
  maximum identifier length: 18

# mandatory configuration if super class is not defined
sql top select: limit

# mandatory configuration if super class is not defined
sql clauses:
# Default value for each capability is false
sql capabilities:
sql functions:
# mandatory configuration if super class is not defined
column types:
# If section "date format codes" is not defined, then this section must exist
date and time sql tokens:
date format codes:
```

Dialect YML- “sql top select”

- Specify what pagination mechanism to use when querying this database.

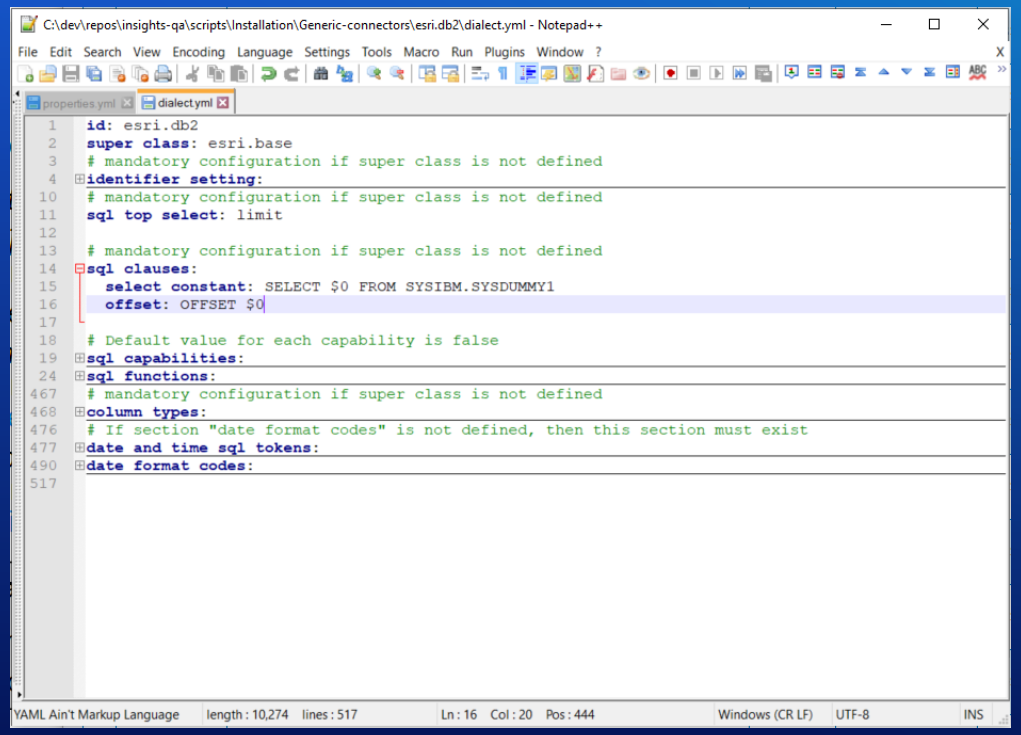
- There are 4 possible values for this key:
  - ‘Top’ – uses select top * clause
    - DB2 : not supported
  - ‘Limit’ – uses the limit expression
    - DB2: supported
  - ‘Fetch next’- uses the fetch next cursor mechanism
    - DB2: supported
  - ‘Row number’– uses the ROW NUMBER mechanism
    - DB2: supported

- .



Dialect YML- “sql clauses”

- **These setting allows the administrator to specify the settings required to run calculations .**
- **‘select from dual” :**
  - **queries** that do not need data from any tables – this is used to perform calculations in the query
    - DB2: SELECT $0 FROM SYSIBM.SYSDUMMY1
  - **‘offset’ – offset syntax**
    - **used in unison with the fetch next select top setting**
    - **DB2 : Offset $0**



Dialect YML- “sql capabilities”

- Defines the current capabilities of this connector supports
- Performance improvements

```yaml
# Default value for each capability is false
sql capabilities:
  window functions: true
  window function in where clause: false
  pagination without order by: true
  common table expression: true
```

Connector type bundle – SQLDialect YML File

- 4 Remaining key sections
  - sql functions
  - Column types
  - Date and time sql tokens
  - Date format codes

- .

properties.yml | dialect.yml

```yml
id: esri.db2
super class: esri.base
# mandatory configuration if super class is not defined
identifier setting:
# mandatory configuration if super class is not defined
sql top select: limit

# mandatory configuration if super class is not defined
sql clauses:
# Default value for each capability is false
sql capabilities:
sql functions:
# mandatory configuration if super class is not defined
column types:
# If section "date format codes" is not defined, then this section must exist
date and time sql tokens:
date format codes:
```

“sql functions” – Aggregate functions group

- Here aggregation functions are defined

