CIM Database Model White Paper

CIM Version 2.10

Document Version 1.5 October 04, 2005

Abstract

The **DMTF Common Information Model (CIM)** is a conceptual information model for describing computing and business entities in enterprise and Internet environments. It provides a consistent definition and structure of data, using object-oriented techniques. The CIM Schema establishes a common conceptual framework that describes the managed environment.

The CIM database model describes the common management characteristics of a database environment. The model includes the common classes and properties that are independent of database organization or vendor implementation. The classes include the database system, which represents the application software aspects of a database environment; the common database, which represents a logical unit of inter-related data; and the database service, which represents the entity that performs tasks for a database.

This paper describes the background and motivation for defining the CIM database model, the contents of the model, and how it relates to other CIM schemas and database management standards. A use case is included that describes how the database model can be used for management purposes.

Notices

DSP0133 Status: Informational

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents for uses consistent with this purpose, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

Implementation of certain elements of this standard or proposed standard may be subject to third-party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third-party patent rights, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third-party patent rights, or for such party's reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

For information about patents held by third parties that have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit http://www.dmtf.org/about/policies/disclosures.php.

Table of Contents

1 Introduction

The **Common Information Model (CIM)** provides consistent information models with well-defined associations that capture management content for applications, systems, networks, devices, and other technology-focused management domains. CIM models establish a common conceptual framework that enables both hardware and software providers to consistently represent management information across vendor boundaries.

The benefits of CIM and the motivation behind extending the CIM information models into additional technology and vendor-agnostic domains are described in the *CIM Core Model White Paper*. The same motivations apply for the database management domain. When management data is unified across the enterprise, there is significant value to both the customer and the solution provider.

1.1 Overview

The database model that is described in this white paper extends the scope of CIM to include the database management domain. Three primary components are used to model a database environment:

1. The database system software
2. The common database entity
3. The database services

The concepts and associations defined by the database model are intended to be independent of any particular database type or vendor implementation. Future versions of the model may extend these classes to include additional entities and associations for specific types of environments, such as relational databases. Database software vendors may extend the database model to include vendor-specific content.

The primary focus of the DMTF Database Working Group in CIM version 2.7 was to model the database management entities and properties that are defined in the SNMP RDBMS MIB as specified in RFC 1679. In CIM version 2.8, the database model has been extended to model an initial set of classes that represent database storage.

1.2 Terminology

Table 1 shows the terminology that is used in this paper and database model. Readers of this document should be familiar with CIM, the existing models, and database technology.

Table 1. Terminology

Term	Definition
Database	A collection of interrelated data, treated as a unit, which is organized into one or more schemas.
Database Environment	A database system, one or more databases, and the services that control the administration, usage, monitoring, and maintenance of a database.
Database Server	The SNMP RDBMS term for the entity that provides access to the database. In CIM database model terminology, this entity is referred to as the database service.
Database Service	The entity that performs tasks for a database, such as providing user access to the information within the database. Database services may be implemented as one or more processes.
Database System	The entity that represents the application software aspects of a database environment.
Relational Database	A database in which the schemas are organized based on the relational model.
Schema	A collection of related database objects that reside in a database.
Storage Area	A logical container in which database information is stored.

2 Database Model

This section describes the classes and associations that are defined for the database model, and the relationships between the database model and other CIM models.

2.1 Background and Assumptions

The CIM Database Working Group was formed in 1997 to model the general management characteristics of a relational database environment. The initial model included the relational database software system, storage management, catalogs and schemas, users, transactions, and security. The result was a complex database model that was never completely specified.

The working group was reinstated in January of 2002. The goals and scope of the new working group were modified to include the content of RFC 1697, the SNMP RDBMS MIB specification. In addition, the working group agreed that the common database model content must be independent of any particular database organization. The database model supports hierarchical, relational, object-oriented, mixed, and other database implementations.

The initial model and white paper provided an excellent starting point for discussing how database entities could be represented in a model and database vendor-agnostic way. The RFC 1697 specification provided boundaries for the modeling effort. This specification identifies database entities and properties that are common across relational databases. The Database Working Group mapped these entities and properties to the CIM schema and abstracted the concepts to include any type of database.

2.1.1 Motivation for Developing a Database Model

A number of key factors contributed to the reinstatement of the Database Working Group. These factors are discussed in this section. The most important motivator is that CIM end users want a comprehensive model for all relevant management information. A common database model has been missing from CIM. The database model brings CIM a step closer to modeling end-to-end manageability from an enduser perspective. In addition, consistent management content that crosses vendor and platform boundaries enables more advanced management capabilities, such as cross-vendor diagnostics.

Management solution providers also benefit from the development of a common database model. The common model content provides a blueprint of database entities and properties that span vendor implementations. Solution providers can write management applications that have a consistent look-and-feel using the common content. Vendor-specific model extensions can also be integrated as needed.

Database vendors benefit from the development of a common database model. The CIM model allows the relationships between the database as a component of other systems, or the components required for a database environment, to be formally defined. This relationship benefits database users and helps to build a common platform for database manageability. Because CIM is object-oriented, database vendors can extend the model to include vendor-specific content by extending from the common aspects of the database model. From an end-user perspective, database vendor extensions are an integrated part of the complete model, rather than a separate, proprietary interface that requires special handling. As a result, database vendors can

write a single API that uses a consistent management information model across the complete software and hardware stack.

2.1.2 Assumptions

In CIM version 2.7, the database model focuses on the database entities and properties that have been defined in RFC 1697, as mapped to the CIM schema. The working group assumes that the content of this specification defines common database management content that spans database vendor implementations.

The Database Working Group has defined an initial set of common classes and associations in CIM 2.8 that model database storage. These classes use the work that was done in the Systems and Devices Working Group to map storage entities at the system level. The working group assumes that this lower level of mapping provides a common base for all users of the database model.

2.2 Conceptual Areas Addressed by the Model

The CIM database model defines management components for a database environment. Three major entities are modeled:

1. The database system represents the software application aspects of the database environment
2. The common database is a logical entity that represents the unit of inter-related, organized data
3. The database service represents the process or processes that perform tasks for the database, such as providing user access

In addition, a number of supportive classes represent configuration parameters, resources, and statistics. Figure 1 provides a conceptual representation of a database environment.

The database model is described in more detail in the following sections.

2.3 Database System

The database system represents the software application deployment aspects of the database environment. Database system software controls the organization, retrieval, storage, security, and maintenance of a database. It includes the software inventory information for the database environment, the software features that are meaningful from a user's perspective, and the software elements that are part of the database software. For background material on the concepts and schema details for application modeling, see the *Understanding the Application Management Model* white paper.

In Figure 2, a system administrator installs the database software onto the host operating system. The information associated with the installation process—the files and programs installed, how the files and programs are associated with the database system, the user features that are supplied by the database software, product and component versioning details—have all been defined as part of the application model. A CIM_DatabaseSystem is the logical entity that represents a manageable instance of the software aspects of a database environment in the database model.

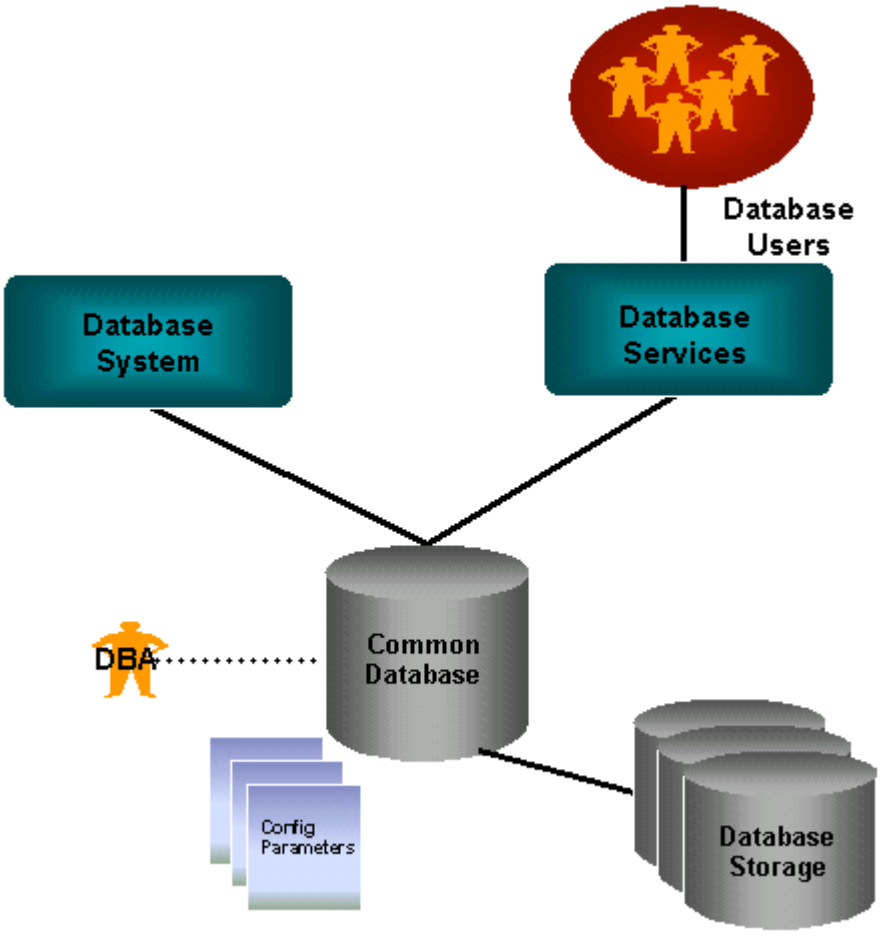

Figure 1. Conceptual Representation of the Database Model

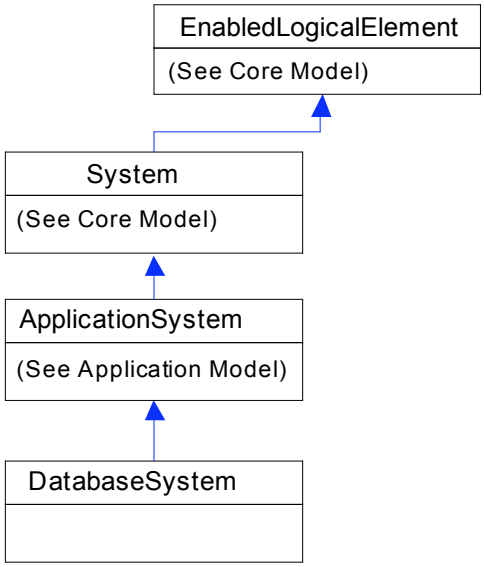

Figure 2. CIM_DatabaseSystem Class

From a modeling perspective, CIM_DatabaseSystem is a subclass of CIM_ApplicationSystem. In CIM 2.8, the Applications Working Group began modeling the runtime aspects of an application system using the CIM_RuntimeApplicationSystem subclass of CIM_ApplicationSystem. The database model does not directly use this subclass and its associations at this point in time. We expect that a future version of the database model will leverage relevant aspects of this portion of the application model.

One instance of CIM_DatabaseSystem is created for each database system installation that needs to be individually managed. For example, assume that a system administrator installs a database system into DB_HOME1. If the administrator installs another database product into DB_HOME1, a second instance of CIM_DatabaseSystem does not need to be created. Because the instance represents the existence of the database system as a manageable entity, the addition or removal of individual database products does not have an impact on the instances in the CIM_DatabaseSystem class.

