Data Guide

Including

- Data Sources: Understanding Data Sources and options to define them.
- About Data Blocks: Understanding Data Blocks and how to create them.
- Data Blocks Security: Granting access to the data retrieved by a data block.
- SQL Parameters: About SQL Parameters and their Data Types.
- Conditional SQL: Creating SQL containing optional filters.
- Data Block Includes: Creating Data Blocks from other Data Blocks.
- Drupal Entities: Create Data Blocks that load Drupal entities.
- Raw Mode Queries: Reduce memory consumption when exporting data.

Data Sources

Data sources define a connection to a database and a collection of data blocks (paramterized sql and/or xml files).

Part of the definition of a data source, is the location of a collection of files (sometimes called a repository), which is a directory on the web server that will contain all data block files related to that data source. This directory should not be in a place inside of your web servers document root, but if you do, take steps to make sure that files in this directory is not writable by the web user.

Defining such data source can be done in one of three ways:

- Defined Forena configuration data tab.
- Defined within settings.php files which allows for including PHP logic.
- Defined by a module using a forna_alter_repos hook.

Once the data sources are defined, you can start building data blocks using these data sources.

Data Sources Settings

Visit the Data tab within Forena Configuration to define data sources (database connections) using administration (configuration) screens. Note however that prior to defining a new database connection, you'll need to create a folder to store the corresponding data blocks (SQL files) on the file system.

The following table illustrates the options provided:

Name	Machine readable name, which is used in referencing all data blocks used by this data source and which should not contain any special characters or spaces.
Title	Human readable name that describes the data source, and which is primarily used in error messages where the data source cannot be accessed.
Enabled	Check-box which, if unchecked (disabled), will cause all queries related to this data block to return no data.
Debug	Set to TRUE to write the raw data queries and their results to the watchdog log. This can be useful for debugging data blocks in a development environment.
Path	Repository directory containing the data block files.

Current user	The function that will be used to determine the current user (i.e. Drupal's UID, Drupal's User Name, or none). The **current_user** argument is passed to all data blocks so that you can write queries that are specific to the user logged in (by using **current_user**) preceded by a "**:**".
Data security method	The function that will be used to check permissions. Supported options are: • **Use drupal permissions**, which will use Drupal security access rights (Access callback = user_access). • **Match values provided by a data block**, which uses a data block providing permissions list to interpret permissions and which should return a single column of permissions based on the current user (may be provided by another repository). **Note**: Custom modules may provide their own functions for checking security.
Driver	The name of the class that will be used to provide the data blocks (= the data provider). Data engine plugins delivered with Forena include: • FrxDrupal - Drupal • FrxOracle - Oracle Database • FrxPDO - PDO other than Drupal • FrxPostgres - Postgres Database • FrxMSSQL - MSSQL Database • FrxFiles - XML Files
uri	The connection uri for the data block (not for FrxDrupal driver).
User	The user name used to establish the connection to the data provider (only for FrxOracle, FrxPDO or FrxMSSQL driver).
Password	The password used to establish a connection to data provider (only for FrxOracle, FrxPDO, FrxPostgres or FrxMSSQL driver).
Database	The database used to establish a connection to data provider (only for FrxDrupal or FrxMSSQL driver).
Character Set	Leave blank for default character set (only for FrxOracle driver).
Oracle native XML	Enable this option (by checking the check-box) if you want to use Oracle's native XML functions (only for FrxOracle driver). In order to use this you must first install a function called f_forena_xml into your database schema. You'll find the .sql file which can be used to install this function included in the Forena distribution.
Postgres native XML	Enable this option (by checking the check-box) if you want to use Postgres native XML support, which requires Postgres 8.3 or better (only for FrxPostgres driver).
Microsoft SQL native XML	Enable this option (by checking the check-box) if you want to use XML auto queries to generate XML (only for FrxMSSQL driver).

For more info about this topic, checkout Basic Reporting on External Data - Quick introduction to reporting using the Northwind sample database (i.e. what is explained from about 03:00 to 05:45, and which also explains how to enable the Northwind Extended database in MySQL format).

Data Sources in settings.php files

Additional data block repositories can be created to allow Forena to report against most applications. To create repositories you'll need to perform these steps:

1. Edit your Drupal site's settings.php to specify the locations of the additional data block repositories.
2. Manually create the directory.
3. Create new data block files as necessary to be used in reports.

