****

**PViMS**

**Database Manual**

****

Contents

[1 Introduction 4](#_Toc455571559)

[1.1 Document Overview 4](#_Toc455571560)

[1.2 Purpose of the Document 4](#_Toc455571561)

[1.3 Audience 4](#_Toc455571562)

[2 Database Entities 5](#_Toc455571563)

[2.1 Work Plan Entities 5](#_Toc455571564)

[2.1.1 Entity Relationship Diagram 5](#_Toc455571565)

[2.1.2 Reference Entities 5](#_Toc455571566)

[2.1.3 Transaction Entities 6](#_Toc455571567)

[2.2 Dataset Entities 7](#_Toc455571568)

[2.2.1 Entity Relationship Diagram 7](#_Toc455571569)

[2.2.2 Reference Entities 8](#_Toc455571570)

[2.2.3 Transaction Entities 8](#_Toc455571571)

[2.3 Customisation Entities 12](#_Toc455571572)

[2.3.1 Entity Relationship Diagram 12](#_Toc455571573)

[2.3.2 Reference Entities 13](#_Toc455571574)

[2.3.3 Transaction Entities 14](#_Toc455571575)

[2.4 Patient Entities 17](#_Toc455571576)

[2.4.1 Entity Relationship Diagram 17](#_Toc455571577)

[2.4.2 Reference Entities 18](#_Toc455571578)

[2.4.3 Transaction Entities 18](#_Toc455571579)

[2.5 Encounter Entities 24](#_Toc455571580)

[2.5.1 Entity Relationship Diagram 24](#_Toc455571581)

[2.5.2 Transaction Entities 24](#_Toc455571582)

[2.6 Other Entities 27](#_Toc455571583)

[2.6.1 Reference Entities 27](#_Toc455571584)

[3 Database Security 34](#_Toc455571585)

[3.1 Database Owner and Role 34](#_Toc455571586)

[3.2 Database User and Role 39](#_Toc455571587)

[4 Database Maintenance 39](#_Toc455571588)

[4.1 Full Recovery Backups 39](#_Toc455571589)

[4.2 Log Backups 40](#_Toc455571590)

[4.3 Continuity Planning 41](#_Toc455571591)

[1.1.1 Introduction 41](#_Toc455571592)

[1.1.2 Objectives (Best Practices) 41](#_Toc455571593)

[1.1.3 Solution 43](#_Toc455571594)

[4.4 Index Rebuilding 45](#_Toc455571595)

[5 Database Backup 46](#_Toc455571596)

[5.1 Preparation 46](#_Toc455571597)

[5.2 Create Maintenance Procedure 46](#_Toc455571598)

[5.3 Update System Database Backup Script 48](#_Toc455571599)

[5.4 Update User Database Backup Script 49](#_Toc455571600)

[5.5 Update Batch Jobs 49](#_Toc455571601)

[5.6 Configure a daily task 50](#_Toc455571602)

[5.7 Configure a weekly task 53](#_Toc455571603)

[6 Database Restore 56](#_Toc455571604)

[6.1 Procedures 56](#_Toc455571605)

# Introduction

## Document Overview

This document focuses on the technical aspects of the PViMS database implementation. Due to the use of an Object Relational Mapping tool (Entity Framework), PViMS is effectively database agnostic. However, the focal RDBMS of this implementation is MS SQL Server and as such this document is prepared with SQL Server in mind.

## Purpose of the Document

The purpose of the document is to describe the technical implementation of the PViMS database component. This includes: -

* Entity relationship diagrams per core area
* Data dictionary for all entities
* Description of database security, owner vs user model
* Database maintenance
* Continuity planning

## Audience

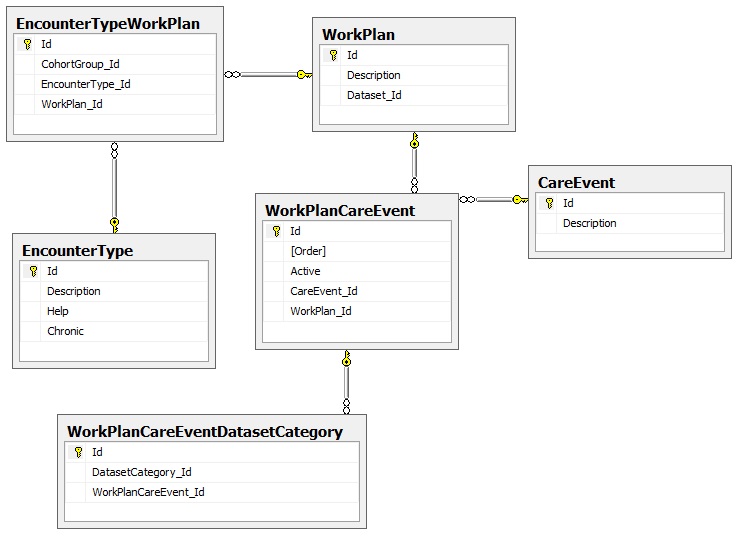
This document is targeted towards database administrators.

# Database Entities

## Work Plan Entities

|  |
| --- |
| This section contains all entities that govern the conceptual process that effectively allow the underlying dataset elements to be rendered to the end user. In essence, the base World Health Organisation record for TB cohort monitoring is managed through the implementation of a work plan. |

### Entity Relationship Diagram



### Reference Entities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | CareEvent | | | | |
| **Description** | Contains a list of care events. A care event is defined as an interaction between a health professional and a patient. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The name of the care event | Nvarchar | Max length 50 | N/A | No |
| **Entity** | EncounterType | | | | |
| **Description** | Contains a list of encounter types. An encounter type is defined as a reason a patient is attending a facility. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The name of the encounter type | Nvarchar | Max length 50 | N/A | No |
| Help | Additional help that supplements the description of the encounter type. | Nvarchar | Max length 250 | N/A | Yes |
| Chronic | Is this a chronic based encounter type. Dataset elements will be rendered based on the configuration of this encounter type. | Bit | N/A | N/A | No |

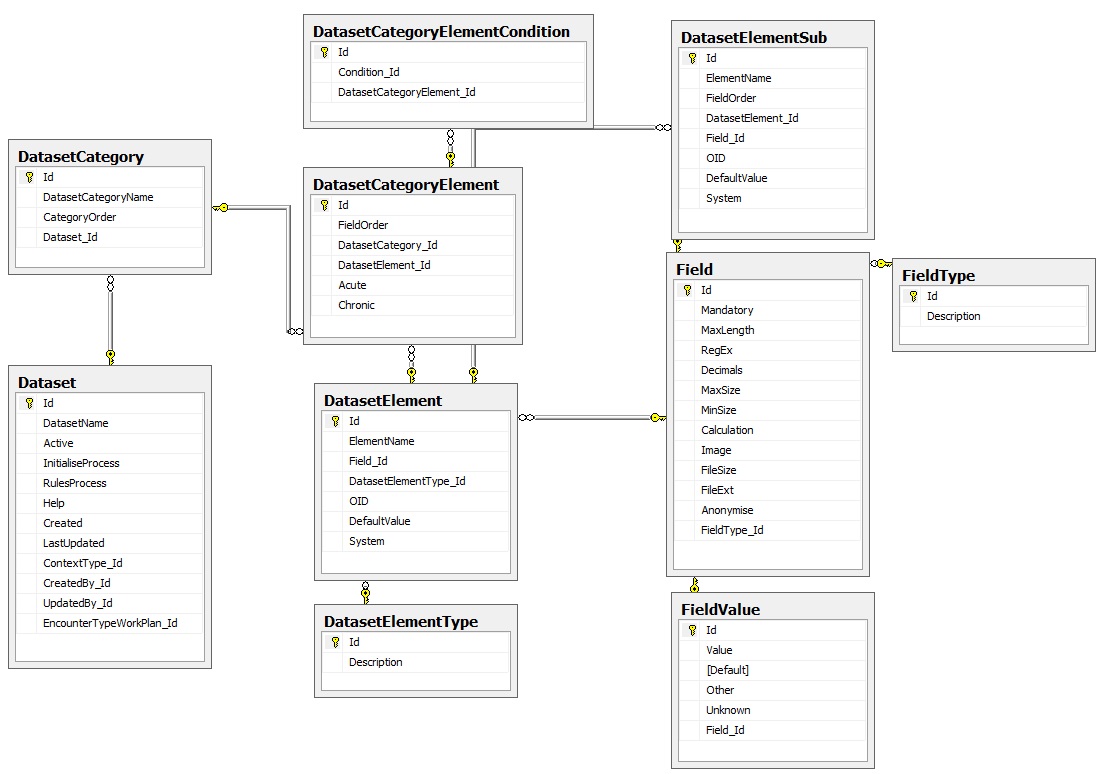
### Transaction Entities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | WorkPlan | | | | |
| **Description** | Contains a list of work plans. A work plan constitutes the fundamental approach a facility will undertake to treat a patient. For instance, a chronic repeat visit work plan can be designed to treat chronic patients who are visiting the facility to collect medication only. A work plan is made up of several care events. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The name of the work plan | Nvarchar | Max length 50 | N/A | No |
| Dataset\_Id | The associated dataset which contains all data elements that will be rendered if this work plan is selected | Integer | FK to Dataset | Foreign | Yes |
| **Entity** | EncounterTypeWorkPlan | | | | |
| **Description** | Bridging table that links an encounter type to a work plan. In theory more than work plan can be defined per encounter type, but practically the system implements this as a one to one relationship. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| CohortGroup\_Id | Is this work plan implementation of an encounter type linked to a cohort study. This work plan will only be activated for patients on a particular study. | Integer | FK to CohortGroup | Foreign | Yes |
| EncounterType\_Id | The encounter type the work plan is responsible for | Integer | FK to EncounterType | Foreign | No |
| WorkPlan\_Id | The work plan that defines the approach to the encounter | Integer | FK to WorkPlan | Foreign | No |
| **Entity** | WorkPlanCareEvent | | | | |
| **Description** | Bridging table that links a work plan to its underlying care events. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Order | The order of the care event in relation to the work plan. Care events run sequentially. Please note that this order is not currently enforced in PViMS. This is creating scalability for future development. | Integer |  | N/A | No |
| Active | Is this care event currently active for this work plan. | Bit | None | N/A | No |
| CareEvent\_Id | The care event that has been linked to the work plan | Integer | FK to CareEvent | Foreign | No |
| WorkPlan\_Id | The work plan that the care event has been linked to | Integer | FK to WorkPlan | Foreign | No |
| **Entity** | WorkPlanCareEventDatasetCategory | | | | |
| **Description** | The dataset categories that have been linked to a specific care event which is implemented for a specified work plan | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| DatasetCategory\_Id | The dataset category that has been linked to the work plan care event. The dataset category must be linked to the dataset that has been assigned to the work plan | Integer | FK to DatasetCategory | Foreign | No |
| WorkPlanCareEvent\_Id | The work plan care event that the dataset category has been linked to | Integer | FK to WorkPlanCareEvent | Foreign | No |

## Dataset Entities

|  |
| --- |
| This section contains all entities that govern the implementation of clinical data within the system. All clinical data can be distilled into the concept of a dataset element which forms the logical unit that comprises a dataset which in turn is rendered in accordance with a work plan entity. |

### Entity Relationship Diagram



### Reference Entities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | FieldType | | | | |
| **Description** | Contains a list of field types. A field type governs the type of data that is stored for a specified dataset element | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The name of the field type | Nvarchar | Max length 50 | N/A | No |
|  | Field Types   * Listbox * DropDownList * AlphaNumericTextbox * NumericTextbox * YesNo * Date * Table * System |  |  |  |  |
| **Entity** | DatasetElementType | | | | |
| **Description** | Contains a list of dataset element types. This will be used for future scalability. Primary definition of a data type is currently defined through the FieldType element. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The name of the dataset element type | Nvarchar | Max length 50 | N/A | No |