If the system administrator installs a new database system into DB_HOME2, an instance needs to be created in CIM_DatabaseSystem to represent this new database system. Because the new database system is an individually named manageable entity, it would require its own CIM_DatabaseSystem instance.

The properties and associations for the CIM_DatabaseSystem class are inherited from its parents in the CIM schema hierarchy. The database model does not define additional properties that are specific to a database system. Having a separate class that represents a database system within the model has two important advantages.

1. The CIM_DatabaseSystem class groups the application systems that are database systems
2. Associations that are specific to database systems can be modeled

Table 2 describes the properties of CIM_DatabaseSystem. For more information, see the MOF and white papers for the class from which the property was inherited.

2.4 Common Database: CIM_CommonDatabase

The common database describes the vendor and database organization-agnostic properties of a database. It is a logical entity that names a specific, manageable organized body of related information. The SNMP RFC 1697 specification defines a database as an inter-related unit of data that is organized into a schema. The working group did not create a specific definition that mapped across vendor implementations.

The DMTF Database Working Group chose the class name CIM_CommonDatabase to represent the logical database entity. At this level of the database schema, instances of CIM_CommonDatabase span database organizations. The class supports hierarchical, relational, object-oriented, mixed, and other database model implementations. Each separately manageable database, whether it was created through a SQL command, vendor database creation utility, or was preexisting on the operating system, should have an instance in the CIM_CommonDatabase class.

Table 2. CIM_DatabaseSystem Properties

Property Name	Inherited From Class	Description
CreationClassName	CIM_System	The name of the class or subclass used when the instance was created. In this case, the value is CIM_DatabaseSystem. This property is used in conjunction with name to uniquely identify instances within the class.
Name	CIM_System	The name of the database system.
NameFormat	CIM_System	A string that identifies how the database system name was generated.
PrimaryOwnerContact	CIM_System	A string that identifies how the database system owner can be contacted (for example, email address, phone number, pager, and so on).
PrimaryOwnerName	CIM_System	The name of the primary owner of the system.
Roles	CIM_System	An array of strings that specifies the roles that the database system plays in the IT environment.
OperationalStatus	CIM_ManagedSystemElement	The status of the database system.
InstallDate	CIM_ManagedSystemElement	The date and time when the database system was first installed.
Description	CIM_ManagedElement	A longer textual description of the database system.
Caption	CIM_ManagedElement	A short textual description of the database system.
ElementName	CIM_ManagedElement	A user-friendly name for the database system.
EnabledStatus	CIM_EnabledLogicalElement	A status indicating whether the database system is enabled or in a disabled state.

Several properties are defined for a database at this level of the schema. A number of other properties are inherited. The following properties are defined for this class:

- **InstanceID**

  The InstanceID property opaquely identifies a specific instance of CIM_CommonDatabase. It must be unique within a namespace. Without this property, unique database naming would need to be managed by provider writers who extend from CIM_CommonDatabase.

  Because the contents of InstanceID are opaque, clients that reference this property should not require the contents to be written in a specific format.

- **SizeAllocated**

  The SizeAllocated property contains the estimated amount of disk space, in SizeUnits, that has been reserved for database use. The value of this property is not expected to

change frequently. The SizeAllocated property maps to the rdbmsDbInfoSizeAllocated variable in the SNMP RDBMS MIB.**SizeUnits**

The SizeUnits property identifies the units for the SizeAllocated property and for the SizeUsed property that are defined in the CIM_CommonDatabaseStatistics class. The mapping for the units is:

1. Bytes
2. Kilobytes
3. Megabytes
4. Gigabytes
5. Terabytes

The SizeUnits property maps to the rdbmsDbInfoSizeUnits variable in the SNMP RDBMS MIB.

- **LastBackup**

  The LastBackup property identifies the date and time when the latest complete or partial backup of the database was performed. If the database has never been backed up, then this property has no meaning. The value of this property should be set to all zeros in interval format if a backup operation has never been performed for the database. The LastBackup property maps to the rdbmsDbInfoLastBackup variable in the SNMP RDBMS MIB.

- **DatabaseVersion**

  The DatabaseVersion property identifies the version number for the database. If the version is not relevant for a specific vendor implementation, the value for this property must be set to NULL. The database version property maps to the rdbmsDbInfoVersion variable in the SNMP RDBMS MIB.

Table 3 provides a brief summary of the inherited properties. For more information, see the MOF where the referenced class is defined.

Table 3. Inherited Properties for CIM_CommonDatabase

Property Name	Inherited From Class	Description
Name	CIM_ManagedSystemElement	The database name.
OperationalStatus	CIM_ManagedSystemElement	The status of the database.
InstallDate	CIM_ManagedSystemElement	The date and time when the database was created.
Description	CIM_ManagedElement	A longer textual description of the database.
Caption	CIM_ManagedElement	A short textual description of the database.
ElementName	CIM_ManagedElement	A user-friendly name for the database.
EnabledStatus	CIM_EnabledLogicalElement	A status indicating whether the database is enabled or in a disabled state.

Figure 3 illustrates the CIM_AssociatedDatabaseSystem association, which represents the relationship between a database system where the database software has been deployed and the databases it controls.

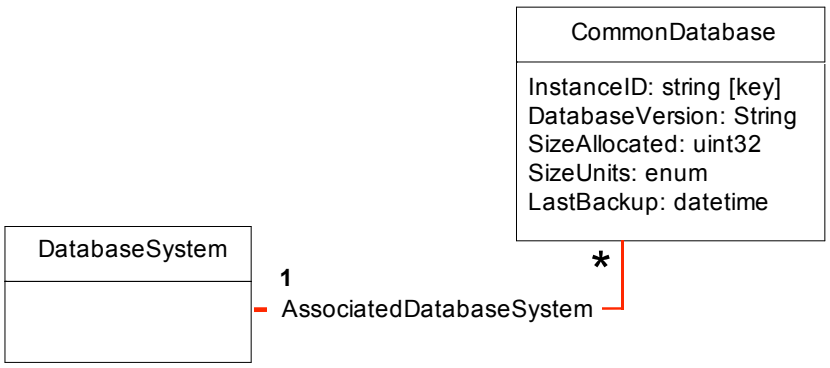

Figure 3. CIM_AssociatedDatabaseSystem Association

A single database system can be associated to zero or more databases. An instance of the database system class must exist for this association to be valid.

This association can be used to identify the database home for a specific common database instance. It also relates a specific common database instance to the database system that controls it. As a result, it is possible to list the common database instances that are controlled by a specific database system.

CIM_CommonDatabase relates a database to the user entity that administers the database. A database does not require an administrator. A database may have multiple administrators. Figure 4 illustrates the UML for the CIM_DatabaseAdministrator association.

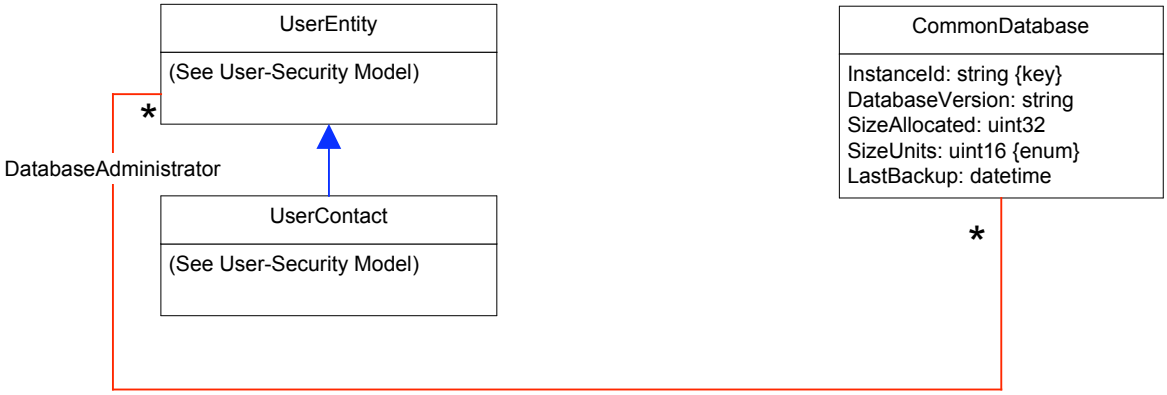

Figure 4. CIM_DatabaseAdministrator Association

CIM 2.8 introduces the CIM_UserContact concrete class. This class contains the detailed user contact information needed for a database administrator. This class was not defined in the CIM 2.7 model, so the Database Working Group could not use it for the DatabaseAdministrator association. In CIM2.8, the working group cannot change the reference in the database administrator association to use the CIM_UserContact class because it might break existing vendor implementations if the vendor has created its own concrete class.

To transition to the next major release of CIM, when CIM_UserEntity will become concrete and include the properties defined in CIM_UserContact, the Database Working Group suggests that

the CIM_UserContact class be used to create a vendor-specific concrete class in order to define database administrator contact information.

The CIM_DatabaseAdministrator association maps to the rdbmsDbContact variable in the SNMP RDBMS MIB.

2.4.1 Database Storage

The Database model in CIM 2.8 includes an initial model for database storage. The Database and SYSDEV working groups discussed the aspects of database storage that are specific to a database and the aspects that are applicable to file systems. General classes and associations were added to the System or Device model and the database-specific content was included in the Database model.

The goal for the initial model for database storage was to include the primary management entities that are common across database vendors that represent database storage, and to map the database specific entities down to the lower level system and device information. The database-specific content that is modeled includes both user storage content and any files that are created for use by the database.

A database storage area is a container for logically organizing and storing database information. A database may have multiple storage areas. A database storage area is a type of file system that is created and controlled by the database system. Figure 5 illustrates a number of the more important file system properties.

Many file system properties and associations are relevant to a database storage area.

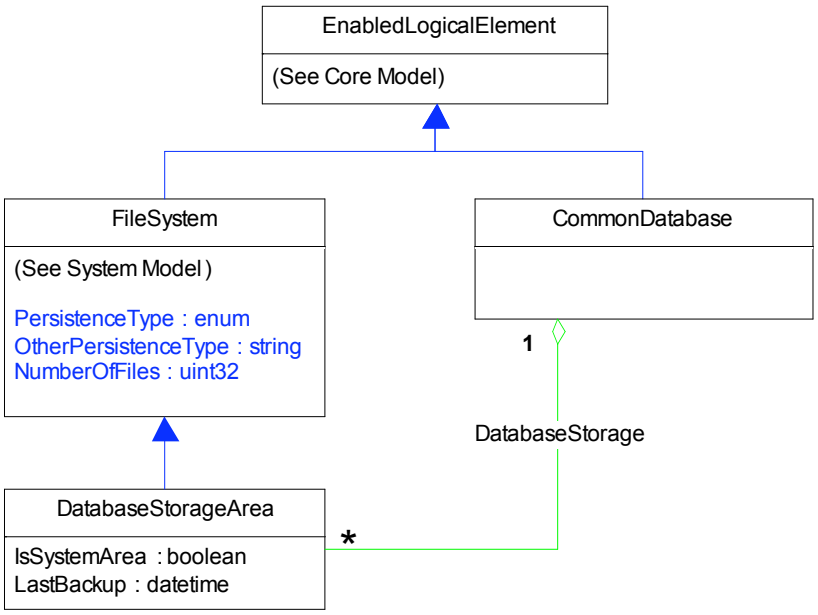

Figure 5. Database Storage

Table 4 provides a brief summary of the inherited file system properties. See the system and device models for more information on these properties and associations and the core model for information on the properties and associations inherited from higher-level classes.