Step 1: Edit your Drupal site's settings.php file

The list of additional repositories is stored in a php global variable called **$_forena_repositories**. The following example code illustrates the lines that would need to be added to your Drupal site's settings.php file to create a new data block repository using **local** as the machine readable name (which is used in referencing all data blocks used by this data source and which should not contain any special characters or spaces):

```
global $_forena_repositories;
$_forena_repositories['local'] = array(
  'path' => 'sites/default/local_blocks',
  'title' => 'Site Specific Data Blocks' );
```

The following table illustrates the options provided:

path	Repository directory containing the data block files.
title	Human readable name that describes the data source, and which is primarily used in error messages where the data source cannot be accessed.
debug	Set to TRUE to write the raw data queries and their results to the watchdog log. This can be useful for debugging data blocks in a development environment.
user callback	The function that will be used to determine the current user (i.e. Drupal's UID, Drupal's User Name, or none). The **current_user** argument is passed to all data blocks so that you can write queries that are specific to the user logged in (by using **current_user**) preceded by a "**:**".
access callback	The function that will be used to check permissions. To use Drupal security access rights specify a value of **user_access**. Custom modules may provide their own functions for checking security.
data provider	The name of the class that will be used to provide the data blocks. Data engine plugins delivered with Forena include: • FrxDrupal - Drupal • FrxOracle - Oracle Database • FrxPDO - PDO other than Drupal • FrxPostgres - Postgres Database • FrxMSSQL - MSSQL Database • FrxFiles - XML Files
uri	The connection uri for the data block (not for FrxDrupal driver).
user	The user name used to establish the connection to the data provider (only for FrxOracle, FrxPDO or FrxMSSQL driver).
password	The password used to establish a connection to data provider (only for FrxOracle, FrxPDO, FrxPostgres or FrxMSSQL driver).
database	The database used to establish a connection to data provider (only for FrxDrupal or FrxMSSQL driver).
character_set	Leave blank for default character set (only for FrxOracle driver).
oracle_xml	Set to true if you want to use Oracle's native XML functions (only for FrxOracle driver). In order to use this you must first install a function called f_forena_xml into your database schema. You'll find the .sql file which can be used to install this function included in the Forena distribution.
postgres_xml	Set to true if you want to use Postgres native XML support, which requires Postgres 8.3 or better (only for FrxPostgres driver).

mssql_xml	Set to true if you want to use XML auto queries to generate XML (only for FrxMSSQL driver).

Step 2: Manually create the directory

Create the directory corresponding to the **path** specified in the previous step.

Data Blocks (SQL Queries)

Data blocks are files that are located in a **database repository**, which is a (secured) directory on the web server that contains all data block files related to a specific data source. The actual format of these files is dependent on which data provider or driver is being used for the data source:

- the most common format for these files is SQL, used for all supported data engines except for FrxFiles (XML Files). Here is a sample of such SQL file:

```
--ACCESS=access administration pages
SELECT type, count(type) as typecount
FROM
GROUP BY type
ORDER BY type asc
LIMIT 50
```

- an alternative format for these files is XML.

SQL and XML files live on the file system on the web server and can be created using the SQL or XML editor of your choice.

To create new data blocks in SQL format, you may want to consider using the Create New SQL Query link on the Structuring Forena Data screen. This will launch the **query writing tool** that comes with Forena, which is called the **Forena Query Builder**. This is a separated module that comes with Forena. It can be enabled as per standard Drupal instructions to enable an additional module. For a video tutorial about this query writing tool, checkout Define data blocks with optional filters.

Data Block Security

Data Blocks Security is a technique used to secure a selected data block. It is an additional (but optional) security layer on top of the Drupal permission "access **repository name** data" right (whereas "repository name" is the name of the data block's repository). Using this technique it is possible to enforce additional Drupal permissions such as:

- access content
- access administration pages

- administer content
- administer users
- administer permissions
- ... (the (Drupal permissions) sky is the limit)

Data Blocks Security is actually implemented by means of a comment containing **ACCESS=*some_permission***. It is added near the top of a data block which indicates the Drupal permission required to access the data retrieved by the data block. Its format depends on the format of the Data Block (which can be either in SQL format or in XML format), as further detailed below.

Note: While working on these permissions, it may be a good idea to enable the Masquerade module to experience the effects of granting some permission to various types of user roles.

SQL Data Blocks Security

To specifying Data Blocks Security for a data block in SQL format, use an SQL comment starting with **ACCESS=*some_permission***, as in this example:

```
--ACCESS=administer users
SELECT u.uid,u.name
FROM {role} r
  JOIN {users_roles} ur ON r.rid=ur.rid
  JOIN users u ON ur.uid=u.uid WHERE r.rid = :role
--INFO
type[role]=int
```

XML Data Blocks Security

To specifying Data Blocks Security for a data block in XML format, use an XML comment line containing **ACCESS=*some_permission***.

Custom Data Blocks Security

The security for data block repository is configurable and pluggable, meaning that developers may create functions that determine how the permission is checked. In the Drupal repository delivered with Forena, access permissions are tested using the Drupal user_access() function, so the value should match a Drupal permission. If no value is provided, then all users with the "access **repository name** data" right (whereas "repository name" is the name of the data block's repository) will be authorized to access this data.

In Drupal 7, Drupal permissions passed to the user_access() function are string keys that are usually lower cased versions of the rights found on the Drupal permission tab. However, module and core developers may use any string that they want in creating rights. There is unfortunately no easy way

in Drupal to list module permissions, but after the permission has been granted to a role, you may use the Roles sample report to determine a listing of rights that may be used to identify permissions.