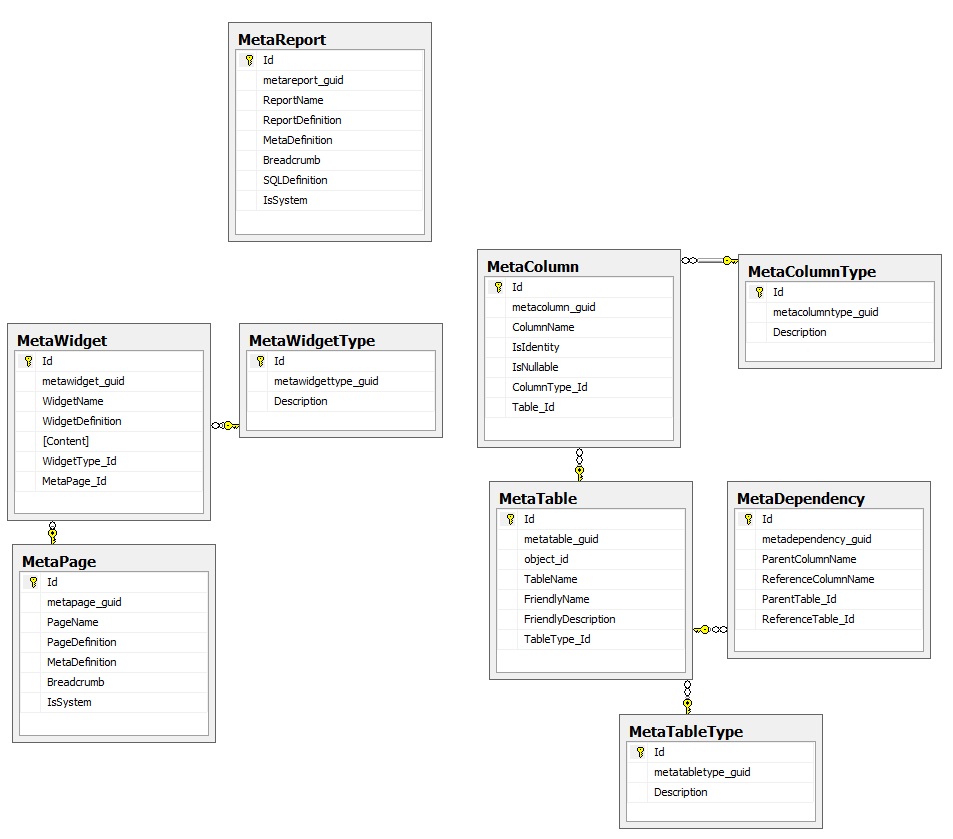
### Transaction Entities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | Dataset | | | | |
| **Description** | Contains a list of datasets. A dataset is effectively a collection of dataset categories which in turn includes a list of dataset elements per category. In other words, a dataset is a collection of dataset elements that should be captured in conjunction with each other. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| DatasetName | The name of the dataset | Nvarchar | Max length 50 | N/A | No |
| Active | Is this dataset currently active for data collection | Bit | None | N/A | No |
| InitialiseProcess | A c# method that must be called to initialize the dataset on creation | Nvarchar | Max length 100 | N/A | Yes |
| RulesProcess | A c# method that must be called to instantiate business rules against the dataset | Nvarchar | Max length 100 | N/A | Yes |
| Help | Additional description of the dataset | Nvarchar | Max length 250 | N/A | Yes |
| Created | AUDIT: Date record was created | Datetime | None | N/A | No |
| LastUpdated | AUDIT: Date record was last updated | Datetime | None | N/A | Yes |
| ContextType\_Id | The type of entity that this dataset associated to | Integer | FK to ContextType | Foreign | Yes |
| CreatedBy\_Id | AUDIT: The user that created this record | Integer | FK to User | Foreign | Yes |
| UpdatedBy\_Id | AUDIT: The user that last updated this record | Integer | FK to User | Foreign | Yes |
| EncounterTypeWorkPlan\_Id | The work plan that has been used to implement this dataset | Integer | FK to EncounterTypeWorkPlan | Foreign | Yes |
| **Entity** | DatasetCategory | | | | |
| **Description** | Contains a list of categories associated to a dataset. A dataset category is effectively a collection of dataset elements that are related to each other through the context of the category. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| DatasetCategoryName | The name of the category | Nvarchar | Max length 50 | N/A | No |
| CategoryOrder | The order of the category in relation to the dataset as a whole | Smallint | None | N/A | No |
| Dataset\_Id | The dataset the category is associated to | Integer | FK to Dataset | Foreign | No |
| **Entity** | DatasetCategoryElement | | | | |
| **Description** | Contains a list of elements associated to a category. Each record effectively corresponds to a clinical element that will ultimately be rendered to the end user. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| FieldOrder | The order of the element in relation to the category as a whole | Smallint | None | N/A | No |
| DatasetCategory\_Id | The dataset category that the element has been linked to. Each element can be linked to more than one category across multiple datasets. | Integer | FK to DatasetCategory | Foreign | No |
| DatasetElement\_Id | The element that is being linked to the dataset category | Integer | FK to DatasetElement | Foreign | No |
| Acute | Display element for acute encounter types (non chronic) | Bit | None | N/A | No |
| Chronic | Display this element for chronic encounter types | Bit | None | N/A | No |
| **Entity** | DatasetCategoryElementCondition | | | | |
| **Description** | Contains a list of chronic conditions that an element has been associated to (relevant if the chronic field has been set to true for the associated element) | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Condition\_Id | The chronic condition the patient must have for the element to be rendered | Integer | None | N/A | No |
| DatasetCategoryElement\_Id | The dataset category element that the chronic condition has been linked to | Integer | FK to DatasetCategoryElement | Foreign | No |
| **Entity** | DatasetElement | | | | |
| **Description** | Contains a list of unique dataset elements defined within the system. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| ElementName | The unique name of the dataset element (e.g. Weight) | Nvarchar | Max length 100 | N/A | No |
| Field\_Id | The field definition for this dataset element. Contains rules that define the data integrity for the element. | Integer | FK to Field | Foreign | No |
| DatasetElementType\_Id | The dataset element type | Integer | FK to DatasetElementType | Foreign | Yes |
| OID | The OID for the dataset element. Used for E2B extracts. | Nvarchar | Max length 50 | N/A | Yes |
| DefaultValue | The default value for an OID field in an E2B extract. | Nvarchar | Max length MAX | N/A | Yes |
| System | Is this a system defined field | Bit | None | N/A | No |
| **Entity** | DatasetElementSub | | | | |
| **Description** | Contains a list of unique dataset elements for the TABLE field type | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| ElementName | The unique name of the dataset element (e.g. Weight) | Nvarchar | Max length 100 | N/A | No |
| FieldOrder | The order of the element in relation to the table as a whole | Smallint | None | N/A | No |
| DatasetElement\_Id | The dataset element that the sub element is linked to | Integer | FK to DatasetElement | Foreign | No |
| Field\_Id | The field definition for this dataset element. Contains rules that define the data integrity for the element. | Integer | FK to Field | Foreign | No |
| OID | The OID for the dataset element. Used for E2B extracts. | Nvarchar | Max length 50 | N/A | Yes |
| DefaultValue | The default value for an OID field in an E2B extract. | Nvarchar | Max length MAX | N/A | Yes |
| System | Is this a system defined field | Bit | None | N/A | No |
| **Entity** | Field | | | | |
| **Description** | Contains the data integrity rules for the associated dataset element | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Mandatory | Is the dataset element mandatory. Applicable for all field types. | Bit | None | N/A | No |
| MaxLength | The maximum length of the element value. Applicable to the AlphaNumericTextbox. | Smallint | None | N/A | Yes |
| RegEx | Is there a regular expression that should be executed when validating the element. THIS IS CURRENTLY NOT IMPLEMENTED | Nvarchar | Max length 100 | N/A | Yes |
| Decimals | The number of decimals for a numerical value. Applicable to the NumericTextbox | Smallint | None | N/A | Yes |
| MaxSize | The maximum value for a numerical value. Applicable to the NumericTextbox | Decimal(18.2) | None | N/A | Yes |
| MinSize | The minimum value for a numerical value. Applicable to the NumericTextbox | Decimal(18.2) | None | N/A | Yes |
| Anonymise | If the element is included in any extract or report, should this field be anonymized | Bit | None | N/A | No |
| FieldType\_Id | The type of field that | Integer | FK to FieldType | Foreign | No |
| **Entity** | FieldValue | | | | |
| **Description** | Contains a list of drop down list values for DropDownList field types | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Value | The value of the drop down list | Nvarchar | Max length 100 | N/A | No |
| Default | Is this the default value for the drop down list | Bit | None | N/A | No |
| Other | Drop down list value for OTHER | Bit | None | N/A | No |
| Unknown | Drop down list value for UNKNOWN | Bit | None | N/A | No |
| Field\_Id | The field that the list of values is linked to | Integer | FK to Field | Foreign | No |

## Customisation Entities

|  |
| --- |
| This section contains all entities that govern the implementation of a dynamic framework that allows customization of reports and content within the publication portal. |

### Entity Relationship Diagram



### Reference Entities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | MetaWidgetType | | | | |
| **Description** | Contains a list of meta widget types. A meta widget type governs the type of widgets that can be added to the publication portal | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| GUID | A globally unique identifier for this widget type | UniqueIdentifier | None | N/A | No |
| Description | The name of the meta widget type | Nvarchar | Max length 50 | N/A | No |
|  | Meta Widget Types   * Report * Content.General * Content.FAQ * Content.CaseStudy |  |  |  |  |
| **Entity** | MetaColumnType | | | | |
| **Description** | Contains a list of meta column types. A meta column type governs the type of columns that can be defined in the reporting portal | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| GUID | A globally unique identifier for this widget type | UniqueIdentifier | None | N/A | No |
| Description | The name of the meta column type | Nvarchar | Max length 50 | N/A | No |
|  | Meta Column Types   * bigint * binary * bit * char * date * datetime * decimal * image * int * nchar * nvarchar * smallint * time * tinyint * uniqueidentifier * varbinary * varchar |  |  |  |  |
| **Entity** | MetaTableType | | | | |
| **Description** | Contains a list of meta table types. A meta table type governs the type of tables that can be defined in the reporting portal | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| GUID | A globally unique identifier for this widget type | UniqueIdentifier | None | N/A | No |
| Description | The name of the meta table type | Nvarchar | Max length 50 | N/A | No |
|  | Meta Table Types   * Core * CoreChild * Child * History * Lookup |  |  |  |  |