Table 4. Inherited File System Properties

Property Name	Inherited From Class	Description
PersistenceType	CIM_FileSystem	An enumerated value representing the persistence characteristics of the information contained in the file system. Values include persistent, temporary, external, unknown, or 'Other'.
OtherPersistenceType	CIM_FileSystem	A string containing the value of the persistence type when the persistence type enumeration is 'Other'.
NumberOfFiles	CIM_FileSystem	The number of files contained in the file system.

In addition to the inherited properties, the following properties, which are specific to a database, are defined in the CIM_DatabaseStorageArea class:

- **IsSystemArea**

  The IsSystemArea property is a Boolean value that indicates whether the storage area is a system storage area. Some database systems may designate a storage area as a system storage area. A system storage area contains information that is owned by the database system, such as the data dictionary for the database. It is recommended that system and user storage areas should not be combined.

- **LastBackup**

  The LastBackup property is a timestamp that contains the date and time when the last backup of the DatabaseStorageArea successfully completed. Some database systems allow database storage areas to be individually backed up.

  The CIM_DatabaseStorage association can be used to identify the storage areas that belong to a particular database. A storage area cannot belong to more than one database.

Two levels of storage-related settings are defined in the Database model. The database-level settings that are defined in the CIM_CommonDatabaseSettingData class control aspects of database configuration, such as recovery. The CIM_DatabaseSegmentSettingData class defines settings that control how a database segment is created or extended. Instances within this class can also be associated to a storage area through the ScopedSetting association to specify the default settings for a database storage area. Figure 6 illustrates these two levels of storage area settings.

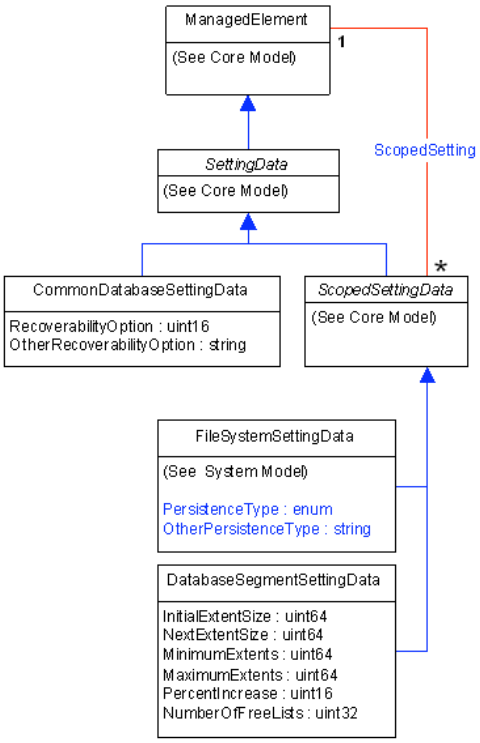

Figure 6. Database Storage Area Settings

Settings can be inherited from the file system level or can be database-system specific. The database-specific settings allow the following recoverability options for a storage area to be specified:

- **RecoverabilityOption**

  The RecoverabilityOption setting determines the level of recoverability for the database.

- **OtherRecoverabilityOption**

  The OtherRecoverabilityOption is a string that contains the value representing the recoverability option when the setting has been set to 'Other'.

The setting class has a related capability class that defines which settings are applicable for a given database.

Database systems may create special purpose files for recovery, transaction control, or to maintain state information for the database. These files may contain information that is relevant from a management perspective. They may require backup, special placement, or specific administration for routine database operation. From a file-system perspective, these database files are logical files, just like any other file that is created. The Database working group has not defined any properties that are specific to a database file at this time, so a subclass was not needed. Figure 7 illustrates these database files.

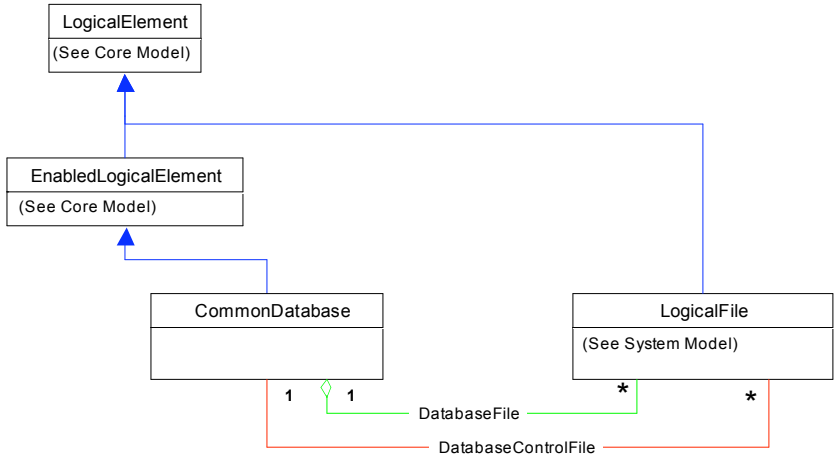

Figure 7. Database Files

The CIM_DatabaseFile association can be used to identify the database files that belong to a particular database, such as redo log files. A database file cannot belong to more than one database.

The CIM_DatabaseControlFile association can be used to identify the control files for the database. The CIM_DatabaseSegment class is the area of database storage that is modeled in CIM 2.8. Figure 8 illustrates the database segment.

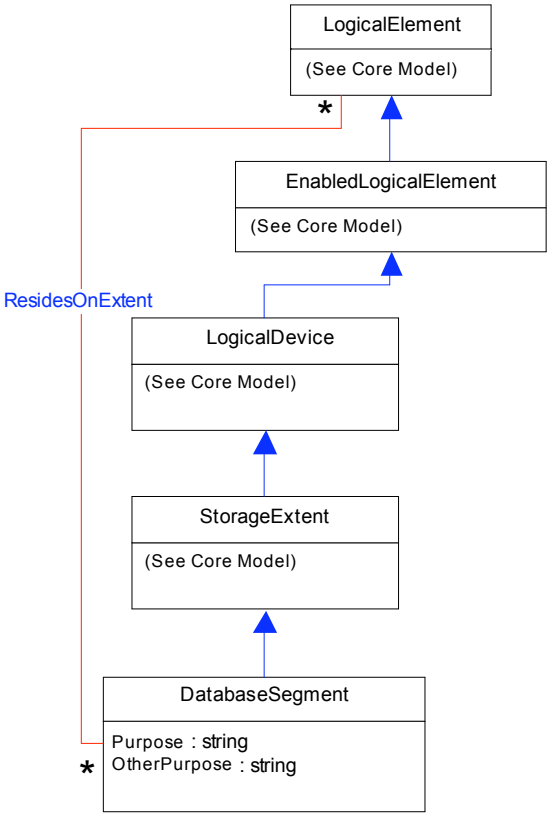

Figure 8. Database Segment

A database segment is a logical storage entity composed of one or more storage extents, each of which may have one or more database blocks. Database segments have various types depending on their purpose. Because a database segment is a type of storage extent, the working group extended from the CIM_StorageExtent class and overrode the purpose property to describe the database-specific extent usage detail.

This level of storage within the database model uses the existing classes and associations from the system model.
Note: The order of extents within a DatabaseSegment should be represented in the BasedOn dependency of CIM_StorageExtent using the BasedOn.OrderIndex attribute.

The following properties within the CIM_DatabaseSegment class override the inherited properties from CIM_StorageExtent:

- **Purpose**

  This property identifies how the segment is used by the database. For example, the database segment may contain user data information, index information, or temporary information.

- **OtherPurpose**

  This property contains the value representing the purpose when the purpose is set to 'Other'.

In addition to these properties, the working group modeled a number of properties as settings. They are used to configure the database segment. The properties would be redundant if included in the database segment class, because the current values are already reflected in the properties inherited from the storage extent level.

- **InitialExtentSize**

  This setting represents the size, in bytes, of the first extent to be allocated when a database segment is created.

- **NextExtentSize**

  This setting represents the size, in bytes, that is used for the next incremental extent for the database segment. A value of 0 indicates that the value for the next extent will be determined through the PercentIncrease setting.

- **PercentIncrease**

  This setting specifies the percentage by which the next incremental extent will increase over the previously allocated size of all extents for the database segment. A PercentIncrease value of 0 indicates that all incremental extents will be the same size, as specified by the NextExtentSize setting. This value is ignored and should be set to 0 if the NextExtentSize setting has a value other than 0.

- **MinimumExtentSize**

  This setting determines the total number of extents that are allocated when the database segment is created. It is used to create a large initial segment allocation at creation time. This makes it more likely that the space will be contiguous.

- **MaximumExtentSize**

  This setting places an upper limit on the number of extents that can be allocated for a database segment.

- **NumberOfFreeLists**

  This setting identifies the number of freelists that are defined for the database segment. This value is typically set to the expected number of concurrent inserts for the segment.

  A freelist is a list of the free blocks that are associated with a database segment. The freelist is used to determine which segments are eligible for accepting data when a new insert or update request is processed.

2.5 Database Service: CIM_DatabaseService

The database service describes the process or set of processes that performs tasks for the database. The database service is referred to as a database server in the RFC 1697 specification. By either name, this class defines the process or processes that coordinate user access to the database. Some database services perform other tasks, such as user authentication, authorization, concurrency control, data manipulation, integrity verification and data recovery.

Figure 9 illustrates the CIM_DatabaseService class.

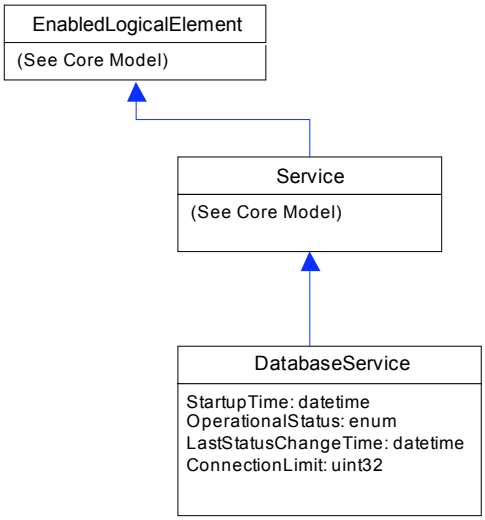

Figure 9. CIM_DatabaseService Class

The CIM_DatabaseService class contains instances of the manageable database service entities. Several properties are defined for a database at this level of the schema. A number of other properties are inherited. The properties defined for this class include:

- **StartupTime**

  The StartupTime property contains the date and time when the database service was last started. A value of all zeros indicates that the database service has never been started. This property maps to the rdbmsSrvInfoStartupTime variable in the SNMP RDBMS MIB.

- **OperationalStatus**

  The OperationalStatus property overrides the description that was inherited from CIM_ManagedSystemElement. The values and value map for the property are inherited from CIM_ManagedSystemElement.

