The AR System data dictionary ............................................. 24
Initial table ........................................................................ 27
Tables for forms ................................................................ 28
Tables for fields ................................................................ 29
Tables for menus ................................................................ 29
Tables for filters ................................................................ 30
Tables for escalations .......................................................... 30
Tables for active links ........................................................ 30
Tables for mapping workflow ............................................... 31
Tables for containers .......................................................... 31
Creating tables for forms ...................................................... 32
Main data table ................................................................... 32
Status history table ............................................................ 33
Attachment tables .............................................................. 34
Currency table .................................................................... 35
Indexing ............................................................................. 36
SQL views .......................................................................... 37
Updating tables when AR System forms change ......................... 38
Adding fields ..................................................................... 39
Deleting fields ................................................................... 39
Changing character field lengths .......................................... 40
Related information ............................................................ 43
DB2 .................................................................................. 43
Informix ............................................................................. 43
Oracle ............................................................................... 43
Sybase .............................................................................. 43
Microsoft SQL ................................................................. 43
Unicode database support ................................................... 44
Unicode compliance versus Unicode database support ............... 44
Creating a Unicode database ................................................. 45
Migrating existing databases to Unicode .................................. 46
Chapter 2 SQL Definitions of the data dictionary tables .................. 49
DB2 Universal .................................................................. 50
Informix ............................................................................. 71
Database Reference

Oracle .......................................................... 89
Sybase and Microsoft SQL Server ..................... 106

Appendix A
Database user names, passwords, and dates .......... 129
Changing the AR System database user name and password .... 130
Converting AR System dates to database dates ......... 131

Index ............................................................. 133
Important: The compatibility information listed in the product documentation is subject to change. See the compatibility matrix at http://supportweb.remedy.com for the latest, most complete information about what is officially supported.

Carefully read the system requirements for your particular operating system, especially the necessary patch requirements.

Audience

This guide is written for database administrators who are responsible for overseeing the interaction between the BMC® Remedy® Action Request System® (AR System®) and specific databases. This guide is also intended to provide information for AR System administrators who are responsible for defining and changing the structure of AR System forms.

This guide assumes knowledge of database administration and familiarity with the operating system platform you are using. You should be familiar with BMC Remedy Administrator before you begin.
AR System documents

The following table lists documentation available for AR System products. Unless otherwise noted, online documentation in Adobe Acrobat (PDF) format is available on AR System product installation CDs, on the Customer Support site (supportweb.remedy.com), or both.

You can access product Help through each product’s Help menu or by clicking on Help links.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
<th>Audience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts</td>
<td>Overview of AR System architecture and features with in-depth examples; includes information about other AR System products as well as a comprehensive glossary for the entire AR System documentation set.</td>
<td>Everyone</td>
</tr>
<tr>
<td>Installing</td>
<td>Procedures for installing AR System.</td>
<td>Administrators</td>
</tr>
<tr>
<td>Getting Started</td>
<td>Introduces topics that are usually only learned when first starting to use the system, including logging in, searching for objects, and so on.</td>
<td>Everyone</td>
</tr>
<tr>
<td>Form and Application Objects</td>
<td>Describes components necessary to build applications in AR System, including applications, fields, forms, and views.</td>
<td>Developers</td>
</tr>
<tr>
<td>Workflow Objects</td>
<td>Contains all of the workflow information.</td>
<td>Developers</td>
</tr>
<tr>
<td>Configuring</td>
<td>Contains information about configuring AR System servers and clients, localizing, importing and exporting data, and archiving data.</td>
<td>Administrators</td>
</tr>
<tr>
<td>Installing and Administering</td>
<td>Contains information about the mid tier, including mid tier installation and configuration, and web server configuration.</td>
<td>Administrators</td>
</tr>
<tr>
<td>BMC Remedy Mid Tier</td>
<td>Discussed integrating AR System with external systems using plug-ins and other products, including LDAP, OLE, and ARDBC.</td>
<td>Administrators</td>
</tr>
<tr>
<td>Integrating with Plug-ins and</td>
<td>Server administration topics and technical essays related to monitoring and maintaining AR System for the purpose of optimizing performance and troubleshooting problems.</td>
<td>Administrators</td>
</tr>
<tr>
<td>Third-Party Products</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimizing and Troubleshooting</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Database Reference

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
<th>Audience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Reference</td>
<td>Database administration topics and rules related to how AR System interacts with specific databases; includes an overview of the data dictionary tables.</td>
<td>Administrators</td>
</tr>
<tr>
<td>Administering BMC Remedy DSO</td>
<td>Server administration and procedures for implementing a distributed AR System server environment with the BMC Remedy Distributed Server Option (DSO).</td>
<td>Administrators</td>
</tr>
<tr>
<td>Administering BMC Remedy Flashboards</td>
<td>Flashboards administration and procedures for creating and modifying flashboards and flashboards components to display and monitor AR System information.</td>
<td>Administrators /Programmers</td>
</tr>
<tr>
<td>C API Reference</td>
<td>Information about AR System data structures, C API function calls, and OLE support.</td>
<td>Administrators /Programmers</td>
</tr>
<tr>
<td>C API Quick Reference</td>
<td>Quick reference to C API function calls.</td>
<td>Administrators /Programmers</td>
</tr>
<tr>
<td>Java API *</td>
<td>Information about Java classes, methods, and variables that integrate with AR System.</td>
<td>Administrators /Programmers</td>
</tr>
<tr>
<td>Administering BMC Remedy Email Engine</td>
<td>Procedures for installing, configuring, and using the BMC Remedy Email Engine.</td>
<td>Administrators</td>
</tr>
<tr>
<td>Error Messages</td>
<td>List and expanded descriptions of AR System error messages.</td>
<td>Administrators /Programmers</td>
</tr>
<tr>
<td>Master Index</td>
<td>Combined index of all books.</td>
<td>Everyone</td>
</tr>
<tr>
<td>Release Notes</td>
<td>Information about new features list, compatibility lists, international issues, and open and fixed issues.</td>
<td>Everyone</td>
</tr>
<tr>
<td>BMC Remedy Import Help</td>
<td>Procedures for using BMC Remedy Import.</td>
<td>Administrators</td>
</tr>
<tr>
<td>BMC Remedy Administrator Help</td>
<td>Procedures for creating and modifying an AR System application for tracking data and processes.</td>
<td>Administrators</td>
</tr>
<tr>
<td>BMC Remedy Alert Help</td>
<td>Procedures for using BMC Remedy Alert.</td>
<td>Everyone</td>
</tr>
<tr>
<td>BMC Remedy Mid Tier Configuration Tool Help</td>
<td>Procedures for configuring the BMC Remedy Mid Tier.</td>
<td>Administrators</td>
</tr>
</tbody>
</table>

* A JAR file containing the Java API documentation is installed with the AR System server. Typically, it is stored in `C:\Program Files\AR System\Arserver\Api\doc\ardoc70.jar` on Windows and `/usr/ar/<server_name>/api/doc/ardoc70.jar` on UNIX.
Learn about the AR System Developer Community

If you are interested in learning more about AR System, looking for an opportunity to collaborate with fellow AR System developers, and searching for additional resources that can benefit your AR System solution, then this online global community sponsored by BMC Remedy is for you.

In the Developer Community, you will find collaboration tools, product information, resource links, user group information, and be able to provide BMC Remedy with feedback.

The Developer Community offers the following tools and information:
- Community message board
- Community Downloads
- AR System Tips & Tricks
- Community recommended resources
- Product information
- User Experience Design tips

Why should you participate in the Developer Community?

You can benefit from participating in the Developer Community for the following reasons:
- The community is a direct result of AR System developer feedback.
- BMC Remedy provides unsupported applications and utilities by way of Community Downloads, an AR System application.
- BMC Remedy posts the latest AR System product information in the Developer Community to keep you up to date.
- It is an opportunity to directly impact product direction through online and email surveys.
- It’s free!

How do you access the Developer Community?

Go to supportweb.remedy.com, and click the Developer Community link.
This chapter describes how AR System 7.0 interacts with DB2 Universal, Informix, Oracle, Sybase, and Microsoft SQL Server database systems.

The following topics are provided:
- The database structure and AR System (page 12)
- The AR System database (page 12)
- Using relational databases with AR System (page 13)
- Database types and data types (page 20)
- The AR System data dictionary (page 24)
- Creating tables for forms (page 32)
- SQL views (page 37)
- Updating tables when AR System forms change (page 38)
- Related information (page 43)
- Unicode database support (page 44)

**Note:** If you are upgrading from a previous version of AR System, the data dictionary will be restructured. This chapter describes changes that occur during installation, and changes that occur as new data is stored in the database.
The database structure and AR System

In general, AR System hides the underlying database from the user. The AR System server interacts with the database and provides information to the user independent of the underlying database. All access through the API supplied with the product goes through this server and is independent of the database.

AR System supports read access directly from the tables but does not support update access to any of the AR System tables directly through SQL. You must go through the AR System API for update access.

Note: BMC Remedy reserves the right to change the structure of the AR System database with any release. If the structure is changed, the database version number will be updated to indicate a change.

The AR System database

Other than AR System data, AR System and its installation do not interact with or affect other data in the database. The only exception is data that is referenced by using the Direct SQL capability within workflow or by using a view form. See the Workflow Objects guide for more information about this function.

WARNING: Because AR System passes SQL commands to the database without checking the syntax, all commands are submitted to the database. Make sure all submitted commands achieve the desired result. Your SQL commands should comply with ANSI SQL standards, so that single quotes are reserved for strings and double quotes are reserved for use with database object names only.

When you install AR System over a relational database, an AR System database is created. By default, this database is named ARSystem, and the user ARAdmin is defined. You can choose other values during installation. This document refers to the default values, so if you changed these during installation, substitute your database and user names for ARSystem and ARAdmin. The characteristics of the AR System database vary depending on the type of underlying relational database.
You can perform any system administrator activity on the database or on any of the tables it contains. This includes performing regular backups, creating more tablespaces to be added to the AR System database, and adding more containers to tablespaces. With a Sybase or Microsoft SQL database, flush the transaction log (or configure it to autoflush) as part of your regular backup strategy.

After the AR System database is created, AR System creates a series of tables that form its data dictionary. See “The AR System data dictionary” on page 24 for information.

Using relational databases with AR System

Each type of relational database behaves differently in regards to search qualifications, wildcards, and so forth. The following sections describe these differences. Inform your users of the requirements for successful searches on your database type.

For information about different behaviors and requirements for installing AR System with specific databases, see the Installing guide.

For information about configuration options and parameters associated with specific databases, see the `ar.conf` or `ar.cfg` file documentation in the Configuring guide.

Using IBM DB2 Universal Database with AR System

DB2 behaviors that you need to consider are described in the following sections.

User name and password

- When the DB2 database resides on the same machine as the AR System server, `ARAdmin` user is not used. You can run the AR System server installer as root or any other user, as long as that user has administrator privileges for the specific DB2 instance on which you install AR System database.
When the DB2 database resides on a different machine than the AR System server, the database user name, `aradmin`, must be created in lowercase before installing AR System server. The database user name is associated with the operating system. For overwrite and new installations (but not for upgrade installations), this operating system account must exist before installing AR System server. The password must be `AR#Admin#`. After the AR System server is installed, you can change the password. See “Changing the AR System database user name and password” on page 130.

Because the database user name is associated with the operating system, you must make password changes in the operating system and set the new password in the Server Information dialog box in BMC Remedy Administrator.

Form and field limits

When you create a form, there is a size limit. The total size of all data fields in a form cannot exceed 16 KB with the installed AR System database. This is due to a DB2 limitation that creates a database with a tablespace that has a 16KB page size. If you create a form that exceeds 16 KB, then you must create a tablespace with a large page size before you create such a form.

To create a tablespace with a larger page size for a particular form

1. Stop the AR System server.
2. Create a tablespace with 32 KB page size. (You might want to name the tablespace something like `TBS32K`.)
3. Start the AR System server.
4. In BMC Remedy Administrator, open the Server Information dialog box.
5. On the Database tab, add the following options to the database configuration file.

```
Form: <form_name>
Clause: IN TBS32K
```

This causes the table for the `<form_name>` form to be created in the tablespace of 32 KB.
You can also specify the clause as follows:

Form:
Clause: IN TBS32K

This causes the table for all the forms to be created in the tablespace of 32 KB.

6 Click OK to save this server information.

7 Create the form.

If this procedure does not work, you might need to change some of the character fields (these use the varchar datatype) to 256 or more bytes, so that a different datatype (long varchar) is used in the underlying DB2 database. The long varchar datatype takes up much less space in the main data table than the varchar datatype.

The following limits pertain to the size of attachments and fields:

- The character field length is limited to 1 MB.
- The attachment size is limited to 1 GB.
- You cannot sort character fields greater than 254 bytes.
- You cannot store background bitmaps larger than 1 MB.

**LIKE predicate**

DB2 does not support using a column reference on the right side (or pattern) of the LIKE predicate. Only character-value references are supported. For example, the following query returns an error message because DB2 does not support using a field ID on the right of the LIKE predicate.

```
"Demo" LIKE 'Submitter'
```

This might affect the functionality of BMC Remedy applications.
Using Informix with AR System

Informix behaviors that you need to consider are described in the following sections.

Diary and character field size limit

When specifying query criteria, you cannot use diary fields or character fields that contain more than 254 characters. The database system does not support qualifications on these field types. If you specify a qualification for one of these field types, you will receive an error.

If your site has purchased the Full Text Search (FTS) capability of AR System, you can perform searches on fields that are enabled and indexed for FTS.

Supported wildcards

The only wildcard characters supported in the LIKE comparison are the percent symbol (%) and the underscore (_). If you want to search for these characters, include a backslash (\) before the character (for example, \%). There is no support for sets or ranges of values.

This limitation applies only to queries that search for entries in the database. Wildcards are fully supported in filter, escalation, and active link qualifications and in pattern specifications for character fields.

Modulo operator

The modulo operator (%) is not supported and cannot be used in any arithmetic operations that search for entries in the database. The modulo operator is fully supported in filter, escalation, and active link qualifications and set field values.

Maximum number of database connections

You are limited to the maximum connections configured on your Informix database. If you are operating in a multiprocess server environment, be aware that each server process uses a connection.
Shared libraries

Because the AR System uses shared libraries on all platforms when using Informix, ESQL/C must be installed prior to AR System installation. Additionally, you must manually specify the path to the ESQL/C libraries by setting the shared library path equal to the paths in the following examples:

- **HP-UX:**
  ```bash
  $INFORMIXDIR/lib:$INFORMIXDIR/lib/esql:$SHLIB_PATH
  ```

- **Solaris:**
  ```bash
  $INFORMIXDIR/lib:$INFORMIXDIR/lib/esql:$LD_LIBRARY_PATH
  ```

- **AIX:**
  ```bash
  $INFORMIXDIR/lib:$INFORMIXDIR/lib/esql:$LIBPATH
  ```

Accessing external databases with Direct SQL

If you are using an Informix database on your AR System server to access an external Informix database through direct SQL, both databases must have the same options set. The AR System is installed with log options and non-ANSI options by default.

Using Microsoft SQL Server with AR System

Microsoft SQL Server behaviors that you need to consider are described in the following sections.

Diary and Character field qualifications

When you specify search criteria for a field that contains more than 8000 characters or a diary field, you must use the `LIKE` operator. If you use any other relational operator, you will receive an error.

Case sensitivity in queries

By default, Microsoft SQL Server search criteria is in dictionary order and is case-insensitive. You can, however, specify an option that enables case-sensitive searches. For more information, see your Microsoft SQL Server documentation.
Using Oracle with AR System

When specifying search criteria, you cannot use diary fields or character fields that contain more than 4000 characters. The database system does not support qualifications on these field types. If you specify a qualification for one of these field types, you will receive an error. An exception to this rule is if you change the Oracle-Search-On-Clob setting option in the ar.conf (ar.cfg) file. If you set this option to true, you can perform a string search (without wildcards) on these field types. For more information about the ar.conf or ar.cfg file, see the Configuring guide.

For searches on database entries, the only wildcard characters supported in the LIKE comparison are the percent symbol (%) and the underscore (_). There is no support for sets or ranges of values. Wildcards are fully supported in filter, escalation, and active link qualifications and in pattern specifications for character fields.

Using Sybase with AR System

This section describes Sybase behaviors that you need to consider.

Diary and character field qualifications

When you specify query criteria for a field that contains more than 255 characters or a diary field, you must use the LIKE operator. If you use any other relational operator, you will receive an error.

For decimal fields, a NULL value is read from the database as 0.00 and not as a NULL value. This is due to an incorrect return from the Sybase database library.

Case-insensitive queries

By default, query criteria is case sensitive. You can, however, specify an option that allows for case-insensitive queries. For more information, see your Sybase documentation.
Issues with AR System joins

The following issues pertain to AR System joins and Sybase databases:

- With Sybase databases, you cannot nest outer-joined AR System forms.
- When opening an outer join form in modify mode, the database operation might fail. If it does fail, you will receive AR System error message 552 and Sybase error message 4426.
- Sybase does not support long character or diary fields in an outer join form.
- In the database, long character fields and diary fields are implemented as text columns.
- If you try to query on a diary or long text field contained in the inner table of an outer join, Sybase error 7114 will cause arserverd to crash. (Sybase Change Request #122344)

Sybase character sets

The following issues pertain to Sybase database character sets:

- If your Sybase server is configured to use the ISO-8859-1 character set, you must include the following line in your ar.conf file:

  ```
  Sybase-Character-Set: iso_1
  ```

- If you experience character conversion errors, contact Sybase Support for help matching the Sybase client (arserverd process) character set with your Sybase server character set.

- The database removes trailing spaces that you add to names, menu labels, and field labels in BMC Remedy Administrator.

SQL statement length limit

You cannot submit an SQL statement longer than 5197 characters to a Sybase database. If you do, the AR System server will return an error citing incorrect syntax.
Database types and data types

The following sections describe how each database uses data types for its columns.

**DB2 data types**

AR System uses seven different DB2 data types for its columns: `int`, `float`, `varchar`, `longvarchar`, `clob`, `decimal`, and `blob`. AR System fields use these data types as follows:

<table>
<thead>
<tr>
<th>Field type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer, selection, and timestamp</td>
<td><code>int</code></td>
</tr>
<tr>
<td>Real</td>
<td><code>float</code></td>
</tr>
<tr>
<td>Decimal</td>
<td><code>decimal</code></td>
</tr>
<tr>
<td>Character fields, with a defined maximum that is 255 bytes or fewer</td>
<td><code>varchar</code></td>
</tr>
<tr>
<td>Character fields, with a defined maximum from 256 bytes up to 32700 bytes</td>
<td><code>longvarchar</code></td>
</tr>
<tr>
<td>Diary fields and character fields, with no maximum or a maximum over 32700 bytes</td>
<td><code>clob</code> (up to 1 MB)</td>
</tr>
<tr>
<td>Attachment</td>
<td><code>blob</code> (up to 1 GB)</td>
</tr>
</tbody>
</table>

**Note:** Trim, control, table, column, page holder, page, view, and display-only fields do not require any storage in the data tables, so no column is created for them.
Informix data types

AR System uses four different Informix data types for its columns: \texttt{int}, \texttt{float}, \texttt{varchar}, and \texttt{byte}. AR System fields use these data types as follows:

<table>
<thead>
<tr>
<th>Field type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer, selection, and time stamp</td>
<td>\texttt{int}</td>
</tr>
<tr>
<td>Real</td>
<td>\texttt{float}</td>
</tr>
<tr>
<td>Decimal</td>
<td>\texttt{char}</td>
</tr>
<tr>
<td>Character fields, with a defined maximum of 255</td>
<td>\texttt{varchar}</td>
</tr>
<tr>
<td>bytes or fewer</td>
<td></td>
</tr>
<tr>
<td>Diary fields and character fields, with no</td>
<td>\texttt{byte}</td>
</tr>
<tr>
<td>maximum length or a maximum length of more</td>
<td></td>
</tr>
<tr>
<td>than 255 bytes</td>
<td></td>
</tr>
<tr>
<td>Attachment</td>
<td>\texttt{byte} (up to 2 GB)</td>
</tr>
</tbody>
</table>

\textbf{Note:} Trim, control, table, column, pageholder, page, view, and display-only fields do not require any storage in the data tables, so no column is created for them.
**Microsoft SQL data types**

AR System uses five different Microsoft SQL data types for its columns: `int`, `float`, `varchar`, `text`, and `image`. AR System fields use these data types as follows:

<table>
<thead>
<tr>
<th>Field type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer, selection, and time stamp</td>
<td><code>int</code></td>
</tr>
<tr>
<td>Real</td>
<td><code>float</code></td>
</tr>
<tr>
<td>Decimal</td>
<td><code>char</code></td>
</tr>
<tr>
<td>Character fields, with a defined maximum length of 8000 bytes or fewer</td>
<td><code>varchar</code></td>
</tr>
<tr>
<td>Diary fields and character fields, with no maximum length or a maximum length of more than 8000 bytes</td>
<td><code>text</code></td>
</tr>
<tr>
<td>Attachment up to 2 GB</td>
<td><code>image</code></td>
</tr>
</tbody>
</table>

**Note:** Trim, control, table, column, page holder, page, view, and display-only fields do not require any storage in the data tables, so no column is created for them.
Oracle data types

AR System uses five different Oracle data types for columns: `number (15, 0)`, `float`, `varchar`, `blob`, and `clob`. AR System fields use these data types:

<table>
<thead>
<tr>
<th>Field type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer, selection, and time stamp fields</td>
<td><code>number (15, 0)</code></td>
</tr>
<tr>
<td>Real fields</td>
<td><code>float</code></td>
</tr>
<tr>
<td>Decimal</td>
<td><code>char</code></td>
</tr>
<tr>
<td>Character fields, with a defined maximum of 4000 bytes or fewer</td>
<td><code>varchar</code></td>
</tr>
<tr>
<td>Diary fields and character fields, with an unlimited length or a defined maximum length of more than 4000 bytes</td>
<td><code>clob</code></td>
</tr>
<tr>
<td>Attachment fields up to 4 GB</td>
<td><code>blob</code></td>
</tr>
</tbody>
</table>

For the Oracle RDBMS, the default maximum attachment size is 2 GB. (This was increased from 1 MB to 2 GB.) You can adjust the maximum attachment size by updating the `db-max-attach-size` configuration parameter in your `ar.conf` (or `ar.cfg`) file. For more information about the `ar.conf` or `ar.cfg` file, see the Configuring guide.