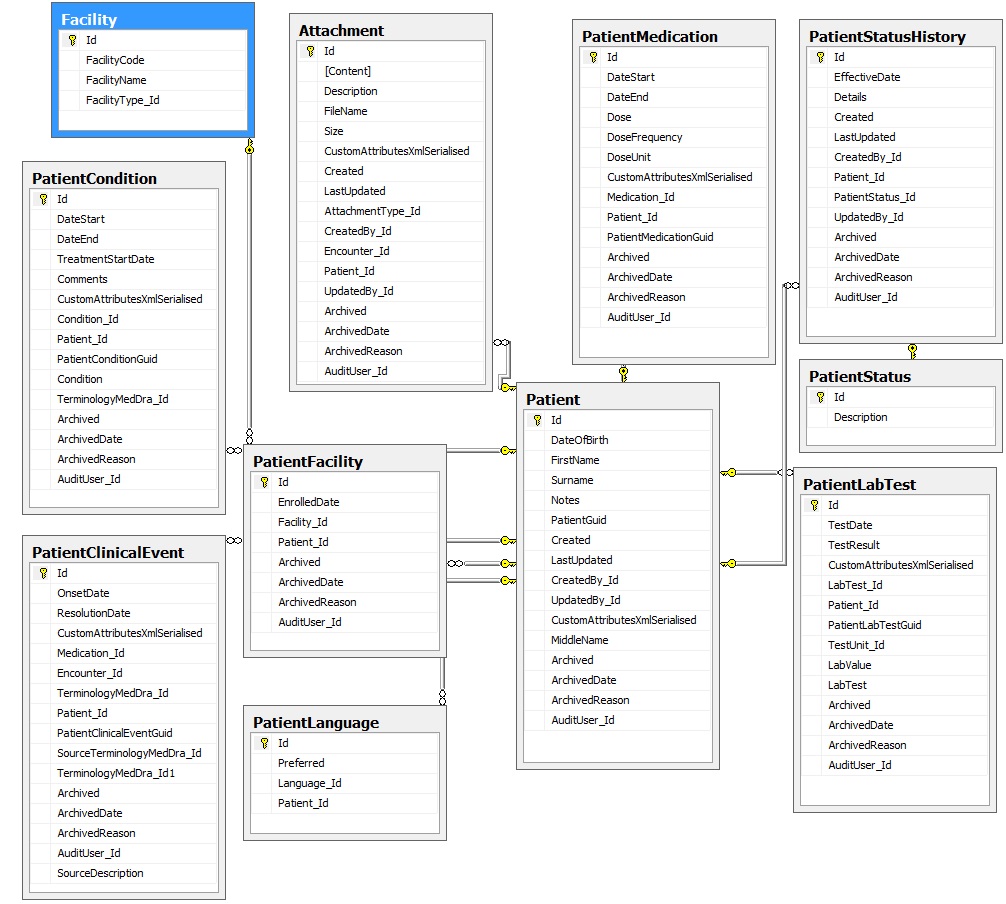
### Transaction Entities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | MetaReport | | | | |
| **Description** | Contains a list of custom defined reports | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| GUID | A globally unique identifier for this report | UniqueIdentifier | None | N/A | No |
| ReportName | The name of the report | Nvarchar | Max length 50 | N/A | No |
| ReportDefinition | A detailed description for the report | Nvarchar | Max length 250 | N/A | Yes |
| MetaDefinition | The configuration of the report | Nvarchar | Max length MAX | N/A | No |
| Breadcrumb | The menu structure for the report | Nvarchar | Max length 250 | N/A | No |
| SQLDefinition | The SQL statement needed to generate the report | Nvarchar | Max length MAX | N/A | No |
| IsSystem | Is this a system defined report. System reports may not be adjusted. | Bit | None | N/A | No |
| **Entity** | MetaTable | | | | |
| **Description** | Contains a list of system generated META tables | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| GUID | A globally unique identifier for this table | UniqueIdentifier | None | N/A | No |
| Object\_id | The SQL defined object ID for the source table | Integer | None | N/A | No |
| TableName | The name of the table | Nvarchar | Max length 50 | N/A | No |
| FriendlyName | Table friendly name | Nvarchar | Max length 100 | N/A | Yes |
| FriendlyDescription | A friendly description of the table | Nvarchar | Max length 250 | N/A | Yes |
| TableType\_Id | The type of table defined | Integer | FK to TableType | Foreign | No |
| **Entity** | MetaColumn | | | | |
| **Description** | Contains a list of system generated META columns | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| GUID | A globally unique identifier for this table | UniqueIdentifier | None | N/A | No |
| ColumnName | The name of the column | Nvarchar | Max length 50 | N/A | No |
| IsIdentity | Is this an identity column | Bit | None | N/A | No |
| IsNullable | Is this column nullable | Bit | None | N/A | No |
| ColumnType\_Id | The type of column defined | Integer | FK to ColumnType | Foreign | No |
| Table\_Id | The table the column is associated to | Integer | FK to Table | Foreign | No |
| **Entity** | MetaDependency | | | | |
| **Description** | Defines dependency between Meta generated tables | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| GUID | A globally unique identifier for this dependency | UniqueIdentifier | None | N/A | No |
| ParentColumnName | The name of the parent column in the relationship | Nvarchar | Max length 50 | N/A | No |
| ReferenceColumnName | The name of the reference column in the relationship | Nvarchar | Max length 50 | N/A | No |
| ParentTable\_Id | The parent table in the relationship | Integer | FK to MetaTable | Foreign | No |
| ReferenceTable\_Id | The reference table in the relationship | Integer | FK to MetaTable | Foreign | No |
| **Entity** | MetaPage | | | | |
| **Description** | Contains a list of system generated META pages in the publication portal | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| GUID | A globally unique identifier for this page | UniqueIdentifier | None | N/A | No |
| PageName | The name of the publication page | Nvarchar | Max length 50 | N/A | No |
| PageDefinition | A detailed description for the page | Nvarchar | Max length 250 | N/A | Yes |
| MetaDefinition | The configuration of the page | Nvarchar | Max length MAX | N/A | No |
| Breadcrumb | The menu structure for the report | Nvarchar | Max length 250 | N/A | No |
| IsSystem | Is this a system defined page. System pages may not be adjusted. | Bit | None | N/A | No |
| **Entity** | MetaWidget | | | | |
| **Description** | Contains a list of system generated META widgets per page in the publication portal | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| GUID | A globally unique identifier for this widget | UniqueIdentifier | None | N/A | No |
| WidgetName | The name of the publication widget | Nvarchar | Max length 50 | N/A | No |
| WidgetDefinition | A detailed description for the widget | Nvarchar | Max length 250 | N/A | Yes |
| Content | The content of the widget | Nvarchar | Max length MAX | N/A | No |
| WidgetType\_Id | The type of widget | Integer | FK to WidgetType | N/A | No |
| MetaPage\_Id | The page the widget is associated to | Integer | FK to MetaPage | N/A | No |

## Patient Entities

|  |
| --- |
| This section contains all entities that relate to the patient specifically. |

### Entity Relationship Diagram



### Reference Entities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | PatientStatus | | | | |
| **Description** | Contains a list of patient statuses | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The name of the patient status | Nvarchar | Max length 50 | N/A | No |
|  | Patient Statuses   * Active * Suspended * Stopped ART * Investigation * LTF * Stopped PMTCT * Transferred Out * Died |  |  |  |  |

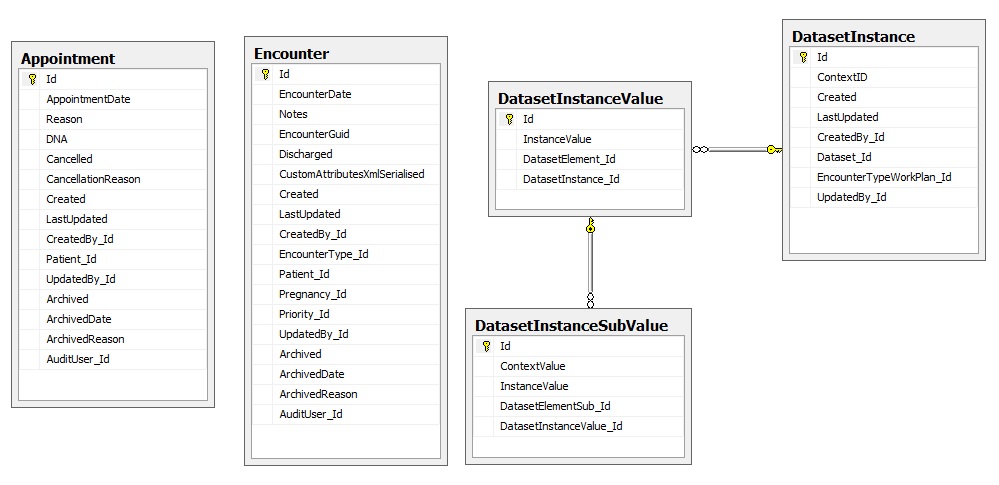
### Transaction Entities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | Patient | | | | |
| **Description** | The core patient demographics table | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| DateOfBirth | The date of birth of the patient | Date | None | N/A | No |
| FirstName | The first name of the patient | Nvarchar | Max length MAX | N/A | No |
| Surname | The surname of the patient | Nvarchar | Max length MAX | N/A | No |
| Notes | Generic patient notes | Nvarchar | Max length MAX | N/A | Yes |
| GUID | A globally unique identifier for this patient | UniqueIdentifier | None | N/A | No |
| Created | AUDIT: Date record was created | Datetime | None | N/A | No |
| LastUpdated | AUDIT: Date record was last updated | Datetime | None | N/A | Yes |
| CreatedBy\_Id | AUDIT: The user that created this record | Integer | FK to User | Foreign | Yes |
| UpdatedBy\_Id | AUDIT: The user that last updated this record | Integer | FK to User | Foreign | Yes |
| CustomAttributesXmlSerialised | Contains instance for all custom attributes | XML | None | N/A | Yes |
| MiddleName | The middle name of the patient | Nvarchar | Max length MAX | N/A | No |
| Archived | ARCHIVE: Is this record archived | Bit | None | N/A | No |
| ArchivedDate | ARCHIVE: The date the record was archived | DateTime | None | N/A | Yes |
| ArchivedReason | ARCHIVE: The reason the record was archived | Nvarchar | Max length 200 | N/A | Yes |
| AuditUser\_Id | ARCHIVE: Who archived the record | Integer | FK to User | Foreign | Yes |
| **Entity** | PatientCondition | | | | |
| **Description** | A list of conditions associated to the patient | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| DateStart | The start date of the condition | Date | None | N/A | Yes |
| DateEnd | The end date of the condition | Date | None | N/A | Yes |
| TreatmentStartDate | The date treatment was started on the condition | Date | None | N/A | Yes |
| Comments | Generic condition notes | Nvarchar | Max length 250 | N/A | Yes |
| CustomAttributesXmlSerialised | Contains instance for all custom attributes | XML | None | N/A | Yes |
| Patient\_Id | The patient the condition is associated to | Integer | FK to Patient | Foreign | No |
| GUID | A globally unique identifier for this patient condition | UniqueIdentifier | None | N/A | No |
| TerminologyMedDRA\_Id | The MedDRA term for the condition | Integer | FK to TerminologyMedDRA | Foreign | No |
| Archived | ARCHIVE: Is this record archived | Bit | None | N/A | No |
| ArchivedDate | ARCHIVE: The date the record was archived | DateTime | None | N/A | Yes |
| ArchivedReason | ARCHIVE: The reason the record was archived | Nvarchar | Max length 200 | N/A | Yes |
| AuditUser\_Id | ARCHIVE: Who archived the record | Integer | FK to User | Foreign | Yes |
| **Entity** | PatientMedication | | | | |
| **Description** | A list of medications associated to the patient | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| DateStart | The start date of the medication | Date | None | N/A | Yes |
| DateEnd | The end date of the medication | Date | None | N/A | Yes |
| Dose | The dose of the medication | Nvarchar | Max length 30 | N/A | Yes |
| DoseFrequency | The frequency of the dose | Nvarchar | Max length 30 | N/A | Yes |
| DoseUnit | The unit of the dose | Nvarchar | Max length 10 | N/A | Yes |
| CustomAttributesXmlSerialised | Contains instance for all custom attributes | XML | None | N/A | Yes |
| Medication\_Id | The medication the patient is taking | Integer | FK to Medication | Foreign | No |
| Patient\_Id | The patient the condition is associated to | Integer | FK to Patient | Foreign | No |
| GUID | A globally unique identifier for this patient condition | UniqueIdentifier | None | N/A | No |
| Archived | ARCHIVE: Is this record archived | Bit | None | N/A | No |
| ArchivedDate | ARCHIVE: The date the record was archived | DateTime | None | N/A | Yes |
| ArchivedReason | ARCHIVE: The reason the record was archived | Nvarchar | Max length 200 | N/A | Yes |
| AuditUser\_Id | ARCHIVE: Who archived the record | Integer | FK to User | Foreign | Yes |
| **Entity** | PatientClinicalEvent | | | | |
| **Description** | A list of clinical events associated to the patient | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| OnsetDate | The onset date of the clinical event | Date | None | N/A | Yes |
| ResolutionDate | The resolution date of the clinical event | Date | None | N/A | Yes |
| CustomAttributesXmlSerialised | Contains instance for all custom attributes | XML | None | N/A | Yes |
| TerminologyMedDRA\_Id | The MedDRA term for the clinical event that is selected by the PV specialist | Integer | FK to TerminologyMedDRA | Foreign | No |
| Patient\_Id | The patient the clinical event is associated to | Integer | FK to Patient | Foreign | No |
| GUID | A globally unique identifier for this patient clinical event | UniqueIdentifier | None | N/A | No |
| SourceTerminologyMedDRA\_Id | The MedDRA term for the clinical event that is selected by the clinician | Integer | FK to TerminologyMedDRA | Foreign | No |
| Archived | ARCHIVE: Is this record archived | Bit | None | N/A | No |
| ArchivedDate | ARCHIVE: The date the record was archived | DateTime | None | N/A | Yes |
| ArchivedReason | ARCHIVE: The reason the record was archived | Nvarchar | Max length 200 | N/A | Yes |
| AuditUser\_Id | ARCHIVE: Who archived the record | Integer | FK to User | Foreign | Yes |
| SourceDescription | The clinical event description as per the patient | Nvarchar | Max length 500 | N/A | Yes |
| **Entity** | PatientLabTest | | | | |
| **Description** | A list of lab tests associated to the patient | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| TestDate | The test date of the lab test | Date | None | N/A | Yes |
| TestResult | The result of the test | Nvarchar | Max length 50 | N/A | Yes |
| CustomAttributesXmlSerialised | Contains instance for all custom attributes | XML | None | N/A | Yes |
| LabTest\_Id | The lab test conducted | Integer | FK to LabTest | Foreign | No |
| Patient\_Id | The patient the lab test is associated to | Integer | FK to Patient | Foreign | No |
| GUID | A globally unique identifier for this patient lab test | UniqueIdentifier | None | N/A | No |
| TestUnit\_Id | The unit of the test | Integer | FK to LabTestUnit | Foreign | Yes |
| LabValue | The value of the lab test | Decimal(18.2) | None | N/A | No |
| Archived | ARCHIVE: Is this record archived | Bit | None | N/A | No |
| ArchivedDate | ARCHIVE: The date the record was archived | DateTime | None | N/A | Yes |
| ArchivedReason | ARCHIVE: The reason the record was archived | Nvarchar | Max length 200 | N/A | Yes |
| AuditUser\_Id | ARCHIVE: Who archived the record | Integer | FK to User | Foreign | Yes |
| **Entity** | PatientFacility | | | | |
| **Description** | A list of facilities associated to the patient | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| EnrolledDate | The enrolled date into the facility | Date | None | N/A | No |
| Facility\_Id | The facility the patient is linked to | Integer | FK to Facility | Foreign | No |
| Patient\_Id | The patient the facility is associated to | Integer | FK to Patient | Foreign | No |
| Archived | ARCHIVE: Is this record archived | Bit | None | N/A | No |
| ArchivedDate | ARCHIVE: The date the record was archived | DateTime | None | N/A | Yes |
| ArchivedReason | ARCHIVE: The reason the record was archived | Nvarchar | Max length 200 | N/A | Yes |
| AuditUser\_Id | ARCHIVE: Who archived the record | Integer | FK to User | Foreign | Yes |
| **Entity** | PatientLanguage | | | | |
| **Description** | A list of languages associated to the patient | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Preferred | Is this the preferred language of the patient | Bit | None | N/A | No |
| Language\_Id | The language associated to the patient | Integer | FK to Language | Foreign | No |
| Patient\_Id | The patient the language is associated to | Integer | FK to Patient | Foreign | No |
| **Entity** | Attachment | | | | |
| **Description** | A list of attachments associated to the patient | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Content | The content of the attachment | Varbinary | Max length Max | N/A | No |
| Description | A friendly description of the attachment | Nvarchar | Max length 100 | N/A | Yes |
| FileName | The filename for the attachment | Nvarchar | Max length 50 | N/A | No |
| Size | The size of the attachment | Bigint | None | N/A | No |
| CustomAttributesXmlSerialised | Contains instance for all custom attributes | XML | None | N/A | Yes |
| Created | AUDIT: Date record was created | Datetime | None | N/A | No |
| LastUpdated | AUDIT: Date record was last updated | Datetime | None | N/A | Yes |
| CreatedBy\_Id | AUDIT: The user that created this record | Integer | FK to User | Foreign | Yes |
| UpdatedBy\_Id | AUDIT: The user that last updated this record | Integer | FK to User | Foreign | Yes |
| AttachmentType\_Id | The type of the attachment | Integer | FK to AttachmentType | Foreign | Yes |
| Patient\_Id | The patient the attachment is associated to | Integer | FK to Patient | Foreign | No |
| Archived | ARCHIVE: Is this record archived | Bit | None | N/A | No |
| ArchivedDate | ARCHIVE: The date the record was archived | DateTime | None | N/A | Yes |
| ArchivedReason | ARCHIVE: The reason the record was archived | Nvarchar | Max length 200 | N/A | Yes |
| AuditUser\_Id | ARCHIVE: Who archived the record | Integer | FK to User | Foreign | Yes |
| **Entity** | PatientStatusHistory | | | | |
| **Description** | A list of status updates associated to the patient | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| EffectiveDate | The effective date of the status change | Datetime | None | N/A | Yes |
| Details | Details associated to the status change | Nvarchar | Max length 100 | N/A | Yes |
| Created | AUDIT: Date record was created | Datetime | None | N/A | No |
| LastUpdated | AUDIT: Date record was last updated | Datetime | None | N/A | Yes |
| CreatedBy\_Id | AUDIT: The user that created this record | Integer | FK to User | Foreign | Yes |
| UpdatedBy\_Id | AUDIT: The user that last updated this record | Integer | FK to User | Foreign | Yes |
| PatientStatus\_Id | The status of the patient | Integer | FK to PatientStatus | Foreign | Yes |
| Patient\_Id | The patient the attachment is associated to | Integer | FK to Patient | Foreign | No |
| Archived | ARCHIVE: Is this record archived | Bit | None | N/A | No |
| ArchivedDate | ARCHIVE: The date the record was archived | DateTime | None | N/A | Yes |
| ArchivedReason | ARCHIVE: The reason the record was archived | Nvarchar | Max length 200 | N/A | Yes |
| AuditUser\_Id | ARCHIVE: Who archived the record | Integer | FK to User | Foreign | Yes |