  This property contains the operational status of the database service. These status values are:

  - "OK" means that the database service is operational and available for general use.
  - "Stopped" means that the service is unavailable and cannot be used.
  - "In Service" implies an administrative state of unavailability.
  - "Stressed" means that the database service is operating at a less than optimal level.
  - "Starting" means that the database service is in the process of becoming operational.

  Table 5 provides the status values that map to the RFC 1697 rdbmsSrvInfoOperStatus values.

Table 5. CIM Operational Status and RFC 1697 OperStatus Value Mapping

CIM OperationalStatus Value	RFC 1697 OperStatus Value
OK	UP
Stopped	DOWN
In Service	HALTED
Stressed	CONGESTED
Starting	RESTARTING

  Additional values are CIM OperationalStatus specific. See the definition of OperationalStatus in the CIM_ManagedSystemElement class for more information on the additional values.

- **LastStatusChangeTime**

  The LastStatusChangeTime property contains the date and time when the operational status of the database service last changed. This property maps to the rdbmsSrvInfoLastChange variable in the SNMP RDBMS MIB.

- **ConnectionLimit**

  This property contains the maximum number of active inbound connections that can be concurrently open for the database service. This property maps to the rdbmsSrvInfoMaxInboundAssociations variable in the SNMP RDBMS MIB.

Table 6 provides a brief summary of the inherited properties. See the MOF where the referenced class is defined for more information.

Table 6. Inherited Properties for DatabaseService

Property Name	Inherited From Class	Description
SystemCreationClassName	CIM_Service	The creation class name for the

Property Name	Inherited From Class	Description
		system where the database service is running.
SystemName	CIM_Service	The name of the system where the database service is running.
CreationClassName	CIM_Service	The name of the class or subclass used when the instance was created. In this case, the value is CIM_DatabaseService. This property is used in conjunction with name to uniquely identify instances within the class.
Name	CIM_Service	The name of the service.
StartMode	CIM_Service	This property indicates whether the database service startup is manual or automatic.
Started	CIM_Service	A Boolean value that indicates whether the database service is started or stopped.
PrimaryOwnerName	CIM_Service	The name of the primary owner of the database service. This name maps to the owner name content found in the rdbmsSrvContact property in the RDBMS SNMP MIB.
PrimaryOwnerContact	CIM_Service	The contact information for the primary owner of the database service. This information maps to the owner contact content found in the rdbmsSrvContact property in the RDBMS SNMP MIB.
InstallDate	CIM_ManagedSystemElement	The date and time when the database service was created.
Description	CIM_ManagedElement	A longer textual description of the database service.
Caption	CIM_ManagedElement	A short textual description of the database service.
ElementName	CIM_ManagedElement	A user-friendly name for the database service. In some cases, this may be the same value as the name property.
EnabledStatus	CIM_EnabledLogicalElement	A status indicating whether the database service is enabled or in a disabled state.

The CIM_ServiceAvailableToDatabase association relates database services to databases. A database can have zero or more database services that are available to service it. A database service can serve zero or more databases.

The relationship between a database and its database services is determined by the architecture of the vendor implementation. Because the model supports the case where a single database service can be used with multiple databases, properties are placed on the association to represent the relationship between a service and its availability to a specific database. Figure 10 illustrates the CIM_ServiceAvailableToDatabase association.

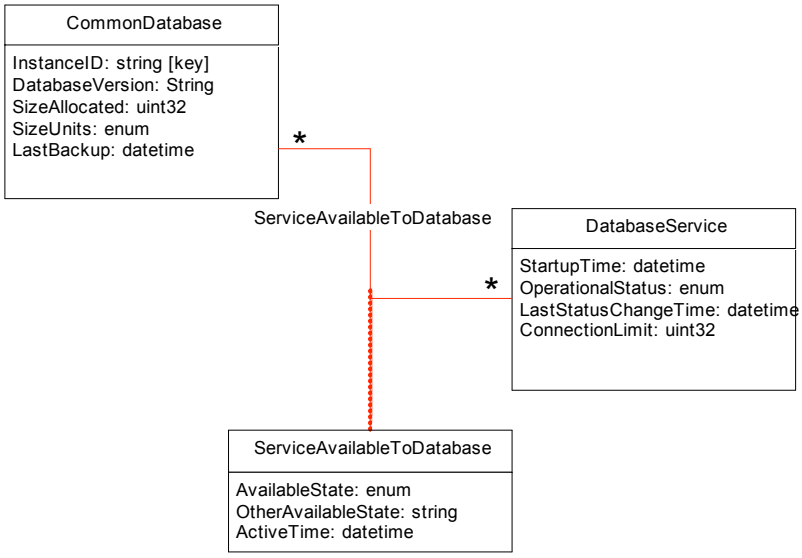

Figure 10. CIM_ServiceAvailableToDatabase Association

This relationship can be used to identify the number of database services that provide access to a specific database, the status of the service, and other useful information, such as the host where the service is running.

When a database service is actively serving a database, the following properties are defined:

- **AvailableState**

  The AvailableState property indicates the current state of a database service regarding its ability to access a specific database. The mapping from the numeric state representations is:

  1. **Other** – See OtherAvailableState for more details.
  2. **Active** – The service is actively using the database.
  3. **Available** – The service is waiting for a task to perform.
  4. **Restricted** – The service is less than completely available for use by the database.
  5. **Unavailable** – The service is not available for use by the database.

  The AvailableState property maps to the rdbmsRelState variable in the SNMP RDBMS MIB.

- **OtherAvailableState**

  The OtherAvailableState property contains information that describes the ability of the server to access the database when the AvailableState is 'Other'.

- **ActiveTime**

  The ActiveTime property contains the time when the database was made active by this service. If the AvailableState property is not 'Active', then the ActiveTime property must be set to 0. The ActiveTime property maps to the rdbmsRelActiveTime variable in the SNMP RDBMS MIB.

2.6 Database Parameters: CIM_DatabaseParameter

The database parameter class is an abstract class that represents the database and service configuration parameter settings. Configuration settings are name-value pairs that allocate specific resources, such as the number of database buffers for a database or service. These settings are collected into a profile that identifies the parameters for a specific database or service.

Vendor implementations can associate configuration settings with the database or the database service, and allow configuration settings for both the database and the services. The CIM database model is flexible enough to support all these scenarios.

In CIM version 2.7, the initial focus of the CIM Database Working Group was to map the contents from the SNMP RDBMS MIB. The RDBMS MIB defines database and service parameters as generic name/value pairs. The CIM convention is to define specific named properties within the class, rather than to create a generic parameter class.

To accommodate SNMP mapping, the CIM_DatabaseParameter class was created as an abstract class. A CIM_SNMPDatabaseParameter class was created as a subclass to provide the explicit SNMP mapping. In the future, the Database Working Group will investigate whether common database parameter properties can be specified at this level of the model. If parameters can be identified, a new subclass will be created that extends from CIM_DatabaseParameter to include the common properties.

Another advantage of making CIM_DatabaseParameter an abstract class is that it provides a class from which vendors can extend to provide vendor-specific named parameter classes. This is recommended, because vendor implementations from this class will help the working group refine the common properties that span vendor and database organizations in a future release of the schema. Figure 11 illustrates the CIM_DatabaseParameter class and associations.

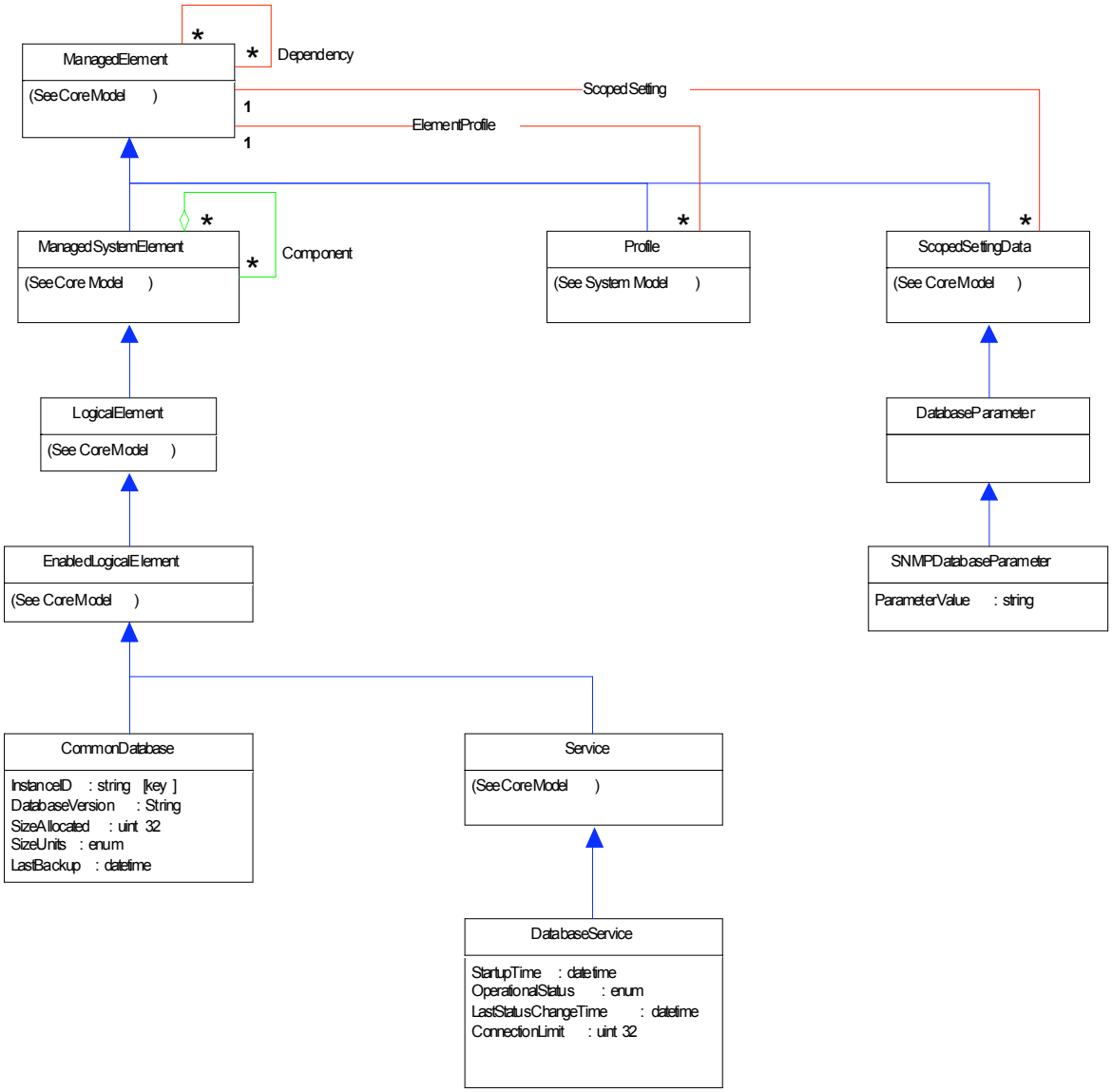

Figure 11. CIM_DatabaseParameter Class and Associations

The CIM_CommonDatabase and CIM_DatabaseService parameter settings are grouped using the CIM_ElementProfile association that is defined in the Core schema. This association can be used to identify the specific set of database parameter settings for a given database, or the set of database service parameter settings for a specific database service.