The maximum value allowed is limited by your server operating system and configuration.

**Note:** Attachment fields created by a pre-7.0 AR System server will remain as `long raw` data type. New attachment fields will use the Oracle `blob` type.

**Note:** Trim, control, table, column, page holder, page, view, and display-only fields do not require any storage in the data tables, so no column is created for them.
Chapter 1—Using relational databases with AR System

The AR_SERVER_INFO_ORACLE_CLOB_STORE_IN_ROW server information setting controls the Oracle CLOB storage. The default value of this setting is \textit{FALSE}, which causes new LOBs to be created “out row.” If the setting is \textit{TRUE}, all CLOBs to be created are “in row.” The corresponding AR configuration setting is \texttt{Oracle-Clob-Storage-In-Row}.

\section*{Sybase data types}

AR System uses five different Sybase data types for its columns: \texttt{int}, \texttt{float}, \texttt{varchar}, \texttt{text}, and \texttt{image}. AR System fields use these data types as follows:

<table>
<thead>
<tr>
<th>Field type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer, selection, and time stamp fields</td>
<td>\texttt{int}</td>
</tr>
<tr>
<td>Real fields</td>
<td>\texttt{float}</td>
</tr>
<tr>
<td>Decimal</td>
<td>\texttt{char}</td>
</tr>
<tr>
<td>Character fields, with a defined maximum length of 255 bytes or fewer</td>
<td>\texttt{varchar}</td>
</tr>
<tr>
<td>Diary fields and character fields, with no maximum length or a maximum length of more than 255 bytes</td>
<td>\texttt{text}</td>
</tr>
<tr>
<td>Attachment fields up to 2 GB</td>
<td>\texttt{image}</td>
</tr>
</tbody>
</table>

Trim, control, table, column, page holder, page, view, and display-only fields do not require any storage in the data tables, so no column is created for them.

\section*{The AR System data dictionary}

The AR System data dictionary is composed of tables that contain the structural definitions of all the forms, filters, escalations, active links, character menus, and containers that are entered into the system (see Figure 1-1 on page 25, Figure 1-2 on page 26, and Figure 1-3 on page 27). Together, these tables contain the complete definition of the structures and workflow in your implementation of AR System. As you add new structures or alter existing structures in your system, appropriate updates are made to these tables to reflect the changes.
Figure 1-2: AR System Data Dictionary: Active Links, Filters, and Escalations

arschema
  name
  schemaId
  nextId
  schemaType

actlink
  actlinkId

actlink_gotoaction
  actlinkId
  actionIndex

actlink_goto
  actlinkId
  actionIndex

actlink_wait
  actlinkId
  actionIndex

actlink_gotoaction
  actlinkId
  actionIndex

actlink_set
  actlinkId
  actionIndex

actlink_process
  actlinkId
  actionIndex

actlink_message
  actlinkId
  actionIndex

actlink_call
  actlinkId
  actionIndex

actlink_close
  actlinkId
  actionIndex

actlink_commit
  actlinkId
  actionIndex

actlink_auto
  actlinkId
  actionIndex

actlink_sql
  actlinkId
  actionIndex

actlink_dde
  actlinkId
  actionIndex

actlink_push
  actlinkId

actlink_set_char
  actlinkId
  actionIndex

actlink_open
  actlinkId
  actionIndex

support_file
  id

See Figure 1-1

BMC Remedy Action Request System 7.0

Chapter 1—Using relational databases with AR System
The first table is named `control`, and it contains one row. The columns contain information about the version of the database, `dbVersion`, and a set of numbers identifying the next available ID for the various structure items that can be created.
Tables for forms

A set of tables is used to define the form (known as schema in the database tables). The arschema table contains information about the form definitions. The four main fields in the arschema table are:

- **schemaId** — The unique internal ID for the form (which does not change, regardless of changes to the form).
- **name** — The administrator name for the form.
- **schemaType** — The type of form (regular, join, view, or vendor).
- **nextId** — The next available ID for a new entry for that form.

The following set of tables holds information associated with the form definition:

- **schema_group_ids** — Defines which groups have access to the form.
- **schema_vendor** — Defines how the form is attached to a database table.
- **schema_list_fields** — Defines how the form is attached to an ARDBC data source.
- **schema_join** — Defines how the form is joined, if applicable.
- **schema_index** — Defines the indexes for the form.
- **schema_sort** — Defines the default sort order for the form.
- **schema_archive** — Defines the archive information for the form.

Every form contains at least one view user interface (VUI) that represents the various layouts and fields that hold the data for the form. The vui table contains information about each VUI in each form. Every VUI is identified by the combination of the schemaId that connects the VUI to a form, and the vuid that identifies that VUI within the form.
Tables for fields

The `field` table contains all of the information (except for the display information) about each field in each form. Every field is identified by the combination of the `schemaId` that connects the field to a form and the `fieldId` that identifies the field within the form.

The `vuiId` and `fieldId` are unique within the form, so a single ID identifies either a VUI or a field. The `field_dispprop` table contains information used to define how the field is displayed in the form. The `field_permissions` table contains information about the permissions of various groups to the individual fields. The following series of additional tables hold information that is specific to each data type: `field_int`, `field_real`, `field_char`, `field_diary`, `field_dec`, `field_curr`, `field_date`, `field_enum`, `field_attach`, `field_table`, `field_column`, `field_view`, `field_display`, and `field_enum_values` (there is no additional data for timestamp, trim, or control fields).

If a field is located in a join form, there is an additional entry in the `join_mapping` table. This entry contains the definition of how this field is connected to the field in the base forms that make up the join form.

If a field is located in a view form, there is an additional entry in the `view_mapping` table. This entry contains the definition of how this field is connected to the field in the base forms that make up the view form.

If a field is located in a vendor form, there is an additional entry in the `vendor_mapping` table. This entry contains the definition of how this field is connected to the field in the base forms that make up the vendor form.

Tables for menus

The `char_menu` table contains an entry for each menu, and tags each with a `charMenuId`. A set of tables associated with the `char_menu` table (linked by `charMenuId`) provides the details about the various types of character menus:

- `char_menu_list`
- `char_menu_query`
- `char_menu_file`
- `char_menu_sql`
- `char_menu_dd`
Tables for filters

The filter table contains an entry for each filter, and tags each with a filterId. Tables associated with the filter table (linked by filterId) provide the details about the various actions defined for each filter:

- filter_notify
- filter_notify_ids
- filter_message
- filter_log
- filter_set
- filter_process
- filter_push
- filter_sql
- filter_gotoaction
- filter_call
- filter_exit
- filter_goto

Tables for escalations

The escalation table contains an entry for each escalation, and tags each with an escalationId. Because escalations and filters are so tightly linked, the information about actions for escalations is stored in the same tables as the filter actions. The escalationId and the filterId are unique within the table, so a single ID identifies either a filter or an escalation.

Tables for active links

The actlink table contains an entry for each active link, and tags each with an actlinkId. Tables associated with the actlink table (linked by actlinkId) provide the details about the various actions that are defined for each active link:

- actlink_macro
- actlink_macro_parm
- actlink_set
- actlink_process
- actlink_message
- actlink_set_char
- actlink_dde
- actlink_gotoaction
- actlink_wait
- actlink_goto
- actlink_exit
- actlink_call
- actlink_close
- actlink_commit
- actlink_open
- actlink_sql
- actlink_push
- actlink_auto

The support_file table stores report definitions. Finally, the table actlink_group_ids contains the list of groups that can execute the active link.
Tables for mapping workflow

A set of mapping tables associates each filter, escalation, or active link with all its forms, allowing administrators to create shared workflow. The `filter_mapping` table contains the `filterId` and `schemaId` for each entry, creating a link between each filter and form. The `escal_mapping` table associates escalations with forms by storing the `escalationId` and `schemaId` for each entry. In a similar way, the `actlink_mapping` table associates active links with forms by storing the `actlinkId` and `schemaId` for each entry.

Tables for containers

The `arcontainer` table contains an entry for each container, and tags each with an `containerId`. Containers are used to define guides, applications, packing lists, workspaces, and web services. The three main fields in the table are:

- `containerId` — The unique internal ID for the container.
- `name` — The administrator name for the container.
- `containerType` — The type of container.

The following set of tables holds information associated with the container definition:

- `arctr_group_ids` — Defines which groups have access to the container.
- `arctr_subadmin` — Defines which groups have subadministrator access to the container for containers that are not owned.
- `arreference` — Defines the references for each container.
- `arref_group_ids` — Lists the owners for the container.
- `cntnr_ownr_obj` — Defines group access permissions for external references.

A list of references defines the components that belong to each container. For example, a container might reference forms, workflow objects, and other internal and external objects that make up an application or guide. Each container can have zero, one, or multiple references. Each reference is identified by the `containerId` of the container to which it belongs, and by the `referenceId` that identifies the object itself.
All references are described by reference type, data type, reference order number, label, and description. Internal references store the referenceObjId. External references store a short value or long value that describes the external reference. The arref_group_ids table can have zero, one, or multiple group entries that define group access permissions for each external reference. Each entry describes a groupId permitted to access an external reference.

For more information about using containers to create guides, see the Workflow Objects guide. For more information about the data structures used to define containers, see the C API Reference guide.

Creating tables for forms

The arschema table holds information about each form, including form name, schema ID and next request ID. When a new regular form is created, three or more of the following tables are created in the database to hold the information (requests) for that form:

- "Main data table" on page 32
- "Status history table" on page 33
- "Attachment tables" on page 34
- "Currency table" on page 35
- "Indexing" on page 36

Main data table

Each form has an associated main data table that holds all the information for that form. The main data table contains a column for each field except Attachments and Status History. Each main data table or view (for join forms) is named with a T followed by the unique ID (schemaId) for the form (for example, T3). You can find the ID by searching the arschema table by the name column and retrieving the schemaId value. The ID does not change regardless of changes made to the form, so the table name remains the same. In Figure 1-1 on page 25, the main data table is labeled Tn.
All columns in each table or view are named with a C followed by the unique ID for the field within the form. For example, the Submitter field is C2. The ID for the field does not change, the creator of the field can assign the ID. Every ID is unique within a form, so there is never an issue with duplicate names. After an ID has been assigned, it cannot be changed, regardless of any changes to the field. For information about reserved and core IDs, see the Form and Application Objects guide.

For join forms, if there is an attachment field on the form, a column is added to the Main Data view. The contents of this column are a concatenation of the C, CO, and CC columns of the Attachment Details table. If new attachments are added to the base form, the view is updated. See “Attachment tables” on page 34.

Because AR System must retain the IDs of the requests in the underlying table to form the ID of a join form entry, there are a few extra columns and some special handling for column C1. AR System creates a series of columns for each regular form that is involved in the join tree. The columns are named with an E followed by a zero-based index (three regular tables would be named E0, E1, and E2). These columns point to the corresponding entry IDs (column C1) of the regular forms. The C1 column for the join form is determined by concatenating the entry IDs of the regular forms (in the E columns) separated by vertical bars (|).

**Status history table**

The status history table contains all the information for the Status History field. Each status history table or view (for join forms) is named with an H followed by the unique ID for the form (for example, H3). The ID is the same ID that the main data table or view uses, and the name of each also remains unchanged. Every main data table has an associated status history table. In Figure 1-1 on page 25, the status history table is labeled Hn.

The most important column in this table is the entryId. It provides a reference to the C1 column of the main data table. (Column C1 is always the RequestID.) This column is followed by a series of one or more column pairs. There is one pair for each state defined for the Status field. The columns are named with a prefix followed by the numeric representation for each state. The prefixes are U for the user name and T for the time the entry was last changed to the corresponding state. The numeric value is zero-indexed. For example, a form with three states for the Status field would yield a table with seven columns: entryId, U0, T0, U1, T1, U2, and T2.
If new status values are added, appropriate columns are added to this table to reflect the new states. If states are deleted, the columns are left in the table, enabling the states to be added again in the future. The data for the status values is stored in the database as an integer that relates to the order of the choices. If you add values at the beginning or in the middle of existing values, other values in the list might change.

Unlike in regular forms, for join forms, the Status History field is optional. If it is present, the Status and Status History fields must be from the same base table. If there is no Status History field in the form, the Status History table does not exist. If a Status History field is present, it is defined as an exact duplicate view of the status history table or view of the base form to which it is connected. The only difference is the name of the view. For more information about the Status History field, see the Form and Application Objects guide.

**Note:** View and vendor forms do not have corresponding status history tables.

---

**Attachment tables**

There are two attachment tables: the attachment details table and the attachment data table.

**Attachment details table**

The Attachment details table contains information for the properties of Attachment fields. For every Attachment field in the form, a separate table is created to store the attachment value.

The Attachment details table is named with a B followed by the unique ID for the form (for example, B3). In Figure 1-1 on page 25, the attachment details table is labeled Bn. An attachment details table with one column (C1) is created with every form.

For every attachment field added to any attachment pool on the form, three new columns are added. Each column is named with C, CO, or CC, followed by the attachment field ID. For example, the three columns added for one attachment might be called C536870920, CO536870920, and CC536870920, where 536870920 is the attachment field ID.
The `c` column stores the full path name of the attached file. The `co` column stores the original size (in bytes) of the attached file. The `cc` column stores the compressed size (in bytes) of the attachment file.

**Attachment data table**

For each attachment field on a form, an attachment data table is created. The attachment data table is named with a `B` followed by the unique ID for the form, followed by `c`, followed by the attachment field ID. For example, the attachment data table might be called `B7C536870920`, where `7` is the schema ID, and `536870920` is the attachment field ID. In Figure 1-1 on page 25, the attachment data table is labeled `BnC<fieldID>`.

The Attachment data table has two columns: one that holds the RequestID (`entryId`) and one that holds the data from the file. The column holding the data is named with a `C` followed by the attachment field ID. For example, the data column might be named `C536870920`, where `536870920` is the attachment field ID.

**Currency table**

Where a field in a form typically has one corresponding column in the main data table, the currency field has several columns and, therefore, a unique naming convention to distinguish the extra columns. Whereas typical fields follow the naming convention described in "Main data table" on page 32 (all columns in each table or view are named with a `C` followed by the unique ID for the field within the form), the currency field is named with a `C` followed by the unique ID for the currency field and a unique suffix for each additional currency column stored in the database.

The currency suffixes used to name the additional currency columns are defined in the following table.

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Currency Column Represented</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>Decimal value</td>
</tr>
<tr>
<td>C</td>
<td>Code associated with decimal value</td>
</tr>
<tr>
<td>D</td>
<td>Timestamp or Date established as the conversion date</td>
</tr>
<tr>
<td>&lt;Type of currency being used&gt; (USD, EUR, JPY, and so on)</td>
<td>Value of specified type of functional currency</td>
</tr>
</tbody>
</table>
For example, the columns for a currency field might be called C536870913V, C536870913C, C536870913D, or C536870913USD.

Indexing

Indexes are automatically maintained for all the tables created by AR System. Some are defined by AR System, and others are defined by an administrator. If a table is restructured through AR System, all indexes are recreated for the new table.

The main data table has an index supported by AR System defined for the C1 column. This column corresponds to the Request ID field of the form. (In Microsoft SQL databases, the table is created using a primary key, which enables database replication.) The index is a unique index and is used extensively as the main index of the table.

For the main data table, the administrator can create additional indexes for the form. The indexes are unique only if defined as such. These additional indexes are not clustered because there can be only one clustered index, and it is reserved for the main index supported by AR System.

The status history table has an index supported by AR System defined on the entryId column. This column also corresponds to the Request ID field of the form. The index is a unique clustered index and is the main index of the table. AR System does not create additional indexes for the status history table.

The Attachment Data and the Attachment Details tables each have unique indexes supported by AR System. For the Attachment Data table, the index is defined on the entryId column, and for the Attachment Details table, the index is defined on the C1 column. These columns correspond to the Request ID field of the form. The administrator cannot create additional indexes.

Indexing a currency field requires special considerations. Because a currency field is represented by multiple columns in the main data table, multiple columns are indexed. Standard queries against a currency field could potentially use any of several different columns, depending on the currency type specified. To provide comprehensive coverage, indexing a currency field requires an index for the value column, the type column, and for each functional currency column. This can produce significant overhead for the main data table. Therefore, consider indexing a currency field carefully before doing so.
Note: Indexes cannot be created for join forms. The form definition is just a view and the database does not support indexes for views. Indexes defined for the underlying tables are available and are used when performing operations against the join form.

For view forms, you must create indexes within the database. The AR System cannot create indexes on the view of the external database’s table.

For vendor forms, the administrator who implemented the ARDBC data source must define and document a mechanism to establish indexes on the underlying data. For more information about ARDBC, see the C API Reference guide.

SQL views

For each table that is built in the system (except for the attachment tables), an SQL view is automatically created. This view uses the form name as the view name and the field names (not a display label in one of the views) as the column names. The names are created by using the following rules:

- All alphabetic and numeric characters remain as defined.
- All other characters are converted to an underscore (_).
- If the first character is not alphanumeric, a leading $A$ is added to the name.
- If the name of a field is blank, a field name with a leading $A$ followed by the $fieldId$ is used.
- If the name is one of the reserved words for the database, the string $_x$ is appended.

The name of the table must be unique among all the table names after the conversion. If it is not unique, a set of three digits is added to the end of the name (with the name truncated, if necessary, to fit the maximum length allowed for an SQL name). First, the digits $001$ are tried. If that is unique, the new name contains $001$ at the end. If $001$ does not make the name unique, $002$ is tried, then $003$, and so on until a unique name is found. Column names must also be unique, so the same naming convention is used.
The SQL view of the status history table follows the same strategy as the SQL view of the base table. The name of the table is created by adding SH_ to the front of the name of the base table view. The column names are mapped to the name of the Request ID field and the names of each of the Status values with _TIME and _USER appended. So, a form with two states, New and Closed, would end up with columns in the view named Entry_Id, New_USER, New_TIME, Closed_USER, and Closed_TIME.

These SQL views are recreated whenever the name for the field is changed or when a change is made to the form that affects the underlying table (deleting a field, adding a field, or changing the length of a field).

You can use the view or the base tables to read data from the database. The SQL views are especially useful when using a third-party report writer, because the names of the various tables and columns are easier to use than the internal, numeric representations used in the base tables.

**Updating tables when AR System forms change**

When you restructure a regular form by adding new fields, deleting old fields, or changing the length of existing fields, AR System restructures the underlying database to reflect those changes. This section covers the following topics:

- “Adding fields” on page 39.
- “Deleting fields” on page 39.
- “Changing character field lengths” on page 40.

**Note:** This section does not apply to join forms. Adding or deleting a field from a join form simply adds or removes the reference to the field in the underlying form. You cannot change the length of a field, because it is defined by the underlying form.

For view forms, the database view is recreated when any fields are added or removed. The database is not recreated if field properties (for example, length) are changed.
Important: Consider performance when you restructure your database. When a table is restructured, the performance impact of the operation is dependent on the amount of data in the affected table. If the table contains a large amount of data, the restructuring operation might take a long time, and it might take a large amount of log and data space within the system. Accordingly, plan updates to occur during hours when access to data in the system is not critical.

Adding fields

When you add a new field to a form, a new column is added to the main data table by using the ALTER TABLE command. The structure of the database is changed to add the new column according to the rules stated in “Creating tables for forms” on page 32.

The data for the new field for any existing entries is NULL even if it is a required field. You can change these values at any time. When the field is added, it can be used for all existing or future entries. Use the BMC User Modify All operation to assign a default value for the field.

Deleting fields

Deleting a field from a form physically removes the field from the database. The corresponding column and all data that is associated with the field are removed. The following sections describe how each database deletes fields.

DB2

In a DB2 database, the following syntax is used to build a new table that contains all the structure and data of the original table except for the deleted column:

```
CREATE TABLE <new table, excluding the field being deleted>
INSERT INTO <new table> AS SELECT <all fields, excluding the field being deleted> FROM <old table>
```

After the new table is created, the original is deleted:

```
DELETE TABLE <old table>
```

Any indexes that are defined as part of the form definition are recreated on the rebuilt table.
Informix, Oracle, Sybase, and Microsoft SQL

In the Informix, Oracle, Sybase, and Microsoft SQL databases, the `ALTER TABLE ... DROP ...` syntax is used to remove the column from the table.

Changing character field lengths

The following sections describe how each database changes the length of a character field.

**Note:** The operation of changing character field lengths logs the entire table that is being modified. If this table is large, it consumes a large amount of log space. You might need to expand your system's log space.

**DB2**

In a DB2 database, the length of a character field is changed in one of the following ways:

- If the new length and old length are both <= 255 bytes, the `ALTER TABLE` command is used to change the columns. Neither the table nor the index are recreated.
- For any other change in length, a new column is created with the new length restriction. Then, all the data is copied from the original column to the new column and the original column is deleted from the main data table.