## Encounter Entities

|  |
| --- |
| This section contains all entities that relate to the clinical data collected in context of a patient encounter. A patient’s longitudinal clinical history is effectively made up of a series of encounters. |

### Entity Relationship Diagram



### Transaction Entities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | Encounter | | | | |
| **Description** | A list of encounters associated to the patient | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| EncounterDate | The date of the encounter | Datetime | None | N/A | No |
| GUID | A globally unique identifier for this encounter | UniqueIdentifier | None | N/A | No |
| Discharged | Is the encounter closed | Bit | None | N/A | No |
| CustomAttributesXmlSerialised | Contains instance for all custom attributes | XML | None | N/A | Yes |
| Created | AUDIT: Date record was created | Datetime | None | N/A | No |
| LastUpdated | AUDIT: Date record was last updated | Datetime | None | N/A | Yes |
| CreatedBy\_Id | AUDIT: The user that created this record | Integer | FK to User | Foreign | Yes |
| UpdatedBy\_Id | AUDIT: The user that last updated this record | Integer | FK to User | Foreign | Yes |
| EncounterType\_Id | The type of encounter | Integer | FK to EncounterType | Foreign | Yes |
| Patient\_Id | The patient the encounter is associated to | Integer | FK to Patient | Foreign | No |
| Priority\_Id | The priority of the encounter | Integer | FK to Priority | Foreign | No |
| Archived | ARCHIVE: Is this record archived | Bit | None | N/A | No |
| ArchivedDate | ARCHIVE: The date the record was archived | DateTime | None | N/A | Yes |
| ArchivedReason | ARCHIVE: The reason the record was archived | Nvarchar | Max length 200 | N/A | Yes |
| AuditUser\_Id | ARCHIVE: Who archived the record | Integer | FK to User | Foreign | Yes |
| **Entity** | DatasetInstance | | | | |
| **Description** | A list of dataset instances | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| ContextID | The unique id of the associated entity | Integer | None | N/A | No |
| Created | AUDIT: Date record was created | Datetime | None | N/A | No |
| LastUpdated | AUDIT: Date record was last updated | Datetime | None | N/A | Yes |
| CreatedBy\_Id | AUDIT: The user that created this record | Integer | FK to User | Foreign | Yes |
| UpdatedBy\_Id | AUDIT: The user that last updated this record | Integer | FK to User | Foreign | Yes |
| Dataset\_Id | The dataset associated to the instance | Integer | FK to Dataset | Foreign | No |
| EncounterTypeWorkPlan\_Id | The encounter type work plan that resulted in the instantiation of the dataset | Integer | FK to EncounterTypeWorkPlan | Foreign | No |
| **Entity** | DatasetInstanceValue | | | | |
| **Description** | A list of clinical values per dataset instance | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| InstanceValue | The value of the associated clinical element | Nvarchar | Max length 10 | N/A | No |
| DatasetElement\_Id | The dataset element that defines the corresponding value | Integer | FK to DatasetElement | Foreign | No |
| DatasetInstance\_Id | The dataset instance that contains the collection of instance values | Integer | FK to DatasetInstance | Foreign | No |
| **Entity** | DatasetInstanceSubValue | | | | |
| **Description** | A list of clinical values per dataset element table | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| ContextValue | A globally unique identifier for this sub table row | UniqueIdentifier | None | N/A | No |
| InstanceValue | The value of the associated clinical element | Nvarchar | Max length 10 | N/A | No |
| DatasetElementSub\_Id | The dataset sub-element that defines the corresponding value | Integer | FK to DatasetElementSub | Foreign | No |
| DatasetInstanceValue\_Id | The dataset instance valuethat contains the collection of sub instance values | Integer | FK to DatasetInstanceValue | Foreign | No |
| **Entity** | Appointment | | | | |
| **Description** | A list of appointments associated to the patient | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| AppointmentDate | The date of the appointment | Datetime | None | N/A | No |
| Reason | The reason for the appointment | Nvarchar | Max length 250 | N/A | No |
| DNA | The patient did not arrive for their appointment | Bit | None | N/A | No |
| CancellationReason | The reason the appointment was cancelled | Nvarchar | Max length 250 | N/A | Yes |
| Created | AUDIT: Date record was created | Datetime | None | N/A | No |
| LastUpdated | AUDIT: Date record was last updated | Datetime | None | N/A | Yes |
| CreatedBy\_Id | AUDIT: The user that created this record | Integer | FK to User | Foreign | Yes |
| UpdatedBy\_Id | AUDIT: The user that last updated this record | Integer | FK to User | Foreign | Yes |
| Patient\_Id | The patient the appointment is associated to | Integer | FK to Patient | Foreign | No |
| Archived | ARCHIVE: Is this record archived | Bit | None | N/A | No |
| ArchivedDate | ARCHIVE: The date the record was archived | DateTime | None | N/A | Yes |
| ArchivedReason | ARCHIVE: The reason the record was archived | Nvarchar | Max length 200 | N/A | Yes |
| AuditUser\_Id | ARCHIVE: Who archived the record | Integer | FK to User | Foreign | Yes |

## Other Entities

|  |
| --- |
| This section contains a list of the remaining entities. |

### Reference Entities

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entity** | ContextType | | | | |
| **Description** | Contains a list of context types. A context type is defined as the type of entity a dataset will connect to | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The name of the context type | Nvarchar | Max length 50 | N/A | No |
|  | Context types:   * Encounter * Patient * Pregnancy * Global * PatientClinicalEvent * DatasetInstance |  |  |  |  |
| **Entity** | TerminologyMedDRA | | | | |
| **Description** | Contains a list of all MedDRA definitions | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| MedDraTerm | The MedDRA term | Nvarchar | Max length 100 | N/A | No |
| MedDraCode | The MedDRA code | Nvarchar | Max length 10 | N/A | No |
| MedDraTermType | The MedDRA term type | Nvarchar | Max length 4 | N/A | No |
|  | MedDRA Term Types:   * LLT * HLGT * SOC * PT * HLT |  |  |  |  |
| Parent\_Id | The parent of this MedDRA term | Integer | FK to TerminologyMedDRA | Foreign | Yes |
| **Entity** | Medication | | | | |
| **Description** | Contains a list of medication | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| DrugName | The name of the drug | Nvarchar | Max length 100 | N/A | No |
| Active | Is the medication active and available for use in the system | Bit | None | N/A | No |
| PackSize | The pack size of the medication | Integer | None | N/A | No |
| Strength | The strength of the medication | Nvarchar | None | N/A | No |
| CatalogNo | The catalog number of the medication | Nvarchar | Max length 10 | N/A | Yes |
| MedicationForm\_Id | The form of the medication | Integer | FK to MedicationForm | Foreign | Yes |
| **Entity** | MedicationForm | | | | |
| **Description** | Contains a list of medication forms. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The name of the medication form | Nvarchar | Max length 50 | N/A | No |
|  | Medication Forms:   * Tablets * Ear Drops * Suspension * Lozenges * Cream * Ointment * Gel * Oral Gel * Lotion * Vaginal Cream * Eye Ointment * Syrup * Suppository * Nose Drops * Eye Drops * Capsules * Soap * Inhaler * Spray * Ampule * Amber * Crystals * Drops * Infusion * Injection * Oil * Powder * Sachets * Shampoo * Solution * Strips * Vial * Wax * Unknown * Tab/Cap |  |  |  |  |
| **Entity** | LabTest | | | | |
| **Description** | Contains a list of lab tests | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Active | Is the lab test currently active for selection | Bit | None | N/A | No |
| Description | The name of the lab test | Nvarchar | Max length 50 | N/A | No |
|  | Lab tests:   * AFB Smear Result * Albumin * ALT (SGPT) * Amylase * AST (SGOT) * Audiometry * Blood Glucose * Chest X-Ray * CD4 Count * Creatinine Clearance * Culture Results * QTc interval * ESR * Glucose * Heart Rate * Haemoglobin * Hepatitis B Virus * Hepatitis C Virus * HIV Antibody * Lactic acid * Lipase * Neurological Exam * Platelet Count * Pregnancy Test * PR Interval * QRS Duration * QT Duration * RBC Count * Serum Creatinine (SCr) * Serum Potassium (K+) * Serum Magnesium (Mg++) * Serum Calcium (Ca++) * Smear * Total WBC * TSH * Visual acuity * WBC Count |  |  |  |  |
| **Entity** | LabTestUnit | | | | |
| **Description** | Contains a list of lab test units | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Active | Is the lab test unit currently active for selection | Bit | None | N/A | No |
| Description | The name of the lab test unit | Nvarchar | Max length 50 | N/A | No |
|  | Lab test unit:   * N/A * mg/dL * µmol/L * % * % hearing loss left ear * % hearing loss right ear * µg/dL * µg/L * beats per minute * breaths per minute * cavities * cells/mm 3 * g/dL * g/L * IU/L * kg/m 2 * mEq/L * mg/24 hr * min * mL/min * mm Hg * mm/h * mmol/kg * mmol/L * mOsm/kg * ms * ng/dL * ng/L * ng/mL * ng/mL/hr * nmol/L * pg/mL * pH * pmol/L * sec * U/L * X 10 3 /mm 3 * X 10 6 /mm 3 * X 10 9 /L * μg/dL * μg/L * μmol/L * μU/L * μU/mL |  |  |  |  |
| **Entity** | Language | | | | |
| **Description** | Contains a list of languages. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The name of the context type | Nvarchar | Max length 20 | N/A | No |
| **Entity** | Facility | | | | |
| **Description** | Contains a list of facilities. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| FacilityCode | The unique code of the facility | Nvarchar | Max length 10 | N/A | No |
| FacilityName | The name of the facility | Nvarchar | Max length 50 | N/A | No |
| FacilityType\_Id | The type of facility | Integer | None | N/A | Yes |
| **Entity** | FacilityType | | | | |
| **Description** | Contains a list of facility types | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The name of the facility type | Nvarchar | Max length 50 | N/A | No |
|  | Facility types:   * Unknown * Hospital * CHC * PHC |  |  |  |  |
| **Entity** | AttachmentType | | | | |
| **Description** | Contains a list of attachment types | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The name of the attachment type | Nvarchar | Max length 50 | N/A | No |
| Key | The name of the attachment type | Nvarchar | Max length 4 | N/A | No |
|  | Attachment types:   * MS Word 2003-2007 Document * MS Excel 2003-2007 Document * MS Word Document * MS Excel Document * Portable Document Format * Image | JPEG * Image | JPEG * Image | PNG * Image | BMP |  |  |  |  |
| **Entity** | User | | | | |
| **Description** | Contains a list of system users. | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Email | The email address of the user | Nvarchar | Max length 256 | N/A | Yes |
| EmailConfirmed | Is email address confirmed | Bit | None | N/A | No |
| PasswordHash | Encrypted user password | Nvarchar | Max length MAX | N/A | Yes |
| SecurityStamp | Invalidate existing sign ins if user details changed | Nvarchar | Max length MAX | N/A | Yes |
| PhoneNumber | The phone number of the user | Nvarchar | Max length MAX | N/A | Yes |
| PhoneNumberConfirmed | Is phone number confirmed | Bit | None | N/A | No |
| TwoFactorEnabled | Two forms of security validation | Bit | None | N/A | No |
| LockoutEndDateUtc | Locked out date | Datetime | None | N/A | Yes |
| LockoutEnabled | User has been locked out of account | Bit | None | N/A | No |
| AccessFailedCount | Number of times user has failed security validations | Integer | None | N/A | No |
| UserName | The user name of the user | Nvarchar | Max length 256 | N/A | No |
| FirstName | The first name of the user | Nvarchar | Max length MAX | N/A | Yes |
| LastName | The last name of the user | Nvarchar | Max length MAX | N/A | Yes |
| Active | User is currently active | Bit | None | N/A | No |
| CurrentContext | The portal the user is currently logged into | Nvarchar | Max length MAX | N/A | Yes |
| **Entity** | Role | | | | |
| **Description** | Contains a list of security roles | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Name | The name of the role | Nvarchar | Max length 30 | N/A | No |
| Key | The unique key for this role | Nvarchar | Max length 30 | N/A | No |
|  | Roles:   * Administrator * Registration Clerk * Data Capturer * Clinician * Analytics * Reporter * Publisher * Reporter Administrator * Publisher Administrator |  |  |  |  |
| **Entity** | Priority | | | | |
| **Description** | Contains a list of priorities | | | | |
| **Field Name** | **Field Description** | **Data Type** | **Field Constraints** | **Primary/**  **Foreign** | **Nullable** |
| Id | Unique auto-incremented seed for table | Integer | None | Primary | No |
| Description | The description of the priority | Nvarchar | Max length 50 | N/A | No |
|  | Priorities:   * Urgent * High * Medium * Low |  |  |  |  |

