IRIS Warehouse User Guide

Last Saved: July 28, 2020

Table of Contents

Contents

[1 Introduction 3](#_Toc46774657)

[2 Accessing and Understanding the Warehouse 3](#_Toc46774658)

[2.1 Access 3](#_Toc46774659)

[2.2 General Rules for Use 3](#_Toc46774660)

[2.3 Warehouse Theories, Practices and Patterns 5](#_Toc46774661)

[2.4 When Dimensions and Facts Are Built 6](#_Toc46774662)

[2.5 Warehouse Patterns 6](#_Toc46774663)

[2.6 Warehouse Metadata 8](#_Toc46774664)

[2.7 How to Query the Warehouse 10](#_Toc46774665)

[2.7.1 Using Tables and Views 10](#_Toc46774666)

[2.7.2 Using OLAP Cubes 11](#_Toc46774667)

[3 Adding New Data to the Warehouse 14](#_Toc46774668)

[3.1 Considerations 14](#_Toc46774669)

[3.2 Steps to Adding Data Objects to the Warehouse 15](#_Toc46774670)

[3.2.1 Planning 15](#_Toc46774671)

[3.2.2 Define Initial ETL Process 16](#_Toc46774672)

[3.2.3 Formal Quality Assurance (QA) 16](#_Toc46774673)

[3.2.4 Documentation 17](#_Toc46774674)

[3.2.5 Closeout 17](#_Toc46774675)

[3.3 Dimensions 17](#_Toc46774676)

[4 Data Workflows 17](#_Toc46774678)

[4.1 ETL Process to Load Dimensions and Facts 18](#_Toc46774679)

[4.1.1 Dimensions 18](#_Toc46774680)

[4.1.2 Facts 18](#_Toc46774681)

[4.2 Choreographing the Loading of Core and Program-Specific Tables 18](#_Toc46774682)

[5 Change Management 19](#_Toc46774683)

[6 Configurations and Permissions 20](#_Toc46774684)

[6.1 SQL-Server Agent 20](#_Toc46774685)

[7 Appendix 20](#_Toc46774686)

[7.1 Example Query from Warehouse 20](#_Toc46774687)

1. Introduction

IRIS Warehouse (The Warehouse), is an Enterprise Data Warehouse[[1]](#footnote-2), or a…

***…centrally managed repository of integrated data from one or more disparate sources for the purpose of analyzing and reporting quality information quickly and easily across the USFWS designed for use by all USFWS staff.***

IRIS Warehouse is read-only and indented should be fast, accurate, timely and easy to use. Longer term, the intent is that this warehouse becomes to the source of truth for all reporting and analysis, regardless of where and how that is accomplished.

Currently, IRIS Warehouse is a collaboration among the following three “data-rich” programs within the USFWS:

* Refuges (Warehouse Lead: Brent Frakes)
* Migratory Birds (Warehouse: Lead: Nathan Zimpfer)
* Ecological Services (Warehouse Lead: Jorge Bravo)

This guide is intended to serve all types of users ranging from the beginners to the staff managing The Warehouse. Section 2 documents how to access The Warehouse and provides more guidance on its use and design.

1. Accessing and Understanding the Warehouse
   1. Access

Currently, read-only access the beta version of IRIS Warehouse can be obtained by contacting Brent Frakes ([brent\_frakes@fws.gov](mailto:brent_frakes@fws.gov)). Once you have permission, you can login here:

**Server**: ifw9ecos-bvdb11.fws.doi.net\ecos\_beta

**Database**: IRIS\_DataWarehouse

We are asking that staff register with the Warehouse to ensure that we have a complete list of all users which makes it easier to communicate changes, bug fixes, and system outages.

* 1. General Rules for Use

When using the Warehouse, we want you to always keep in mind the following:

* The Warehouse is a **read-only repository** for using the data we generate. It does not replace the systems used to manage the data. If you find inaccurate data in the Warehouse, please go back and fix it in the source.
* The structure of the Warehouse differs significantly from the source information systems. Data is structured around dimensions and facts which are optimized for reporting and anaysis. Within each dimension and fact, the data may originate from many different sources.
* All table rows[[2]](#footnote-3) have a row access level, which indicates whether information may be shared with the public, internally only, or intended for use by specific individuals.
* All table rows have a row quality level, which indicates the overall quality of the information on a scale from 1 (poorest quality) to 1000 (highest quality) where -9999 = (unknown quality)
* Always check the table dbo.\_WarehouseStatus. It will tell you when the facts and dimensions are last synchronized with the sources. IRIS is automatically set to synch nightly at 3AM, unless otherwise noted.
* **Always** start your queries with fact tables, **never** with Dimensions – Fact tables ensure the cardinality is correct.
* **Never join dimensions together** – you are likely to get inaccurate and unexpected answers.
* The suite of fact tables should let you answer most, if not all, of the common questions you have. If there is not a fact table which can answer you question, please communicate to the Warehouse team about adding a new fact table.
* To answer unique and novel questions, **you may join the fact tables together using the foreign keys to the dimensions**. However, you must be extremely careful of any difference in grain between the fact tables to ensure your answers are correct.
* You may only share information with the public if it is designated with a row access level of Public. In other words, internal or unknown records may only be used internally.
* If the data in a fact/dimension appears incorrect, please confirm with the source:
  + If the source is incorrect, correct it there (or contact the person who can fix it)
  + If a fact or dimension is incorrect, let a Warehouse Lead know
* Never query all of the information from the Warehouse and stage it locally to query from your computer. Doing so will take a lot of time, is a severe burden on server resources, and affects other users. In short, pull only what you need – nothing more.