**Informix**

In an Informix database, the length of a character field is changed in one of the following ways:

- If the original size is <= 255 bytes and you decrease the length, no change is made to the table.
- If the original size and the new size are both <= 255 bytes and you increase the length, the `ALTER TABLE ... MODIFY ...` command syntax is used.
If the original size is <= 255 bytes and the new size is > 255 bytes, a new column is created with the new length restriction. Then, all the data is copied from the original column to the new column, the original column is deleted from the main data table, and the column type is changed from varchar to byte.

If the original size is > 255 bytes and the new size is <= 255 bytes, a new column is created with the new length restriction. Then, all the data from the original column is copied to the new column, the original column is deleted, and the data type of the column is changed from byte to varchar.

If the original size and the new size are both > 255 bytes, no change is made to the table, whether you have decreased or increased the length.

**Microsoft SQL**

In a Microsoft SQL database, if the field is created in AR System 5.1 and later, the length of a character field is changed in one of the following ways:

- If the original size is <= 8000 bytes and you decrease the length, no change is made to the table.
- If the original size is > 8000 bytes and the new length is > 8000 bytes, no change is made to the table.
- For any other change in length, a new column is created with the new length restriction. Then, all data from the original column is copied to the new column and the original column is deleted from the main table.

If the field is created in a version of AR System earlier than 5.1, the length of a character field is changed in one of the following ways:

- If the original size is <= 255 bytes and the new length is <=8000 bytes, no change is made to the table.
- If the original size is > 255 bytes and the new length is > 8000 bytes, no change is made to the table.
- For any other change in length, a new column is created with the new length restriction. Then, all data from the original column is copied to the new column and the original column is deleted from the main data table.
Oracle

Table 1-1 shows the changes that AR System makes to an Oracle database when you change the length of character fields. Note that the handling of field length changes depends on the initial size of the field, and whether the field was created in the current version or a previous version of AR System.

Table 1-1: Changing character field lengths for Oracle

<table>
<thead>
<tr>
<th>Administrator Action</th>
<th>AR System Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decreases the length of a field from &gt; 4000 bytes to &lt;= 4000 bytes.</td>
<td>Adds a new varchar column to the main data table; copies the data from the clob column to the new column; deletes the old column.</td>
</tr>
<tr>
<td>Decreases the length of a field from &lt;= 4000 bytes to less than 4000 bytes.</td>
<td>No restructuring performed.</td>
</tr>
<tr>
<td>Increases the length of a field from &lt;= 4000 bytes to &gt; 4000 bytes.</td>
<td>Adds a new clob column to the main data table; copies the data from the varchar column to the new column; deletes the old column.</td>
</tr>
<tr>
<td>Increases the length of a field from &gt; 4000 bytes to another value also &gt; 4000 bytes.</td>
<td>No restructuring performed.</td>
</tr>
</tbody>
</table>

Sybase

In a Sybase database, the length of a character field is changed in one of the following ways:

- If the original size is <= 255 bytes and you decrease the length, no change is made to the table.
- If the original size is > 255 bytes and the new length is > 255 bytes, no change is made to the table.
- For any other change in length, a new column is created with the new length restriction. Then, all data from the original column is copied to the new column and the original column is deleted from the main data table.
Related information

For general information about relational databases, see Introduction to Database Systems, by C.J. Date. The following sections also offer suggested reading for the databases that AR System supports. Depending on the version of relational database you are using, the titles of the following books might differ slightly.

**DB2**

A Complete Guide to DB2 Universal Database by Don Chamberlin

**Informix**

- Informix Guide to SQL: Tutorial
- Informix Guide to SQL: Reference and Syntax
- Informix Guide to SQL: Reference and Using Triggers

For a discussion of the structure used by previous versions of AR System for the Informix database, see the technical notes available at the Customer Support website (http://supportweb.remedy.com).

**Oracle**

- SQL Reference Manual
- Oracle Administrator’s Guide

**Sybase**

- Sybase Commands Reference Manual
- Sybase Administration Guide

**Microsoft SQL**

- Transact-SQL Desk Reference: For Microsoft SQL Server
- Microsoft SQL Server 2000 Administrator’s Companion
Unicode database support

The Unicode database option provides support for Unicode, giving you the option to have multi- and single-byte forms, data, and workflow stored in the same database or database instance.

**Note:** To use AR System with the Unicode database option to support multiple languages using a shared database, you must install an English version of AR System server before you install multi-language AR System servers.

The Unicode support model in the 7.0 architecture allows you to use multiple languages on one AR System server. You are no longer restricted by the locale of the OS that you are running on. See the Unicode white paper at [http://supportweb.remedy.com](http://supportweb.remedy.com) for more information about using AR System server with Unicode.

Unicode compliance versus Unicode database support

A product that provides Unicode compliance is written using the Unicode character coding system, which means Unicode data can flow from database to client without any conversion occurring. A product that provides Unicode database support allows the database to contain Unicode characters, but converts them between the database and the product. Any product accessing the database, however, still uses a native character set.

AR System, AR System server, and AR System clients do not provide Unicode compliance. The AR System server provides Unicode database support, making it possible to store AR System objects and data in a Unicode database, while still using a native character set.

This means you might need more than one AR System server to support multiple languages.

**WARNING:** For version 6.x, if you modify records using an AR System server that uses a different locale than the one used to store the data, you will corrupt those records. For example, if you store data using German characters from a German version of AR System server, and then modify those records using Japanese characters from a Japanese version of AR System server, the data will be corrupted for the German system.
**WARNING:** For version 7.0, if you modify definitions using an AR System server that uses a different locale than the one used to store the data, you will corrupt those definitions. For example if you modify a German view with an Admin Tool running on a Japanese machine you will corrupt those characters.

---

**Creating a Unicode database**

Each database type supported by AR System supports Unicode at one of the following levels:

<table>
<thead>
<tr>
<th>Unicode support level</th>
<th>Database type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database instance</td>
<td>Sybase, Informix, Oracle</td>
<td>For these database types, you need to configure the database instance before you install the new AR System database.</td>
</tr>
<tr>
<td>Specific column type</td>
<td>Microsoft SQL</td>
<td>Uses Unicode data types nchar, nvarchar, and ntext.</td>
</tr>
<tr>
<td>Database</td>
<td>DB2</td>
<td></td>
</tr>
</tbody>
</table>

During installation, the AR System installer gives you the option of creating a Unicode database. You can safely do this if you meet the following two requirements:

- You are not installing on an existing AR System database.
- Your database supports Unicode at the column or database level, or you have configured your database instance for databases that support Unicode at the database-instance level.

See Installing AR System for detailed overwrite, installation, and upgrade information for Unicode databases.

**WARNING:** If you have an existing AR System database, you must first migrate it to Unicode before upgrading your AR System server. If you choose the Unicode database option during an Upgrade install against a non-Unicode AR System database, you will corrupt your database. See the next section, “Migrating existing databases to Unicode.”
Migrating existing databases to Unicode

If you are upgrading an AR System that already has a database, you must migrate the existing database to Unicode before proceeding with the AR System upgrade installation. This ensures your data integrity.

See your database documentation for database migration procedures. See Installing AR System for detailed overwrite, installation, and upgrade information for Unicode databases.

Migrating an Oracle database

To migrate an Oracle database, follow these general steps. See your Oracle documentation for detailed information.

Step 1  Confirm your Unicode Oracle database is using character set AL32UTF8.

The character set is defined during the creation of the new Unicode database. There is no change on a character set for an existing database. During the creation of the database, the response to the prompt for character set is AL32UTF8. The Oracle database engine will take care of any conversion required during import of the original (non-Unicode) into the new database.

Step 2  Perform a full export and import on the whole database. See your Oracle documentation for more information.

Note: UTF-8 columns usually store fewer characters compared to non-Unicode columns. In these cases, you might be introducing data truncation. See your Oracle documentation for more information.
Migrating a Microsoft SQL Server database

To migrate an Microsoft SQL Server database, follow these general steps. See your Microsoft SQL Server documentation for detailed information.

**Step 1** Create new columns in the target database that correspond to the source database as shown in the following table:

<table>
<thead>
<tr>
<th>Source column type</th>
<th>Target column type</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>nchar</td>
</tr>
<tr>
<td>varchar</td>
<td>nvarchar</td>
</tr>
<tr>
<td>text</td>
<td>ntext</td>
</tr>
</tbody>
</table>

**Step 2** Migrate your data on a column-by-column basis.

**WARNING:** AR System will not work if you have both Unicode and non-Unicode columns in the database.

Migrating a DB2 database

For DB2 databases, the database character set is a configuration parameter that you cannot update. Therefore, existing non-Unicode DB2 AR System databases cannot be migrated to Unicode.

Migrating a Sybase database

To migrate a Sybase database, follow these general steps. See your Sybase documentation for detailed information.

**Step 1** Export the existing database.

**Step 2** Change the Sybase default character set to UTF8.

**Step 3** Import the data back into the Sybase database.

**Step 4** Update the tables containing text columns.
Migrating an Informix database

To migrate an Informix database, follow these general steps. See your Informix documentation for detailed information.

**Step 1** Set the `DB_LOCALE` Informix environment variable to `xx_xx.utf8`, where `xx_xx` is the language and territory.

**Step 2** Set the `CLIENT_LOCALE` AR System server environment variable to reflect the correct Informix client locale on the machine that runs AR System server. See your Informix documentation for details.

**WARNING:** AR System will not work if you have both Unicode and non-Unicode columns in the database.
This chapter includes sets of SQL commands that define the AR System data dictionary for the databases supported by AR System.

The following topics are provided:

- DB2 Universal (page 50)
- Informix (page 71)
- Oracle (page 89)
- Sybase and Microsoft SQL Server (page 106)
The following set of SQL commands define the AR System data dictionary for DB2 Universal. For an explanation of these commands, see A Complete Guide to DB2 Universal Database.

```sql
CREATE TABLE control
   (dbVersion       int           not null,
    schemaId        int           not null,
    filterId        int           not null,
    serverId        int           not null,
    containerId     int           not null,
    actlinkId       int           not null,
    adminExtId      int           not null,
    charMenuId      int           not null)

CREATE TABLE arschema
   (name            varchar(254)  not null,
    schemaId        int           not null,
    schemaType      int           not null,
    timestamp       int           not null,
    owner           varchar(254)  not null,
    lastChanged     varchar(254)  not null,
    coreVersion     int           not null,
    numFields       int           not null,
    numVuis         int           not null,
    defaultVui      varchar(254)  not null,
    nextId          int           not null,
    nextFieldId     int           not null,
    maxStatEnums    int           not null,
    upgrdVersion    int           ,
    safeGuard       varchar(254)  not null,
    changeDiary     clob(1M)      ,
    helpText        clob(1M)      ,
    objProp         clob(1M)      ,
    version         varchar(32)   ,
    smObjProp       clob(1M)      )

CREATE UNIQUE INDEX schema_ind
   ON arschema (name) CLUSTER;
CREATE UNIQUE INDEX schema_id_ind
   ON arschema (schemaId)

CREATE TABLE schema_group_ids
   (schemaId       int           not null,
    groupId         int           not null,
    permission      int           not null)

CREATE INDEX schemaGroupIdInd
   ON schema_group_ids (schemaId) CLUSTER;

CREATE TABLE schema_group_ids
   (schemaId       int           not null,
    groupId         int           not null,
    permission      int           not null)

CREATE INDEX schemaGroupInd
   ON schema_group_ids (schemaId)
   CLUSTER;
```
CREATE TABLE subadmin_group
    (schemaId       int           not null,
     groupId         int           not null)
;
CREATE INDEX subadmin_group_ind
    ON subadmin_group (schemaId)
    CLUSTER;
CREATE TABLE schema_list_fields
    (schemaId       int           not null,
     listIndex       int           not null,
     fieldId         int           not null,
     columnWidth     int           not null,
     separatorLen    int           not null,
     separator       varchar(10)   )
;
CREATE INDEX schemaListFieldInd
    ON schema_list_fields (schemaId)
    CLUSTER;
CREATE TABLE schema_sort
    (schemaId        int           not null,
     listIndex       int           not null,
     fieldId         int           not null,
     sortOrder       int           not null)
;
CREATE INDEX schema_sort_ind
    ON schema_sort (schemaId)
    CLUSTER;
CREATE TABLE schema_archive
    (schemaId          int           not null,
     enable            int           not null,
     archiveType       int           not null,
     archiveToForm     int           ,
     archiveToFile     varchar(255)  ,
     queryShort        varchar(255)  ,
     queryLong         clob(1M)      ,
     monthday          int           not null,
     weekday           int           not null,
     hourmask          int           not null,
     minute            int           not null,
     archiveFromForm   int           )
;
CREATE INDEX schema_archive_ind
    ON schema_archive (schemaId)
    CLUSTER;
CREATE TABLE schema_audit
    (schemaId          int           not null,
     enable            int           not null,
     style             int           not null,
     form              int           ,
     queryShort        varchar(255)  ,
     queryLong         clob(1M)      )
;
CREATE  INDEX schema_audit_ind
ON schema_audit (schemaId)
CLUSTER;

CREATE TABLE schema_index
(schemaId           int           not null,
listIndex         int           not null,
numFields         int           not null,
uniqueFlag        int           not null,
indexName         varchar(254)  not null,
f1                int           not null,
f2                int           ,
f3                int           ,
f4                int           ,
f5                int           ,
f6                int           ,
f7                int           ,
f8                int           ,
f9                int           ,
f10               int           ,
f11               int           ,
f12               int           ,
f13               int           ,
f14               int           ,
f15               int           ,
f16               int           )
;

CREATE  INDEX schema_index_ind
ON schema_index (schemaId)
CLUSTER;

CREATE TABLE schema_join
(schemaId          int           not null,
memberA           varchar(254)  not null,
memberB           varchar(254)  not null,
options           int           ,
queryShort        varchar(255)  ,
queryLong         clob(1M)      )
;

CREATE UNIQUE INDEX schema_join_ind
ON schema_join (schemaId)
;

CREATE TABLE schema_view
(schemaId          int           not null,
tableName         clob(1M)      ,
keyField          varchar(254)  not null,
queryShort        varchar(255)  ,
queryLong         clob(1M)      )
;

CREATE UNIQUE INDEX schema_view_ind
ON schema_view (schemaId)
;

CREATE TABLE schema_vendor
(schemaId          int           not null,
vendorName        varchar(254)  not null,
tableName         clob(1M)      )
;

Chapter 2—SQL Definitions of the data dictionary tables
CREATE UNIQUE INDEX schema_vendor_ind ON schema_vendor (schemaId);

CREATE TABLE field
(schemaId int not null,
fieldId int not null,
fieldName varchar(254) not null,
fieldType int not null,
timestamp int not null,
owner varchar(254) not null,
lastChanged varchar(254) not null,
datatype int not null,
option int not null,
createMode int not null,
fbOption int ,
defaultValue varchar(255) ,
changeDiary clob(1M) ,
helpText clob(1M) )
;

CREATE UNIQUE INDEX field_ind ON field (schemaId, fieldId) CLUSTER;
CREATE INDEX field_schema_ind ON field (schemaId);

CREATE TABLE vui
(schemaId int not null,
vuiId int not null,
vuiName varchar(254) not null,
locale varchar(30) ,
vuiType int ,
timestamp int not null,
owner varchar(254) not null,
lastChanged varchar(254) not null,
changeDiary clob(1M) ,
helpText clob(1M) )
;

CREATE UNIQUE INDEX vui_ind ON vui (schemaId, vuiId) CLUSTER;
CREATE INDEX vui_schema_ind ON vui (schemaId);

CREATE TABLE field_dispprop
(schemaId int not null,
fieldId int ,
listIndex int not null,
vuiId int ,
propShort varchar(255) ,
propLong clob(10M) )
;
CREATE UNIQUE INDEX field_disprop_ind ON field_disprop (schemaId, fieldId, listIndex, vuiId);

CREATE TABLE field_int
(schemaId int not null,
fieldId int not null,
rangeLow int ,
rangeHigh int )

CREATE UNIQUE INDEX field_int_ind ON field_int (schemaId, fieldId)
CLUSTER ;

CREATE TABLE field_real
(schemaId int not null,
fieldId int not null,
rangeLow float ,
rangeHigh float ,
arprecision int )

CREATE UNIQUE INDEX field_real_ind ON field_real (schemaId, fieldId)
CLUSTER ;

CREATE TABLE field_diary
(schemaId int not null,
fieldId int not null,
fullTextOptions int )

CREATE UNIQUE INDEX field_diary_ind ON field_diary (schemaId, fieldId)
CLUSTER ;

CREATE TABLE field_char
(schemaId int not null,
fieldId int not null,
maxLength int ,
qbeMatchOp int ,
menuStyle int ,
charMenu varchar(254) ,
pattern varchar(255) ,
fullTextOptions int )

CREATE UNIQUE INDEX field_char_ind ON field_char (schemaId, fieldId)
CLUSTER ;

CREATE TABLE field_enum
(schemaId int not null,
fieldId int not null,
maxEnum int not null,
enumStyle int ,
schemaName varchar(254) ,
serverName varchar(64) ,
nameField int ,
numberField int ,
queryShort varchar(255) ,
queryLong clob(1M) )
CREATE UNIQUE INDEX field_enum_ind ON field_enum (schemaId, fieldId) CLUSTER;

CREATE TABLE field_enum_values
(schemaId int not null,
fieldId int not null,
enumId int not null,
value varchar(254) not null)
;

CREATE INDEX field_enum_val_ind ON field_enum_values (schemaId, fieldId) CLUSTER;

CREATE TABLE field_permissions
(schemaId int not null,
fieldId int not null,
groupId int not null,
permission int not null)
;

CREATE INDEX fieldPermissionInd ON field_permissions (schemaId, fieldId) CLUSTER;

CREATE TABLE field_attach
(schemaId int not null,
fieldId int not null,
maxSize int not null,
attachType int not null,
fullTextOptions int)
;

CREATE UNIQUE INDEX field_attach_ind ON field_attach (schemaId, fieldId) CLUSTER;

CREATE TABLE field_table
(schemaId int not null,
fieldId int not null,
numColumns int not null,
maxRetrieve int not null,
tfSchema varchar(254) not null,
tfServer varchar(64) not null,
queryShort varchar(255),
queryLong clob(1M),
sampleSchema varchar(254),
sampleServer varchar(64))
;

CREATE UNIQUE INDEX field_table_ind ON field_table (schemaId, fieldId) CLUSTER;

CREATE TABLE field_column
(schemaId int not null,
fieldId int not null,
parent int not null,
dataField int not null,
colLength int not null,
dataSource int not null,
dataField int not null,
parent int not null)
CREATE UNIQUE INDEX field_column_ind
ON field_column (schemaId, fieldId)
CLUSTER;

CREATE TABLE field_dec
(schemaId int not null,
fieldId int not null,
rangelow varchar(64),
rangohigh varchar(64),
arprecision int)

CREATE UNIQUE INDEX field_dec_ind
ON field_dec (schemaId, fieldId)
CLUSTER;

CREATE TABLE field_curr
(schemaId int not null,
fieldId int not null,
rangelow varchar(64),
rangohigh varchar(64),
arprecision int,
funcCurr clob(1M),
allowCurr clob(1M))

CREATE UNIQUE INDEX field_curr_ind
ON field_curr (schemaId, fieldId)
CLUSTER;

CREATE TABLE join_mapping
(schemaId int not null,
fieldId int not null,
memberIndex int not null,
mfieldId int not null)

CREATE TABLE field_view
(schemaId int not null,
fieldId int not null,
maxLength int)

CREATE UNIQUE INDEX field_view_ind
ON field_view (schemaId, fieldId)
CLUSTER;

CREATE TABLE field_display
(schemaId int not null,
fieldId int not null,
maxLength int)

CREATE UNIQUE INDEX field_display_ind
ON field_display (schemaId, fieldId)
CLUSTER;

CREATE TABLE field_date
(schemaId int not null,
fieldId int not null,
minDate int,
maxDate int)