```yaml
24  sql functions:
25    # aggregate functions group
26    - name: count
29    - name: count distinct
32    - name: sum
35    - name: avg
38    - name: max
41    - name: median
44    - name: percentile
45      sql: PERCENTILE_CONT($0) WITHIN GROUP (ORDER BY $1)
46
47    - name: min
50    - name: stddev
53    - name: variance
```

“sql functions” – String functions group

properties.yml | dialect.yml

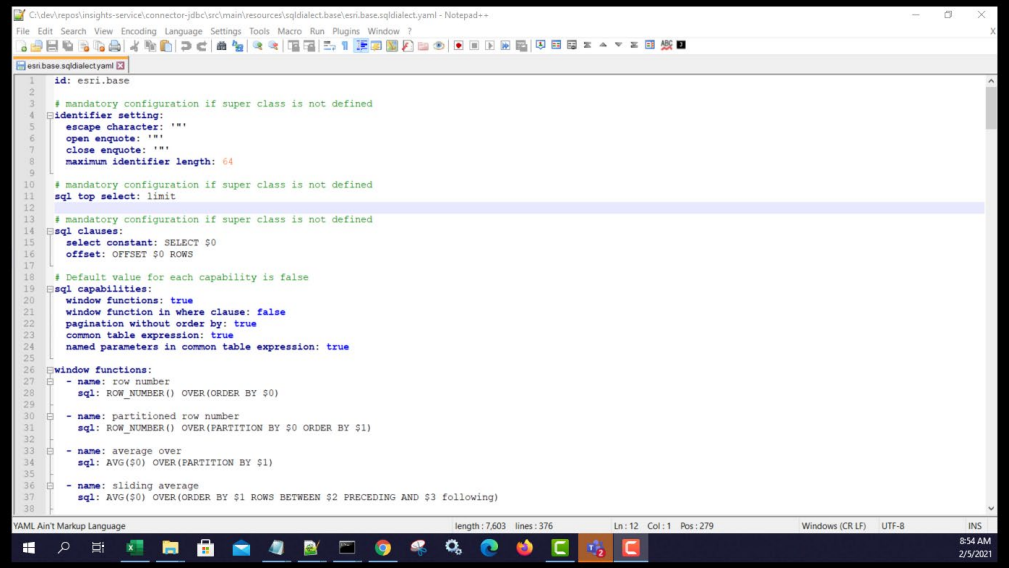
```yaml
# string functions group
- name: trim
  sql: TRIM(BOTH $0 FROM $1)

- name: left trim
  sql: TRIM(LEADING $0 FROM $1)

- name: right trim
  sql: TRIM(TRAILING $0 FROM $1)

- name: lower
  sql: LOWER($0)

- name: upper
  sql: UPPER($0)

- name: string concatenation
  sql: CONCAT($0, $1)

- name: string position
  sql: POSSTR($1, $0)

- name: string length
  sql: LENGTH($0)

- name: string substitute
- name: substring
- name: left
- name: right
- name: like
```

“sql functions” – Date and Time functions group

- Here date and time functions are defined

properties.yml | dialect.yml

```yaml
# date time functions group
- name: extract year
- name: extract quarter
- name: extract month
- name: extract day of month
- name: extract day of week
- name: extract hour
- name: extract minute
- name: extract second
- name: datedif year
- name: datedif month
- name: datedif day
- name: datedif hour
- name: datedif minute
- name: datedif second
- name: create timestamp from milliseconds
  sql: TIMESTAMP ('1970-01-01-00.00.00.00000') + ($0/1000) SECOND

- name: create timestamp from components
- name: create timestamp from text
- name: create time from milliseconds
- name: create time from components
- name: create date from milliseconds
- name: create date from components
- name: current timestamp
- name: current time
- name: current date
- name: now
```



Configuring date and time functions

Chaka Clarke

SQL Dialect YML- Conditional Function Group

- Specify the SQL Functions the Database uses to perform conditional functions.

properties.yml | dialect.yml

```yml
# conditional functions group

- name: case when else
  sql: CASE WHEN $0 THEN $1 ELSE $2 END

- name: case when
  sql: CASE WHEN $0 THEN $1 END

- name: is null
  sql: $0 IS NULL

- name: is not null
  sql: $0 IS NOT NULL

- name: nullif
  sql: nullif($0, $1)

- name: coalesce
  sql: COALESCE($0, $1)
```

- DB2 all these functions are supported. Users will be alerted of the removal of these functions

SQL Dialect YML- Logical Operators Group

- These settings allows the administrator to specify the SQL operators the Database uses to perform queries.

properties.yml | dialect.yml