Furthermore, as of this date, version 1.0 of IRIS Warehouse is **NOT** **in production** mode. Because it is, you are using this Warehouse knowing the following the following risks:

* **Data May Be Inaccurate** – While we have been careful, our logic to extract and transform the information from the sources has not been fully verified. **If you do find errors, please report them to us immediately so the errors can be resolved.**
* **Facts and Dimensions May Change** – Although generally quite stable, table name, field names and field types may change without notice. We would like to have all of this finalized now to avoid dealing with so many changes once the Warehouse is in productions
* **Facts and Dimensions May Be Periodically Empty**- Because we are actively working on the version 1.0 release, we periodically re-create tables and lose the information. While we attempt to refresh the tables as quickly as possible, there is a risk you may query one while empty.
* **Fact Tables May Point to Invalid Rows in Dimension Tables** – If we are actively synching with the sources, there is a chance you may query a fact table during this time and get odd results. Generally, however, we try to limit synching to the evening, night and early mornings.

Although you do need to be aware of these risks, also know that The Warehouse is not a chaotic mess that is ever in rapid flux.

* 1. Warehouse Theories, Practices and Patterns

Development of the Warehouse was guided by the theory, practices and patterns of Kimball (<https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/>) . Ralph Kimball pioneered this technique, which has been since adopted and used by most data-rich corporations internationally.

A warehouse is built on the premise that there are an unlimited number of questions FWS staff will ask, but there are a common set of dimensions around which they frame their questions. For example, everyone has questions around date, time, staff, organization (e.g., refuge or office). Questions around these dimensions are hampered by the fact that every information system tracks these concepts in different ways, making it difficult to ask cross-cutting questions. Thus, aligning our information to these dimensions ensures a high level of interoperability. Furthermore, dimensions are quality rich and frequently offer other possible ways of representing information that would have been a challenge in the original information systems. Dimension tables almost always contain the qualitative information that describes the different types of objects/things/concepts described by are various information system

In addition to dimensions, The Warehouse is comprised of fact tables which join the various dimensions together for the purpose of answering a suite of questions (aka Data Mart). Fact table are either just foreign keys (i.e., pointers back to each dimension) or other statistics that are used to answer questions and make decisions. Fact table ensure that the dimensions are joined properly and that answers are always correct.

DIMENSION

FACT

DIMENSION

DIMENSION

DIMENSION

1. Star schema creating by joining a fact to its related dimensions.
   1. When Dimensions and Facts Are Built

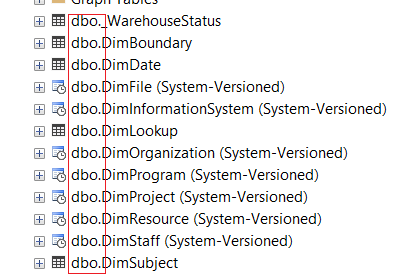
Building new dimensions and fact tables always begins with the identification of the information needed by USFWS staff to accomplish their job duties. For example, the Chief Data Officer needs to understand what information system are managed by the UFWS. Example questions related to this general need would relate to **when** the systems were built **who** is responsible for maintaining them. From these two example questions come three dimensions: staff, date and information system. To help the CDO answer questions, these dimensions can be joined together in a fact table that provides an inventory of these systems.

* 1. Warehouse Patterns

To be usable, intuitive, and consistent, The Warehouse follows a number of standard design patterns. Some of these patterns apply to all tables while others are specific to facts and dimensions.

All tables follow these design patterns:

* Dimensions are always prefixed with Dim; Fact tables are always prefixed with Fact. Tables table names with an underscore (e.g., \_WarehouseStatus) are metadata tables about the warehouse.
* Dimension are always singular (e.g., DimTribe); fact tables are always plural (e.g., FactTribes)
* Dimensions and facts in the dbo schema are relevant and readable by the entire USFWS.