56 Chapter 2—SQL Definitions of the data dictionary tables
CREATE UNIQUE INDEX field_date_ind
ON field_date (schemaId, fieldId) CLUSTER;
CREATE UNIQUE INDEX join_mapping_ind
ON join_mapping (schemaId, fieldId);
CREATE TABLE view_mapping
(schemaId int not null,
fieldId int not null,
extField varchar(254) not null)
CREATE UNIQUE INDEX view_mapping_ind
ON view_mapping (schemaId, fieldId);
CREATE TABLE vendor_mapping
(schemaId int not null,
fieldId int not null,
extField varchar(254) not null)
CREATE UNIQUE INDEX vendor_mapping_ind
ON vendor_mapping (schemaId, fieldId);
CREATE TABLE char_menu
(name varchar(254) not null,
charMenuId int not null,
timestamp int not null,
owner varchar(254) not null,
lastChanged varchar(254) not null,
refreshCode int not null,
menuType int not null,
safeGuard varchar(254) not null,
changeDiary clob(1M),
helpText clob(1M),
objProp clob(1M),
version varchar(32),
smObjProp clob(1M))
CREATE UNIQUE INDEX char_menu_ind
ON char_menu (name) CLUSTER;
CREATE UNIQUE INDEX char_menu_id_ind
ON char_menu (charMenuId);
CREATE TABLE char_menu_list
(charMenuId int not null,
path varchar(30) not null,
label varchar(254) not null,
childType int not null,
value varchar(255))
CREATE INDEX char_menu_list_ind
ON char_menu_list (charMenuId) CLUSTER;
CREATE TABLE char_menu_query
  (charMenuId       int           not null,
   path             varchar(30)   not null,
   arschema         varchar(254)  not null,
   server           varchar(255)  not null,
   labelField       int           not null,
   labelField2      int           ,
   labelField3      int           ,
   labelField4      int           ,
   labelField5      int           ,
   valueField       int           not null,
   sortOnLabel      int           not null,
   queryShort       varchar(255)  ,
   queryLong        clob(1M)      ,
   keywordList      clob(1M)      ,
   parameterList    clob(1M)      ,
   externList       clob(1M)      ,
   sampleSchema     varchar(254)  ,
   sampleServer     varchar(64)   )
;
CREATE INDEX char_menu_qry_ind
  ON char_menu_query (charMenuId)
CLUSTER;
CREATE TABLE char_menu_file
  (charMenuId       int           not null,
   path             varchar(30)   not null,
   fileLocation     int           not null,
   filename         varchar(255)  not null)
;
CREATE INDEX char_menu_file_ind
  ON char_menu_file (charMenuId)
CLUSTER;
CREATE TABLE char_menu_sql
  (charMenuId       int           not null,
   path             varchar(30)   not null,
   server           varchar(255)  not null,
   labelIndex       int           not null,
   labelIndex2      int           ,
   labelIndex3      int           ,
   labelIndex4      int           ,
   labelIndex5      int           ,
   valueIndex       int           not null,
   sqlCmdShort      varchar(255)  ,
   sqlCmdLong       clob(1M)      ,
   keywordList      clob(1M)      ,
   parameterList    clob(1M)      ,
   externList       clob(1M)      )
;
CREATE INDEX char_menu_sql_ind
  ON char_menu_sql (charMenuId)
CLUSTER;
CREATE TABLE char_menu_dd
  (charMenuId int not null,
   path varchar(30) not null,
   server varchar(64) not null,
   structType int not null,
   nameType int not null,
   valueFormat int not null,
   structSubtype int,
   arschema varchar(254),
   hiddenToo int )
;

CREATE INDEX char_menu_dd_ind
ON char_menu_dd (charMenuId)
CLUSTER ;

CREATE TABLE arcontainer
  (name varchar(254) not null,
   containerId int not null,
   containerType int not null,
   timestamp int not null,
   owner varchar(254) not null,
   lastChanged varchar(254) not null,
   numReferences int not null,
   label varchar(255),
   safeGuard varchar(254) not null,
   description clob(1M),
   changeDiary clob(1M),
   helpText clob(1M),
   objProp clob(1M),
   version varchar(32),
   smObjProp clob(1M) )
;

CREATE UNIQUE INDEX arctr_ind
ON arcontainer (name) CLUSTER ;
CREATE UNIQUE INDEX arctr_id_ind
ON arcontainer (containerId);

CREATE TABLE arctr_group_ids
  (containerId int not null,
   groupId int not null,
   permission int not null)
;

CREATE INDEX arctr_group_ind
ON arctr_group_ids (containerId)
CLUSTER ;

CREATE TABLE arctr_subadmin
  (containerId int not null,
   groupId int not null)
;

CREATE INDEX arctr_subadmin_ind
ON arctr_subadmin (containerId)
CLUSTER ;
CREATE TABLE cntnr_ownr_obj
  (containerId       int           not null,
   ownerObjType      int           not null,
   ownerObjId        int           not null,
   objIndex          int           not null)
;
CREATE INDEX cntnr_ownr_id_ind
  ON  cntnr_ownr_obj (containerId)
;
CREATE INDEX cntnr_ownr_obj_ind
  ON  cntnr_ownr_obj (ownerObjType, ownerObjId)
;
CREATE UNIQUE INDEX cntnr_ownr_ind
  ON  cntnr_ownr_obj (containerId, ownerObjType, ownerObjId)
;
CREATE TABLE arreference
  (containerId     int           not null,
   referenceId     int           not null,
   referenceType   int           not null,
   dataType        int           not null,
   referenceOrder  int           not null,
   referenceObjId  int           ,
   valueShort      varchar(255)  ,
   label           varchar(255)  ,
   valueLong       clob(1M)      ,
   description     clob(1M)      )
;
CREATE UNIQUE INDEX arref_ind
  ON arreference (containerId, referenceId) CLUSTER ;

CREATE TABLE arref_group_ids
  (containerId     int           not null,
   referenceId     int           not null,
   groupId         int           not null)
;
CREATE INDEX arref_group_ind
  ON arref_group_ids (containerId, referenceId) CLUSTER ;

CREATE TABLE filter
  (name varchar(254) not null,
   filterId       int           not null,
   timestamp      int           not null,
   owner          varchar(254)  not null,
   lastChanged    varchar(254)  not null,
   wkConnType     int           not null,
   fOrder         int           not null,
   opSet          int           not null,
   enable         int           not null,
   numActions     int           not null,
   numElses       int           not null,
   safeGuard      varchar(254)  not null,
   queryShort     varchar(255)  ,
CREATE TABLE filter_notify
(filt erId int not null,
actionIndex int not null,
userName varchar(255) not null,
notifyText varchar(255),
priority int not null,
mechanism int not null,
mechXRef int not null,
fieldIdCode int not null,
subjectText varchar(255),
behavior int,
permission int,
fromUser varchar(255),
replyTo varchar(255),
cc varchar(255),
bcc varchar(255),
organization varchar(255),
mailboxName varchar(255),
headerTemplate varchar(255),
footerTemplate varchar(255),
contentTemplate varchar(255),
notifyTextLong clob(1M)) ;

CREATE INDEX filter_notify_ind
ON filter_notify (filterId) CLUSTER ;

CREATE TABLE filter_notify_ids
(filt erId int not null,
actionIndex int not null,
fieldId int not null) ;

CREATE INDEX filterNotifyIdsInd
ON filter_notify_ids (filterId, actionIndex) CLUSTER ;

CREATE TABLE filter_message
(filt erId int not null,
actionIndex int not null,
msgType int not null,
msgNum int not null,
msgText varchar(255) not null) ;
CREATE INDEX filter_message_ind
    ON filter_message (filterId)
    CLUSTER;

CREATE TABLE filter_log
    (filterId int not null,
     actionIndex int not null,
     logFile     varchar(255) )
    CLUSTER;

CREATE INDEX filter_log_ind
    ON filter_log (filterId)
    CLUSTER;

CREATE TABLE filter_set
    (filterId int not null,
     actionIndex int not null,
     fieldId    int not null,
     assignShort varchar(255) ,
     assignLong  clob(1M)      ,
     sampleSchema varchar(254) ,
     sampleServer varchar(64)   )
    CLUSTER;

CREATE INDEX filter_set_ind
    ON filter_set (filterId)
    CLUSTER;

CREATE TABLE filter_process
    (filterId int not null,
     actionIndex int not null,
     command    varchar(255) not null)
    CLUSTER;

CREATE INDEX filter_process_ind
    ON filter_process (filterId)
    CLUSTER;

CREATE TABLE filter_push
    (filterId int not null,
     actionIndex int not null,
     fieldId    int not null,
     assignShort varchar(255) ,
     assignLong  clob(1M)      ,
     sampleSchema varchar(254) ,
     sampleServer varchar(64)   )
    CLUSTER;

CREATE INDEX filter_push_ind
    ON filter_push (filterId)
    CLUSTER;

CREATE TABLE filter_sql
    (filterId int not null,
     actionIndex int not null,
     assignShort varchar(255) ,
     assignLong  clob(1M)      )
    CLUSTER;

CREATE INDEX filter_sql_ind
    ON filter_sql (filterId)
    CLUSTER;
CREATE TABLE filter_gotoaction
  (filterId      int           not null,
   actionIndex   int           not null,
   tag           int           not null,
   fieldIdOrValue int default 0 )
;
CREATE  INDEX filter_gotoa_ind
  ON filter_gotoaction (filterId)
  CLUSTER ;

CREATE TABLE filter_call
  (filterId      int           not null,
   actionIndex   int           not null,
   serverName    varchar(64)   not null,
   guideName     varchar(254)  not null,
   guideMode     int           not null,
   guideTableId  int          ,
   assignShort   varchar(255) ,
   assignLong    clob(1M)     ,
   sampleServer  varchar(64)  ,
   sampleGuide   varchar(254) )
;
CREATE  INDEX filter_call_ind
  ON filter_call (filterId)
  CLUSTER ;

CREATE TABLE filter_exit
  (filterId      int           not null,
   actionIndex   int           not null,
   closeAll      char          )
;
CREATE  INDEX filter_exit_ind
  ON filter_exit (filterId)
  CLUSTER ;

CREATE TABLE filter_goto
  (filterId      int           not null,
   actionIndex   int           not null,
   label         varchar(128)  not null)
;
CREATE  INDEX filter_goto_ind
  ON filter_goto (filterId)
  CLUSTER ;

CREATE TABLE filter_mapping
  (schemaId      int           not null,
   objIndex      int           not null,
   filterId      int           not null)
;
CREATE UNIQUE  INDEX filter_mapping_ind
  ON filter_mapping (schemaId, filterId)
;

CREATE TABLE escalation
```sql
BMC Remedy Action Request System 7.0

<table>
<thead>
<tr>
<th>Column</th>
<th>Data Type</th>
<th>Nullable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>varchar(254)</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>escalationId</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>timestamp</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>owner</td>
<td>varchar(254)</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>lastChanged</td>
<td>varchar(254)</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>wkConnType</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>numActions</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>numEIses</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>firetmType</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>tmInterval</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>monthday</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>weekday</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>hourmask</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>minute</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>enable</td>
<td>int</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>safeGuard</td>
<td>varchar(254)</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>queryShort</td>
<td>varchar(255)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>queryLong</td>
<td>clob(1M)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>changeDiary</td>
<td>clob(1M)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>helpText</td>
<td>clob(1M)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>objProp</td>
<td>clob(1M)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>version</td>
<td>varchar(32)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>smObjProp</td>
<td>clob(1M)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CREATE UNIQUE INDEX escalation_ind ON escalation (name) CLUSTER;
CREATE UNIQUE INDEX escalation_id_ind ON escalation (escalationId);
CREATE TABLE escal_mapping
(schemaid int not null,
 objIndex int not null,
 escalationId int not null)
CREATE UNIQUE INDEX escal_mapping_ind ON escal_mapping (schemaid, escalationId);
CREATE TABLE actlink
(name varchar(254) not null,
 actlinkId int not null,
 timestamp int not null,
 owner varchar(254) not null,
 lastChanged varchar(254) not null,
 wkConnType int not null,
 alOrder int not null,
 executeMask int not null,
 controlfieldId int,
 fieldId int not null,
 enable int not null,
 numActions int not null,
 numEIses int not null,
 safeGuard varchar(254) not null,
 queryShort varchar(255),
)

64 Chapter 2—SQL Definitions of the data dictionary tables
CREATE UNIQUE INDEX actlink_ind ON actlink (name) CLUSTER;
CREATE UNIQUE INDEX actlink_id_ind ON actlink (actlinkId);

CREATE TABLE actlink_group_ids
(actlinkId int not null,
groupId int not null)

CREATE INDEX actLinkGroupIdsInd ON actlink_group_ids (actlinkId) CLUSTER;

CREATE TABLE actlink_macro
(actlinkId int not null,
actionIndex int not null,
macroName varchar(254) not null,
shortText varchar(255) ,
longText clob(1M) )

CREATE INDEX actlink_macro_ind ON actlink_macro (actlinkId) CLUSTER;

CREATE TABLE actlink_macro_parm
(actlinkId int not null,
actionIndex int not null,
name varchar(254) not null,
value varchar(255) not null)

CREATE INDEX alk_ma_parm_ind ON actlink_macro_parm (actlinkId, actionIndex) CLUSTER;

CREATE TABLE actlink_set
(actlinkId int not null,
actionIndex int not null,
fieldId int not null,
assignShort varchar(255) ,
assignLong clob(1M) ,
keywordList clob(1M) ,
parameterList clob(1M) ,
sampleSchema varchar(254) ,
sampleServer varchar(64) )

CREATE INDEX actlink_set_ind ON actlink_set (actlinkId) CLUSTER;
CREATE TABLE actlink_process
(
  actlinkId      int           not null,
  actionIndex    int           not null,
  command        varchar(255)  not null,
  keywordList    varchar(255)  ,
  parameterList  varchar(255)  
) ;

CREATE INDEX actLinkProcessInd
ON actlink_process (actlinkId)
CLUSTER ;

CREATE TABLE actlink_message
(
  actlinkId      int           not null,
  actionIndex    int           not null,
  msgType        int           not null,
  msgNum         int           not null,
  msgText        clob(1M)      not null,
  msgPane        char          default '0' 
) ;

CREATE INDEX actLinkMessageInd
ON actlink_message (actlinkId)
CLUSTER ;

CREATE TABLE actlink_set_char
(
  actlinkId      int           not null,
  actionIndex    int           not null,
  fieldId        int           not null,
  charMenu       varchar(254)  ,
  propShort      varchar(255)  ,
  propLong       clob(1M)      ,
  focus          int           ,
  accessOpt      int           default 0 )

CREATE INDEX actlink_schar_ind
ON actlink_set_char (actlinkId)
CLUSTER ;

CREATE TABLE actlink_dde
(
  actlinkId      int           not null,
  actionIndex    int           not null,
  serviceName    varchar(64)   not null,
  topic          varchar(64)   not null,
  action         int           not null,
  path           varchar(255)  not null,
  command        varchar(255)  not null,
  item           clob(1M)      
) ;

CREATE INDEX actlink_dde_ind
ON actlink_dde (actlinkId)
CLUSTER ;

CREATE TABLE actlink_auto
(
  actlinkId      int           not null,
  actionIndex    int           not null,
  autoServerName varchar(255)  not null,
  clsId          varchar(128)  not null,
  isVisible      char          not null,
  isSensitive    varchar(100)  not null,
  isTranslated   varchar(100)  not null,
  isClosed       int           not null,
  currentAuto    int           not null,
  isCurrentAuto  int           not null,
  isSynchronous  int           not null,
  isAutoServer   int           not null,
  isAutoService  int           not null,
  isAutoItem     int           not null,
  isAutoTopic    int           not null,
  isAutoAction   int           not null,
  isAutoPath     int           not null,
  isAutoCommand  int           not null,
  isAutoItem     int           not null,
  isAutoTopic    int           not null,
  isAutoAction   int           not null,
  isAutoPath     int           not null,
  isAutoCommand  int           not null,
  isAutoItem     int           not null,
  isAutoTopic    int           not null,
  isAutoAction   int           not null,
  isAutoPath     int           not null,
  isAutoCommand  int           not null,
  isAutoItem     int           not null,
  isAutoTopic    int           not null,
  isAutoAction   int           not null,
  isAutoPath     int           not null,
  isAutoCommand  int           not null,
  isAutoItem     int           not null,
  isAutoTopic    int           not null,
  isAutoAction   int           not null,
  isAutoPath     int           not null,
  isAutoCommand  int           not null,
  isAutoItem     int           not null,
  isAutoTopic    int           not null,
  isAutoAction   int           not null,
  isAutoPath     int           not null,
  isAutoCommand  int           not null,
  isAutoItem     int           not null,
  isAutoTopic    int           not null,
  isAutoAction   int           not null,
  isAutoPath     int           not null,
  isAutoCommand  int           not null,
  isAutoItem     int           not null,
  isAutoTopic    int           not null,
  isAutoAction   int           not null,
  isAutoPath     int           not null,
  isAutoCommand  int           not null,
  isAutoItem     int           not null,
  isAutoTopic    int           not null,
  isAutoAction   int           not null,
  isAutoPath     int           not null,
  isAutoCommand  int           not null,
  isAutoItem     int           not null,
  isAutoTopic    int           not null,
  isAutoAction   int           not null,
  isAutoPath     int           not null,
  isAutoCommand  int           not null,
  isAutoItem     int           not null,
  isAutoTopic    int           not null,
  isAutoAction   int           not null,
  isAutoPath     int           not null,
CREATE TABLE actlink_open
(
  actlinkId      int           not null,
  actionIndex    int           not null,
  serverName     varchar(64)   not null,
  schemaName     varchar(254)  not null,
  vuiLabel       varchar(254)  ,
  closeBox       char          ,
  assignShort    varchar(255)  ,
  assignLong     clob(1M)      ,
  windowMode     int           ,
  noMatchCtnu    char          ,
  pollIntval     int           ,
  sortList       varchar(255)  ,
  queryShort     varchar(255)  ,
  queryLong      clob(1M)      ,
  msgType        int           ,
  msgNum         int           ,
  msgText        clob(1M)      ,
  msgPane        char          ,
  reportstr      clob(1M)      ,
  supresEptyList char          ,
  targetLocation varchar(255)  
)
CREATE INDEX actlink_open_ind ON actlink_open (actlinkId) CLUSTER;

CREATE TABLE actlink_commit
  (actlinkId     int           not null,
   actionIndex   int           not null) CLUSTER;

CREATE INDEX actlink_commit_ind ON actlink_commit (actlinkId) CLUSTER;

CREATE TABLE actlink_close
  (actlinkId     int           not null,
   actionIndex   int           not null,
   closeAll char ) CLUSTER;

CREATE INDEX actlink_close_ind ON actlink_close (actlinkId) CLUSTER;

CREATE TABLE actlink_call
  (actlinkId     int           not null,
   actionIndex   int           not null,
   serverName    varchar(64)   not null,
   guideName     varchar(254)  not null,
   guideMode     int           not null,
   guideTableId  int           default 0 not null,
   assignShort   varchar(255)  ,
   assignLong    clob(1M)      ,
   sampleServer  varchar(64)   ,
   sampleGuide   varchar(254)  ) CLUSTER;

CREATE INDEX actlink_call_ind ON actlink_call (actlinkId) CLUSTER;

CREATE TABLE actlink_exit
  (actlinkId     int           not null,
   actionIndex   int           not null,
   closeAll char ) CLUSTER;

CREATE INDEX actlink_exit_ind ON actlink_exit (actlinkId) CLUSTER;

CREATE TABLE actlink_goto
  (actlinkId     int           not null,
   actionIndex   int           not null,
   label         varchar(128)  not null) CLUSTER;
CREATE INDEX actlink_goto_ind
ON actlink_goto (actlinkId)
CLUSTER;

CREATE TABLE actlink_wait
(actlinkId int not null,
actionIndex int not null,
buttonTitle varchar(64) default 'Continue'
);

CREATE INDEX actlink_wait_ind
ON actlink_wait (actlinkId)
CLUSTER;

CREATE TABLE actlink_gotoaction
(actlinkId int not null,
actionIndex int not null,
tag int not null,
fieldIdOrValue int default 0
);

CREATE INDEX actlink_gotoa_ind
ON actlink_gotoaction (actlinkId)
CLUSTER;

CREATE TABLE actlink_mapping
(schemaid int not null,
objIndex int not null,
actlinkId int not null)
;

CREATE UNIQUE INDEX actlink_mapping_ind
ON actlink_mapping (schemaid, actlinkId)
;

CREATE TABLE alert_user
(username varchar(254) not null,
clientIPAddr varchar(16) not null,
actualIPAddr varchar(16) not null,
serverIPAddr varchar(16) not null,
clientPort int not null,
regFlags int not null,
clientVersion int not null,
regTime int not null,
clientCodeSet int not null)
;

CREATE UNIQUE INDEX alert_user_ind
ON alert_user (username, clientIPAddr, clientPort)
;

CREATE TABLE alert_time
(username varchar(254) not null,
checkpointTime int not null)
;

CREATE UNIQUE INDEX alert_time_ind
ON alert_time (username)
CREATE TABLE support_file
  (fileType       int           not null,
   id             int           not null,
   id2            int           not null,
   fileId         int           not null,
   timestamp      int           not null,
   fileContent    blob(1G)      );

CREATE UNIQUE INDEX support_file_ind
  ON support_file (fileType, id, id2, fileId)
  CLUSTER ;

CREATE TABLE servgrp_config
  (name varchar(64),
   checkInterval int           not null)
  ;

CREATE TABLE servgrp_op_mstr
  (operation      varchar(255)  not null,
   opNum          int           not null,
   configLabel    varchar(255),
   configCommand  varchar(50),
   categoryStrs   varchar(255)
  );

CREATE TABLE ft_pending
  (serverName     varchar(64)   not null,
   schemaId       int           not null,
   fieldId        int           not null,
   entryId        varchar(15),
   operationType  int           not null,
   updateTime     int           ,
   seqNum         int           not null)
  ;

CREATE INDEX ft_pending_ind
  ON ft_pending (seqNum)
  CLUSTER ;
The following set of SQL commands define the AR System data dictionary for Informix databases. For an explanation of the commands, see the Informix Guide to SQL: Reference and Syntax.