# Database Security

Most SQL Server databases have a number of users viewing and accessing data, which makes security a major concern for the administrator. The smart administrator will take full advantage of SQL Server security roles, which grant and deny permissions to groups of users, greatly reducing the security workload.

The first step in protecting your organisations data is determining which users need to view which data and then allowing access to only those users. For example, a registration clerk probably views demographic data while clinicians have access to clinical data.

You must also decide which users can change the data.

Accommodating a number of users could be a huge task if it weren’t for the Windows security model, which can easily accommodate many users with one role. A role defines what a user can and can’t do within a database, and multiple users can share the same role.

The benefits of using roles

Roles are a part of the tiered security model:

* Login security—Connecting to the server
* Database security—Getting access to the database
* Database objects—Getting access to individual database objects and data

First, the user must log in to the server by entering a password. Once connected to the server, access to the stored databases is determined by user accounts. After gaining access to an actual database, the user is restricted to the data he or she can view and modify.

The main benefit of roles is efficient management. Imagine a group of 1,000 users suddenly needing to view or modify new data. Using Windows security, you simply select an existing Windows group and assign it to a SQL Server role—instead of modifying 1,000 user accounts. To clarify, Windows groups consist of users with access to the Windows network, but SQL Server roles belong strictly to SQL Server. You’re simply granting permissions to SQL Server data and objects to valid Windows users.

This section will describe the roles in more detail below.

## Database Owner and Role

We'll start with the biggest role: db\_owner.

A member of the **db\_owner** role can do anything inside the database. Now there is a difference between a member of the db\_owner role and the dbo user. That difference is that if someone maps into the database as the dbo user, that person bypasses all security checks. An example of this is anyone who is a member of the sysadmin fixed server role. They map in as dbo. And as a result, they don't receive security checks.

If a user is not dbo but is a member of the db\_owner role, it does receive a security check. Of course, unless you've explicitly used DENY to block access, that user can do what he or she wants. The DENY stops them cold (it does not stop dbo). However, a member of the db\_owner role could remove the DENY, so effectively that person can do anything, even if you put roadblocks in place. Therefore, the db\_owner role should be given out only when necessary. Some applications will require it, which is a headache, but rarely will actual people need it (unless the application is connecting using their credentials). So you should be able to keep a reasonable tight control over this role. Like sysadmin, which is returned as a member of every fixed server role if you use the IS\_SRVROLEMEMBER() function, if you query for someone who is a member of the db\_owner role to determine if that user is a member of any other fixed database role, it will return true, even if the user is not explicitly a member of that role. For instance, a user who is a member of db\_owner but who is not a member of db\_securityadmin will still return a 1 if you execute the following query:

SELECT IS\_MEMBER('db\_securityadmin');

**Things to remember:**

* The db\_owner role allows a user to do anything within the database.
* DBAs who are already members of the sysadmin fixed server role come in as dbo and don't need this role explicitly granted to them.
* Normal users should not be a member of this role.
* Applications might require their user account to be a member of this role.
* Members of the db\_owner fixed database role can perform all configuration and maintenance activities on the database, and can also drop the database.
* db\_owner is a database role. Databases are owned by logins. Whatever login owns the database is aliased as dbo inside the database. You can change the database owner by using the sp\_changedbowner system stored procedure.
* Since it's little used, you should audit its membership for exceptions.

**db\_accessadmin**

The db\_accessadmin role also manages security, but handles access to the database, as the name implies. The db\_accessadmin role grants, denies, or revokes permission to enter the database for logins. Combined with db\_securityadmin, and you can completely manage security into and throughout the database. Like db\_securityadmin, though, access into the database is usually handled by DBAs. If they aren't members of the sysadmin fixed server role, they are members of the securityadmin fixed server role. As a result, this role should also be rarely used.

The db\_accessadmin role can allow access into or block access to the database for logins.

Again, since DBAs usually manage security and have an appropriate server-level role, this role is little used.

Normal users should not be a member of this role.

Applications should tend not to need this role.

This is another role you should audit for membership exceptions.

**db\_backupoperator**

The db\_backupoperator allows a member of the role to take backups of the database. However, it's only going to allow native backups, as in the standard backups through SQL Server itself. If you're using a third party product, chances are it is usually the methods which allow for high speed backups. Unfortunately, these methods require the login executing them to be a member of the sysadmin fixed server role. As a result, this role tends to be of limited usefulness. Add to it that you're backing up to a local drive, and it's rare to see a non-DBA having this level of access, even in a development system. Because of all these things, this is another role that is typically not used much.

The db\_backupoperator role allows a user to take backups of the database.

Most 3rd party backup utilities utilize methods that require sysadmin rights, which this doesn't give.

Another role that is little used because this functionality is usually handled by DBAs or a service account.

Normal users should not be a member of this role.

Applications should tend not to need this role, though I have seen exceptions.

**db\_ddladmin**

The db\_ddladmin is another powerful role because it allows a user to create, drop, or modify any objects within a database, regardless of who owns it. So a user could alter a stored procedure owned by dbo, for instance. This role is sometimes given to developers on non-production systems as they built custom applications. However, there is typically no reason anyone should be a member of this role on a production database. One thing the db\_ddladmin does not do is allow the user to alter permissions on the objects. So a member of this role can create or modify the object, such as a stored procedure, but not alter the permissions on it unless he or she is the owner. So, for instance, a member of this role could create a stored procedure in a schema owned by dbo, but couldn't grant the ability to execute it.

The db\_ddladmin role can create, drop, and alter objects within the database, regardless of who the owner is.

The db\_ddladmin role cannot alter security.

It is not unusual to grant this role to developers in a non-production environment.

Normal users should not be a member of this role.

Applications should not need this role.

No one should normally be a member of this role on a production database.

**db\_datareader**

The db\_datareader role allows a user to be able to issue a SELECT statement against all tables and views in the database. DENY for a user (or a role the user is a member of) will still block the SELECT, however. But if there are no permissions set, whatsoever, the user will have the ability to SELECT against the table or view. The catch with this role is that the permission is implicit. That means if you query sys.database\_permissions, you will not see any permission granted, either to the db\_datareader role or directly to the user. Therefore, if you need to audit for everyone who has SELECT access to particular tables in a database, you'll have to query the membership of this group via the use of sp\_helprolemember:

EXEC sp\_helprolemember 'db\_datareader';

It is not unusual to see the db\_datareader role used in databases. It's an easy way to grant SELECT permissions to everything without having to worry about it. However, due to the fact that it uses implicit permissions, I prefer to create a user-defined database role and explicitly grant permissions. With that said, here are things to remember:

The db\_datareader role gives implicit access to SELECT against all tables and views in a database.

In SQL Server 2005 and up, an explicit DENY will block access to objects.

It is not unusual to see this role used in production for developers.

It is not unusual to see this role used in production for normal users.

Applications will occasionally need this role.

Creating a user-defined database role and explicitly defining permissions is still preferred over the use of this role.

**db\_datawriter**

The db\_datawriter role is like the db\_datareader role in that it gives implicit access to tables and views within a database. It also can be blocked by an explicit DENY for the user or for a role the user is a member of. Unlike db\_datareader, however, db\_datawriter gives INSERT, UPDATE, and DELETE permissions . Again, since the permission is implicit, you will not see these rights show up in sys.database\_permissions. And like with db\_datareader, you'll have to check the membership of this role to determine actual permissions in the event of an audit.

The db\_datawriter role gives implicit access to INSERT, UPDATE, and DELETE against all tables and views in a database.

In SQL Server 2005 and up, an explicit DENY will block access to objects.

Typically developer are not members of this role in production unless all users are.

While less common than with db\_datareader, it is not all that unusual to see this role used in production for normal users.

Applications will occasionally need this role.

Creating a user-defined database role and explicitly defining permissions is still preferred over the use of this role.

**db\_denydatareader**

Unlike the previous two roles, db\_denydatareader denies access. In this case, the db\_denydatareader is the same as having a DENY for SELECT on all tables and views in the database. Because DENY trumps everything else, this is not a role I've seen used frequently. If there are no permissions for a given user on an object, such as the user has no SELECT permissions on a table, then SQL Server blocks access. Therefore, if a user doesn't have SELECT permission on TableA, then the user cannot successfully issue a SELECT query against TableA. An explicit DENY is not needed. And since this affects all tables and views, that adds to the reason this database role is typically not used. And like db\_datareader and db\_datawriter, the DENY is implicit, meaning you'll have to query for membership in this role to determine who is affected.

The db\_denydatareader role is denied access to SELECT against any table or view in the database.

Typically this role is not used.

The DENY is implicit.

Creating a user-defined database role and explicitly defining permissions is still preferred over the use of this role.

**db\_denydatawriter**

Wrapping up our list of roles is db\_denydatawriter. The db\_denydatawriter has an implicit DENY on INSERT, UPDATE, and DELETE for all tables and views in the database. Again, this is not a role that sees much use, for the same reasons as db\_denydatareader.

The db\_denydatawriter role is denied access to INSERT, UPDATE, or DELETE against all tables and views in the database.

Typically this role is not used.

The DENY is implicit.

Creating a user-defined database role and explicitly defining permissions is still preferred over the use of this role.

## Database User and Role

All objects in a database are owned by a **user**. Users that are members of the db\_owner role, among other permissions, are allowed to create objects owned by dbo. If a user is not a member of db\_owner, but has some create permissions (e.g. Create Table), then any objects they create will be owned by the user that created them. You can change the ownership of an object using sp\_changeobjectowner system stored procedure.

Here is the list of the roles that can be assigned:

**db\_datawriter** Members of the db\_datawriter fixed database role can add, delete, or change data in all user tables.

**db\_datareader** Members of the db\_datareader fixed database role can read all data from all user tables.

The reason we only assign the above mentioned roles are because of security purposes. The database user should only be able to read and write to the database, all other privileges will belong to the the db\_owner.