CIM_ScopedSetting is an inherited association that is used to relate a specific setting data instance to its managed element. In this case, the association relates settings to the appropriate database or service.

The CIM_DatabaseParameter class does not define any common database parameter properties at this time. A number of properties are inherited from its parent classes in the CIM schema. Table 7 provides a brief description of these properties. See the MOF where these properties are defined for more information.

Table 7. Inherited Properties for the CIM_DatabaseParameter Class

Property Name	Inherited From Class	Description
InstanceID	CIM_SettingData	An opaque ID that identifies a specific database parameter instance.
Name	CIM_SettingData	A u-friendly name for the database parameter.
Description	CIM_ManagedElement	A longer textual description of the database parameter.
Caption	CIM_ManagedElement	A short textual description of the database parameter.
ElementName	CIM_ManagedElement	A user-friendly name for the database parameter. In some cases, this may be the same value as the name property.

2.6.1 SNMP Database and Service Parameter Mapping

The CIM_SNMPDatabaseParameter class extends from CIM_DatabaseParameter. This class was created in the database schema to provide a mapping to the rdbmsDbParamTable and the rdbmsSrvParamTable entities in the SNMP RDBMS MIB.

In addition to the properties that are inherited from CIM_DatabaseParameter, the CIM_SNMPDatabaseParameter class defines the following property:

- **ParameterValue**

  The ParameterValue property is a string representation of the value of the database or service parameter. The ParameterValue property maps to the rdbmsDbParamCurrValue variable in the SNMP RDBMS MIB.

2.7 Software and Statistics

The software and statistics portion of the database model focuses primarily on the statistics that have been defined in RFC 1697. No additional classes or associations have been added for database software modeling. Model extensions to support specific database software capabilities may be included in a future database model release.

Three primary classes of statistics are defined in the database model. These classes include 1) Database Resource Statistics (CIM_DatabaseResourceStatistics), 2) Common Database Statistics (CIM_CommonDatabaseStatistics), and 3) Database Service Statistics (CIM_DatabaseServiceStatistics). This section describes these classes, along with their properties and associations.

Each class inherits properties through the CIM_StatisticalData class. Table 8 provides a brief summary of the inherited properties. See the MOF where these properties are defined for more information.

Each class of database statistics that is defined in the schema uses the CIM_ElementStatisticalData association from the Core schema to relate an instance of statistical information to its associated managed entity.

Table 8. Inherited Properties for the CIM_DatabaseServiceStatistics Class

Property Name	Inherited From Class	Description
InstanceID	CIM_StatisticalData	An opaque ID that identifies a specific statistics instance.
Name	CIM_StatisticalData	A user-friendly name for the database statistic.
Description	CIM_ManagedElement	A longer textual description of the database statistic.
Caption	CIM_ManagedElement	A short textual description of the database statistic.
ElementName	CIM_ManagedElement	A user-friendly name for the database statistic. In some cases, this may be the same value as the name property.

2.7.1 Database Resource Statistics

The CIM_DatabaseResourceStatistics class contains statistics on resources that have limits that are enforced by either the database or a database service. One instance exists in this class for each database or database service resource that has a resource limit. For example, the database may impose a limit on the number of locks, or the amount of disk space that can be allocated for a database partition. This class maps to the rdbmsLimitedResourceTable and the rdbmsSrvLimitedResourceTable that are defined in the SNMP RDBMS MIB.

The following properties are defined in this class:

- **Current**

  The Current property contains the current value of the limited resource. This property maps to the rdbmsDbLimitedResourceCurrent variable and to the rdbmsSrvLimitedResourceCurrent variable in the SNMP RDBMS MIB.

- **Limit**

  The Limit property contains the maximum value that the database resource can attain. For example, if a resource is defined to limit the number of database locks, the limit might be set to 10,000 locks. As a result, no more than 10,000 locks could be held at any one time for database usage. This property maps to the rdbmsDbLimitedResourceLimit variable and to the rdbmsSrvLimitedResourceLimit variable in the SNMP RDBMS MIB.

- **Highwater**

  The Highwater property contains the maximum value for the database resource measured from the time when the first database service was started for the database. This property maps to the rdbmsDbLimitedResourceHighwater variable and to the rdbmsSrvLimitedResourceHighwater variable in the SNMP RDBMS MIB.

- **Failures**

  The Failures property contains a count of the number of times that the database resource limit would have been exceeded if the resource were allowed to be consumed beyond the limit. This property maps to the rdbmsDbLimitedResourceFailures variable and to the rdbmsSrvLimitedResourceFailures variable in the SNMP RDBMS MIB.

Figure 12 illustrates the CIM_Database ResourceStatistics class.

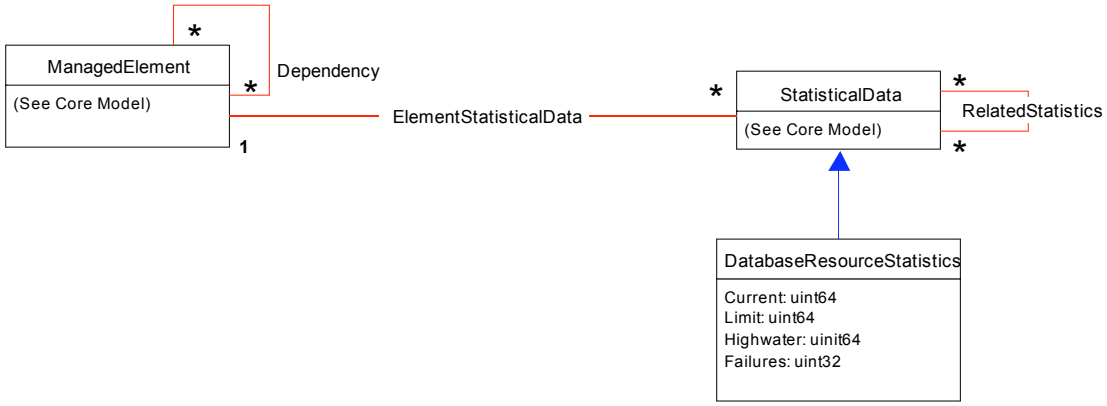

Figure 12. CIM_DatabaseResourceStatistics Class

2.7.2 Common Database Statistics

The CIM_CommonDatabaseStatistics class contains the statistics for a database that span model organization and vendor implementation. This class contains the following property:

- **SizeUsed**

  The SizeUsed property contains the estimated amount of disk space that is currently used by the database. The unit of this property is specified in the SizeUnits property from the CIM_CommonDatabase class. The same units must be used for this property and the SizeAllocated property that is defined in the CIM_CommonDatabase class.The SizeUsed property maps to the rdbmsDbInfoSizeUsed variable in the SNMP RDBMS MIB

Figure 13**Error! Reference source not found.** illustrates the CIM_CommonDatabaseStatistics class.

2.7.3 Database Service Statistics

The CIM_DatabaseServiceStatistics class contains the database service statistics that span model organization and vendor implementation. The UML representation of this class is illustrated in **Error! Reference source not found.** at the end of this section. The class contains the following properties:

- **LastActivity**

  The LastActivity property contains the date and time when the most recent inbound activity was started for the database service. A value of all zeros indicates that no inbound activity has taken place since the service was started. This property maps to the rdbmsSrvInfoLastInboundActivity variable in the SNMP RDBMS MIB.

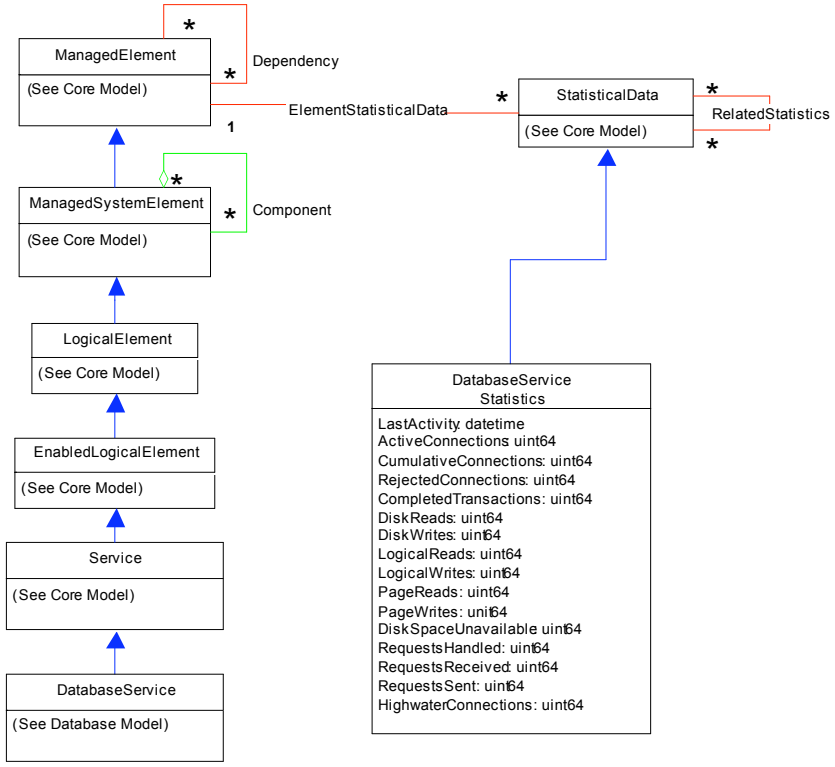

Figure 13. CIM_CommonDatabaseStatistics Class

- **ActiveConnections**

  The ActiveConnections property is a counter of the number of active inbound connections that are using the database service. This property maps to the rdbmsSrvInfoapplInboundAssociation variable in the SNMP RDBMS MIB.

- **CumulativeConnections**

  The CumulativeConnections property is a counter of the total number of inbound connections to the service from the time that the service was started. This property maps to the rdbmsSrvInfoapplAccumulatedInboundAssociations variable in the SNMP RDBMS MIB.

- **RejectedConnections**

  The RejectedConnections property is a counter of the total number of inbound connections that were rejected by the service from the time that the service was started. This property maps to the rdbmsSrvInfoapplRejectededInboundAssociations variable in the SNMP RDBMS MIB.

- **CompletedTransactions**

  The CompletedTransactions property is a counter of the total number of transactions that have been completed by a commit or abort. Some database operations, such as read-only queries, may not create a transaction. This property maps to the rdbmsSrvInfoFinishedTransactions variable in the SNMP RDBMS MIB.

- **DiskReads**

  The DiskReads property is a counter of the total number of database file reads that were issued by the service since it was started. This property maps to the rdbmsSrvInfoDiskReads variable in the SNMP RDBMS MIB.

- **DiskWrites**

  The DiskWrites property is a counter of the total number of database file writes that were issued by the service since it was started. This property maps to the rdbmsSrvInfoDiskWrites variable in the SNMP RDBMS MIB.

- **LogicalReads**

  The LogicalReads property is a counter of the total number of logical database file reads that were issued by the service since it was started. Database implementations cache information in memory. By comparing the DiskReads and the LogicalReads properties, the client can determine how many of the reads were satisfied through a cache in contrast with the more expensive "read from file." This property maps to the rdbmsSrvInfoLogicalReads variable in the SNMP RDBMS MIB.