```
DATABASE ARSystem;
CREATE TABLE control
    (dbVersion int          not null,
     schemaid int          not null,
     filterId int          not null,
     serverId int          not null,
     containerId int        not null,
     actlinkId int          not null,
     adminExtId int         not null,
     charMenuId int         not null);
CREATE TABLE arschema
    (name         varchar(254) not null,
     schemaid     int          not null,
     schemaType   int          not null,
     timestamp    int          not null,
     owner        varchar(254) not null,
     lastChanged  varchar(254) not null,
     coreVersion  int          not null,
     numFields    int          not null,
     numVuis      int          not null,
     defaultVui   varchar(254) not null,
     nextId       int          not null,
     nextFieldId  int          not null,
     maxStatEnums int          not null,
     upgrdVersion int                  ,
     safeGuard    varchar(254) not null,
     changeDiary byte                 ,
     helpText     byte                 ,
     objProp      byte                 ,
     version      varchar(32)          ,
     smObjProp    byte                 );
CREATE UNIQUE CLUSTER INDEX schema_ind
    ON arschema (name);
CREATE UNIQUE INDEX schema_id_ind
    ON arschema (schemaid);
CREATE TABLE schema_group_ids
    (schemaid     int          not null,
     groupId      int          not null,
     permission   int          not null);
CREATE CLUSTER INDEX schema_group_ind
    ON schema_group_ids (schemaid);
CREATE TABLE subadmin_group
    (schemaid     int          not null,
     groupId      int          not null);
CREATE CLUSTER INDEX subadmin_group_ind
    ON subadmin_group (schemaid);
```
CREATE TABLE schema_list_fields
(schemaid int not null,
listIndex int not null,
fieldId int not null,
columnWidth int not null,
separatorLen int not null,
separator varchar(10));
CREATE CLUSTER INDEX schema_list_f_ind
ON schema_list_fields (schemaid);
CREATE TABLE schema_sort
(schemaid int not null,
listIndex int not null,
fieldId int not null,
sortOrder int not null);
CREATE CLUSTER INDEX schema_sort_ind
ON schema_sort (schemaid);
CREATE TABLE schema_archive
(schemaid int not null,
enable int not null,
archiveType int not null,
archiveToForm int,
archiveToFile varchar(255),
queryShort varchar(255),
queryLong byte,
monthday int not null,
weekday int not null,
hourmask int not null,
minute int not null,
archiveFromForm int)
CREATE CLUSTER INDEX schema_archive_ind
ON schema_archive (schemaid);
CREATE TABLE schema_audit
(schemaid int not null,
enable int not null,
style int not null,
form int,
queryShort varchar(255),
queryLong byte)
CREATE CLUSTER INDEX schema_audit_ind
ON schema_audit (schemaid);
CREATE TABLE schema_index
(schemaid int not null,
listIndex int not null,
numFields int not null,
uniqueFlag int not null,
indexName varchar(254) not null,
f1 int not null,
f2 int,
f3 int,
f4 int,
f5 int,
f6 int,
f7 int,
f8 int,
CREATE CLUSTER INDEX schema_index_ind ON schema_index (schemaId);
CREATE TABLE schema_join
(schemaId int not null,
memberA varchar(254) not null,
memberB varchar(254) not null,
options int ,
queryShort varchar(255) ,
queryLong byte );
CREATE UNIQUE INDEX schema_join_ind ON schema_join (schemaId);
CREATE TABLE schema_view
(schemaId int not null,
tableName byte ,
keyField varchar(254) not null,
queryShort varchar(255) ,
queryLong byte );
CREATE UNIQUE INDEX schema_view_ind ON schema_view (schemaId);
CREATE TABLE schemaVendor
(schemaId int not null,
vendorName varchar(254) not null,
tableName byte );
CREATE UNIQUE INDEX schemaVendor_ind ON schemaVendor (schemaId);
CREATE TABLE field
(schemaId int not null,
fieldId int not null,
fieldName varchar(254) not null,
fieldType int not null,
timestamp int not null,
owner varchar(254) not null,
lastChanged varchar(254) not null,
datatype int not null,
ofOption int not null,
createMode int not null,
fbOption int ,
defaultValue varchar(255) ,
changeDiary byte ,
helpText byte );
CREATE UNIQUE CLUSTER INDEX field_ind ON field (schemaId, fieldId);
CREATE INDEX field_schema_ind ON field (schemaId);
CREATE TABLE vui
(schmaId      int          not null,
 vuiId        int          not null,
 vuiName      varchar(254) not null,
 locale       varchar(30)  ,
 vuiType      int          ,
timestamp    int          not null,
owner        varchar(254) not null,
lastChanged  varchar(254) not null,
changeDiary  byte                 ,
helpText     byte                 );
CREATE UNIQUE CLUSTER INDEX vui_ind
ON vui (schmaId, vuiId);
CREATE INDEX vui_schema_ind
ON vui (schmaId);
CREATE TABLE field_dispprop
(schmaId      int          not null,
 fieldId      int          ,
 listIndex    int          not null,
vuiId         int          ,
propShort     varchar(255)         ,
propLong      byte                 );
CREATE UNIQUE INDEX field_dispprop_ind
ON field_dispprop (schmaId, fieldId, listIndex, vuiId);
CREATE TABLE field_int
(schmaId      int          not null,
 fieldId      int          not null,
rangeLow     int          ,
rangeHigh    int          );
CREATE UNIQUE CLUSTER INDEX field_int_ind
ON field_int (schmaId, fieldId);
CREATE TABLE field_real
(schmaId      int          not null,
 fieldId      int          not null,
rangeLow     float                ,
rangeHigh    float                ,
arprecision  int                  );
CREATE UNIQUE CLUSTER INDEX field_real_ind
ON field_real (schmaId, fieldId);
CREATE TABLE field_diary
(schmaId      int          not null,
 fieldId      int          not null,
fullTextOptions int               );
CREATE UNIQUE CLUSTER INDEX field_diary_ind
ON field_diary (schmaId, fieldId);
CREATE TABLE field_char
(schmaId      int          not null,
 fieldId      int          not null,
maxLength    int          ,
qbeMatchOp   int          ,
menuStyle    int          ,
charMenu     varchar(254)  ,
pattern      varchar(255)  ,
fullTextOptions int               );
CREATE UNIQUE CLUSTER INDEX field_char_ind
ON field_char (schemaId, fieldId);
CREATE TABLE field_enum
(schemaId     int          not null,
fieldId      int          not null,
maxEnum      int          not null,
enumStyle    int          ,
schemaName   varchar(254)         ,
serverName   varchar(64)          ,
nameField    int          ,
numberField  int          ,
queryShort   varchar(255)         ,
queryLong    byte          );
CREATE UNIQUE CLUSTER INDEX field_enum_ind
ON field_enum (schemaId, fieldId);
CREATE TABLE field_enum_values
(schemaId     int          not null,
fieldId      int          not null,
enumId       int          not null,
value        varchar(254) not null);
Chapter 2—SQL Definitions of the data dictionary tables

BMC Remedy Action Request System 7.0

```sql
CREATE TABLE field_dec
(schemald int not null,
  fieldId int not null,
  rangeLow varchar(64),
  rangeHigh varchar(64),
  aprecision int);
CREATE UNIQUE CLUSTER INDEX field_dec_ind
ON field_dec (schemald, fieldId);
CREATE TABLE field_curr
(schemald int not null,
  fieldId int not null,
  rangeLow varchar(64),
  rangeHigh varchar(64),
  aprecision int,
  funcCurr byte,
  allowCurr byte);
CREATE UNIQUE CLUSTER INDEX field_curr_ind
ON field_curr (schemald, fieldId);
CREATE TABLE field_view
(schemald int not null,
  fieldId int not null,
  maxLength int);
CREATE UNIQUE CLUSTER INDEX field_view_ind
ON field_view (schemald, fieldId);
CREATE TABLE field_display
(schemald int not null,
  fieldId int not null,
  maxLength int);
CREATE UNIQUE CLUSTER INDEX field_display_ind
ON field_display (schemald, fieldId);
CREATE TABLE field_date
(schemald int not null,
  fieldId int not null,
  minDate int,
  maxDate int);
CREATE UNIQUE CLUSTER INDEX field_date_ind
ON field_date (schemald, fieldId);
CREATE TABLE join_mapping
(schemald int not null,
  fieldId int not null,
  memberIndex int,
  mfieldId int);
CREATE UNIQUE INDEX join_mapping_ind
ON join_mapping (schemald, fieldId);
```
CREATE TABLE view_mapping
  (schemaId int not null,
   fieldId int not null,
   extField varchar(254) not null);
CREATE UNIQUE INDEX view_mapping_ind
  ON view_mapping (schemaId, fieldId);
CREATE TABLE vendor_mapping
  (schemaId int not null,
   fieldId int not null,
   extField varchar(254) not null);
CREATE UNIQUE INDEX vendor_mapping_ind
  ON vendor_mapping (schemaId, fieldId);
CREATE TABLE char_menu
  (name varchar(254) not null,
   charMenuId int not null,
   timestamp int not null,
   owner varchar(254) not null,
   lastChanged varchar(254) not null,
   refreshCode int not null,
   menuType int not null,
   safeGuard varchar(254) not null,
   changeDiary byte ,
   helpText byte ,
   objProp byte ,
   version varchar(32) ,
   smObjProp byte ,
   )
CREATE UNIQUE CLUSTER INDEX char_menu_ind
  ON char_menu (name);
CREATE UNIQUE INDEX char_menu_id_ind
  ON char_menu (charMenuId);
CREATE TABLE char_menu_list
  (charMenuId int not null,
   path varchar(254) not null,
   label varchar(254) not null,
   childType int not null,
   value varchar(255) )
CREATE CLUSTER INDEX char_menu_list_ind
  ON char_menu_list (charMenuId);
CREATE TABLE char_menu_query
  (charMenuId int not null,
   path varchar(30) not null,
   arschema varchar(254) not null,
   server varchar(255) not null,
   labelField int not null,
   labelField2 int ,
   labelField3 int ,
   labelField4 int ,
   labelField5 int ,
   valueField int not null,
   sortOnLabel int not null,
   queryShort varchar(255) ,
   queryLong byte ,
   keywordList lvarchar ,
   parameterList lvarchar ,
BMC Remedy Action Request System 7.0

externList lvarchar,
sampleSchema varchar(254),
sampleServer varchar(64));

CREATE CLUSTER INDEX char_menu_qry_ind
ON char_menu_query (charMenuId);

CREATE TABLE char_menu_file
(charMenuId int not null,
path varchar(30) not null,
fileName varchar(255) not null);

CREATE CLUSTER INDEX char_menu_file_ind
ON char_menu_file (charMenuId);

CREATE TABLE char_menu_sql
(charMenuId int not null,
path varchar(30) not null,
server varchar(255) not null,
labelIndex int not null,
lableIndex2 int,
lableIndex3 int,
lableIndex4 int,
lableIndex5 int,
valueIndex int not null,
sqlCmdShort varchar(255),
sqlCmdLong byte,
keywordList lvarchar,
parameterList lvarchar,
externList lvarchar);

CREATE CLUSTER INDEX char_menu_sql_ind
ON char_menu_sql (charMenuId);

CREATE TABLE char_menu_dd
(charMenuId int not null,
path varchar(30) not null,
server varchar(64) not null,
structType int not null,
nameType int not null,
valueFormat int not null,
structSubtype int,
arschema varchar(254),
hiddenToo int);

CREATE CLUSTER INDEX char_menu_dd_ind
ON char_menu_dd (charMenuId);

CREATE TABLE arcontainer
(name varchar(254) not null,
containerId int not null,
containerType int not null,
timestamp int not null,
owner varchar(254) not null,
lastChanged varchar(254) not null,
numReferences int not null,
lable varchar(255),
safeGuard varchar(254) not null,
description byte,
changeDiary byte);
helpText byte,
objProp byte,
version varchar(32),
smObjProp byte);

CREATE UNIQUE CLUSTER INDEX arctr_ind
ON arcontainer (name);
CREATE UNIQUE INDEX arctr_id_ind
ON arcontainer (containerId);

CREATE TABLE arctr_group_ids
  (containerId int      not null,
   groupId int      not null,
   permission int      not null);
CREATE CLUSTER INDEX arctr_group_ind
ON arctr_group_ids (containerId);

CREATE TABLE arctr_subadmin
  (containerId int      not null,
   groupId int      not null);
CREATE CLUSTER INDEX arctr_subadmin_ind
ON arctr_subadmin (containerId);

CREATE TABLE cntnr_ownr_obj
  (containerId int      not null,
   ownerObjType int      not null,
   ownerObjId int      not null,
   objIndex int      not null);
CREATE INDEX cntnr_ownr_id_ind
ON cntnr_ownr_obj (containerId);
CREATE INDEX cntnr_ownr_obj_ind
ON cntnr_ownr_obj (ownerObjType, ownerObjId);
CREATE UNIQUE INDEX cntnr_ownr_ind
ON cntnr_ownr_obj (containerId, ownerObjType, ownerObjId);

CREATE TABLE arreference
  (containerId int      not null,
   referenceId int      not null,
   referenceType int      not null,
   dataType int      not null,
   referenceOrder int      not null,
   referenceObjId int      ,
   valueShort varchar(255),
   label varchar(255),
   valueLong byte,
   description byte);
CREATE UNIQUE CLUSTER INDEX arref_ind
ON arreference (containerId, referenceId);

CREATE TABLE arref_group_ids
  (containerId int      not null,
   referenceId int      not null,
   groupId int      not null);
CREATE CLUSTER INDEX arref_group_ind
ON arref_group_ids (containerId, referenceId);
### Chapter 2—SQL Definitions of the data dictionary tables

```sql
CREATE TABLE filter
(
    name varchar(254) not null,
    filterId int not null,
    timestamp int not null,
    owner varchar(254) not null,
    lastChanged varchar(254) not null,
    wkConnType int not null,
    fOrder int not null,
    opSet int not null,
    enable int not null,
    numActions int not null,
    numElses int not null,
    safeGuard varchar(254) not null,
    queryShort varchar(255),
    queryLong byte,
    changeDiary byte,
    helpText byte,
    objProp byte,
    version varchar(32),
    smObjProp byte
);

CREATE UNIQUE CLUSTER INDEX filter_ind
ON filter (name);

CREATE UNIQUE INDEX filter_id_ind
ON filter (filterId);

CREATE TABLE filter_notify
(
    filterId int not null,
    actionIndex int not null,
    userName varchar(255) not null,
    notifyText varchar(255),
    priority int not null,
    mechanism int not null,
    mechXRef int not null,
    fieldIdCode int not null,
    subjectText varchar(255),
    behavior int,
    permission int,
    fromUser varchar(255),
    replyTo varchar(255),
    cc varchar(255),
    bcc varchar(255),
    organization varchar(255),
    mailboxName varchar(255),
    headerTemplate varchar(255),
    footerTemplate varchar(255),
    contentTemplate varchar(255),
    notifyTextLong byte
);

CREATE CLUSTER INDEX filter_notify_ind
ON filter_notify (filterId);

CREATE TABLE filter_notify_ids
(
    filterId int not null,
    actionIndex int not null,
    fieldId int not null
);