* Tables within the respective program schema are generally relevant to that program and not the USFWS as a whole. Each program (e.g., Refuges, MigBirds, ECOS, etc.) will have two schemas. One is accessible to all USFWS (e.g., Refuges) while the other (e.g., ECOS\_Private) is further restricted to select staff based on the respective programmatic needs.
* Tables never contain NULL or [Whitespace] values. These will be converted into one of the following:
  + **NA** - Not Applicable. In cases of dates, 12-31-9998 = NA and -9998 for numbers.
  + **UNK** – A value is applicable, but unknown. In cases of dates, 12-31-9999 = NA and -9999 for numbers.
* Every table will contain an ID column, which is an integer that uniquely identifies each row. There is no other meaning behind this number. IDs should only be used for linking table (via Primary and Foreign keys).
* Field names should be clear, unambiguous and intuitive. Abbreviations should be minimized unless interpretation is clear. If there is a need to preserve legacy field names, this will be done through the use of view.
* Column names will follow the pascal case patterning (e.g., PascalCase). In cases of acronyms - which should be rare - an underscore will be used as a separator.
* All fact and dimension tables communicate the following:
  + **RowAccessLevel** - Indicates who this information may be shared with. Public, internal or restricted
  + **RowQualityLevel** – What is the quality of information contained with the row.
* Tables with a related “\_History” table are known as system versioned tables. The history tables track older versions of the data, which makes it possible to reconstruct what information was like at a previous point in time.



For dimension tables, the following patterns apply:

* Dimensions always (except in some rare instances) contain the following standard columns:
  + **RowAccessLevel** -This is a textual representation of row access level. Row Access Level is assessed by the information source. For example, information originating from PRIMR is qualified by the system owner based on their own rules.
  + **RowQualityLevel** – This is a number (between 0 and 1) indicating the quality of that row. Quality is assessed by the information source. For example, information originating from PRIMR is qualified by the system owner based on a set of rules specific to PRIMR.
  + **SourceSystemOriginalKey** – The information system acronym and identifier – divided by a pipe - that traces the information back to the source information system. This SourceSystemOriginalKey ensuring tracability back to the source information record.
  + **SourceSystem** - The information system acronym.
  + **OriginalKey** – the system identifier
* To avoid “centipede” dimensions, simple dimensions (e.g., RowAccessLevel, DatePrecision, and Sex) are collapsed into a single dimension called DimLookup. In Data Warehousing practice, DimLookup is also known as a [junk dimension](https://en.wikipedia.org/wiki/Dimension_(data_warehouse)#:~:text=A%20Junk%20Dimension%20is%20a%20dimension%20table%20consisting%20of%20attributes,no%20or%20true%2Ffalse%20indicators.).
* Table names are always singular, not plural (HuntUnit=Yes, HuntUnits=No)

For fact tables, the following patterns apply

* Fact tables always have the first three columns:
  + **ID** - Nothing more than a unique row identifier. Never hard code the ID into any query since it is likely to change over time.
  + **RowAccessLevelLID** – Public/Internal. The rules for determining this are based on the related dimensions and are specific to each fact table
  + **RowQualityLevelLID** – High/Operational/Unknown. The rules for determining this are based on the related dimensions and are specific to each fact table
* The remaining columns of a fact table will either be
  + A link to a dimension – The link is an integer that functions as the foreign key to the ID in the respective dimension
  + Some type of aggregation/fact (e.g., count) – Aggregations will always identify what type of aggregation they are.
* Any fact table having columns ending in '\*LID' (e.g., RowAccessLevelLID) are referencing dbo.DimLookup.
  1. Warehouse Metadata

Four metadata tables further describe The Warehouse, ranging from each dimension and facts, to the numerous information sources contributing to the dimensions. All metadata tables are prefixed with an underscrore (\_) for the purpose of clarity.

The table **\_WarehouseStatus** provides a definition of each dimension and fact, including the following:

|  |  |
| --- | --- |
| **Column** | **Definition** |
| ID | Unique row identifier |
| DateLastUpdated | The date and time the table was last synchronized from the source(s). Most facts and dimensions should update nightly. However, a few, such as date, time, and boundaries, are updated as needed. |
| SchemaName | Indicates whether this table is in the dbo schema (i.e., common to all USFWS) or in a program-specific schema (e.g., Refuges). Schema names ending in Private indicate information that is restricted to staff within the respective programs. |
| TableName | The name of the table |
| TableType | Whether the table is a dimension or fact |
| Description | General description of the table |
| Grain | For fact tables, this is essential to understand what each row represents. |
| UpdateSchedule | Indicate how often the table is updated |
| Notes | Additional notes about the table that are not addressed elsewhere |

The table \_Columns defines each column/field in the respective warehouse tables:

|  |  |
| --- | --- |
| **Column** | **Definition** |
| ID | Unique row identifier |
| \_WarehouseStatusID | FK pointing to the row in the \_WarehouseStatus table. This key joins to the table and the related columns |
| Name | The name of the column |
| Definition | The definition of the column |
| Notes | Additional notes about the column |