- **LogicalWrites**

  The LogicalWrites property is a counter of the total number of database file writes that were issued by the service since it was started. A logical write is a count of the number of times that parts of database files have been marked "dirty" to indicate that they need to be written to disk. This property maps to the rdbmsSrvInfoLogicalWrites variable in the SNMP RDBMS MIB.

- **PageReads**

  The PageReads property is a counter of the total number of database pages that have been read by the service since it was started. This property maps to the rdbmsSrvInfoPageReads variable in the SNMP RDBMS MIB.

- **PageWrites**

  The PageWrites property is a counter of the total number of database pages that have been written by the service since it was started. This property maps to the rdbmsSrvInfoPageWrites variable in the SNMP RDBMS MIB.

- **DiskSpaceUnavailable**

  The DiskSpaceUnavailable property is a counter of the total number of times that the service requested disk space that was not available since the service was started. This property maps to the rdbmsSrvInfoDiskOutOfSpaces variable in the SNMP RDBMS MIB.

- **RequestsHandled**

  The RequestsHandled property is a counter of the total number of requests that have been received from the service since it was started. This property maps to the rdbmsSrvInfoHandledRequests variable in SNMP RDBMS MIB.

- **RequestsReceived**

  The RequestsReceived property is a counter of the total number of receive operations that occurred while processing requests on inbound associations since the service was started. This property maps to the rdbmsSrvInfoRequestRecvs variable in SNMP RDBMS MIB.

- **RequestsSent**

  The RequestsSent property is a counter of the total number of send operations that occurred while processing requests on inbound associations since the service was started. For example, this may correspond to the number of rows returned to the client by a select operation. This property maps to the rdbmsSrvInfoRequestSends variable in SNMP RDBMS MIB.

- **HighwaterConnections**

  The HighwaterConnections property is a counter of the maximum number of concurrent inbound connections to the service since it was started. This property maps to the rdbmsSrvInfoHighwaterInboundAssociations variable in SNMP RDBMS MIB.

Error! Reference source not found.Error! Reference source not found. shows the CIM_DatabaseServiceStatistics class.

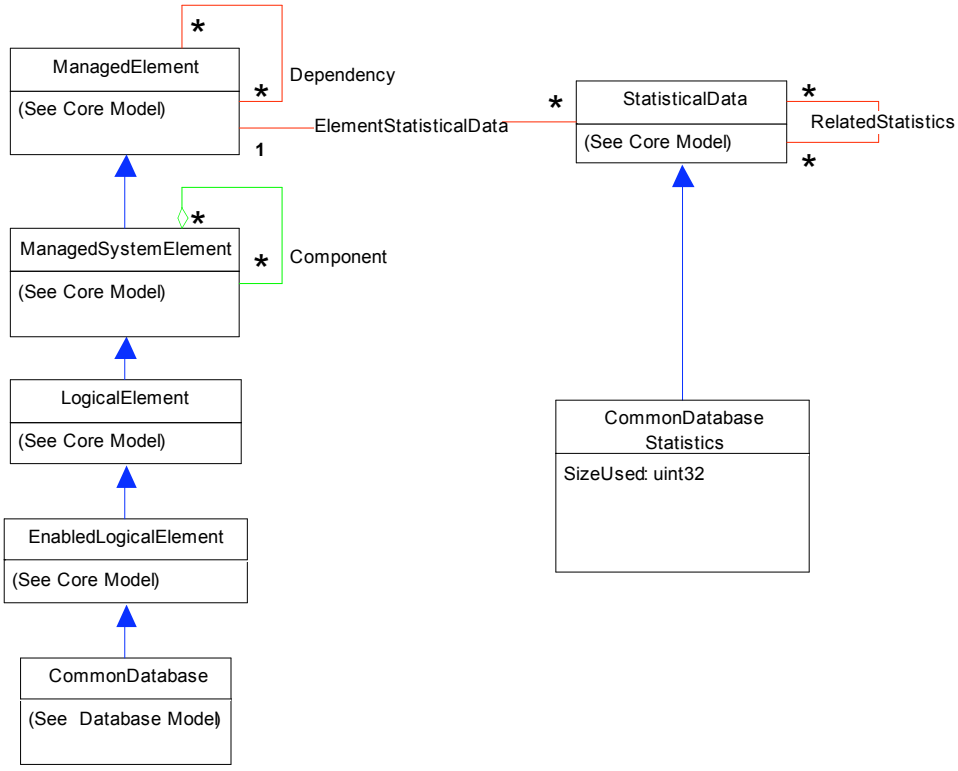

Figure 14. CIM_DatabaseServiceStatistics Class

2.8 Services and Service Access Points

2.8.1 CIM_DatabaseService

The CIM_DatabaseService class inherits from CIM_Service. As a result,, this class inherits all the service-related associations from higher levels of the CIM schema. The database model does not introduce any additional service-related associations. Future versions of the database model may introduce additional associations if the Database Working Group determines that they are specific to the database model and common across database organization and vendor implementations.

3 Relationships to Other Standards and Specifications

3.1 Overlapping Standards and Specifications

This section provides an overview of other standards and specifications that include some level of support for the management of databases. One of the goals of the DMTF Database Working Group is to model the database management content that has been specified in the SNMP RDBMS MIB. As a result, users of management data who are currently using an SNMP-based solution can obtain consistent content from a CIM database model implementation. In a future version of the database model, the working group may investigate ways to incorporate the management content that is defined through the Common Warehouse Metamodel (CWM) specification that was developed by the Object Management Group (OMG).

3.1.1 SNMP RDBMS MIB Specification

The **Simple Network Management Protocol (SNMP) RDBMS MIB** is the management standard that has the widest adoption rate for relational database implementations. The Internet Engineering Task Force (IETF) released it in August 1994. The SNMP RDBMS MIB, as specified in RFC 1697, contains information on installed databases, servers, configuration parameters, and a small number of common statistics and events.

3.1.2 CWM Metamodel

The CWM is a specification that describes the metadata interchange for data warehouses. The OMG released the CWM specification in February 2001. Although the primary focus of this specification is to model warehouse metadata, CWM provides the information that is needed to perform warehouse configuration maintenance operations.

3.2 Mapping of the SNMP RDBMS MIB to the Database Model

The SNMP RDBMS RFC 1697 specification defines a set of common management information that spans relational database systems. The information provided in the common MIB includes a subset of relational database and server properties. The majority of this information is specified in a vendor-private MIB.

3.2.1 Differences Between SNMP RDBMS MIB and CIM Database Models

The CIM database model covers the same content as the SNMP MIB, but there are a number of differences between the two approaches from a management perspective. These differences are:

- The CIM database model is not relational database model-specific.
- The CIM database model defines the relationships between the management entities that are being modeled in addition to defining the common properties. These relationships span the entire CIM model, so management applications can consistently reference information at all levels of the enterprise.
- The CIM database model takes advantage of object-oriented techniques to define abstraction and dependency hierarchies. As a result, more specific management entities

can subclass from a general entity and inherit the properties and relationships of the parent. For example, relational database entities would subclass from the common database model classes. Relational database vendor-specific database entities would subclass from the common relational model entities.

- The CIM database model provides a natural hierarchy that allows management applications to view database information at various levels of detail.

3.2.2 SNMP RDBMS MIB to CIM Database Model Data Mapping

Table 9 describes how the SNMP RDBMS MIB maps to the CIM database model.

Table 9. Mapping of SNMP RDBMS MIB to CIM Database Model

SNMP Variable Name	Pg.	Description	CIM Model Class (parameter)
rdbmsDbVendorName	9	The name of the database vendor.	Product class (Vendor)
rdbmsDbName	9	The name of the database.	CommonDatabase class (Caption), inherited from ManagedSystemElement in the Core schema
rdbmsDbContact	9	The contact person who is responsible for managing the database, along with the information on how to contact the person.	DatabaseAdministrator association
rdbmsDbInfoProductName	9	The product name of the database software.	Product class (Name) in the Core schema
rdbmsDbInfoVersion	9	The product version of the database software.	Product class (Version) in the Core schema
rdbmsDbInfoSizeAllocated	10	The estimated amount of disk space (in units) that has been reserved for database use.	CommonDatabase class (SizeAllocated)
rdbmsDbInfoSizeUnits	10	The units that are used for the database allocated and database size information.	CommonDatabase class (SizeUnits)
rdbmsDbInfoSizeUsed	11	The estimated amount of disk space that is currently used by the database.	CommonDatabaseStatistics class (SizeUsed)
rdbmsDbInfoLastBackup	11	The date and time that the last complete or partial backup operation was performed on the database.	CommonDatabase class (LastBackup)
rdbmsDbParamName	12	The name that identifies a database configuration	DatabaseParameter class (Name), inherited from

SNMP Variable Name	Pg.	Description	CIM Model Class (parameter)
		parameter for a database.	ScopedSettingData in the Core schema
rdbmsDbParamCurrValue	13	The current value of the database configuration parameter.	SNMPDatabaseParameter class (ParameterValue)
rdbmsDbParamComment	14	A description that explains the purpose of the database configuration parameter.	DatabaseParameter class (Description), inherited from ManagedElement in the Core schema
rdbmsDbLimitedResourceName	15	The name of a resource for which the database enforces a maximum usage limit.	DatabaseResourceStatistics class (Name), inherited from StatisticalData in the Core schema
rdbmsDbLimitedResourceLimit	15	The value for the limit imposed by the database for a limited resource.	DatabaseResourceStatistics class (Limit)
rdbmsDbLimitedResourceCurrent	16	The current value for a limited database resource.	DatabaseResourceStatistics class (Current)
rdbmsDbLimitedResourceHighwater	16	The maximum value for a limited database resource that has been viewed since the database was opened by a database service.	DatabaseResourceStatistics class (Highwater)
rdbmsDbLimitedResourceFailures	16	The number of times the database wanted to exceed the resource limit since the database was opened by a database service.	DatabaseResourceStatistics class (Failures)
rdbmsDbLimitedResourceDescription	16	A description of the resource, along with the meaning for the units for the statistics associated with the resource.	DatabaseResourceStatistics class (Description), inherited from ManagedElement in the Core schema
rdbmsSrvVendorName	18	The name of the vendor whose database software provides access to the database.	SoftwareIdentity (Manufacturer), using the ElementSoftwareIdentity Association from the Core schema
rdbmsSrvProductName	18	The vendor-specific product name for the database service.	SoftwareIdentity (Name), using the ElementSoftwareIdentity Association from the Core schema
rdbmsSrvContact	18	The contact person who is responsible for managing the database service, along with the	PrimaryOwnerName, PrimaryOwnerContact in the DatabaseService class,