CREATE CLUSTER INDEX filter_notify_ids_ind
ON filter_notify_ids (filterId, actionIndex);
```
CREATE TABLE filter_message
  (filterId int          not null,
   actionIndex int          not null,
   msgType     int          not null,
   msgNum      int          not null,
   msgText     varchar(255) not null);

CREATE CLUSTER INDEX filter_message_ind
  ON filter_message (filterId);

CREATE TABLE filter_log
  (filterId int          not null,
   actionIndex int          not null,
   logFile     varchar(255)         );

CREATE CLUSTER INDEX filter_log_ind
  ON filter_log (filterId);

CREATE TABLE filter_set
  (filterId int          not null,
   actionIndex int          not null,
   fieldId     int         not null,
   assignShort varchar(255)        ,
   assignLong  byte                ,
   sampleSchema varchar(254)       ,
   sampleServer varchar(64)        );

CREATE CLUSTER INDEX filter_set_ind
  ON filter_set (filterId);

CREATE TABLE filter_process
  (filterId int          not null,
   actionIndex int          not null,
   command     varchar(255) not null);

CREATE CLUSTER INDEX filter_process_ind
  ON filter_process (filterId);

CREATE TABLE filter_push
  (filterId int          not null,
   actionIndex int          not null,
   fieldId     int         not null,
   assignShort varchar(255)        ,
   assignLong  byte                ,
   sampleSchema varchar(254)       ,
   sampleServer varchar(64)        );

CREATE CLUSTER INDEX filter_push_ind
  ON filter_push (filterId);

CREATE TABLE filter_sql
  (filterId int          not null,
   actionIndex int          not null,
   assignShort varchar(255)        ,
   assignLong  byte                );

CREATE CLUSTER INDEX filter_sql_ind
  ON filter_sql (filterId);

CREATE TABLE filter_gotoaction
  (filterId int          not null,
   actionIndex int          not null,
   tag         int         not null,
   fieldIdOrValue int       default 0 );
CREATE CLUSTER INDEX filter_gotoa_ind
ON filter_gotoaction (filterId);

CREATE TABLE filter_call
(ffilterId int not null,
actionIndex int not null,
serverName varchar(64) not null,
guideName varchar(254) not null,
guideMode int not null,
guideTableId int,
assignShort varchar(255) ,
assignLong byte ,
sampleServer varchar(64) ,
sampleGuide varchar(254) );

CREATE CLUSTER INDEX filter_call_ind
ON filter_call (filterId);

CREATE TABLE filter_exit
(filterId int not null,
actionIndex int not null,
closeAll char );

CREATE CLUSTER INDEX filter_exit_ind
ON filter_exit (filterId);

CREATE TABLE filter_goto
(filterId int not null,
actionIndex int not null,
label varchar(128) not null);

CREATE CLUSTER INDEX filter_goto_ind
ON filter_goto (filterId);

CREATE TABLE filter_mapping
(schemaId int not null,
objIndex int not null,
filterId int not null);

CREATE UNIQUE INDEX filter_mapping_ind
ON filter_mapping (schemaId, filterId);

CREATE TABLE escalation
(name varchar(254) not null,
estId int not null,
timestamp int not null,
owner varchar(254) not null,
lastName varchar(254) not null,
wkConnType int not null,
numActions int not null,
umElses int not null,
firetType int not null,
tminterval int not null,
monthday int not null,
weekday int not null,

82 Chapter 2—SQL Definitions of the data dictionary tables
hourmask     int          not null,
minute       int          not null,
enable       int          not null,
safeGuard   varchar(254) not null,
queryShort   varchar(255) ,
queryLong    byte                 ,
changeDiary  byte                 ,
helpText     byte                 ,
objProp      byte                 ,
version      varchar(32)          ,
smObjProp    byte                 );
CREATE UNIQUE CLUSTER INDEX escalation_ind
ON escalation (name);
CREATE UNIQUE INDEX escalation_id_ind
ON escalation (escalationId);
CREATE TABLE escal_mapping
(schemaId        int          not null,
objIndex        int          not null,
escalationId    int          not null);
CREATE UNIQUE INDEX escal_mapping_ind
ON escal_mapping (schemaId, escalationId);
CREATE TABLE actlink
(name         varchar(254) not null,
actlinkId    int          not null,
timestamp    int          not null,
owner        varchar(254)  not null,
lastChanged  varchar(254)  not null,
wkConnType   int          not null,
alOrder      int          not null,
executeMask  int          not null,
controlfieldId int                ,
fieldId      int          not null,
enable       int          not null,
numActions   int          not null,
numElses     int          not null,
safeGuard    varchar(254) not null,
queryShort   varchar(255) ,
queryLong    byte                 ,
changeDiary  byte                 ,
helpText     byte                 ,
objProp      byte                 ,
version      varchar(32)          ,
smObjProp    byte                 );
CREATE UNIQUE CLUSTER INDEX actlink_ind
ON actlink [name];
CREATE UNIQUE INDEX actlink_id_ind
ON actlink [actlinkId];
CREATE TABLE actlink_group_ids
(actlinkId   int         not null,
groupId     int         not null);
CREATE CLUSTER INDEX actlink_group_ind
ON actlink_group_ids [actlinkId];
CREATE TABLE actlink_macro
  (actlinkId int not null,
   actionIndex int not null,
   macroName varchar(254) not null,
   shortText varchar(255),
   longText byte);
CREATE CLUSTER INDEX actlink_macro_ind
  ON actlink_macro (actlinkId);
CREATE TABLE actlink_macro_parm
  (actlinkId int not null,
   actionIndex int not null,
   name varchar(254) not null,
   value varchar(255) not null);
CREATE CLUSTER INDEX alk_ma_parm_ind
  ON actlink_macro_parm (actlinkId, actionIndex);
CREATE TABLE actlink_set
  (actlinkId int not null,
   actionIndex int not null,
   fieldId int not null,
   assignShort varchar(255),
   assignLong byte,
   keywordList lvarchar,
   parameterList lvarchar,
   sampleSchema varchar(254),
   sampleServer varchar(64));
CREATE CLUSTER INDEX actlink_set_ind
  ON actlink_set (actlinkId);
CREATE TABLE actlink_process
  (actlinkId int not null,
   actionIndex int not null,
   command varchar(255) not null,
   keywordList varchar(255),
   parameterList varchar(255));
CREATE CLUSTER INDEX actlink_process_in
  ON actlink_process (actlinkId);
CREATE TABLE actlink_message
  (actlinkId int not null,
   actionIndex int not null,
   msgType int not null,
   msgNum int not null,
   msgText byte,
   msgPane char default '0');
CREATE CLUSTER INDEX actlink_msg_ind
  ON actlink_message (actlinkId);
CREATE TABLE actlink_set_char
  (actlinkId int not null,
   actionIndex int not null,
   fieldId int not null,
   charMenu varchar(254),
   propShort varchar(255),
   propLong byte,
   focus int,
   }
CREATE CLUSTER INDEX actlink_schar_ind ON actlink_set_char (actlinkId);

CREATE TABLE actlink_dde
  (actlinkId int not null,
   actionIndex int not null,
   serviceName varchar(64) not null,
   topic varchar(64) not null,
   action int not null,
   path varchar(255) not null,
   command varchar(255) not null,
   item byte);

CREATE CLUSTER INDEX actlink_dde_ind ON actlink_dde (actlinkId);

CREATE TABLE actlink_auto
  (actlinkId int not null,
   actionIndex int not null,
   autoServerName varchar(255) not null,
   clsId varchar(128) not null,
   isVisible char not null,
   actionShort varchar(255) ,
   actionLong byte ,
   COMShort varchar(255) ,
   COMLong byte);

CREATE INDEX actlink_auto_ind ON actlink_auto (actlinkId);

CREATE TABLE actlink_push
  (actlinkId int not null,
   actionIndex int not null,
   fieldId int not null,
   assignShort varchar(255) ,
   assignLong byte ,
   sampleSchema varchar(254) ,
   sampleServer varchar(64));

CREATE CLUSTER INDEX actlink_push_ind ON actlink_push (actlinkId);

CREATE TABLE actlink_sql
  (actlinkId int not null,
   actionIndex int not null,
   assignShort varchar(255) ,
   assignLong byte ,
   keywordList lvarchar ,
   parameterList lvarchar );

CREATE CLUSTER INDEX actlink_sql_ind ON actlink_sql (actlinkId);

CREATE TABLE actlink_open
  (actlinkId int not null,
   actionIndex int not null,
   serverName varchar(64) not null,
   schemaName varchar(254) not null,
   vuiLabel varchar(254) ,
   closeBox char ,
   ...
assignShort varchar(255) ,
assignLong byte ,
windowMode int ,
noMatchCtnu char ,
pollIntval int ,
sortlist varchar(255) ,
queryshort varchar(255) ,
querylong byte ,
msgType int ,
msgNum int ,
msgText byte ,
msgPane char ,
reportstr byte ,
supresEptyLst char ,
targetLocation varchar(255) );

CREATE CLUSTER INDEX actlink_open_ind
ON actlink_open (actlinkId);

CREATE TABLE actlink_commit
  (actlinkId int not null,
   actionIndex int not null);

CREATE CLUSTER INDEX actlink_commit_ind
ON actlink_commit (actlinkId);

CREATE TABLE actlink_close
  (actlinkId int not null,
   actionIndex int not null,
   closeAll char );

CREATE CLUSTER INDEX actlink_close_ind
ON actlink_close (actlinkId);

CREATE TABLE actlink_call
  (actlinkId int not null,
   actionIndex int not null,
   serverName varchar(64) not null,
   guideName varchar(254) not null,
   guideMode int not null,
   guideTableId int ,
   assignShort varchar(255) ,
   assignLong byte ,
   sampleServer varchar(64) ,
   sampleGuide varchar(254) );

CREATE CLUSTER INDEX actlink_call_ind
ON actlink_call (actlinkId);

CREATE TABLE actlink_exit
  (actlinkId int not null,
   actionIndex int not null,
   closeAll char );

Chapter 2—SQL Definitions of the data dictionary tables
CREATE CLUSTER INDEX actlink_exit_ind
ON actlink_exit (actlinkId);

CREATE TABLE actlink_goto
  (actlinkId int not null,
   actionIndex int not null,
   label varchar(128) not null);

CREATE CLUSTER INDEX actlink_goto_ind
ON actlink_goto (actlinkId);

CREATE TABLE actlink_wait
  (actlinkId int not null,
   actionIndex int not null,
   buttonTitle varchar(64) default 'Continue');

CREATE CLUSTER INDEX actlink_wait_ind
ON actlink_wait (actlinkId);

CREATE TABLE actlink_gotoaction
  (actlinkId int not null,
   actionIndex int not null,
   tag int not null,
   fieldIdOrValue int default 0);

CREATE CLUSTER INDEX actlink_gotoa_ind
ON actlink_gotoaction (actlinkId);

CREATE TABLE actlink_mapping
  (schemaId int not null,
   objIndex int not null,
   actlinkId int not null);

CREATE UNIQUE INDEX actlink_maping_ind
ON actlink_mapping (schemaId, actlinkId);

CREATE TABLE alert_user
  (username varchar(254) not null,
   clientIPAddr varchar(16) not null,
   actualIPAddr varchar(16) not null,
   serverIPAddr varchar(16) not null,
   clientPort int not null,
   regFlags int not null,
   clientVersion int not null,
   regTime int not null,
   clientCodeSet int not null);

CREATE UNIQUE INDEX alert_user_ind
ON alert_user (username, clientIPAddr, clientPort);

CREATE TABLE alert_time
  (username varchar(254) not null,
   checkpointTime int not null);

CREATE UNIQUE INDEX alert_time_ind
ON alert_time (username);
CREATE TABLE support_file
  (fileType int not null,
   id int not null,
   id2 int not null,
   fileId int not null,
   timestamp int not null,
   fileContent byte)
CREATE UNIQUE CLUSTER INDEX support_file_ind
  ON support_file (fileType, id, id2, fileId);

CREATE TABLE servgrp_config
  (name varchar(64),
   checkInterval int not null);

CREATE TABLE servgrp_op_mstr
  (operation varchar(255) not null,
   opNum int not null,
   configLabel varchar(255),
   configCommand varchar(50),
   categoryStrs varchar(255));

CREATE TABLE ft_pending
  (servername varchar(64) not null,
   schemaId int not null,
   fieldId int not null,
   entryId varchar(15),
   operationType int not null,
   updateTime int,
   seqNum int not null);
CREATE CLUSTER INDEX ft_pending_ind
  ON ft_pending (seqNum);
The following set of SQL commands define the AR System data dictionary for Oracle databases. For an explanation of these commands, see the Oracle SQL Reference Manual.

CREATE TABLE control
    (dbVersion number(15,0) not null,
     schemaId number(15,0) not null,
     filterId number(15,0) not null,
     serverId number(15,0) not null,
     containerId number(15,0) not null,
     actlinkId number(15,0) not null,
     adminExtId number(15,0) not null,
     charMenuId number(15,0) not null);

CREATE TABLE arschema
    (name varchar(254) not null,
     schemaId number(15,0) not null,
     schemaType number(15,0) not null,
     timestamp number(15,0) not null,
     owner varchar(254) not null,
     lastChanged varchar(254) not null,
     coreVersion number(15,0) not null,
     numFields number(15,0) not null,
     numVuis number(15,0) not null,
     defaultVui varchar(254) not null,
     nextId number(15,0) not null,
     nextFieldId number(15,0) not null,
     maxStatEnums number(15,0) not null,
     upgrdVersion number(15,0) not null,
     safeGuard varchar(254) not null,
     helpText clob null,
     changeDiary clob null,
     objProp clob null,
     version varchar(32) null,
     smObjProp clob null);

CREATE UNIQUE INDEX schema_ind
    ON arschema (name);

CREATE UNIQUE INDEX schema_id_ind
    ON arschema (schemaId);

CREATE TABLE schema_group_ids
    (schemaId number(15,0) not null,
     groupId number(15,0) not null,
     permission number(15,0) not null);

CREATE INDEX schema_group_ids_ind
    ON schema_group_ids (schemaId);

CREATE TABLE subadmin_group
    (schemaId number(15,0) not null,
     groupId number(15,0) not null);

CREATE INDEX subadmin_group_ind
    ON subadmin_group (schemaId);
CREATE TABLE schema_list_fields
    (schemaId number(15,0) not null,
     listIndex number(15,0) not null,
     fieldId number(15,0) not null,
     columnWidth number(15,0) not null,
     separatorLen number(15,0) not null,
     separator varchar(10) null);
CREATE INDEX schema_list_fields_ind
    ON schema_list_fields (schemaId);

CREATE TABLE schema_sort
    (schemaId number(15,0) not null,
     listIndex number(15,0) not null,
     fieldId number(15,0) not null,
     sortOrder number(15,0) not null);
CREATE INDEX schema_sort_ind
    ON schema_sort (schemaId);

CREATE TABLE schema_archive
    (schemaId number(15,0) not null,
     enable number(15,0) not null,
     archiveType number(15,0) not null,
     archiveToForm number(15,0) null,
     archiveToFile varchar(255) null,
     queryShort varchar(255) null,
     queryLong clob null,
     monthday number(15,0) not null,
     weekday number(15,0) not null,
     hourmask number(15,0) not null,
     minute number(15,0) not null,
     archiveFromForm number(15,0) null);
CREATE INDEX schema_archive_ind
    ON schema_archive (schemaId);

CREATE TABLE schema_audit
    (schemaId number(15,0) not null,
     enable number(15,0) not null,
     style number(15,0) not null,
     form number(15,0) null,
     queryShort varchar(255) null,
     queryLong clob null);
CREATE INDEX schema_audit_ind
    ON schema_audit (schemaId);

CREATE TABLE schema_index
    (schemaId number(15,0) not null,
     listIndex number(15,0) not null,
     numFields number(15,0) not null,
     uniqueFlag number(15,0) not null,
     indexName varchar(254) not null,
     f1 number(15,0) not null,
     f2 number(15,0) null,
     f3 number(15,0) null,
     f4 number(15,0) null,
     f5 number(15,0) null,
     f6 number(15,0) null,
     f7 number(15,0) null,
     f8 number(15,0) null,
     f9 number(15,0) null,
CREATE INDEX schema_index_ind
ON schema_index (schemaId);

CREATE TABLE schema_join
(schemaId     number(15,0) not null,
 memberA      varchar(254) not null,
 memberB      varchar(254) not null,
 options      number(15,0)     null,
 queryShort   varchar(255)     null,
 queryLong    clob             null);

CREATE UNIQUE INDEX schema_join_ind
ON schema_join (schemaId);

CREATE TABLE schema_view
(schemaId     number(15,0) not null,
 tableName    clob             null,
 keyField     varchar(254) not null,
 queryShort   varchar(255)     null,
 queryLong    clob             null);

CREATE UNIQUE INDEX schema_view_ind
ON schema_view (schemaId);

CREATE TABLE schema_vendor
(schemaId     number(15,0)  not null,
 vendorName   varchar(254)  not null,
 tableName    clob              null);

CREATE UNIQUE INDEX schema_vendor_ind
ON schema_vendor (schemaId);

CREATE TABLE field
(schemaId     number(15,0) not null,
 fieldId      number(15,0) not null,
 fieldName    varchar(254) not null,
 fieldType    number(15,0) not null,
 timestamp    number(15,0) not null,
 owner        varchar(254)  not null,
 lastChanged  varchar(254)  not null,
 datatype     number(15,0) not null,
 fOption      number(15,0) not null,
 createMode   number(15,0) not null,
 fbOption     number(15,0) null,
 defaultValue varchar(255)     null,
 helpText     clob             null,
 changeDiary  clob             null);

CREATE UNIQUE INDEX field_ind
ON field (schemaId, fieldId);

CREATE INDEX field_schema_ind
ON field (schemaId);
CREATE TABLE vui
(schemald  number(15,0) not null,
vuid      number(15,0) not null,
vuiName   varchar(254) not null,
locale    varchar(30)      null,
vuiType   number(15,0)     null,
timestamp number(15,0) not null,
owner     varchar(254)  not null,
lastChanged varchar(254) not null,
helpText  clob             null,
changeDiary clob            null);
CREATE UNIQUE INDEX vui_ind
ON vui (schemald, vuid);
CREATE INDEX vui_schema_ind
ON vui (schemald);

CREATE TABLE field_dispprop
(schemald  number(15,0) not null,
fieldId   number(15,0)     null,
listIndex number(15,0) not null,
vuid      number(15,0)     null,
propShort varchar(255)     null,
propLong  clob             null);
CREATE UNIQUE INDEX field_dispprop_ind
ON field_dispprop (schemald, fieldId, listIndex, vuid);

CREATE TABLE field_int
(schemald  number(15,0) not null,
fieldId   number(15,0) not null,
rangeLow  number(15,0)     null,
rangeHigh number(15,0)     null);
CREATE UNIQUE INDEX field_int_ind
ON field_int (schemald, fieldId);

CREATE TABLE field_real
(schemald  number(15,0) not null,
fieldId   number(15,0) not null,
rangelow  float            null,
rangehigh float            null,
arprecision number(15,0)     null);
CREATE UNIQUE INDEX field_real_ind
ON field_real (schemald, fieldId);

CREATE TABLE field_diary
(schemald  number(15,0) not null,
fieldId   number(15,0) not null,
fullTextOptions number(15,0)  null,
isLong    number(15,0)     null);
CREATE UNIQUE INDEX field_diary_ind
ON field_diary (schemald, fieldId);

CREATE TABLE field_char
(schemald  number(15,0) not null,
fieldId   number(15,0) not null,
maxLength number(15,0)     null,
qbeMatchOp number(15,0)     null,
menuStyle number(15,0)     null,
charMenu  varchar(254)     null,
...
pattern varchar(255) null,
fullTextOptions number(15,0) null,
isLong number(15,0) null;

CREATE UNIQUE INDEX field_char_ind
ON field_char (schemaId, fieldId);

CREATE TABLE field_enum
(schemaid number(15,0) not null,
fieldId number(15,0) not null,
maxEnum number(15,0) not null,
enumStyle number(15,0) null,
schemaName varchar(254) null,
serverName varchar(64) null,
nameField number(15,0) null,
numberField number(15,0) null,
queryShort varchar(255) null,
queryLong clob null);

CREATE UNIQUE INDEX field_enum_ind
ON field_enum (schemaId, fieldId);

CREATE TABLE field_enum_values
(schemaid number(15,0) not null,
fieldId number(15,0) not null,
enumId number(15,0) not null,
value varchar(254) not null);

CREATE INDEX field_enum_val_ind
ON field_enum_values (schemaId, fieldId);

CREATE TABLE field_permissions
(schemaid number(15,0) not null,
fieldId number(15,0) not null,
groupId number(15,0) not null,
permission number(15,0) not null);

CREATE INDEX field_permissions_ind
ON field_permissions (schemaId, fieldId);

CREATE TABLE field_attach
(schemaid number(15,0) not null,
fieldId number(15,0) not null,
maxSize number(15,0) not null,
attachType number(15,0) not null,
fullTextOptions number(15,0) null);

CREATE UNIQUE INDEX field_attach_ind
ON field_attach (schemaId, fieldId);

CREATE TABLE field_table
(schemaid number(15,0) not null,
fieldId number(15,0) not null,
umColumns number(15,0) not null,
maxRetrieve number(15,0) not null,

tfSchema varchar(254) not null,
tfServer varchar(64) not null,
queryShort varchar(255) null,
queryLong clob null,
sampleSchema varchar(254) null,
sampleServer varchar(64) null);

CREATE UNIQUE INDEX field_table_ind
ON field_table (schemaId, fieldId);
CREATE TABLE field_column
  (schemaId number(15,0) not null,
   fieldId number(15,0) not null,
   parent number(15,0) not null,
   dataField number(15,0) not null,
   colLength number(15,0) not null,
   dataSource number(15,0) null);
CREATE UNIQUE INDEX field_column_ind
  ON field_column (schemaId, fieldId);
CREATE TABLE field_dec
  (schemaId number(15,0) not null,
   fieldId number(15,0) not null,
   rangeLow varchar(64) null,
   rangeHigh varchar(64) null,
   arprecision number(15,0) null);
CREATE UNIQUE INDEX field_dec_ind
  ON field_dec (schemaId, fieldId);
CREATE TABLE field_curr
  (schemaId number(15,0) not null,
   fieldId number(15,0) not null,
   rangeLow varchar(64) null,
   rangeHigh varchar(64) null,
   arprecision number(15,0) null,
   funcCurr clob null,
   allowCurr clob null);
CREATE UNIQUE INDEX field_curr_ind
  ON field_curr (schemaId, fieldId);
CREATE TABLE field_view
  (schemaId number(15,0) not null,
   fieldId number(15,0) not null,
   maxLength number(15,0) null);
CREATE UNIQUE INDEX field_view_ind
  ON field_view (schemaId, fieldId);
CREATE TABLE field_display
  (schemaId number(15,0) not null,
   fieldId number(15,0) not null,
   maxLength number(15,0) null,
   isLong number(15,0) null);
CREATE UNIQUE INDEX field_display_ind
  ON field_display (schemaId, fieldId);
CREATE TABLE field_date
  (schemaId number(15,0) not null,
   fieldId number(15,0) not null,
   minDate number(15,0) null,
   maxDate number(15,0) null);
CREATE UNIQUE INDEX field_date_ind
  ON field_date (schemaId, fieldId);
CREATE TABLE join_mapping
  (schemaId number(15,0) not null,
   fieldId number(15,0) not null,
   memberIndex number(15,0) not null,
   mfieldId number(15,0) not null);
CREATE UNIQUE INDEX join_mapping_ind
  ON join_mapping (schemaId, fieldId);
CREATE TABLE view_mapping
    (schemaId number(15,0) not null,
     fieldId number(15,0) not null,
     extField varchar(254) not null);
CREATE UNIQUE INDEX view_mapping_ind
    ON view_mapping (schemaId, fieldId);
CREATE TABLE vendor_mapping
    (schemaId number(15,0) not null,
     fieldId number(15,0) not null,
     extField varchar(254) not null);
CREATE UNIQUE INDEX vendor_mapping_ind
    ON vendor_mapping (schemaId, fieldId);
CREATE TABLE char_menu
    (name varchar(254) not null,
     charMenuId number(15,0) not null,
     timestamp number(15,0) not null,
     owner varchar(254) not null,
     lastChanged varchar(254) not null,
     refreshCode number(15,0) not null,
     menuType number(15,0) not null,
     safeGuard varchar(254) not null,
     helpText clob null,
     changeDiary clob null,
     objProp clob null,
     version varchar(32) null,
     smObjProp clob null);
CREATE UNIQUE INDEX char_menu_ind
    ON char_menu (name);
CREATE UNIQUE INDEX char_menu_id_ind
    ON char_menu (charMenuId);
CREATE TABLE char_menu_list
    (charMenuId number(15,0) not null,
     path varchar(30) not null,
     label varchar(254) not null,
     childType number(15,0) not null,
     value varchar(255) null);
CREATE INDEX char_menu_list_ind
    ON char_menu_list (charMenuId);
CREATE TABLE char_menu_query
    (charMenuId number(15,0) not null,
     path varchar(30) not null,
     arschema varchar(254) not null,
     server varchar(255) not null,
     labelField number(15,0) not null,
     labelField2 number(15,0) null,
     labelField3 number(15,0) null,
     labelField4 number(15,0) null,
     labelField5 number(15,0) null,
     valueField number(15,0) not null,
     sortOnLabel number(15,0) not null,
     queryShort clob null,
     queryLong clob null,
     keywordList clob null,
CREATE TABLE char_menu_query
    (charMenuId number(15,0) not null, path varchar(30) not null, fileName varchar(255) not null);

CREATE TABLE char_menu_file
    (charMenuId number(15,0) not null, path varchar(30) not null, fileLocation number(15,0) not null, filename varchar(255) not null);

CREATE TABLE char_menu_sql
    (charMenuId number(15,0) not null, path varchar(30) not null, server varchar(255) not null, labelIndex number(15,0) not null, valueIndex number(15,0) not null, sqlCmdShort varchar(255) null, sqlCmdLong clob null, keywordList clob null, parameterList clob null, externList clob null);

CREATE TABLE char_menu_dd