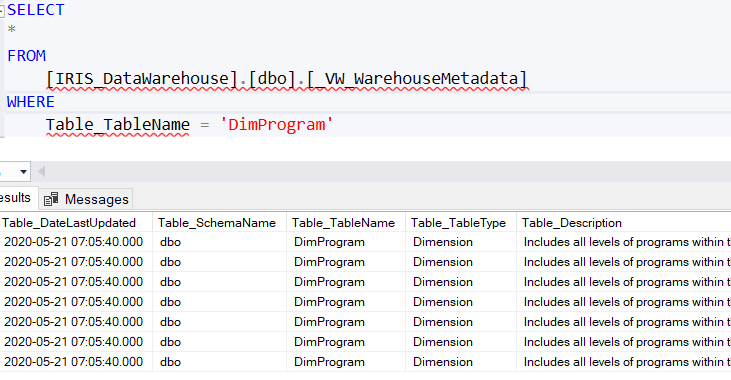
The table \_Sources documents the information source(s) used to populate each dimension:

|  |  |
| --- | --- |
| Column | Definition |
| ID | Unique row identifier |
| \_WarehouseStatusID | FK pointing to the respective dimension in the \_WarehouseStatus table |
| InformationSystemID | FK pointing to the respective information system in the InformationSystem dimension |
| ImportOrder | Indicates the order that sources are imported. This has relevance when information can potentially come from two sources but one take precedence over the other. |
| RowAccesLevel | Describes the logic used to define row access level |
| RowQualityLevel | Describes the logic used to define row quality level |
| ImportNotes | Additional notes |

Finally, the table \_SourceColumns describes which columns in the source tables are used to populate the warehouse dimensions and fact tables:

|  |  |
| --- | --- |
| Column | Definition |
| ID | Unique row identifier |
| \_ColumnID | FK pointing to the respective column |
| \_SourceID | FK pointing to the source used to populate each column |
| SourceTable | Name of table from the information source |
| SourceColumn | Name of the column from the information source |
| SourceColumnAlias | Sometime the column may have an alias. For example, the user interface may label the column differently |
| ImportNotes | Any additional notes |

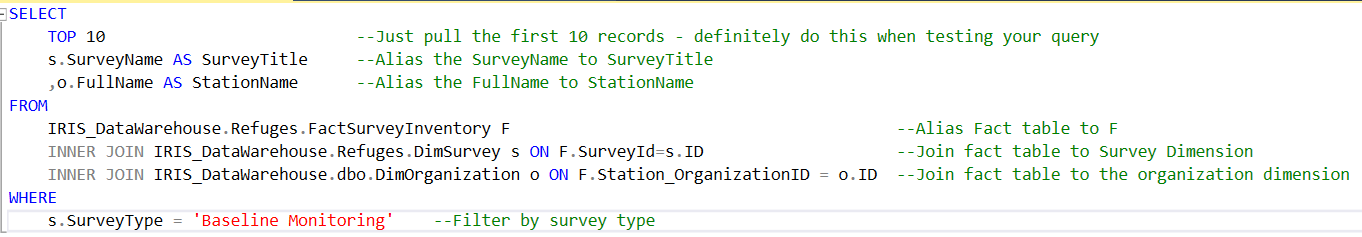
For convenience, the four metadata tables are also integrated as a single view named **\_VW\_WarehouseMetadata**. There is a fair amount of redundancy per row, but it nevertheless is informative and precludes the requirement to join tables.



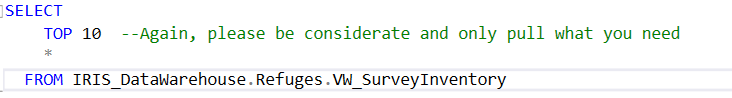
* 1. How to Query the Warehouse
     1. Using Tables and Views

To become acquainted with the Warehouse, we recommend that you install SQL-Server Management Studio. This software is free to use and is designed specifically to work with SQL-Sever.

If you wish to query the Warehouse directly, you have two options. The most flexible and powerful way is to query using a fact table using TSQL. The example below shows an example of how you would do this (code as text is in the Appendix):



If this is too complicated, we will support views that do the joining for you. The downside with views is that you have less flexibility with what columns are available from the related dimensions. Below is an example of querying the SurveyInventory view:



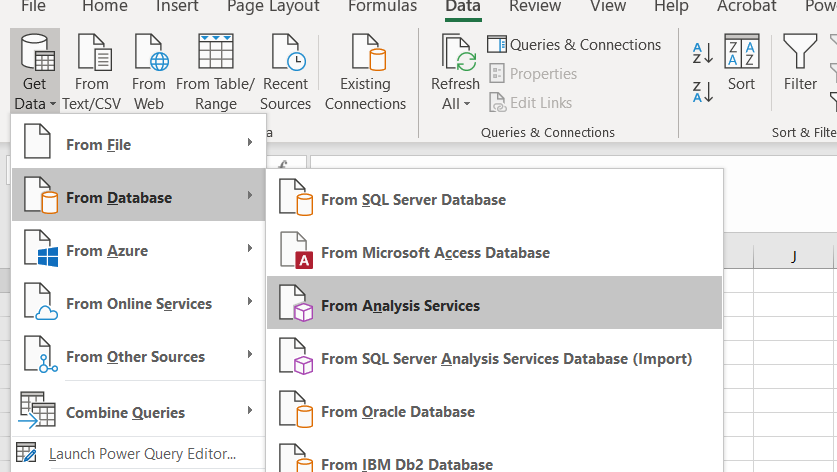
* + 1. Using OLAP Cubes

As needed, we will build OLAP cubes around a fact table that enables rapid and comprehensive querying using Excel pivot tables and/or MS PowerBI. OLAP cubes pre-calculate metrics (e.g. counts, sums, etc.) *a priori* so any query is incredibly fast.