```yaml
# logical operators
- name: and
  sql: $0 AND $1

- name: or
  sql: $0 OR $1

- name: between
  sql: $0 BETWEEN $1 AND $2

- name: not
  sql: NOT($0)
```

- DB2 all these operators are supported. Users will be alerted of the removal of these operators if it is required for a function in insights

SQL Dialect YML- Arithmetic functions Group

- These settings allows the administrator to specify the SQL functions the Database uses to perform queries.

properties.yml | dialect.yml

```yaml
# arithmetic functions group
- name: abs
  sql: ABS($0)

- name: acos
  sql: ACOS($0)

- name: asin
  sql: ASIN($0)

- name: atan
  sql: ATAN($0)

- name: sin
  sql: SIN($0)

- name: sinh
  sql: SINH($0)

- name: cos
  sql: COS($0)

- name: cosh
  sql: COSH($0)

- name: tan
  sql: TAN($0)

- name: tanh
  sql: TANH($0)

- name: ln
  sql: LN(CASE WHEN $0 <= 0 THEN NULL ELSE $0 END)

- name: log
  sql: LOG(CASE WHEN $0 <= 0 THEN NULL ELSE $0 END)

- name: log10
  sql: LOG10(CASE WHEN $0 <= 0 THEN NULL ELSE $0 END)
```

```yaml
- name: negation
  sql: -($0)

- name: add
  sql: ($0) + ($1)

- name: subtract
  sql: ($0) - ($1)

- name: multiply
  sql: $0 * $1

- name: divide
  sql: $0 / $1

- name: power
  sql: POWER($0, $1)

- name: mod
  sql: MOD($0, $1)

- name: round
  sql: ROUND($0)

- name: round to
  sql: ROUND($0, CAST($1 AS INT))

- name: ceiling
  sql: CEILING($0)

- name: floor
  sql: FLOOR($0)

- name: sign
  sql: SIGN($0)

- name: truncate
  sql: TRUNC($0, %1)
```

SQL Dialect YML

- Conversion functions
  - Configure the column conversion syntax
- Other functions
  - Configures the Lag function syntax.

properties.yml | dialectyml

```yaml
294
295     # conversion functions
296     - name: cast
297       sql: CAST($0 AS $1)
298
299     # other functions group
300     - name : lag
301       sql: lag($0)
```

SQLDialect YML- “column types”

- Specify how various column types are converted using the cast command previously shown
- DB2 uses all defaults except for the float conversion

properties.yml | dialect.yml

```
301     sql: lag($0)
302
303 # mandatory configuration if super class is not defined
304 column types:
305   integer: int
306   float: float
307   string: varchar(255)
308   date: date
309   timestamp: timestamp
310   time: time
```

SQLDialect YML- “date and time sql tokens”

- Configure how various date parts are expressed when indicating a date time format string
- DB2

properties.yml | dialect.yml