    (charMenuId number(15,0) not null, path varchar(30) not null, server varchar(64) not null, structType number(15,0) not null, nameType number(15,0) not null, valueFormat number(15,0) not null, structSubtype number(15,0) null, arschema varchar(254) null, hiddenToo number(15,0) null);

CREATE TABLE arcontainer
    (name varchar(254) not null, containerId number(15,0) not null, containerType number(15,0) not null, timestamp number(15,0) not null, owner varchar(254) not null, lastChanged varchar(254) not null, numReferences number(15,0) not null, label varchar(255) null, safeGuard varchar(254) not null, description varchar(2000) null,
CREATE UNIQUE INDEX arctr_ind ON arcontainer (name);
CREATE UNIQUE INDEX arctr_id_ind ON arcontainer (containerId);
CREATE TABLE arctr_group_ids
  (containerId number(15,0) not null,
   groupId number(15,0) not null,
   permission number(15,0) not null);
CREATE INDEX arctr_group_ind ON arctr_group_ids (containerId);
CREATE TABLE arctr_subadmin
  (containerId number(15,0) not null,
   groupId number(15,0) not null);
CREATE INDEX arctr_subadmin_ind ON arctr_subadmin (containerId);
CREATE TABLE cntnr_ownr_obj
  (containerId number(15,0) not null,
   ownerObjType number(15,0) not null,
   ownerObjId number(15,0) not null,
   objIndex number(15,0) not null);
CREATE INDEX cntnr_ownr_id_ind ON cntnr_ownr_obj (containerId);
CREATE INDEX cntnr_ownr_obj_ind ON cntnr_ownr_obj (ownerObjType, ownerObjId);
CREATE UNIQUE INDEX cntnr_ownr_ind ON cntnr_ownr_obj (containerId, ownerObjType, ownerObjId);
CREATE TABLE arreference
  (containerId number(15,0) not null,
   referenceId number(15,0) not null,
   referenceType number(15,0) not null,
   dataType number(15,0) not null,
   referenceOrder number(15,0) not null,
   referenceObjId number(15,0) null,
   valueShort varchar(255) null,
   label varchar(255) null,
   valueLong clob null,
   description varchar(2000) null);
CREATE UNIQUE INDEX arref_ind ON arreference (containerId, referenceId);
CREATE TABLE arref_group_ids
  (containerId number(15,0) not null,
   referenceId number(15,0) not null,
   groupId number(15,0) not null);
CREATE INDEX arref_group_ind ON arref_group_ids (containerId, referenceId);
CREATE TABLE filter
(name varchar(254) not null,
filterId number(15,0) not null,
timestamp number(15,0) not null,
owner varchar(254) not null,
lastChanged varchar(254) not null,
wkConnType number(15,0) not null,
foOrder number(15,0) not null,
opSet number(15,0) not null,
enable number(15,0) not null,
nmActions number(15,0) not null,
nmElbes number(15,0) not null,
safeGuard varchar(254) not null,
queryShort varchar(255) null,
queryLong clob null,
changeDiary clob null,
helpText clob null,
objProp clob null,
version varchar(32) null,
smObjProp clob null);
CREATE UNIQUE INDEX filter_ind
ON filter (name);
CREATE UNIQUE INDEX filter_id_ind
ON filter (filterId);
CREATE TABLE filter_notify
(filterId number(15,0) not null,
actionIndex number(15,0) not null,
userName varchar(255) not null,
notifyText varchar(255) null,
priority number(15,0) not null,
mechanism number(15,0) not null,
mechXRef number(15,0) not null,
fieldIdCode number(15,0) not null,
subjectText varchar(255) null,
behavior number(15,0) null,
permission number(15,0) null,
fromUser varchar(255) null,
replyTo varchar(255) null,
cc varchar(255) null,
bcc varchar(255) null,
organization varchar(255) null,
mailboxName varchar(255) null,
headerTemplate varchar(255) null,
footerTemplate varchar(255) null,
contentTemplate varchar(255) null,
notifyTextLong clob null);
CREATE INDEX filter_notify_ind
ON filter_notify (filterId);
CREATE TABLE filter_notify_ids
(filterId number(15,0) not null,
actionIndex number(15,0) not null,
fieldId number(15,0) not null);
CREATE INDEX filter_notify_ids_ind
ON filter_notify_ids (filterId, actionIndex);
CREATE TABLE filter_message
  (filterId number(15,0) not null,
   actionIndex number(15,0) not null,
   msgType number(15,0) not null,
   msgNum number(15,0) not null,
   msgText varchar(255) not null);
CREATE INDEX filter_message_ind
ON filter_message (filterId);

CREATE TABLE filter_log
  (filterId number(15,0) not null,
   actionIndex number(15,0) not null,
   logFile varchar(255) null);
CREATE INDEX filter_log_ind
ON filter_log (filterId);

CREATE TABLE filter_set
  (filterId number(15,0) not null,
   actionIndex number(15,0) not null,
   fieldId number(15,0) not null,
   assignShort varchar(255) null,
   assignLong clob null,
   sampleSchema varchar(254) null,
   sampleServer varchar(64) null);
CREATE INDEX filter_set_ind
ON filter_set (filterId);

CREATE TABLE filter_process
  (filterId number(15,0) not null,
   actionIndex number(15,0) not null,
   command varchar(255) not null);
CREATE INDEX filter_process_ind
ON filter_process (filterId);

CREATE TABLE filter_push
  (filterId number(15,0) not null,
   actionIndex number(15,0) not null,
   fieldId number(15,0) not null,
   assignShort varchar(255) null,
   assignLong clob null,
   sampleSchema varchar(254) null,
   sampleServer varchar(64) null);
CREATE INDEX filter_push_ind
ON filter_push (filterId);

CREATE TABLE filter_sql
  (filterId number(15,0) not null,
   actionIndex number(15,0) not null,
   assignShort varchar(255) null,
   assignLong clob null);
CREATE INDEX filter_sql_ind
ON filter_sql (filterId);

CREATE TABLE filter_gotoaction
  (filterId number(15,0) not null,
   actionIndex number(15,0) not null,
   tag number (15,0) not null,
   fieldIdOrValue number(15,0) default 0 null);
CREATE INDEX filter_gotoa_ind
ON filter_gotoaction (filterId);
CREATE TABLE filter_call
  (filterId number(15,0) not null,
   actionIndex number(15,0) not null,
   serverName varchar(64) not null,
   guideName varchar(254) not null,
   guideMode number(15,0) not null,
   guideTableId number(15,0) null,
   assignShort varchar(255) null,
   assignLong clob null,
   sampleServer varchar(64) null,
   sampleGuide varchar(254) null);
CREATE INDEX filter_call_ind
  ON filter_call (filterId);

CREATE TABLE filter_exit
  (filterId number(15,0) not null,
   actionIndex number(15,0) not null,
   closeAll char null);
CREATE INDEX filter_exit_ind
  ON filter_exit (filterId);

CREATE TABLE filter_goto
  (filterId number(15,0) not null,
   actionIndex number(15,0) not null,
   label varchar(128) not null);
CREATE INDEX filter_goto_ind
  ON filter_goto (filterId);

CREATE TABLE filter_mapping
  (schemaId number(15,0) not null,
   objIndex number(15,0) not null,
   filterId number(15,0) not null);
CREATE UNIQUE INDEX filter_mapping_ind
  ON filter_mapping (schemaId, filterId);

CREATE TABLE escalation
  (name varchar(254) not null,
   escalationId number(15,0) not null,
   timestamp number(15,0) not null,
   owner varchar(254) not null,
   lastChanged varchar(254) not null,
   wkConnType number(15,0) not null,
   numActions number(15,0) not null,
   numElses number(15,0) not null,
   firetmType number(15,0) not null,
   tminterval number(15,0) not null,
   monthday number(15,0) not null,
   weekday number(15,0) not null,
   hourmask number(15,0) not null,
   minute number(15,0) not null,
   enable number(15,0) not null,
   safeGuard varchar(254) not null,
   queryShort varchar(255) null,
   queryLong clob null,
CREATE UNIQUe INDEX escalation_ind
ON escalation (name);
CREATE UNIQUE INDEX escalation_id_ind
ON escalation (escalationId);
CREATE TABLE escal_mapping
(schemaid number(15,0) not null,
objIndex number(15,0) not null,
escalationId number(15,0) not null);
CREATE UNIQUE INDEX escal_mapping_ind
ON escal_mapping (schemaid, escalationId);
CREATE TABLE actlink
(name varchar(254) not null,
actlinkid number(15,0) not null,
timestamp number(15,0) not null,
owner varchar(254) not null,
lastChanged varchar(254) not null,
wkConnType number(15,0) not null,
alOrder number(15,0) not null,
executeMask number(15,0) not null,
controlfieldId number(15,0) not null,
fieldId number(15,0) not null,
enable number(15,0) not null,
numActions number(15,0) not null,
numElses number(15,0) not null,
safeGuard varchar(254) not null,
queryShort varchar(255) null,
queryLong clob null,
helpText clob null,
changeDiary clob null,
objProp clob null,
version varchar(32) null,
smObjProp clob null);
CREATE INDEX actlink_macro_ind
ON actlink_macro (actlinkId);

CREATE TABLE actlink_macro_parm
(actlinkId number(15,0) not null,
actionIndex number(15,0) not null,
name varchar(254) not null,
value varchar(255) not null);

CREATE INDEX alk_ma_parm_ind
ON actlink_macro_parm (actlinkId, actionIndex);

CREATE TABLE actlink_set
(actlinkId number(15,0) not null,
fieldId number(15,0) not null,
assignShort varchar(255) null,
assignLong clob null,
keywordList clob null,
parameterList clob null,
sampleSchema varchar(254) null,
sampleServer varchar(64) null);

CREATE INDEX actlink_set_ind
ON actlink_set (actlinkId);

CREATE TABLE actlink_process
(actlinkId number(15,0) not null,
actionIndex number(15,0) not null,
command varchar(255) not null,
keywordList varchar(255) null,
parameterList varchar(255) null);

CREATE INDEX actlink過程中
NO actlink_process (actlinkId);

CREATE TABLE actlink_message
(actlinkId number(15,0) not null,
actionIndex number(15,0) not null,
msgType number(15,0) not null,
msgNum number(15,0) not null,
msgText clob not null,
msgPane char default '0' null);

CREATE INDEX actlink_message_ind
ON actlink_message (actlinkId);

CREATE TABLE actlink_set_char
(actlinkId number(15,0) not null,
actionIndex number(15,0) not null,
fieldId number(15,0) not null,
charMenu varchar(254) null,
propShort varchar(255) null,
propLong clob null,
focus number(15,0) null,
accessOpt number(15,0) default 0 null);

CREATE INDEX actlink_schar_ind
ON actlink_set_char (actlinkId);

CREATE TABLE actlink_sde
(actlinkId number(15,0) not null,
actionIndex number(15,0) not null,
serviceName varchar(64) not null,
CREATE INDEX actlink_dde_ind
ON actlink_dde (actlinkId);

CREATE TABLE actlink_auto
(actlinkId number(15,0) not null,
actionIndex number(15,0) not null,
autoServerName varchar(255) not null,
cId varchar(128) not null,
isVisible char not null,
actionShort varchar(255) null,
actionLong varchar(2000) null,
COMShort varchar(255) null,
COMLong clob null);

CREATE INDEX actlink_auto_ind
ON actlink_auto (actlinkId);

CREATE TABLE actlink_push
(actlinkId number(15,0) not null,
actionIndex number(15,0) not null,
fieldId number(15,0) not null,
assignShort varchar(255) null,
assignLong clob null,
sampleSchema varchar(255) null,
sampleServer varchar(64) null);

CREATE INDEX actlink_push_ind
ON actlink_push (actlinkId);

CREATE TABLE actlink_sql
(actlinkId number(15,0) not null,
actionIndex number(15,0) not null,
assignShort varchar(255) null,
assignLong clob null,
keywordList clob null,
parameterList clob null);

CREATE INDEX actlink_sql_ind
ON actlink_sql (actlinkId);

CREATE TABLE actlink_open
(actlinkId number(15,0) not null,
actionIndex number(15,0) not null,
serverName varchar(64) not null,
schemaName varchar(254) not null,
vuiLabel varchar(254) null,
closeBox char null,
assignShort varchar(255) null,
assignLong clob null,
windowMode number(15,0) null,
noMatchCnt nu char null,
pollIntval number(15,0) null,
sortList varchar(255) null,
queryShort varchar(255) null,
queryLong clob null,
msgType number(15,0) null,
CREATE TABLE actlink_open
  (actlinkId   number(15,0) not null,
   actionIndex number(15,0) not null,
   msgNum         number(15,0)     null,
   msgText        clob             null,
   msgPane        char             null,
   reportstr      clob             null,
   supressEptyLst  char             null,
   targetLocation varchar(255)     null);
CREATE INDEX actlink_open_ind
  ON actlink_open (actlinkId);

CREATE TABLE actlink_commit
  (actlinkId   number(15,0) not null,
   actionIndex number(15,0) not null);
CREATE INDEX actlink_commit_ind
  ON actlink_commit (actlinkId);

CREATE TABLE actlink_close
  (actlinkId   number(15,0) not null,
   actionIndex number(15,0) not null,
   closeAll    char             null);
CREATE INDEX actlink_close_ind
  ON actlink_close (actlinkId);

CREATE TABLE actlink_call
  (actlinkId   number(15,0) not null,
   actionIndex number(15,0) not null,
   serverName  varchar(64)  not null,
   guideName   varchar(254) not null,
   guideMode   number(15,0) not null,
   guideTableId number(15,0)    null,
   assignShort varchar(255)     null,
   assignLong  clob             null,
   sampleServer varchar(64)     null,
   sampleGuide  varchar(254)    null);
CREATE INDEX actlink_call_ind
  ON actlink_call (actlinkId);

CREATE TABLE actlink_exit
  (actlinkId   number(15,0) not null,
   actionIndex number(15,0) not null,
   closeAll    char             null);
CREATE INDEX actlink_exit_ind
  ON actlink_exit (actlinkId);

CREATE TABLE actlink_goto
  (actlinkId   number(15,0) not null,
   actionIndex number(15,0) not null,
   label       varchar(128) not null);
CREATE INDEX actlink_goto_ind
  ON actlink_goto (actlinkId);

CREATE TABLE actlink_gotoaction
  (actlinkId    number(15,0)not null,
   actionIndex  number(15,0)not null,
CREATE INDEX actlink_gotoa_ind
  ON actlink_gotoaction (actlinkId);

CREATE TABLE actlink_mapping
  (schemaId   number(15,0) not null,
   objIndex   number(15,0) not null,
   actlinkId   number(15,0) not null);
CREATE UNIQUE INDEX actlink_mapping_ind
  ON actlink_mapping (schemaId, actlinkId);

CREATE TABLE alert_user
  (username      varchar(254) not null,
   clientIPAddr  varchar(16)  not null,
   actualIPAddr  varchar(16)  not null,
   serverIPAddr  varchar(16)  not null,
   clientPort    number(15,0) not null,
   regFlags      number(15,0) not null,
   clientVersion number(15,0) not null,
   regTime       number(15,0) not null,
   clientCodeSet number(15,0) not null);
CREATE UNIQUE INDEX alert_user_ind
  ON alert_user (username, clientIPAddr, clientPort);

CREATE TABLE alert_time
  (username      varchar(254)   not null,
   checkpointTime number(15,0) not null);
CREATE UNIQUE INDEX alert_time_ind
  ON alert_time (username);

CREATE TABLE support_file
  (fileType    number(15,0) not null,
   id          number(15,0) not null,
   id2         number(15,0) not null,
   fileId      number(15,0) not null,
   timestamp   number(15,0) not null,
   fileContent blob         null);
CREATE UNIQUE INDEX support_file_ind
  ON support_file (fileType, id, id2, fileId);

CREATE TABLE servgrp_config
  (name          varchar(64)      null,
   checkInterval number(15,0) not null);

CREATE TABLE servgrp_op_mstr
  (operation      varchar(255) not null,
   opNum          number(15,0) not null,
   configLabel    varchar(255)     null,
   configCommand  varchar(50)      null,
   categoryStrs   varchar(255)     null);
CREATE TABLE ft_pending
(serverName varchar(64) not null,
schemaId number(15,0) not null,
fieldId number(15,0) not null,
entryId varchar(15) null,
operationType number(15,0) not null,
updateTime number(15,0) null,
seqNum number(15,0) not null);
CREATE INDEX ft_pending_ind
ON ft_pending (seqNum);

Sybase and Microsoft SQL Server

The following set of SQL commands define the AR System data dictionary
for Sybase and Microsoft SQL Server databases. For an explanation of these
commands, see the Sybase Commands Reference Manual or the Transact-SQL
Desk Reference: For Microsoft SQL Server.

The data definitions in this section are the same for Microsoft SQL Server
databases configured to use Unicode, except that three column types are
changed to indicate that they are Unicode columns as follows:

<table>
<thead>
<tr>
<th>Standard column type</th>
<th>Unicode column type</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>nchar</td>
</tr>
<tr>
<td>varchar</td>
<td>nvarchar</td>
</tr>
<tr>
<td>text</td>
<td>ntext</td>
</tr>
</tbody>
</table>

use ARSystem
go
CREATE TABLE control
(dbVersion int not null,
schemaId int not null,
filterId int not null,
serverId int not null,
containerId int not null,
actlinkkid int not null,
adminExtId int not null,
charMenulId int not null)
go
CREATE TABLE arschema
(name varchar(254) not null,
schemaId int not null,
schemaType int not null,
timestamp int not null,
owner varchar(254) not null,
CREATE TABLE arschema
(
lastChanged varchar(254) not null,
coreVersion int not null,
umFields int not null,
umVuis int not null,
defaultVui varchar(254) not null,
nextId int not null,
maxStatEnums int not null,
nextFieldId int not null,
upgrdVersion int null,
safeGuard varchar(254) not null,
changeDiary text null,
helpText text null,
objProp text null,
version varchar(32) null,
smObjProp text null
)
go
CREATE UNIQUE INDEX schema_ind
ON arschema (name)
CREATE UNIQUE CLUSTERED INDEX schema_id_ind
ON arschema (schemaId)
go
CREATE TABLE schema_group_ids
(schemaId int not null,
groupId int not null,
permission int not null)
go
CREATE CLUSTERED INDEX schema_group_ids_ind
ON schema_group_ids (schemaId)
go
CREATE TABLE subadmin_group
(schemaId int not null,
groupId int not null)
go
CREATE CLUSTERED INDEX subadmin_group_ind
ON subadmin_group (schemaId)
go
CREATE TABLE schema_list_fields
(schemaId int not null,
listIndex int not null,
fieldId int not null,
columnWidth int not null,
separatorLen int not null,
separator varchar(10) null)
go
CREATE CLUSTERED INDEX schema_list_fields_ind
ON schema_list_fields (schemaId)
go
CREATE TABLE schema_sort
(schemaId int not null,
listIndex int not null,
fieldId int not null,
sortOrder int not null)
go
CREATE CLUSTERED INDEX schema_sort_ind
ON schema_sort (schemaId)
go
CREATE TABLE schema_archive
(schemaId int not null,
 enable int not null,
 archiveType int not null,
 archiveToForm int null,
 archiveToFile varchar(255) null,
 queryShort varchar(255) null,
 queryLong text null,
 monthday int not null,
 weekday int not null,
 hourmask int not null,
 minute int not null,
 archiveFromForm int null)
go
CREATE CLUSTERED INDEX schema_archive_ind
ON schema_archive (schemaId)
go
CREATE TABLE schema_audit
(schemaId int not null,
 enable int not null,
 style int not null,
 form int null,
 queryShort varchar(255) null,
 queryLong text null)
go
CREATE CLUSTERED INDEX schema_audit_ind
ON schema_audit (schemaId)
go
CREATE TABLE schema_index
(schemaId int not null,
 listIndex int not null,
 numFields int not null,
 uniqueFlag int not null,
 indexName varchar(254) not null,
 f1 int not null,
 f2 int null,
 f3 int null,
 f4 int null,
 f5 int null,
 f6 int null,
 f7 int null,
 f8 int null,
 f9 int null,
 f10 int null,
 f11 int null,
 f12 int null,
 f13 int null,
 f14 int null,
 f15 int null,
 f16 int null)

go
108 Chapter 2—SQL Definitions of the data dictionary tables
CREATE CLUSTERED INDEX schema_index_ind
ON schema_index (schemaId)
go
CREATE TABLE schema_join
(schemaId int not null,
memberA varchar(254) not null,
memberB varchar(254) not null,
option int null,
queryShort varchar(255) null,
queryLong text null)
go
CREATE UNIQUE INDEX schema_join_ind
ON schema_join (schemaId)
go
CREATE TABLE schema_view
(schemaId int not null,
tableName text null,
keyField varchar(254) not null,
queryShort varchar(255) null,
queryLong text null)
go
CREATE UNIQUE INDEX schema_view_ind
ON schema_view (schemaId)
go
CREATE TABLE schema_vendor
(schemaId int not null,
vendorName varchar(254) not null,
tableName text null)
go
CREATE UNIQUE INDEX schema_vendor_ind
ON schema_vendor (schemaId)
go
CREATE TABLE field
(schemaId int not null,
fieldId int not null,
fieldName varchar(254) not null,
fieldType int not null,
timestamp int not null,
owner varchar(254) not null,
lastChanged varchar(254) not null,
datatype int not null,
foption int not null,
createMode int not null,
fbOption int null,
defaultValue varchar(255) null,
changeDiary text null,
helpText text null)
go
CREATE UNIQUE CLUSTERED INDEX field_ind
ON field (schemaId, fieldId)
go
CREATE INDEX field_schema_ind
ON field (schemaId)
CREATE TABLE vui
(schemaId int not null,
vuiId int not null,
vuiName varchar(254) not null,
locale varchar(30) null,
vuiType int null,
timestamp int not null,
owner varchar(254) not null,
lastChanged varchar(254) not null,
changeDiary text null,
helpText text null)
go
CREATE UNIQUE CLUSTERED INDEX vui_ind ON vui (schemaId, vuiId)
CREATE INDEX vui_schema_ind ON vui (schemaId)
go
CREATE TABLE field_dispprop
(schemaId int not null,
fieldId int null,
listIndex int not null,
vuiId int null,
propShort varchar(255) null,
propLong text null)
go
CREATE UNIQUE INDEX field_dispprop_ind ON field_dispprop (schemaId, fieldId, listIndex, vuiId)
go
CREATE TABLE field_int
(schemaId int not null,
fieldId int not null,
rangLow int null,
rangHigh int null)
go
CREATE UNIQUE CLUSTERED INDEX field_int_ind ON field_int (schemaId, fieldId)
go
CREATE TABLE field_real
(schemaId int not null,
fieldId int not null,
rangLow float null,
rangHigh float null,
arprecision int null)
go
CREATE UNIQUE CLUSTERED INDEX field_real_ind ON field_real (schemaId, fieldId)
go
CREATE TABLE field_diary
(schemaId int not null,
fieldId int not null,
fullTextOptions int null)
go
CREATE UNIQUE CLUSTERED INDEX field_diary_ind
ON field_diary (schemaId, fieldId)
go
CREATE TABLE field_char
(schemaId int not null,
fieldId int not null,
maxLength int null,
qbeMatchOp int null,
menuStyle int null,
charMenu varchar(254) null,
pattern varchar(255) null,
fullTextOptions int null)
go
CREATE UNIQUE CLUSTERED INDEX field_char_ind
ON field_char (schemaId, fieldId)
go
CREATE TABLE field_enum
(schemaId int not null,
fieldId int not null,
maxEnum int not null,
enumStyle int null,
schemaName varchar(254) null,
serverName varchar(64) null,
nameField int null,
numberField int null,
queryShort varchar(255) null,
queryLong text null)
go
CREATE UNIQUE CLUSTERED INDEX field_enum_ind
ON field_enum (schemaId, fieldId)
go
CREATE TABLE field_enum_values
(schemaId int not null,
fieldId int not null,
enumId int not null,
value varchar(254) not null)
go
CREATE CLUSTERED INDEX field_enum_val_ind
ON field_enum_values (schemaId, fieldId)
go
CREATE TABLE field_permissions
(schemaId int not null,
fieldId int not null,
groupId int not null,
permission int not null)
go
CREATE CLUSTERED INDEX field_permissions_ind
ON field_permissions (schemaId, fieldId)
go
CREATE TABLE field_attach
(schemaId int not null,
fieldId int not null,
maxSize int not null,
CREATE TABLE field_table
(schemaid int not null,
fieldid int not null,
numColumns int not null,
maxRetrieve int not null,
tfSchema varchar(254) not null,
tfServer varchar(64) not null,
queryShort varchar(255) null,
queryLong text null,
sampleSchema varchar(254) null,
sampleServer varchar(64) null)

CREATE UNIQUE CLUSTERED INDEX field_table_ind
ON field_table (schemaid, fieldid)

CREATE TABLE field_column
(schemaid int not null,
fieldid int not null,
parnt int not null,
dataField int not null,
colLength int not null,
dataSource int null)

CREATE UNIQUE CLUSTERED INDEX field_column_ind
ON field_column (schemaid, fieldid)

CREATE TABLE field_dec
(schemaid int not null,
fieldid int not null,
rangLow varchar(64) null,
rangHigh varchar(64) null,
arprecision int null)

CREATE UNIQUE CLUSTERED INDEX field_dec_ind
ON field_dec (schemaid, fieldid)

CREATE TABLE field_curr
(schemaid int not null,
fieldid int not null,
rangLow varchar(64) null,
rangHigh varchar(64) null,
arprecision int null,
funcCurr text null,
allowCurr text null)

CREATE UNIQUE CLUSTERED INDEX field_curr_ind
ON field_curr (schemaid, fieldid)
CREATE TABLE join_mapping
(schemaId int not null,
fieldId int not null,
memberIndex int not null,
mfieldId int not null)

CREATE TABLE field_view
(schemaId int not null,
fieldId int not null,
maxLength int null)

CREATE UNIQUE CLUSTERED INDEX field_view_ind
ON field_view (schemaId, fieldId)

CREATE TABLE field_display
(schemaId int not null,
fieldId int not null,
maxLength int null)

CREATE UNIQUE CLUSTERED INDEX field_display_ind
ON field_display (schemaId, fieldId)

CREATE TABLE field_date
(schemaId int not null,
fieldId int not null,
minDate int null,
maxDate int null)

CREATE UNIQUE CLUSTERED INDEX field_date_ind
ON field_date (schemaId, fieldId)

CREATE UNIQUE INDEX join_mapping_ind
ON join_mapping (schemaId, fieldId)

CREATE TABLE view_mapping
(schemaId int not null,
fieldId int not null,
extField varchar(254) not null)

CREATE UNIQUE INDEX view_mapping_ind
ON view_mapping (schemaId, fieldId)

CREATE TABLE vendor_mapping
(schemaId int not null,
fieldId int not null,
extField varchar(254) not null)

CREATE UNIQUE INDEX vendor_mapping_ind
ON vendor_mapping (schemaId, fieldId)

CREATE TABLE char_menu
(icon varchar(254) not null,
charMenuId int not null)
Chapter 2—SQL Definitions of the data dictionary tables