SNMP Variable Name	Pg.	Description	CIM Model Class (parameter)
		information on how to contact the person.	inherited from Service
rdbmsSrvInfoapplName	19	The product-specific name for the database service.	DatabaseService class (Name), inherited from Service in the Core schema
rdbmsSrvInfoapplVersion	19	The software version number for the database service, in product specific format.	SoftwareIdentity (VersionString), using the ElementSoftwareIdentity Association from the Core schema
rdbmsSrvInfoapplOperStatus	19	The current state of the database service _up or down. “Down” means the service is known but not available. **Note**: The CIM schema supports additional status values that are not available through SNMP.	DatabaseService class (OperationalStatus)
rdbmsSrvInfoapplLastChange	19	The date and time when the database service status last changed.	DatabaseService class (LastStatusChangeTime)
rdbmsSrvInfoapplInboundAssociation	19	The number of active inbound connections using the database service.	DatabaseServiceStatistics class (ActiveConnections)
rdbmsSrvInfoapplAccumulated InboundAssociations	19	The total number of inbound connections since the database service was started.	DatabaseServiceStatistics class (CumulativeConnections)
rdbmsSrvInfoLastInboundActivity	19	The date and time of the most recent inbound connection to the database service	DatabaseServiceStatistics class (LastActivity)
rdbmsSrvInfoapplRejected InboundAssociations	20	A count of the number of inbound connections that were rejected by the database service.	DatabaseServiceStatistics class (RejectedConnections)
rdbmsSrvInfoStartupTime	20	The date and time when the database service was started.	DatabaseService class (StartupTime)
rdbmsSrvInfoFinishedTransactions	21	The number of transactions that have been completed by a commit or abort. Some database operations, such as read-only queries, may not create a transaction.	DatabaseServiceStatistics class (CompletedTransactions)

SNMP Variable Name	Pg.	Description	CIM Model Class (parameter)
rdbmsSrvInfoDiskReads	21	The total number of database file reads issued by the database service since it was started.	DatabaseServiceStatistics class (DiskReads)
rdbmsSrvInfoLogicalReads	21	The total number of logical database file reads issued by the database service since it was started.	DatabaseServiceStatistics class (LogicalReads)
rdbmsSrvInfoDiskWrites	21	The total number of database file writes issued by the database service since it was started.	DatabaseServiceStatistics class (DiskWrites)
rdbmsSrvInfoLogicalWrites	22	The total number of logical database file writes issued by the database service since it was started. A logical write is a count of the number of times parts of database files have been marked “dirty” to indicate they need to be written to disk.	DatabaseServiceStatistics class (LogicalWrites)
rdbmsSrvInfoPageReads	22	The total number of database pages read by the database service since startup.	DatabaseServiceStatistics class (PageReads)
rdbmsSrvInfoPageWrites	22	The total number of database pages written by the database service since startup.	DatabaseServiceStatistics class (PageWrites)
rdbmsSrvInfoDiskOutOfSpaces	22	The total number of times the server requested disk space and it was not available since database service startup.	DatabaseServiceStatistics class (DiskSpaceUnavailable)
rdbmsSrvInfoHandledRequests	23	The number of request that were received by the server since database service startup.	DatabaseServiceStatistics class (RequestsHandled)
rdbmsSrvInfoRequestRecvs	23	The number of receive operations made while processing any requests for the database service since startup.	DatabaseServiceStatistics class (RequestsReceived)
rdbmsSrvInfoRequestSends	23	The number of send operations made while	DatabaseServiceStatistics class (RequestsSent)

SNMP Variable Name	Pg.	Description	CIM Model Class (parameter)
		processing any requests for the database service since startup.	
rdbmsSrvInfoHighwaterInbound Associations	24	The maximum number of active inbound connections that have been concurrently opened by the database service since startup.	DatabaseServiceStatistics class (HighwaterConnections)
rdbmsSrvInfoMaxInboundAssociations	24	The greatest number of active inbound connections that can be concurrently open by the server.	DatabaseService class (ConnectionLimit)
rdbmsSrvParamName	26	The name that identifies a configuration parameter for a database service.	DatabaseParameter class (Name), inherited from ScopedSettingData in the Core schema
rdbmsSrvParamCurrValue	26	The current value of the database service configuration parameter.	SNMPDatabaseParameter class (ParameterValue)
rdbmsSrvParamComment	27	A description that explains the purpose of the database service configuration parameter.	DatabaseParameter class (Description), inherited from ManagedElement in the Core schema
rdbmsSrvLimitedResourceName	28	The name of a resource for which the database service enforces a maximum usage limit.	DatabaseResourceStatistics class (Name), inherited from StatisticalData in the Core schema
rdbmsSrvLimitedResourceLimit	28	The value for the limit imposed by the database service for a limited resource.	DatabaseResourceStatistics class (Limit)
rdbmsSrvLimitedResourceCurrent	29	The current value for a limited database service resource.	DatabaseResourceStatistics class (Current)
rdbmsSrvLimitedResourceHighwater	29	The maximum value for a limited database service resource that has been seen since service startup.	DatabaseResourceStatistics class (Highwater)
rdbmsSrvLimitedResourceFailures	29	The number of times the database service wanted to exceed a resource limit since service startup.	DatabaseResourceStatistics class (Failures)
rdbmsSrvLimitedResourceDescription	29	A description of the resource, along with the meaning for the units for the statistics associated	DatabaseResourceStatistics class (Description), inherited from ManagedElement in the Core schema

SNMP Variable Name	Pg.	Description	CIM Model Class (parameter)
		with the resource.	
rdbmsRelState	30	The current state of the database service's access to the database: • "Active" means the service is actively using the database. • "Available" means the service could use the database if necessary. • "Restricted" means the service is available in a less than complete state. • "Unavailable" means the database is not available through this service. • "Other" means the database service is in some other condition.	ServiceAvailableToDatabase association (AvailableState)
rdbmsRelActiveTime	31	The date and time when the database was made active by the specific server. If the server is not in an active state, this information is not available.	ServiceAvailableToDatabase association (ActiveTime)

3.2.3 SNMP to CIM Database Model Event Mapping

The SNMP RDBMS MIB currently defines the following traps:

- Database/Server state change
- Out of space

SNMP "traps" correspond to "indications" in CIM terminology. Indications can be enabled for properties of managed entities that have been modeled in CIM. See the *CIM Indications* white paper for more information.

3.2.3.1 *State Change Event Mapping*

The rdbmsStateChange trap as defined in RFC 1697 signifies that the rdbmsRelState of the database server has changed in a way that makes the database less accessible for use. In CIM database model terminology, this is indicated when the EnabledStatus of a database service, which is providing access to the database, goes from a state of "Available" or "Active" to some other state.

To directly map from the SNMP state change trap, a CIM client can write an indication filter referencing the CIM_InstModification class using a “where” clause that references the EnabledStatus property from the CIM_DatabaseService class. A simpler filter to use would be the OperationalStatus property from the CIM_DatabaseService class with a value of "degraded".

3.2.3.2 *Out of Space Event Mapping*

The rdbmsOutOfSpace trap detects the condition where a database server was unable to allocate space for the database it is serving. In CIM database model terminology, this corresponds to the condition where a database service that is providing access to the database was unable to allocate space.

To directly map from the SNMP out of space trap, a CIM client can write an indication filter referencing the CIM_InstModification class, using a “where” clause that looks for an increase in the value of the DiskSpaceUnavailable property in the CIM_DatabaseServiceStatistics class.

4 Database Model Use Case

This section provides a use case for the database model. The use case provides an example that illustrates how the database model is used for management purposes.

To demonstrate how the database model can be used in a database organization and vendor-agnostic way, the example is based upon a fictitious database vendor, MyDB Company, who has a database product called MyDB that is organized using an object-relational scheme. The example also assumes that providers have been written for the database schema classes. The use case describes one possible scenario for how these classes can be populated and accessed for enterprise management purposes.

At the beginning of the MyDB product development cycle, the management team at MyDB Company recognized the importance of having consistent management content across the enterprise that included information for the MyDB product. The development staff was instructed to ensure that management content was available for the MyDB product for all aspects of the product lifecycle. CIM was the logical choice to satisfy this directive.

4.1 Creating CIM Class Instances During the MyDB Installation

At install time, the developers at MyDB Company chose to discover and populate the CIM schema used by the CIM Object Manager (CIMOM)–which was already managing the operating system on the host–with content for the MyDB product.

A subset of the MyDB information is used in this example to demonstrate how CIM is used in a database environment. In Table 10 and the other following tables, the Instance Value column represents a class instance that is created at installation time for the MyDB product. Inherited properties that are not explicitly mentioned use the default values.

Table 10. MyDB Product Information

CIM_DatabaseSystem	
Property Name	**Instance Value**
CreationClassName	CIM_DatabaseSystem
Name	MyDB_V10Software
ElementName	MyDB Database Software
PrimaryOwnerName	Cynthia Smith
PrimaryOwnerContact	e-Mail: cynthia.smith@customer.com Phone: (603) 123-4567; Pager: (603) 123-7654
OperationalStatus	3
InstallDate	28-May-2002 11:21.02
Caption	The software for the MyDB database product

The instance added to CIM_DatabaseSystem during the MyDB product installation process is a logical entity representing the MyDB database system. The owner of the software was identified as the person who installed the software, Cynthia Smith. A name that identifies the software is provided in the Name property. The ElementName gives a more user-friendly description.

Because the MyDB product does not have a preconfigured database at installation time, the remaining CIM classes that are part of the common database schema will be populated when a user of the MyDB database system creates a database.

Additional classes from the core, application, and other CIM schemas are populated at installation time to provide complete management content. Most of these classes are not mentioned in this use case.

Table 11 provides an example of instance creation for the CIM_Product class to demonstrate instance population for the MyDB product for a class in the Core schema.

At this point, client applications that are CIM-aware can consistently reference the software installation and product inventory content for MyDB in the same way as other CIM-compliant products.

Table 11. MyDB Instance Population Information

CIM_Product	
Property Name	**Instance Value**
Name	MyDB
IdentifyingNumber	1
Vendor	MyDB Company
Version	V1.0
SKUNumber	1
WarrantyStartDate	28-May-2002 11:21.02
WarrantyDuration	1
Caption	MyDB Database Software Version 1.0

4.2 Creating CIM Class Instances at Database Creation

The MyDB product supports the SQL standard for database creation. At database creation time, an instance is added to the CIM_CommonDatabase class to represent the logical database entity that is created when the "create database" operation has finished. Table 12 provides examples of class instances that are created.

Table 12. Examples of Class Instances Created After Database Creation

CIM_CommonDatabase	
Property Name	**Instance Value**
InstanceID	MyDBGUID (GUID represents a unique ID.)
Name	\\golem\usr\database\mydb\golem.bin
ElementName	Golem
DatabaseVersion	1.0
SizeAllocated	10
SizeUnits	3
OperationalStatus	10
LastBackup	0
InstallDate	31-May-2002 10:16.22
Caption	The Golem database contains Greek Mythology.

In a real-world implementation, the MyDB Company developers may extend from the CIM_CommonDatabase class instead of using the class directly so that vendor-specific properties could be added. This approach also allows clients to distinguish databases from MyDB from those created by other vendors.

In this case, the instance created in CIM_CommonDatabase contains useful management content for the Golem database. This information enables us to determine the following:

- The database is used to store information about Greek mythology.
- The database is version 1.0 .
- The database has an allocated size of 10MB.
- The database was not backed up.
- The database was created on the May 31.
- The database is not running.

The Instanced property uniquely identifies the database instance across the enterprise. The contents of the instance ID include a vendor-specific identifier "MYDB" followed by a unique value, the GUID.

The developers at MyDB Company chose a database name that represents the complete path name to the primary database file. A more user-friendly form of the database name is provided in the ElementName property.

After the CIM_CommonDatabase instance is created, an instance is created in the CIM_AssociatedDatabaseSystem class to represent the relationship between the database that was created and the database system that created and controls the database.