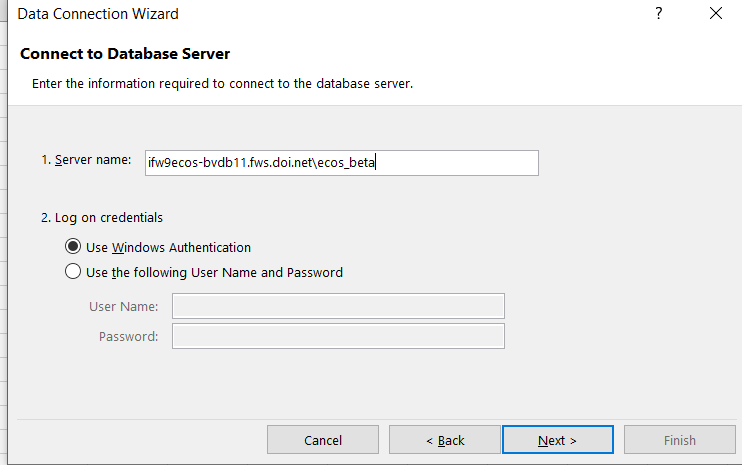
To use the cubes, you will need:

* Office 365 installed on your computer – Although older versions of Office should work, we have been having OLE connection problems.
* Request read-only access to the cubes (contact [brent\_frakes@fws.gov](mailto:brent_frakes@fws.gov)...again) – This is separate from the relational tables. As always, we want to know who is using the cubes.

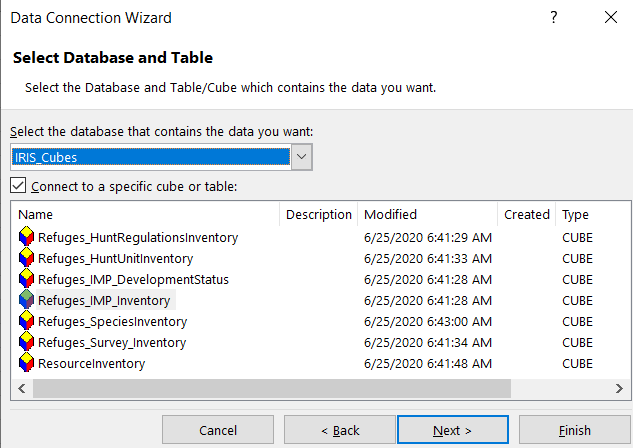
Once you have met the previous criteria, open Excel and select DataFrom DatabaseFrom Analysis Services



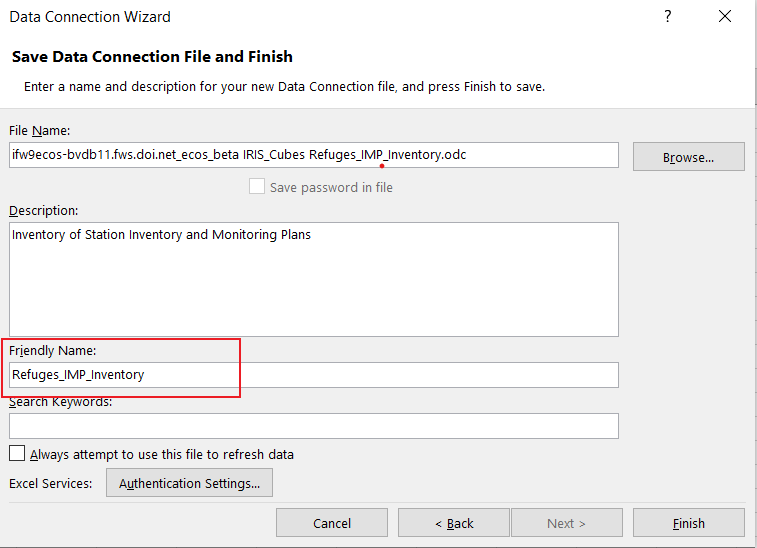
Enter the server name



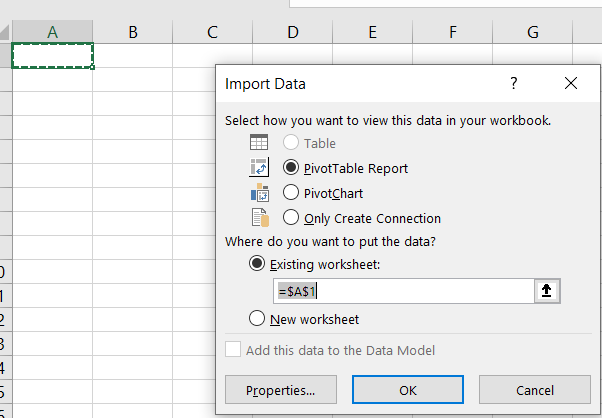
Select a cube



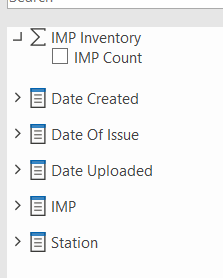
Name the connection and give it a description,