# Database Maintenance

As with the majority of tasks and procedures in the IT world, there isn't an easy one-size-fits-all solution for effective database maintenance, but there are some key areas that nearly always need to be addressed. My top five areas of concern are (in no particular order of importance):

* Data and log file management
* Index fragmentation
* Statistics
* Corruption detection
* Backups

An unmaintained (or poorly maintained) database can develop problems in one or more of these areas, which can eventually lead to poor application performance or even downtime and data loss.

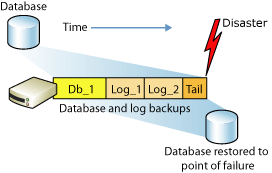
There are however some simple ways to mitigate the problems.

## Full Recovery Backups

The full recovery model uses log backups to prevent data loss in the broadest range of failure scenarios, and backing and restoring the transaction log (log backups) is required.

The advantage of using log backups is that they let you restore a database to any point of time that is contained within a log backup (point-in-time recovery). You can use a series of log backups to roll a database forward to any point in time that is contained in one of the log backups. Be aware that to minimize your restore time, you can supplement each full backup with a series of differential backups of the same data.

Assuming you can back up the active log after a disaster occurs, you can restore the database up to the point of failure without data loss. The disadvantages of using log backups are that they require storage space and increase restore time and complexity.



## Log Backups

The transaction log is a serial record of all the transactions that have been performed against the database since the transaction log was last backed up. With transaction log backups, you can recover the database to a specific point in time (for example, prior to entering unwanted data), or to the point of failure.

Minimally, you must have created at least one full backup before you can create any log backups. After that, the transaction log can be backed up at any time unless the log is already being backed up. We recommend that you take log backups frequently, both to minimize work loss exposure and to truncate the transaction log. Typically, a database administrator creates a full database backup occasionally, such as weekly, and, optionally, creates a series of differential database backup at a shorter interval, such as daily. Independently of the database backups, the database administrator backs up the transaction log at frequent intervals, such as every 10 minutes. For a given type of backup, the optimal interval depends on factors such as the importance of the data, the size of the database, and the workload of the server.

The sequence of transaction log backups is independent of the database backups. You make one sequence of transaction log backups, and then make periodic database backups that are used to start a restore operation. For example, assume the following sequence of events.

|  |  |
| --- | --- |
| **Time** | **Event** |
| 8:00 A.M. | Back up database |
| Noon | Back up transaction log |
| 4:00 P.M. | Back up transaction log |
| 6:00 P.M. | Back up database |
| 8:00 P.M. | Back up transaction log |
| 10:00 P.M. | Failure occurs |

## Continuity Planning

### Introduction

Securing and protecting your firm’s data is essential. Client files, important communications, and valuable work product often exist exclusively in digital format today, and thus a major data loss could have catastrophic professional and ethical ramifications.

### Objectives (Best Practices)

Whether you’re revisiting an existing backup strategy or seriously implementing one for the first time, this five step plan will help make sure you’re covering your bases.

#### STEP 1: ANALYZE

The first step in developing a data backup strategy for MSH is to analyse your current data usage. What data do you store, where do you store it, how often do you access it, and what are the risks and costs associated with losing that data? This is a challenging endeavour in the current computing environment, as data may be spread across numerous devices and services: computers, laptops, tablets, smartphones, firm servers and cloud computing platforms, etc.

Be sure to involve everyone at MSH in this exercise. You’ll probably be surprised to learn where employees are storing valuable data. Use the opportunity to review MSH’s overall handling of sensitive data. If, for example, sensitive documents is being sent to personal email addresses so employees can work from home or over the weekend, you may be facing serious security and confidentiality problems that will need to be addressed along with the backup issues.

In the end, your backup analysis should establish:

* What electronic data MSH currently uses;
* Where that data resides, including the specific vendor/host if it’s held outside of the office
* The approximate amount of data (e.g. 2TB);
* The sensitivity of data, both in terms of confidentiality and time (i.e. urgent matters).

#### STEP 2: PLAN

Once you have a firm grasp of the size and scope of the data you need to backup, you should begin developing an actual backup plan. Your backup plan should provide at least two levels of redundancy (a “belt and suspenders” approach), with both data redundancy (more than one backup of any given file) and geographic redundancy (backups housed in more than one geographic location).

The exact tools and software you use will vary widely depending on the size of the organisation and the complexity of your electronic efforts. In general, you should:

* Focus on business‐grade tools. Popular online backup tools geared towards consumers and less sensitive consumer data may not be appropriate in some instances.
* Plan for where you’ll be, not where you are. The quantity of data you need to backup is only going to increase as time goes by.
* Work with any outside companies that will hold your data. You will need to understand how safely and securely they are holding your data, how you access it, the business terms of your relationship, etc. (See “The Cloud: Due Diligence” below).
* You should also try to keep local copies of any data you store with a third‐party, and you should be sure the third‐party has their own backup strategy.
* Keep security at the front of your mind. Data needs to be backed up, but it also needs to be kept secure.

#### STEP 3: IMPLEMENT

It may seem obvious to say that the next step is to implement your plan, but this is unfortunately where many well‐intentioned backup strategies fall apart. Corners are cut both in cost and time, key efforts are entrusted to people who lack technology expertise, software and hardware is installed but never properly configured, and so forth.

Keep in mind that proper backup is critical to maintaining a healthy, stable, ethical business, and invest in its implementation appropriately. If your organisation lacks the technology know‐how to do this in-house, find an expert to help.

The keys to proper implementation:

Don’t cut corners‐‐follow through on the plan you developed in Step 2.

* That said, stay flexible—you may discover during implementation that you missed something. This is the time to correct the error.
* If necessary, get expert help to implement your backup system correctly.

#### STEP 4: TEST

It’s an all too common horror story: a business has a catastrophic data loss, turns to their backup system to recover the data, and only then discovers there’s a serious flaw in their backup strategy. Maybe data was backing up monthly rather than daily, or key files were being left out of regular backups entirely, or perhaps the backup hard drive itself has failed. There can be many causes, but the results are the same: your backup efforts come to nothing because you’ve failed to test your system and didn’t realize it wasn’t working properly.

As a best practice, you should test your backup solution immediately after implementation and routinely thereafter. Simulate real‐world disaster scenarios, from the major (total loss of a system) to the relatively minor (accidentally erasing a single file).

Not only does regular testing help identify problems in your backup setup, it also has the benefit of training your staff to quickly and efficiently recover files in the event that it’s necessary to do so. This means that if you ever experience a real computer loss and need to restore from your backups, you’ll be prepared to do so.

* Test your setup immediately to be sure it’s working as intended.
* Periodically re‐test your systems to ensure they’re functional and data is being backed up appropriately.
* Prepare to restore data quickly in the event of data loss to minimize impact on your firm.

#### STEP 5: REVIEW

Your data backup strategy will begin to be outdated almost immediately after you implement it. The reason is simple: technology advances at an incredibly rapid rate. New tools, new software, new data— each requires that you adjust your strategy.

* Conduct a full review of your strategy at least annually—more often if your setup is particularly complicated.
* Revisit your backup strategy anytime you make a significant technology investment.

### Solution

VP implements and maintains two utilities for automating the backup and maintenance of databases for Microsoft SQL Server. There is a standalone VB.NET console application and a [SQL stored procedure](http://www.codeplex.com/ExpressMaint/Release/ProjectReleases.aspx?ReleaseId=20228), that automate database backups and maintenance tasks via SMO (SQL Management Objects) or TSQL.

The jobs that can be done with the utility are as follows:

* Full Database Backup
* Differential Database Backup
* Log Backup
* Housekeeping of backup files
* Database Integrity Checks
* Database Index Rebuilds
* Database Index Reorganization
* Database Statistics Update

The databases should be copied to another location routinely in the event that there is a disaster. This could be sent offsite or pushed into the cloud depending on the strategy that the organisation wishes to employ. The most recent copy of the database can be timeously restored to ensure that there is no negative impact on service delivery.

In addition to the database backups there are other critical systems that need to be in tact to deliver a solution with optimal availability. We have outlined the treats and proposed, prevention, response and recovery strategies that need to be considered.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Critical System** | **Threat** | **Prevention Strategy** | **Response Strategy** | **Recovery Strategy** |
| Application Server | System Failure | Secure equipment room, backup server, UPS | Switch over to backup server, validate that UPS is running | Fix/replace primary server, fail back to primary server |
| Server Failure and system outage | Clustered Solution | Failover to backup cluster node | Fix issue and fail back to initial server. |
| System/ Application outage | Scale out virtual server design and load balancing | Direct traffic to additional server capacity | Balance out load and improve capacity. |
| Data Server | System Failure | Secure equipment room, backup server, UPS | Switch over to backup server, validate that UPS is running | Fix/replace primary server, fail back to primary server |
| Corrupted DB | To minimize data corruption, it is recommended to use a db analyse utility regularly. Prevent power outages. Ensure that there is enough disk space | You need to determine the extent of the corruption so that you can figure out what your options are in terms of restore or repair (or potentially just fail over and deal with the corruption offline). | Restore or repair DB |
| Data Loss | Implement RAID system, fault tolerance, and ensure that entire array is available. Check logs, activate automatic notifications so that alerts can be monitored | Restore application data and VM from backups | In the event that a disk has failed, replace disk so that entire array is available |
| Web Server | System Failure | Secure equipment room, backup server, UPS | Switch over to backup server, validate that UPS is running | Fix/replace primary server, fail back to primary server |
| IIS Service Failure | Implement backup server | Failover to backup server | Transfer the affected web server’s resources to a standby system, and then starts up the standby web server. If there is a communications or start-up failure, and local recovery is enabled, then stop and restart the affected web server to see if that corrects the problem. If the restart is unsuccessful, then transfer IIS services to the backup web server. |
| LAN | Downtime | Avoid or mitigate common local area network (LAN) disasters by identifying likely failure points, building redundant systems, and stocking replacement equipment. | Determine if there is a hardware failure | Replace faulty hardware |

## Index Rebuilding

Periodically (daily, weekly, or monthly) perform a database reorganization on all the indexes on all the tables in your database. This will rebuild the indexes so that the data is no longer fragmented.

Fragmented data can cause SQL Server to perform unnecessary data reads, slowing down SQL Server’s performance. It will also update column statistics.

If you do a reorganization on a table with a clustered index, any non-clustered indexes on that same table will automatically be rebuilt.

Database reorganizations can be done using the Maintenance Wizard, or by running your own custom script via the SQL Server Agent.

Questions that need to be considered:

* When should I rebuild the indexes in my relational database (e.g. SQL Server)?

You should rebuild indexes when they become highly fragmented by special events. For example, you perform a large, bulk load of data into an indexed table.

* Is there a case for rebuilding indexes on a regular basis?

There is no such case, but in general, doing Index Maintenance once in a week, over the weekend is the best practice to keep the environment stable.

# Database Backup

|  |
| --- |
| This section describes how to prepare an automated daily backup for the PViMS database. **Please note,** this section targets SQL Express specifically and as such focuses on the limitations inherent to SQL Express in implementing automated backups. SQL Express does not provide a maintenance component as is provided in the Enterprise edition and therefore does not provide the ability to automate tasks through a scheduling service. |

## Preparation

Please ensure you have access to the **Core.PViMS.Backup.zip** file provided as part of system implementation.

* Log on to the SQL server as a local administrator
* Create a folder on the SQL server – **X:\AppData\Backups** where X is the drive on the server that has been allocated for hosting the database
* Create a folder on the SQL server – **X:\AppData\Reports** where X is the drive on the server that has been allocated for hosting the database
* Extract contents of the **Core.PViMS.Backup.zip** zip file to the folder **X:\AppData**

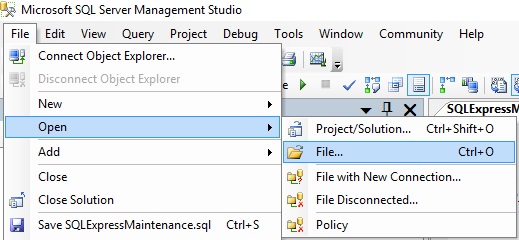
## Create Maintenance Procedure

|  |
| --- |
| Please note, the file **SQLExpressMaintenance.sql** creates a stored procedure that facilitates the actual backup of the database. |

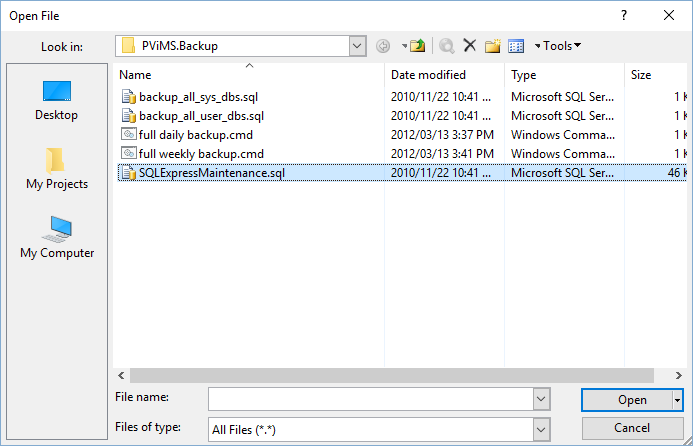
Open SQL Server Management Studio and login to the SQL Server you want to implement the automated backup process on using a SQL server user account with administration permissions.