Because the MyDB product is targeted for small installations that only have a couple of users, the developers did not allow users to control t database configuration.. The developers decided on three database parameters for the MyDB product:

- Number of database buffers
- Database page size
- Maximum number of users

Only the number of database buffers parameter is configurable at the database service level.

The developers of the MyDB product needed to expose the parameters in CIM so that management applications could modify the settings through the CIMOM. In the CIM 2.7 database schema, the developers for the MyDB product had two implementation choices:

- Add instances to the CIM_SNMPDatabaseParameter class to represent the database parameters.
- Create a vendor-specific extension class from CIM_DatabaseParameter that contained their database settings.

The developers at MyDB Company decided that extending from CIM_DatabaseParameter was a more technically appropriate solution, because their parameters would be exposed through CIM as named properties, rather than as name/value strings. The development cost required to create a

vendor-specific parameter class was worth the benefit of having strongly typed database parameter definitions.

The developers created a class called MyDB_DatabaseParameters with three uint32 properties called NumberBuffers, PageSize, and MaximumUsers that extended from CIM_DatabaseParameter.

The MyDB_DatabaseServiceParameters extension class was created to represent that only the number of buffers could be changed by a database service. This class only had one property, called NumberBuffers.

At database creation time, instances are added to the CIM schema to represent the initial database configuration. A combination of classes from the CIM core schema, the CIM database schema, and the MyDB extension classes are populated to represent this information. The following example explains how the instances are added to the CIM schema.

In Table 13, an instance is added to the MyDB_DatabaseParameter extension class to represent the database parameter settings for the GOLEM database.

Table 13. Database Parameter Settings

MyDB_DatabaseParameter	
Property Name	**Instance Value**
InstanceID	MyDBGUID (GUID represents a unique ID.)
Name	\\golem\usr\database\mydb\golem.bin
ElementName	Golem database parameter settings
Caption	Database Parameter settings for the Golem Database
NumberBuffers	100
PageSize	4
MaximumUsers	1

Note: The Name selected for the instance by the MyDB developers is consistent with the Name property used in CIM_CommonDatabase. This is not required, but it allows searches to be constructed across the two classes in the same way.

In Table 14, the database service buffer setting is represented by adding an instance to the MyDB_DatabaseServiceParameter extension class.

Table 14. Database Service Buffer Setting

MyDB_DatabaseServiceParameter	
Property Name	**Instance Value**
InstanceID	MyDBGUID (GUID represents a unique ID.)
Name	\\golem\usr\database\mydb\golem.bin
ElementName	Golem database service parameter settings
Caption	Database Service settings for the Golem Database
NumberBuffers	100

Next, two instances are created for the CIM_ScopedSetting association. The first instance represents the association between the Golem database and its database parameters. The second instance represents the association between the Golem database service and its service parameters.

Instances are created in the CIM_ElementSettingData association to represent default and current setting associations for the Golem database parameter settings and the Golem database service parameter settings. In this example, it is a one-to-one relationship. Because this is the initial database creation, the default parameter settings are also the current settings. The IsDefault property in the association is set to 1. Because some of the parameters may be changed at a future point in time, the IsCurrent property is set to 0.

In the future, if the administrator of the Golem database decides to change one of the database or service parameter values, a new instance would be created in the appropriate CIM_DatabaseParameter class or subclass, and a new association would be formed in CIM_ElementSettingData with the IsCurrent property set to 1 and the IsDefault property set to 2.

Instances in the CIM_ElementProfile class are created to represent the relationship between the Golem database and its database parameters and the Golem database service and its database service parameters. Because the MyDB developers choose to create extension classes, this association is similar to the CIM_ScopedSetting association. However, this association differs in that separate profiles may be created to name and represent the default and current parameter settings. The profile association instances are created to allow management clients to access the Golem database and service parameter content consistently with parameter content that needs to be collected through a profile, such as the instances added to CIM_SNMPDatabaseParameters.

In Table 15, the MyDB product architecture supports only a single service per database. At database creation time, the database service content and the information representing how the service is associated with the database is also populated into the CIM schema.

Table 15. CIM_DatabaseService Settings

CIM_DatabaseService	
Property Name	**Instance Value**
SystemCreationClassName	CIM_System
SystemName	GolemHost
CreationClassName	CIM_DatabaseService
Name	Golem
ElementName	Golem
StartupTime	0
OperationalStatus	10
LastStatusChangeTime	0
ConnectionLimit	1
StartMode	Manual
Started	0
InstallDate	31-May-2002 10:16.22
Caption	The Golem database service

The Golem database service content informs the user that the database service executes on GolemHost, that the service has never been started, that a maximum of one connection per user is allowed, and the time when the database service was created. Because the Golem database service is created as part of the database install, the InstallDate property is the same value as the InstallDate for the Golem database instance in the CIM_CommonDatabase class.

4.3 Using the Database Schema Content for Management

The common database schema contains a number of statistical classes with properties that are useful for monitoring databases. This information can be used to answer management questions about the database environment across the enterprise for configuration management, monitoring, and administration.

This section poses a number of common database questions and explains how the content from the database schema can be used to answer these questions.

- **What database services are running on system GolemHost?**

  The CIM_DatabaseService class filters the instances in the class based on the SystemName property.

- **Which databases are open on system GolemHost?**

  The CIM_DatabaseService class filters the instances in the class based on the SystemName property. From the set of instances returned, the CIM_ServiceAvailableToDatabase association would be accessed, filtering the instances by the property AvailableState with a value equal to 2 or 3. The reference to CIM_CommonDatabase for the filtered set would be traversed, and the distinct set of databases would be displayed using the Name or the Caption property.

- **What are the configuration parameter settings for database X?**

  The CIM_CommonDatabase class is filtered by Name equals database X. The CIM_ElementProfile association is then used to identify the database parameter instances for database X. The corresponding instances from the database parameter class are then displayed.

- **How many connections are currently active on system X?**

  The CIM_DatabaseService is filtered by the SystemName property to include only the database services that are on system X. Next, the CIM_ElementStatisticalData association would be used to reference the CIM_DatabaseServiceStatistics instances that are associated with the filtered list of services. The service name and the active connections properties could then be displayed.

- **What services provide access to database X?**

  CIM_CommonDatabase is filtered by name. The ServiceAvailableToDatabase association would then be used to display the database service names and SystemName properties that are associated with the database.

5 Future Work

The initial model developed by the Database Working Group was determined by the contents specified in RFC 1697. The group focused on a small subset of database management content so that an initial model could be delivered in a relatively short timeframe.

Future extensions to the database model may include the following areas:

- Adding additional common properties that have not been defined in RFC 1697
- Extending the model to cover relational database content
  - SQL
  - Relational schema objects
- Modeling advanced database configurations, such as clustered databases, distributed databases, or standby databases
- Modeling other common database technology areas, such as resource management, queuing, and replication
- Modeling additional common database entities, such as database users
- Modeling common database management workflows, such as backup operations
- Leveraging other modeling efforts, such as CWM meta-model for OMG in the database warehouse domain
- Enabling management capabilities, such as database provisioning

The Database Working Group is also interested in how MOF can be translated to SQL. This translation would allow the static content from a CIMOM to be stored in a relational database.

Appendix A – Change History

Version 1.0	<Initial release>	Initial Draft
Version 1.1	6/25/2002	Changes to reflect member feedback

Appendix B – References

The following background reference material is available:

- "Common Information Model (CIM) Event Model White Paper," at http://www.dmtf.org/standards/published_documents.php
- "Common Information Model (CIM) Specification, V2.2," June 14, 1999, athttp://www.dmtf.org/spec/cims.html
- "Common Warehouse Metamodel (CWM) Specification, V1.0," February 2, 2001, at ftp://ftp.omg.org/pub/docs/ad/01-02-01.pdf.
- Relational Database Management System (RDBMS) Management Information Base (MIB) using SMIv2," Internet Engineering Task Force, August 1994, at http://www.ietf.org/rfc.html
- *A Guide to the SQL Standard, 3rd Edition*, C.J. Date with Hugh Darwen, Addison-Wesley, October 1992
- "Common Information Model (CIM) Application Model White Paper," at http://www.dmtf.org/standards/published_documents.php
- "Unified Modeling Language (UML)," Open Management Group (OMG), at http://www.omg.org/uml/

Appendix C – Extending the Model

The classes and associations that the Database Working Group has defined should be extended by provider writers to include management-related database organization or vendor-specific content. In future versions of the CIM database model, as the model is expanded to include common classes for specific database organizations, the number of database organization-related classes and associations that will need to be defined by provider writers will decrease.

For example, a future version of the CIM database model may include a CIM_RelationalDatabase class that extends from CIM_CommonDatabase and that includes properties that are specific to relational databases.

The current database model is designed so that provider writers can subclass from each of the major classes to include additional vendor-specific content. For example, if additional database properties are relevant for a vendor implementation, the provider writer should include a class that extends from CIM_CommonDatabase to include the additional properties.

Appendix D – Considerations for Implementation

The Database Working Group discussed how to manage statistics that span the period of time when they are collected and are intended to be used together as a group. The statistics in the CIM_DatabaseServiceStatistics class fit this category. Provider writers need to consider the issue that involves the time lapse across individual statistic snapshots.

Table 16 provides an example using CIM_DatabaseServiceStatistics to illustrate the time-lapse impact.

Table 16. Example of Time-Lapse Impact

Time	CompletedTransactions	PageWrites
T1	10	100
T2	20	200

T1 is the time when we start collecting current statistics, T2 the time to complete the "current" collection for the set of DatabaseServiceStatistics by the provider writer.

If the provider is implemented to collect CompletedTransactions at time T1, then time lapses to time T2 before we collect Page Writes, a client trying to calculate PageWrites/Transaction would get a value of 200/10 = 20 PageWrites/Transaction.

On the other hand, if the provider were implemented to collect PageWrites first at time T1, then time lapses to time T2 before we collect CompletedTransactions, a client trying to calculate PageWrites/Transaction would get a value of 100/20 = 5 PageWrites/Transaction.

When the delta between time T1 and time T2 is small, the impact is minimized.

Acknowledgements

The Database Working Group has benefited from the previous work that has been done to identify common database management entities and properties that exist within the industry. The previous work has helped the working group to focus its modeling effort in order to deliver a common database model more quickly. We would like to recognize the members of the initial Database Working Group for their ideas and initial model. We would also like to recognize the members of the IETF Network Working Group who wrote the RFC 1697 standard.

DMTF Database Working Group members who contributed to 2.8 of the database model (listed alphabetically) include:

- Kamesh Aiyer, EMC
- Carl Chan, Sun
- George Ericson, EMC
- Todd Guay, Oracle
- Randy Horman, IBM
- Andreas Koppel, SAP
- Tony Orling, VERITAS
- Darryl Presley, Oracle
- Julie Schott, Cisco
- Andrea Westerinen, Cisco
- Martin White, Oracle

The following members of the DMTF Database Working group that are not already mentioned above (listed alphabetically) contributed to earlier releases of the database model:

- Linda Ball, BMC
- Daniel Forthman, Visionael
- Stefffen Hulegaard, OSA Technologies
- Namik Hrle, IBM
- Charly Jones, Vision Solutions
- Christina Shaw, Hewlett-Packard