SQL Parameters

The **:role** parameter in the data blocks example is a named token that will be replaced in the SQL query with a parameter from the report. The parameter replacement is done by Forena in a way that protects against SQL injection. Although these tokens are modeled after a commonly used database binding syntax, the replacement is done by Forena. Use this syntax instead of the native parameter binding for any database you are accessing with Forena. Tokens may be referenced multiple times within the same SQL query.

SQL Parameter Data Types

All data coming in from parameter forms and from the URL is considered string data. In some cases you may need to make sure that a data block casts the incoming parameters in a particular type. Numeric values should be included in the SQL without surrounding quotes, but in a way that is safe from SQL injection attacks.

You can use the --INFO section of your data block to specify data types for parameters in the data block as in this example:

```
--ACCESS=access content
SELECT nid FROM node
  WHERE promote=1
    AND status=1
  ORDER BY sticky DESC, created
--IF=:limit
LIMIT :limit
--ELSE
LIMIT 10
--END
--INFO
type[limit]=int
```

In the above example (the last line of it), the **:limit** parameter is specified to be of type **int**. The following Parameter Data Types are supported:

int	Convert to an integer. This is useful for limit queries as in the above example.
numeric	Convert to a floating point number (e.g. 6.2).
array	Convert to an array. This is useful for **IN** clauses.
date	Convert to an ISO representation of a date string in YYYY-MM-DD HH:MI:SS format. PHP date creation syntax is supported, so you can use values like now + 1 year.
unixtime	Convert to a UNIX timestamp version of time and treat this as an integer. This is particularly useful for working with Drupal dates. PHP date creation syntax is supported, so you can use values like now + 1 year.

Conditional SQL

When building data blocks you can specify sections of sql that are only included if a particular

parameter is present. This lets you create SQL that has optional filters that can be of significant complexity and don't get included unless needed.

If statements

The **--IF / --ELSE / --END** syntax provides a way to test a value of an incoming parameter to conditionally construct part of an SQL statement. The following example illustrates this technique:

```
SELECT * from
  states
--IF=:state
WHERE code=:state
--ELSE
WHERE code='AL'
--END
ORDER BY NAME
```

In the above example the WHERE clause is only added to the SQL if there is a value specified for the report parameter **:state**. If no value for the :state parameter is provided, then the WHERE clause limits the selection to code of 'AL'. The ORDER BY clause is always included (since it is not part of the --IF / --ELSE / --END construct).

Switch Case Else

Alternatively the **--SWITCH / --CASE / --ELSE / --END** syntax provides a way to test for multiple values of an incoming parameter and conditionally construct part of an SQL statement. The following example illustrates this technique:

```
SELECT * from some_database_table
--SWITCH=:sort
--CASE=code
ORDER BY code
--CASE=total
ORDER BY total
--ELSE
ORDER BY name
--END
```

In the above example, if the report parameter **:sort** passed a value of **code**, it would create an **ORDER BY code** line in the SQL statement. But if it passed a value of **total** it would create an **ORDER BY total** line in the SQL statement. In all other cases, it would create an **ORDER BY name** line in the SQL statement.

Data Block Includes

Data blocks can be build from other data blocks. To accomplish this use the --INCLUDE directive as in this example:

```
--ACCESS=access content
```

```
SELECT * FROM (
--INCLUDE=users_by_state
) t
WHERE state=:state
```

Be aware however of these restrictions / limitations when including other data blocks:

- you can only include blocks within the same repository.
- the security of the including data block is the security that is used for the included data block (the --ACCESS line of the included data block is ignored).

Drupal Entities

When using the Drupal data driver, you can create data blocks that load Drupal entities instead of selecting columns from the database. To do this specify an **entity_type** and **entity_id** column in the --INFO section of your data block as illustrated in this example:

```
--ACCESS=access content
SELECT nid, type, title, uid, sticky, promote FROM node
WHERE type=:content_type AND status=1
ORDER BY title
--INFO
; This demonstrates loading node entities.
entity_type = node
entity_id = nid
```

This example illustrates loading a node, but any entity type (like users, or some custom entity type) may be loaded.

Raw Mode Queries

In some cases where Forena is being used to export data, memory consumption can be reduced by specifying a **return_type** option of **raw** in the data block.

The following example illustrates raw mode used in a Drupal query.

```
SELECT nid, type, title, uid, sticky, promote from node
--INFO
; Use raw mode
return_type = raw
```

Currently this option is only supported in the Drupal and PDO drivers. The one drawback for using raw mode is that all of the XPATH features are disabled for that data query as Forena will not prerender the result set into XML. This means that your iterators can only use an frx:foreach attribute of * and only column names may be used as token replacements in the query. You also will be unable to use XPATH evaluation expressions in your reports that use this feature.