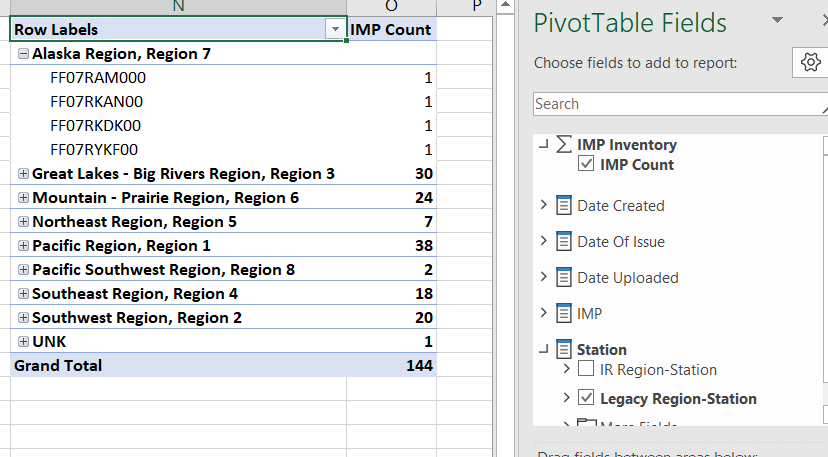
Select one of the view options (e.g., Pivot Table)



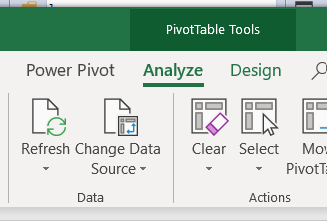
You will then have the option of building the table by selecting at least one metric (under the sigma), and optionally selecting zero or more dimensions.



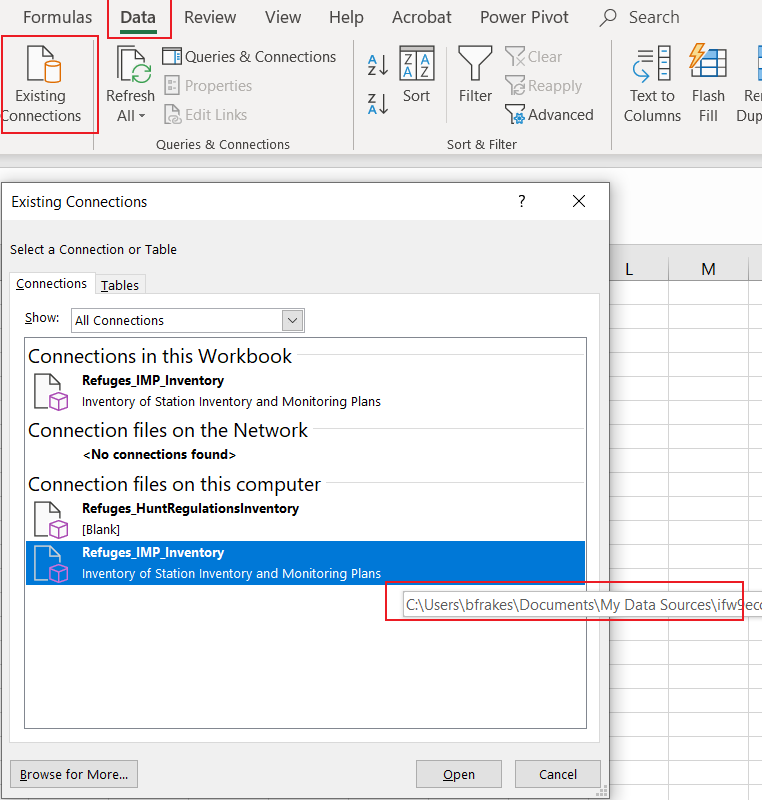
Below is an example of a pivot table showing the number of completed IMPs by legacy region:



When a new pivot is created, data will be refreshed from the server automatically. However, for existing pivot tables where there is no modification to the query, data will need to be refreshed from the server as often as needed by selecting the refresh option:



Likewise, once a connection to a cube is defined, it can be re-used from the Existing Connections option:



Note that hovering over an existing connection indicates where it resides locally on your computer. If you want to remove a connection, you can navigate to that folder and remove it using file explorer.

1. Adding New Data to the Warehouse
   1. Considerations

The Warehouse is intended to be just that, a space to store all types of information, regardless of source and scale. Therefore, all data/information is legitimate to add assuming it meets the following criteria:

* You are the steward of the data or the data is of public domain
* You need this data to accomplish your duties and/or answer questions related to your job
* The data is well-documented
* There is a clear process for accommodating changes to the data

When adding data, we will have a discussion around the following questions:

* Will the information be accessible to the entire USFWS or is it considered private?
* What types of questions are you wanting to answer with the data?
* How should we handle NULL values?
* What is the best way to synchronize with the data source(s)?
* How likely is/are the data source(s) to change in structure?
* Is there a need to track changes over time?
* What program are you in?
* How reliable is the data source for quality? How would one assess each row/record for quality?