```yaml
date and time sql tokens:
    YY: YY
    YYYY: YYYY
    MM: MM
    DD: dd
    hh: hh
    hh24: HH24
    mi: MI
    ss: SS
    ms: MS
    mon: MON
    ampm: a.m.
```

SQLDialect YML- “date format code”

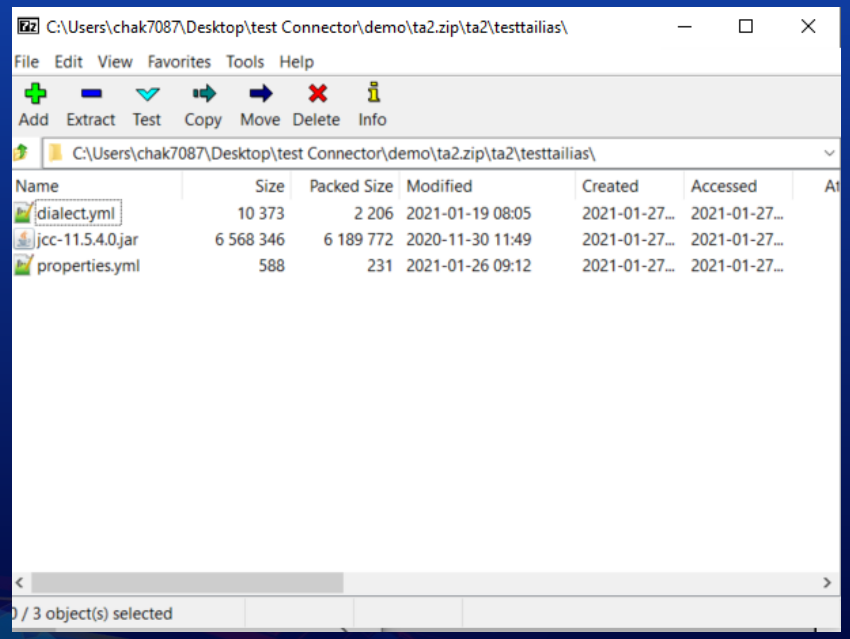
- Optional setting used for databases (MSSQL) that have numerical codes to represent specific format strings
- E.g. For MSSQL
  - Date format codes : #US
    - “mm/dd/yy”:101

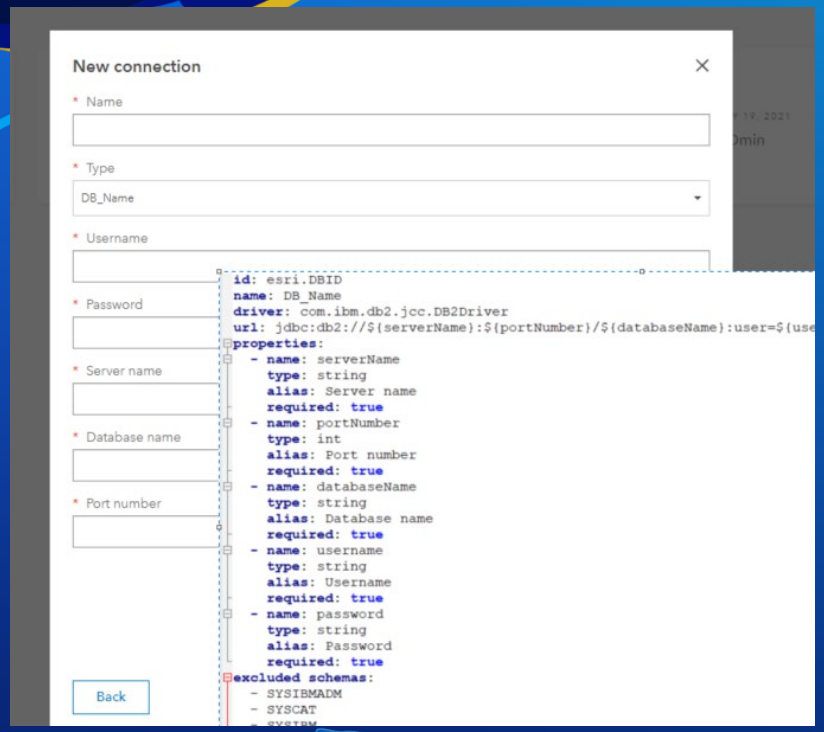
properties.yml | dialect.yml

```yml
date format codes:
  # US
  "mm/dd/yy": mm/dd/yy
  "mm/dd/yyyy": mm/dd/yyyy
  # ANSI
  "yy.mm.dd": yy.mm.dd
  "yyyy.mm.dd": yyyy.mm.dd
  # British/French
  "dd/mm/yy": dd/mm/yy
  "dd/mm/yyyy": dd/mm/yyyy
  # German
  "dd.mm.yy": dd.mm.yy
  "dd.mm.yyyy": dd.mm.yyyy
  # Italian
  "dd-mm-yy": dd-mm-yy
  "dd-mm-yyyy": dd-mm-yyyy
  # Japan
  "yy/mm/dd": yy/mm/dd
  "yyyy/mm/dd": yyyy/mm/dd
  # ISO
  "yymmdd": yymmdd
  "yyyymmdd": yyyymmdd
  # month name
  "dd mon yy": dd mon yy
  "dd mon yyyy": dd mon yyyy
  "Mon dd, yy": mon dd, yy
  "Mon dd, yyyy": mon dd, yyyy
```

Connector Type Bundle

- Once all Configuration is set we can now create a connector type bundle zip containing :
- Dialect.yml, Properties.yml, and Jar of the JDBC Driver.
- In the Following demo we will briefly look at how an administrator can configure to restrict specific functions in insights for a DB2 connector.





Add a new JDBC connector to a DB2 database

Chaka Clarke

Useful resources

- **SharePoint and OneDrive guide**
  - https://doc.arcgis.com/en/insights/latest/get-started/create-sharepoint-connection.htm
  - https://doc.arcgis.com/en/insights/latest/get-started/create-onedrive-connection.htm
- **Supported databases**
  - https://doc.arcgis.com/en/insights/latest/administer/supported-databases.htm
- **Required vendor files**
  - https://doc.arcgis.com/en/insights/latest/administer/vendor-files.htm
- **Insights connector guide**
  - https://links.esri.com/insights-connector-guide
- **JDBC driver repository**
  - http://mvnrepository.com/

Thank you!

Please provide your feedback for this session by clicking on the session survey link directly below the video.

esri®

THE SCIENCE OF WHERE®

Copyright © 2021 Esri. All rights reserved.