Please ensure you have access to the Core.PViMS.Backup.zip file provided as part of system implementation. Extract contents of this zip file to a temporary folder on the SQL Server.

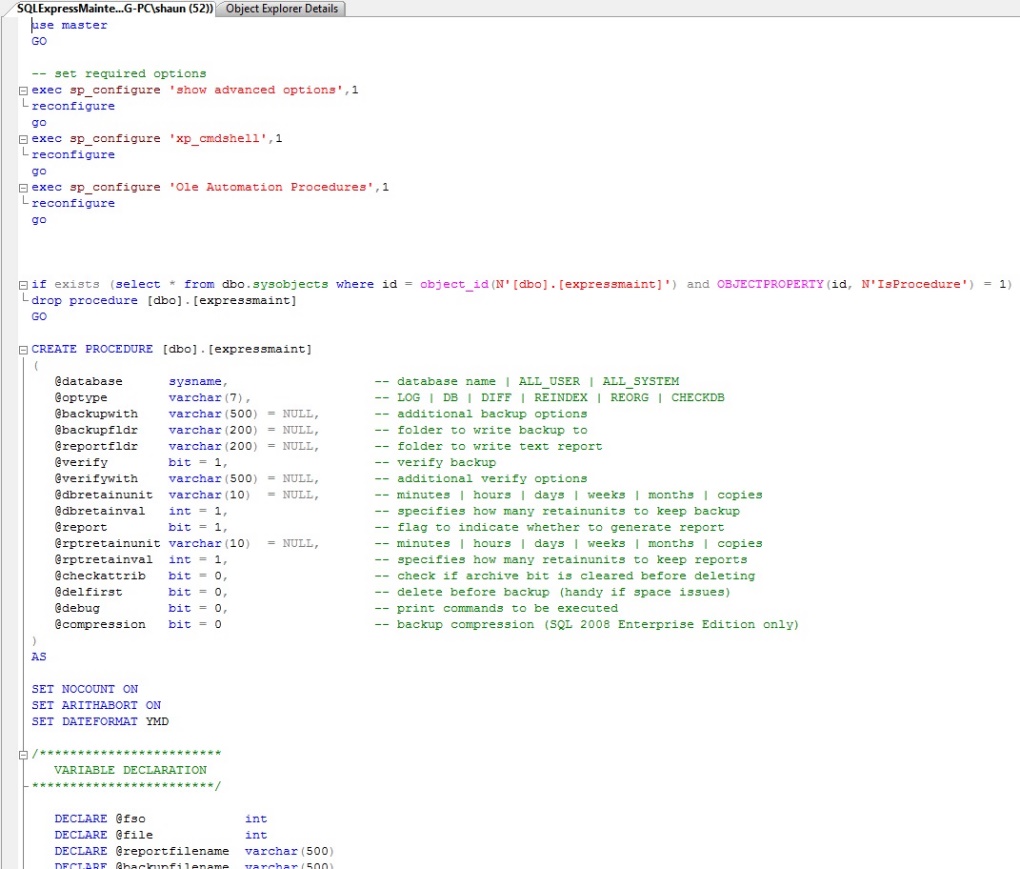
* Select the File 🡪 Open 🡪 File menu option

****

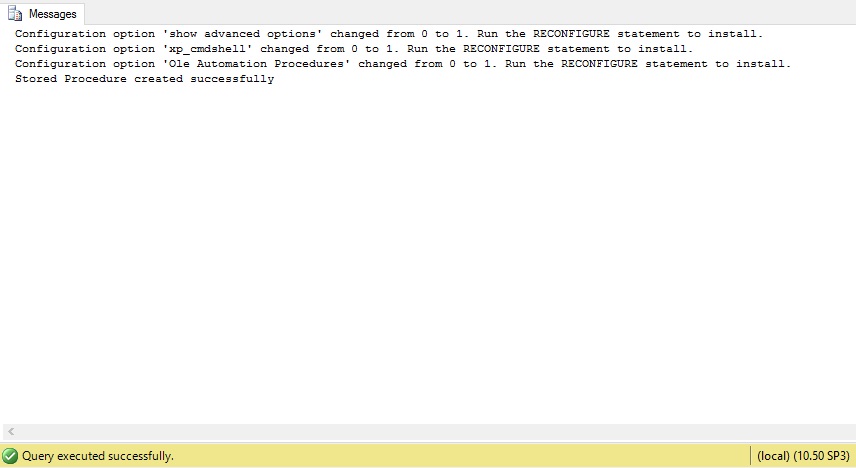
* Browse to the **SQLExpressMaintenance.sql** file which was extracted from the Core.PViMS.Backup.zip file.



* Click open to open this file
* Please note, you will now be able to view the content of this stored procedure in the right hand pane

****

* Press F5 to execute the script and thereby create the stored procedure

****

* Please ensure you receive a Query executed successfully message

## Update System Database Backup Script

|  |
| --- |
| Please note, the file **backup\_all\_sys\_dbs.sql** Is used to execute the maintenance stored procedure and contains all the configuration parameters for this process. The Master Database will be backed up as part of this process.  **Configurable Parameters:**  @backupfldr The path where all backup files will be created  @reportfldr The path where all backup reports will be created  @verify Verify the backup file on completion (0 = no, 1 = yes)  @dbretainval The number of backup files to store on the server |

* Browse to the folder **X:\AppData** and edit the file **backup\_all\_sys\_dbs.sql** using notepad
* Edit the @backupfldr parameter – ensure the specified path is correct (as set up per section 5.1)
* Edit the @reportfldr parameter – ensure the specified path is correct (as set up per section 5.1)
* Edit the @dbretainval parameter – confirm how many backup files you would like to store
* Save any modifications made

## Update User Database Backup Script

|  |
| --- |
| Please note, the file **backup\_all\_user\_dbs.sql** Is used to execute the maintenance stored procedure and contains all the configuration parameters for this process. The PViMS Database will be backed up as part of this process.  **Configurable Parameters:**  @backupfldr The path where all backup files will be created  @reportfldr The path where all backup reports will be created  @verify Verify the backup file on completion (0 = no, 1 = yes)  @dbretainval The number of backup files to store on the server |

* Browse to the folder **X:\AppData** and edit the file **backup\_all\_user\_dbs.sql** using notepad
* Edit the @backupfldr parameter – ensure the specified path is correct (as set up per section 5.1)
* Edit the @reportfldr parameter – ensure the specified path is correct (as set up per section 5.1)
* Edit the @dbretainval parameter – confirm how many backup files you would like to store
* Save any modifications made

## Update Batch Jobs

|  |
| --- |
| Please note, the files **full daily backup.cmd** and **full weekly backup.cmd** are used as part of the tasks scheduler to connect to the PViMS database and call the system or user database backup scripts. |

* Browse to the folder **X:\AppData** and edit the file **full daily backup.cmd** using notepad
* Edit the \*\*SERVERNAME\*\* parameter – capture the name of the SQL server
* Edit the \*\*INSTANCENAME\*\* parameter – capture the instance name of the SQL server
* Browse to the folder **X:\AppData** and edit the file **full weekly backup.cmd** using notepad
* Edit the \*\*SERVERNAME\*\* parameter – capture the name of the SQL server
* Edit the \*\*INSTANCENAME\*\* parameter – capture the instance name of the SQL server
* Save any modifications made

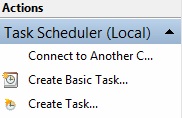
## Configure a daily task

|  |
| --- |
| The daily task will ensure the PViMS database is backed up on a daily basis |

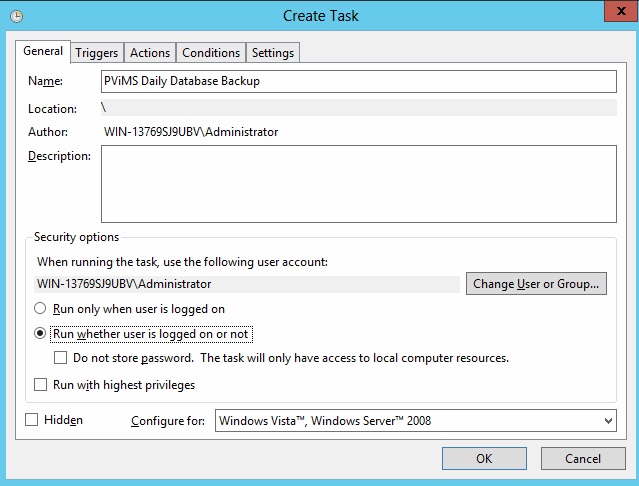
* Log on to the SQL server as a local administrator
* Open Server Manager 🡪 Tools 🡪 Task Scheduler



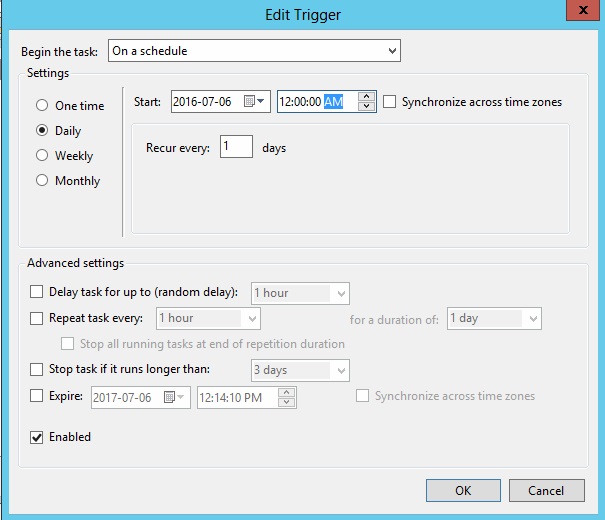
* Create a new task



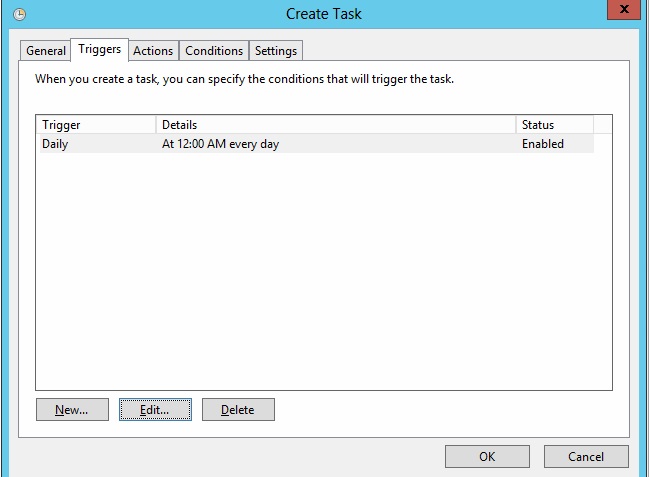
* General Settings
  + Enter a task name | **PViMS Daily Database Backup**
  + Ensure Run when user is logged on or not is selected



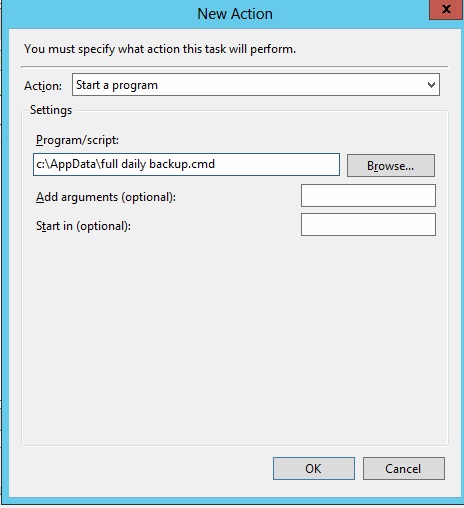
* Trigger Settings
  + Add a new trigger
    - Ensure task is configured to run daily
    - Ensure task is configured to start on the current date plus one day
    - Ensure task is configured to start at 12:00 AM
    - Under advanced settings, ensure task is enabled
    - Click OK to add trigger



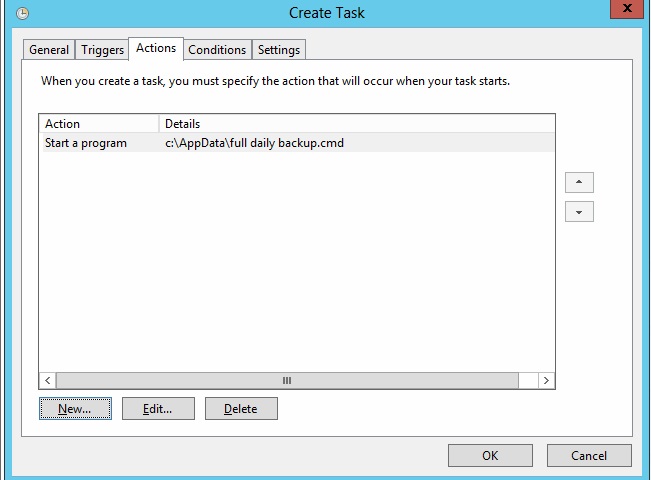
* + - Ensure trigger appears in trigger list



* Action Settings
  + Add a new action
    - Ensure Start a Program is select
    - Browse to the location of the **full daily backup.cmd** file in **X:\AppData**
    - Click OK to add action



* + - Ensure action appears in action list



* Click OK to add this task to the server. You may be prompted to confirm your administrator user name and password to add the task.