Because staff resources are limited, we will prioritize adding new data based on the following considerations:

* Number of users needing the information, including whether the information is considered core or program-specific
* Roles of users within the organization (leadership, refuge staff, office support)
* The complexity of the ETL process
* The space required to manage the data
* The quality of the source information and the cost associated with cleanup
* How the information aligns with the priorities of the USFWS
* Whether there are any legal issues with accessing/using the data
  1. Steps to Adding Data Objects to the Warehouse

The addition of data objects - includes dimensions and facts and may also include views – about some source information, generally adheres to the following steps, recognizing that the steps typically iterative and rarely completely linear.

* + 1. Planning

The planning phase considers:

* Whether a new conformed dimension/facts/views should be added, existing conformed/facts/views, or a program-specific dimensions/facts need to be created to .
* Whether the information that is restricted to specific individuals, internal or public
* If the source data is likely to change over time and if so, whether the changes need to be tracked
* What the primary and natural key of the source data is
* What dimensions are required to build each fact table
* What is the grain of each fact table
* What improvements need to be made to the source data
* Whether the new information added will support end user views/queries
  + 1. Define Initial ETL Process

Once planning is completed, the second step:

* Generates the code to create the staging tables and the dimension and fact tables
* Creates stored procedures and other code to support
  + Truncation – Truncates all data from staging tables
  + Extraction – Extract data from the source(s) and stage it to a staging database
  + Transformation - Change and often improve the integrity of the existing source information
  + Loading – Copy data from staging database to the Warehouse
* Ensure the package is integrated with the broader ETL workflow and the changes are deployed and functional
  + 1. Formal Quality Assurance (QA)

The third step is formal QA, where a rigorous check is done on the data.

The following checks are applied to all tables

* Confirm data types are appropriate and sized appropriately
* Confirm all table patterns are followed (see 2.5)
* NULL and [Whitespace] are converted to UNK or NA

When dimensions are reviewed, the following additional quality checks are applied:

* Text fields are sized appropriately, with minimal use of MAX
* NVARCHAR is only used when support for Unicode is necessary
* Geospatial data is geo-referenced properly as native geometries and/or geographies
* Values match the source values (with exception of cases where cleanup was applied)
* Boolean fields are made more descriptive than True/False or Yes/No
* Numeric values are binned into more meaningful categories
* Type 2, or Slow Changing Dimensions, are not inadvertently adding records
* Natural key constraints are added to prevent duplicity
* The number of rows is consistent with the source(s)
* Whitespace is trimmed around text fields
* UPPERCASE is converted to ProperCase (except for Acronymns or other odd cases)

Reviewing fact tables includes the following additional checks:

* All aggregate columns have the computation as a suffix (e.g., \*\_count)
* The grain matches the metadata description
* The number of rows is consistent with the dimension(s)
* Additive columns are additive
  + 1. Documentation

During the formal documentation, the following are done

* Describe object in the Warehouse Status table
* Describe each column in the \_Columns table
* Describe each source in the \_Sources table. Any modification/filtering of the data will be done here.
* Describe the mapping of each source column to the warehouse column in the \_SourceColumns table
  + 1. Closeout

The final step is closeout, which includes

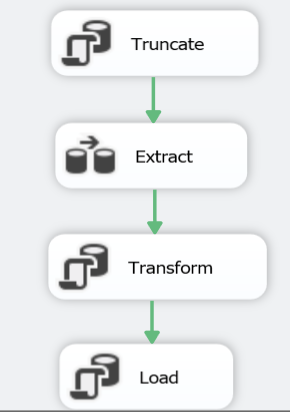
* An expert familiar with the data source confirms the integrity of the data
* An expert familiar with the data confirming row access level
* An expert familiar with the data confirming the row quality level
* Warehouse user leads confirming dimensions/facts/view meet their business requirements
* Signoff
  1. Dimensions

1. Data Workflows

This section explains the process for loading data into the Warehouse. The process is composed of a series of subprocesses, each of which is described below. Section 4.1 describes the standard ETL pattern, which is ubiquitous with loading all data to The Warehouse. Section 4.2 details how the orchestration of the loading for all core and program-specific tables.

* 1. ETL Process to Load Dimensions and Facts
     1. Dimensions

Regardless of the dimension, there are three general steps involved in its creation. These steps are known as an ETL process, or Extract, Transform and Load.



Before being integrated into The Warehouse, all data is first loaded to a dedicated database, known as the staging database. Prior to each load, every table in the staging database is truncated (i.e., cleared of all data), to ensure that they do not contain any stale information. Data is extracted from multiple sources – whether a text file, sharepoint site, database, web service – on a nightly basis (currently 3AM MST). Generally, data is imported to the staging database and tables without any transformation from the source.

Following extraction, the data is transformed to align to the structure of the dimension. During this step, NULL values are set to UNK or NA and other value-added processing steps are applied to the data.