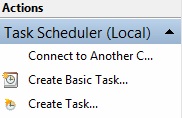
## Configure a weekly task

|  |
| --- |
| The weekly task will ensure the master database is backed up on a weekly basis |

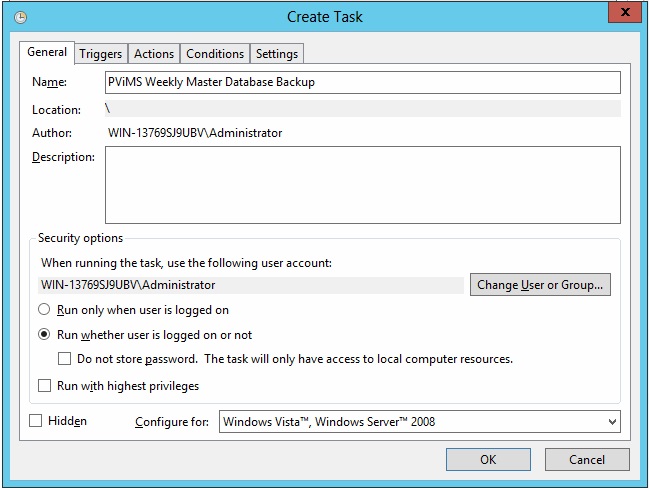
* Log on to the SQL server as a local administrator
* Open Server Manager 🡪 Tools 🡪 Task Scheduler



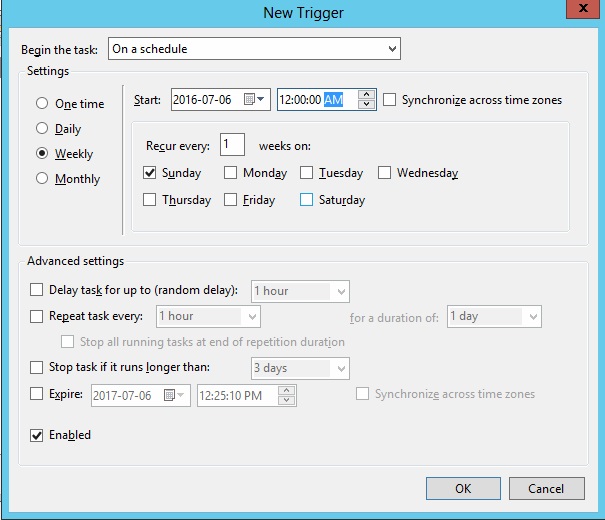
* Create a new task



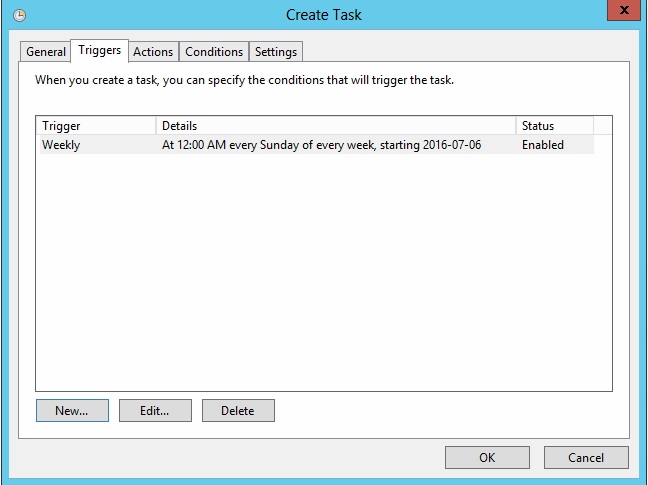
* General Settings
  + Enter a task name | **PViMS Weekly Master Database Backup**
  + Ensure Run when user is logged on or not is selected



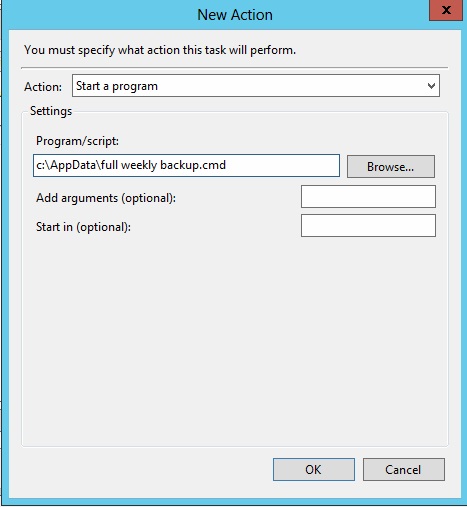
* Trigger Settings
  + Add a new trigger
    - Ensure task is configured to run weekly every Sunday
    - Ensure task is configured to start on the current date plus one day
    - Ensure task is configured to start at 12:00 AM
    - Under advanced settings, ensure task is enabled
    - Click OK to add trigger



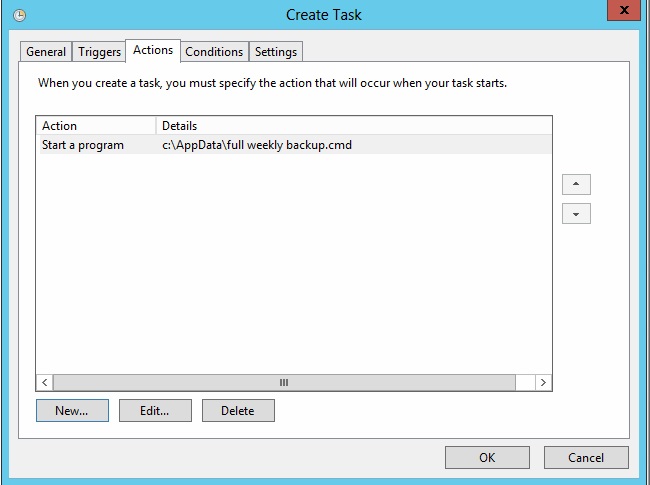
* + - Ensure trigger appears in trigger list



* Action Settings
  + Add a new action
    - Ensure Start a Program is select
    - Browse to the location of the **full weekly backup.cmd** file in **X:\AppData**
    - Click OK to add action



* + - Ensure action appears in action list



* Click OK to add this task to the server. You may be prompted to confirm your administrator user name and password to add the task.

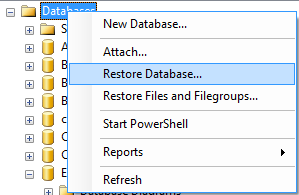
# Database Restore

|  |
| --- |
| This section describes how to restore a full database backup onto the PViMS SQL Server. |

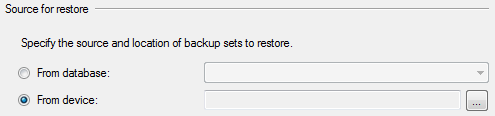
## Procedures

Open SQL Server Management Studio and login to the SQL Server you want to restore the database to. It is best to either login as a Windows Administrator or as the SQL ‘sa’ user.

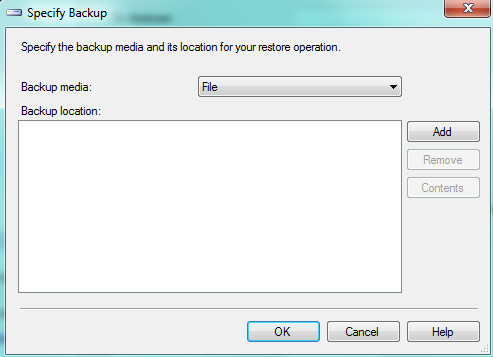
Once logged in, right click on the Databases folder and select ‘Restore Database’.



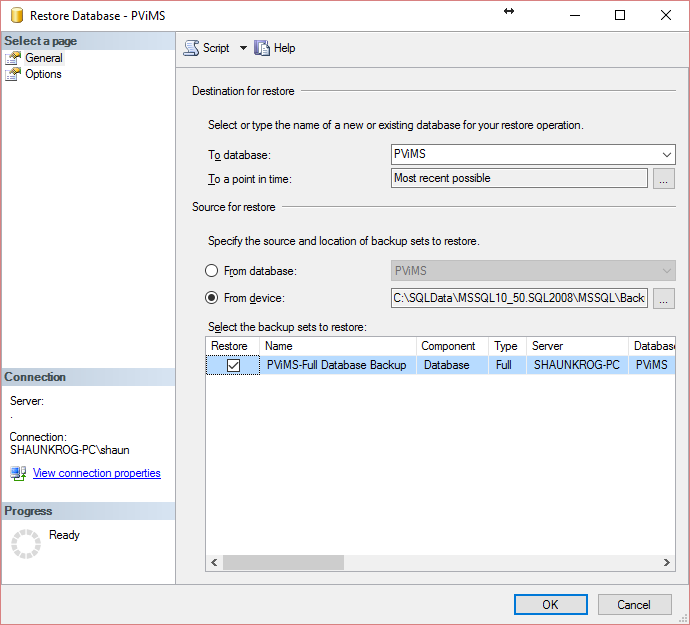
* Click the ellipses button next to ‘From device’ under the ‘Source for restore’ section.



* Set ‘File’ as the backup media and then click ‘Add’.



* Browse to the SQL backup (BAK) file you want to restore.
* In the Restore Database dialog, type or select the name of the database you want this backup restored to.
* If you select an existing database, this database will be replaced by the contents of the backup
* If you type a database name which does not currently exist in your SQL Server installation, a new database will be created.
* Next, select the restore point you want to use. Since a SQL backup file can hold multiple backups you may see more than one restore point listed.



At this point, enough information has been entered for the database to be restored. However, SQL backup files store information about where data files are copied so if there are any file system problems such as the destination directory not existing or conflicting data file names an error will occur. These problems are common when restoring a backup created on a different SQL Server installation.

To review and change the file system settings, click the Options page on the left in the Restore Database dialog.

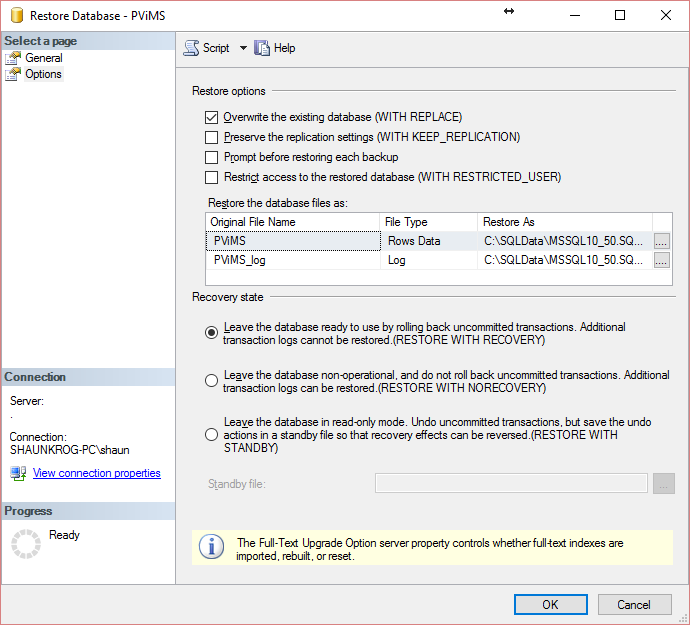


On the options page, you will want to make sure the ‘Restore As’ column points to valid folder locations (you can change them as needed). These files do not have to exist; however, the folder path must exist. If the respective files do exist, SQL Server follows a simple set of rules:

If the ‘To database’ (from the General page) matches the restore database backup (i.e. restoring to matching databases), the respective files will be overwritten as part of the restore.

If the ‘To database’ does not match the restore database backup (i.e. restoring to a different database), the ‘Overwrite the existing database’ will need to be checked for the restore process to complete. Use this function with caution as you can potentially restore database backup information on top of data files from a completely different database.

Generally, you can tell the databases differ based on the ‘Original File Name’ which is the internal name SQL Server uses to reference the respective files.



Once restore options are set, click Ok and the database will be restored.