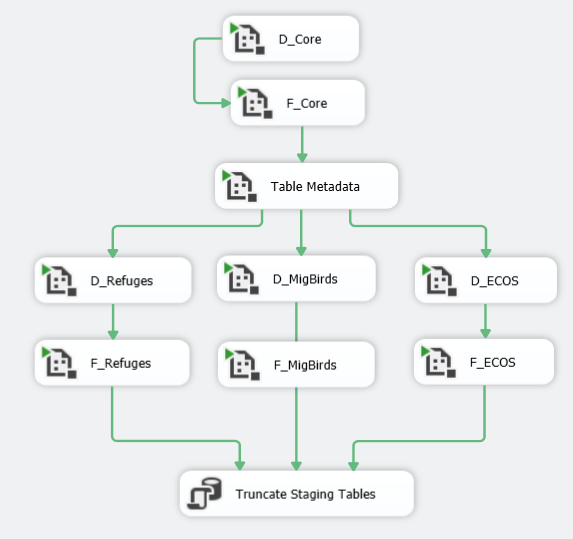
Finally, the data is loaded to the dimension following one of the three following scenarios: There is new data to load to the dimension, the dimension needs to be updated, or there are records in the dimension that must be deleted.

* + 1. Facts

Following creation of all dimensions, the fact tables are then updated. The foreign keys are updated to reflect the new IDs in the related dimensions. Additionally, metrics are calculated, either from the source information or other fact tables.

* 1. Choreographing the Loading of Core and Program-Specific Tables

The following graphic present the high-level workflow of how the Warehouse is synchronized. Core dimensions and facts are first derived since all other tables are dependent on these. Once completed, each program has their dimensions and facts calculated concurrently since there are no dependencies among programs. First, the core dimensions and facts are generated. Once completed, program-specific dimensions and facts are then generated. The final step is cleanup where all staging data is removed.



1. Change Management

IRIS Warehouse is a dynamic system – changes are inevitable and desirable. That said, it is important for end users that these changes do not disrupt their use of the Warehouse. To strike a balance between agility and stability, we make use versioning (major, minor and patch) and views.

Each change to the Warehouse structure (not just a refresh of the data) will be represented by a three-part semantic version (e.g., 1.4.2). This represents levels of versions which include major changes, minor changes and patches. Major versions represent broad changes to the warehouse that are not backwards compatible, meaning that is may not be compatible with code written for prior major versions. Major versions will occur in cases where we make large structural changes to the warehouse in response to user needs and developing best practices. Examples would be the renaming/removal of tables/columns, the change of data types (e.g., from a string to a number), or the modification of domain values. Minor versions are more frequent changes that are backwards compatible. Most often, this includes the addition of tables/columns. Finally, patches represent frequent changes to the Warehouse, most often for the purpose of fixing bugs and minor issues. Examples of patches would be the replacement of NULL values with either an UNK or NA or fixing errors in the ETL process.

Views are virtual tables that are created by joining as a query one or more tables from the Warehouse. Unlike the dimension and fact tables in the Warehouse, a view is not part of the physical schema and has the advantage of hiding the underlying complexity and changes. Views in the Warehouse are managed as the most stable objects and provide the greatest immunity to change.

The following table summarizes the use of versions and views and the lead time staff will have to prepare for the changes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Expected Rate of Change** | **Table Backward Compatibility** | **View Backward Compatibility** | **Lead Time Warning** |
| **Major** | Annually | No | 1 Year from release | 3 months |
| **Minor** | Every 2 months | Yes | Yes | 1 month |
| **Patch** | Bi-Weekly | Yes | Yes | --- |

All information regarding past and future versions is managed in the \_Versions table.

Changes considered to be backward compatible include the following:

* Case changes to table/column/view names
* Addition of a column to a dimension or fact
* Improved documentation that does not change the overall definition of the table/column
* Adding user permissions

Changes that are not backward compatible include:

* Renaming a table/column/view
* Changes to the fact grain
* Removal of a table/column/view
* Changed documentation that alters the meaning and ultimate use of the table/column/view
* Changes to the data types (e.g., from integer to varchar)
* Removing/altering user permissions

1. Configurations and Permissions
   1. SQL-Server Agent

To schedule the synchronization of the warehouse and the processing of the cubes, the user **ifw\ifw9ecos-servcat must be granted**

1. Appendix
   1. Example Query from Warehouse

SELECT

TOP 10

s.SurveyName AS SurveyTitle

,o.FullName AS StationName

FROM

IRIS\_DataWarehouse.Refuges.FactSurveyInventory F

INNER JOIN IRIS\_DataWarehouse.Refuges.DimSurvey s ON F.SurveyId=s.ID

INNER JOIN IRIS\_DataWarehouse.dbo.DimOrganization o ON F.Station\_OrganizationID = o.ID

WHERE

s.SurveyType = 'Baseline Monitoring'

1. This is a real **E**nterprise **D**ata **W**arehouse, not an **e**xtremely **d**umb **w**ay of describing a shared drive [↑](#footnote-ref-2)
2. Exceptions are for common dimensions like date and time [↑](#footnote-ref-3)