```
timestamp    int          not null,
owner        varchar(254) not null,
lastChanged  varchar(254) not null,
refreshCode  int          not null,
menuType     int          not null,
safeGuard    varchar(254) not null,
changeDiary  text             null,
helpText     text             null,
objProp      text             null,
version      varchar(32)      null,
smObjProp    text             null)
go
CREATE UNIQUE CLUSTERED INDEX char_menu_ind
ON char_menu (name)
CREATE UNIQUE INDEX char_menu_id_ind
ON char_menu (charMenuId)
go
CREATE TABLE char_menu_list
(charMenuId   int          not null,
path         varchar(30)  not null,
label        varchar(254) not null,
childType    int          not null,
value        varchar(255)     null)
go
CREATE CLUSTERED INDEX char_menu_list_ind
ON char_menu_list (charMenuId)
go
CREATE TABLE char_menu_query
(charMenuId   int          not null,
path         varchar(30)  not null,
arschema     varchar(254) not null,
server       varchar(255) not null,
labelField   int          not null,
labelField2  int              null,
labelField3  int              null,
labelField4  int              null,
labelField5  int              null,
valueField   int          not null,
sortOnLabel  int          not null,
queryShort   varchar(255)     null,
queryLong    text             null,
keywordList  text             null,
parameterList text            null,
externList   text             null,
sampleSchema varchar(254)     null,
sampleServer varchar(64)      null)
go
CREATE CLUSTERED INDEX char_menu_qry_ind
ON char_menu_query (charMenuId)
go
CREATE TABLE char_menu_file
(charMenuId   int          not null,
path         varchar(30)  not null,
```
CREATE TABLE char_menu_file

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Not Null</th>
</tr>
</thead>
<tbody>
<tr>
<td>charMenuId</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>fileLocation</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>filename</td>
<td>varchar(255)</td>
<td>not null</td>
</tr>
</tbody>
</table>

go

CREATE CLUSTERED INDEX char_menu_file_ind
ON char_menu_file (charMenuId)

go

CREATE TABLE char_menu_sql

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Not Null</th>
</tr>
</thead>
<tbody>
<tr>
<td>charMenuId</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>path</td>
<td>varchar(30)</td>
<td>not null</td>
</tr>
<tr>
<td>server</td>
<td>varchar(255)</td>
<td>not null</td>
</tr>
<tr>
<td>labelIndex</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>sqlCmdShort</td>
<td>varchar(255)</td>
<td>null</td>
</tr>
<tr>
<td>sqlCmdLong</td>
<td>text</td>
<td>null</td>
</tr>
<tr>
<td>keywordList</td>
<td>text</td>
<td>null</td>
</tr>
<tr>
<td>parameterList</td>
<td>text</td>
<td>null</td>
</tr>
<tr>
<td>externList</td>
<td>text</td>
<td>null</td>
</tr>
</tbody>
</table>

go

CREATE CLUSTERED INDEX char_menu_sql_ind
ON char_menu_sql (charMenuId)

go

CREATE TABLE char_menu_dd

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Not Null</th>
</tr>
</thead>
<tbody>
<tr>
<td>charMenuId</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>path</td>
<td>varchar(30)</td>
<td>not null</td>
</tr>
<tr>
<td>server</td>
<td>varchar(64)</td>
<td>not null</td>
</tr>
<tr>
<td>structType</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>nameType</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>valueFormat</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>structSubtype</td>
<td>int</td>
<td>null</td>
</tr>
<tr>
<td>arschema</td>
<td>varchar(254)</td>
<td>null</td>
</tr>
<tr>
<td>hiddenToo</td>
<td>int</td>
<td>null</td>
</tr>
</tbody>
</table>

go

CREATE CLUSTERED INDEX char_menu_dd_ind
ON char_menu_dd (charMenuId)

go

CREATE TABLE arcontainer

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Not Null</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>varchar(254)</td>
<td>not null</td>
</tr>
<tr>
<td>containerId</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>containerType</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>timestamp</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>owner</td>
<td>varchar(254)</td>
<td>not null</td>
</tr>
<tr>
<td>lastChanged</td>
<td>varchar(254)</td>
<td>not null</td>
</tr>
<tr>
<td>numReferences</td>
<td>int</td>
<td>not null</td>
</tr>
<tr>
<td>label</td>
<td>varchar(255)</td>
<td>null</td>
</tr>
<tr>
<td>safeGuard</td>
<td>varchar(255)</td>
<td>null</td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>null</td>
</tr>
<tr>
<td>changeDiary</td>
<td>text</td>
<td>null</td>
</tr>
<tr>
<td>helpText</td>
<td>text</td>
<td>null</td>
</tr>
</tbody>
</table>
CREATE TABLE arctr_group_ids
  (containerId   int         not null,
   groupId       int         not null,
   permission    int         not null)

CREATE CLUSTERED INDEX arctr_group_ind
  ON arctr_group_ids (containerId)

CREATE TABLE arctr_subadmin
  (containerId   int         not null,
   groupId       int         not null)

CREATE CLUSTERED INDEX arctr_subadmin_ind
  ON arctr_subadmin (containerId)

CREATE TABLE cntnr_ownr_obj
  (containerId   int         not null,
   ownerObjType  int         not null,
   ownerObjId    int         not null,
   objIndex      int         not null)

CREATE INDEX cntnr_ownr_id_ind
  ON  cntnr_ownr_obj (containerId)
CREATE INDEX cntnr_ownr_obj_ind
  ON  cntnr_ownr_obj (ownerObjType, ownerObjId)
CREATE UNIQUE INDEX cntnr_ownr_ind
  ON  cntnr_ownr_obj (containerId, ownerObjType, ownerObjId)

CREATE TABLE arreference
  (containerId   int         not null,
   referenceId   int         not null,
   referenceType int         not null,
   dataType      int         not null,
   referenceOrder int         not null,
   referenceObjId int         null,
   valueShort    varchar(255) null,
   label         varchar(255) null,
   valueLong     text         null,
   description   text         null)
CREATE UNIQUE CLUSTERED INDEX arref_ind
ON arreference (containerId, referenceId)
go

CREATE TABLE arref_group_ids
(containerId   int         not null,
referenceId   int         not null,
groupId       int         not null)
go

CREATE CLUSTERED INDEX arref_group_ind
ON arref_group_ids (containerId, referenceId)
go

CREATE TABLE filter
(name         varchar(254) not null,
filterId     int          not null,
timestamp    int          not null,
owner        varchar(254)  not null,
lastChanged  varchar(254)  not null,
wkConnType   int          not null,
order        int          not null,
opSet        int          not null,
enable        int          not null,
umActions     int          not null,
umElses       int          not null,
safeGuard     varchar(254) not null,
queryShort    varchar(255)     null,
queryLong    text             null,
changeDiary  text             null,
helpText     text             null,
objProp      text             null,
version      varchar(32)      null,
smObjProp    text             null)
go

CREATE UNIQUE CLUSTERED INDEX filter_ind
ON filter (name)
CREATE UNIQUE INDEX filter_id_ind
ON filter (filterId)
go

CREATE TABLE filter_notify
(filterId       int          not null,
actionIndex    int          not null,
userName       varchar(255) not null,
notifyText     varchar(255)     null,
priority       int          not null,
mechanism      int          not null,
mechXRef       int          not null,
subjectText    varchar(255)     null,
behavior       int          null,
permission     int          null,
fromUser       varchar(255)     null,
replyTo        varchar(255)     null,
cc             varchar(255)     null,
CREATE CLUSTERED INDEX filter_notify_ind ON filter_notify (filterId)

CREATE TABLE filter_notify_ids
(fildId int not null,
 actionIndex int not null)

CREATE CLUSTERED INDEX filter_notify_ids_ind ON filter_notify_ids (filterId, actionIndex)

CREATE TABLE filter_message
(fildId int not null,
 actionIndex int not null,
 msgType int not null,
 msgNum int not null,
 msgText varchar(255) not null)

CREATE CLUSTERED INDEX filter_message_ind ON filter_message (filterId)

CREATE TABLE filter_log
(fildId int not null,
 actionIndex int not null,
 logFile varchar(255) null)

CREATE CLUSTERED INDEX filter_log_ind ON filter_log (filterId)

CREATE TABLE filter_set
(fildId int not null,
 actionIndex int not null,
 fieldId int not null,
 assignShort varchar(255) null,
 assignLong text null,
 sampleSchema varchar(254) null,
 sampleServer varchar(64) null)

CREATE CLUSTERED INDEX filter_set_ind ON filter_set (filterId)

CREATE TABLE filter_process
(fildId int not null,
 actionIndex int not null,
 command varchar(255) not null)
CREATE CLUSTERED INDEX filter_process_ind ON filter_process (filterId)
go
CREATE TABLE filter_push
  (filterId int not null,
   actionIndex int not null,
   fieldId int not null,
   assignShort varchar(255) null,
   assignLong text null,
   sampleSchema varchar(254) null,
   sampleServer varchar(64) null)
go
CREATE CLUSTERED INDEX filter_push_ind ON filter_push (filterId)
go
CREATE TABLE filter_sql
  (filterId int not null,
   actionIndex int not null,
   assignShort varchar(255) null,
   assignLong text null)
go
CREATE CLUSTERED INDEX filter_sql_ind ON filter_sql (filterId)
go
CREATE TABLE filter_gotoaction
  (filterId int not null,
   actionIndex int not null,
   tag int not null,
   fieldIdOrValue int default 0 null)
go
CREATE CLUSTERED INDEX filter_gotoa_ind ON filter_gotoaction (filterId)
go
CREATE TABLE filter_call
  (filterId int not null,
   actionIndex int not null,
   serverName varchar(64) not null,
   guideName varchar(254) not null,
   guideMode int not null,
   guideTableId int not null,
   assignShort varchar(255) null,
   assignLong text null,
   sampleServer varchar(64) null,
   sampleGuide varchar(254) null)
go
CREATE CLUSTERED INDEX filter_call_ind ON filter_call (filterId)
go
CREATE TABLE filter_exit
  (filterId int not null,
   actionIndex int not null,
   closeAll char null)
CREATE CLUSTERED INDEX filter_exit_ind ON filter_exit (filterId)

CREATE TABLE filter_goto
(filterId int not null,
actionIndex int not null,
label varchar(128) not null)

CREATE CLUSTERED INDEX filter_goto_ind ON filter_goto (filterId)

CREATE TABLE filter_mapping
(schemaId int not null,
objIndex int not null,
filterId int not null)

CREATE UNIQUE INDEX filter_mapping_ind ON filter_mapping (schemaId, filterId)

CREATE TABLE escalation
(name varchar(254) not null,
escalationId int not null,
timestamp int not null,
owner varchar(254) not null,
lastChanged varchar(254) not null,
wkConnType int not null,
umActions varchar(254) not null,
umElses int not null,
firetmType int not null,
tminterval int not null,
monthday int not null,
weekday int not null,
hourmask int not null,
minute int not null,
enable int not null,
safeGuard varchar(254) not null,
queryShort varchar(255) null,
queryLong text null,
changeDiary text null,
helpText text null,
objProp text null,
version varchar(32) null,
smObjProp text null)

CREATE UNIQUE CLUSTERED INDEX escalation_ind ON escalation (name)
CREATE UNIQUE INDEX escalation_id_ind ON escalation (escalationId)
CREATE TABLE escal_mapping
    (schemaId     int          not null,
     objIndex     int          not null,
     escalationId int          not null)

CREATE UNIQUE INDEX escal_mapping_ind
    ON escal_mapping (schemaId, escalationId)

CREATE TABLE actlink
    (name         varchar(254) not null,
     actlinkId    int          not null,
     timestamp    int          not null,
     owner        varchar(254)  not null,
     lastChanged  varchar(254)  not null,
     wkConnType   int          not null,
     alOrder      int          not null,
     executeMask  int          not null,
     controlfieldId      int       null,
     fieldId      int          not null,
     enable       int          not null,
     numActions   int          not null,
     numElses     int          not null,
     safeGuard    varchar(254) not null,
     queryShort   varchar(255)     null,
     queryLong    text             null,
     changeDiary  text             null,
     helpText     text             null,
     objProp      text             null,
     version      varchar(32)      null,
     smObjProp    text             null)

CREATE UNIQUE CLUSTERED INDEX actlink_ind
    ON actlink (name)

CREATE UNIQUE INDEX actlink_id_ind
    ON actlink (actlinkId)

CREATE TABLE actlink_group_ids
    (actlinkId   int         not null,
     groupId     int         not null)

CREATE CLUSTERED INDEX actlink_group_ids_ind
    ON actlink_group_ids (actlinkId)

CREATE TABLE actlink_macro
    (actlinkId   int          not null,
     actionIndex int          not null,
     macroName   varchar(254) not null,
     shortText   varchar(255)     null,
     longText    text             null)
CREATE CLUSTERED INDEX actlink_macro_ind
ON actlink_macro (actlinkId)
go
CREATE TABLE actlink_macro_parm
(actlinkId int not null,
 actionIndex int not null,
 name varchar(254) not null,
 value varchar(255) not null)
go
CREATE CLUSTERED INDEX alk_ma_parm_ind
ON actlink_macro_parm (actlinkId, actionIndex)
go
CREATE TABLE actlink_set
(actlinkId int not null,
 actionIndex int not null,
 fieldId int not null,
 assignShort varchar(255) null,
 assignLong text null,
 keywordList text null,
 parameterList text null,
 sampleSchema varchar(254) null,
 sampleServer varchar(64) null)
go
CREATE CLUSTERED INDEX actlink_set_ind
ON actlink_set (actlinkId)
go
CREATE TABLE actlink_process
(actlinkId int not null,
 actionIndex int not null,
 command varchar(255) not null,
 keywordList varchar(255) null,
 parameterList varchar(255) null)
go
CREATE CLUSTERED INDEX actlink_process_ind
ON actlink_process (actlinkId)
go
CREATE TABLE actlink_message
(actlinkId int not null,
 actionIndex int not null,
 msgType int not null,
 msgNum int not null,
 msgText text not null,
 msgPane char default '0' null)
go
CREATE CLUSTERED INDEX actlink_message_ind
ON actlink_message (actlinkId)
go
CREATE TABLE actlink_set_char
(actlinkId int not null,
 actionIndex int not null,
 fieldId int not null,
 charMenu varchar(254) null,
 propShort varchar(255) null,
 propLong text null,
CREATE CLUSTERED INDEX actlink_schar_ind 
ON actlink_set_char (actlinkId)

CREATE TABLE actlink_dde 
(actlinkId int not null,
actionIndex int not null,
serviceName varchar(64) not null,
topic varchar(64) not null,
action int not null,
path varchar(255) not null,
command varchar(255) not null,
itext text null)

CREATE CLUSTERED INDEX actlink_dde_ind 
ON actlink_dde (actlinkId)

CREATE TABLE actlink_auto 
(actlinkId int not null,
actionIndex int not null,
autoServerName varchar(255) not null,
classId varchar(128) not null,
isVisible char not null,
actionShort varchar(255) null,
actionLong text null,
COMShort varchar(255) null,
COMLong text null)

CREATE CLUSTERED INDEX actlink_auto_ind 
ON actlink_auto (actlinkId)

CREATE TABLE actlink_push 
(actlinkId int not null,
actionIndex int not null,
fieldId int not null,
assignShort varchar(255) null,
assignLong text null,
sampleSchema varchar(255) null,
sampleServer varchar(64) null)

CREATE CLUSTERED INDEX actlink_push_ind 
ON actlink_push (actlinkId)

CREATE TABLE actlink_sql 
(actlinkId int not null,
actionIndex int not null,
assignShort varchar(255) null,
assignLong text null,
keywordList text null,
parameterList text null)

CREATE CLUSTERED INDEX actlink_sql Ind 
ON actlink_sql (actlinkId)
CREATE CLUSTERED INDEX actlink_sql_ind ON actlink_sql (actlinkId)
go

CREATE TABLE actlink_open
  (actlinkId int not null,
   actionIndex int not null,
   serverName varchar(64) not null,
   schemaName varchar(254) not null,
   vuiLabel varchar(254) null,
   closeBox char null,
   assignShort varchar(254) null,
   assignLong text null,
   windowMode int null,
   noMatchCntnu char null,
   pollIntval int null,
   sortlist varchar(255) null,
   queryshort varchar(255) null,
   querylong text null,
   msgType int null,
   msgNum int null,
   msgText text null,
   msgPane char null,
   reportstr text null,
   supresEptyList char null,
   targetLocation varchar(255) null)
go

CREATE CLUSTERED INDEX actlink_open_ind ON actlink_open (actlinkId)
go

CREATE TABLE actlink_commit
  (actlinkId int not null,
   actionIndex int not null)
go

CREATE CLUSTERED INDEX actlink_commit_ind ON actlink_commit (actlinkId)
go

CREATE TABLE actlink_close
  (actlinkId int not null,
   actionIndex int not null,
   closeAll char null)
go

CREATE CLUSTERED INDEX actlink_close_ind ON actlink_close (actlinkId)
go

CREATE TABLE actlink_call
  (actlinkId int not null,
   actionIndex int not null,
   serverName varchar(64) not null,
   guideName varchar(254) not null,
   guideMode int not null,
CREATE TABLE actlink_call
(
guideTableId int null,
assignShort varchar(255) null,
assignLong text null,
sampleServer varchar(64) null,
sampleGuide varchar(254) null
)
go
CREATE CLUSTERED INDEX actlink_call_ind
ON actlink_call [actlinkid]
go
CREATE TABLE actlink_exit
(
actlinkId int not null,
actionIndex int not null,
closeAll char null
)
go
CREATE CLUSTERED INDEX actlink_exit_ind
ON actlink_exit [actlinkid]
go
CREATE TABLE actlink_goto
(
actlinkId int not null,
actionIndex int not null,
label varchar(128) not null
)
go
CREATE CLUSTERED INDEX actlink_goto_ind
ON actlink_goto [actlinkid]
go
CREATE TABLE actlink_wait
(
actlinkId int not null,
actionIndex int not null,
buttonTitle varchar(64) default 'Continue' null
)
go
CREATE CLUSTERED INDEX actlink_wait_ind
ON actlink_wait [actlinkid]
go
CREATE TABLE actlink_gotoaction
(
actlinkId int not null,
actionIndex int not null,
tag int not null,
fieldIdOrValue int default 0 null
)
go
CREATE CLUSTERED INDEX actlink_gotoa_ind
ON actlink_gotoaction [actlinkid]
go
CREATE TABLE actlink_mapping
(
schemaId int not null,
objIndex int not null,
actlinkId int not null
)
go
CREATE UNIQUE INDEX actlink_mapping_ind
ON actlink_mapping (schemaId, actlinkid)
CREATE TABLE alert_user
    (username      varchar(254) not null,
     clientIPAddr varchar(16)  not null,
     actualIPAddr varchar(16)  not null,
     serverIPAddr varchar(16)  not null,
     clientPort   int          not null,
     regFlags     int          not null,
     clientVersion int         not null,
     regTime      int          not null,
     clientCodeSet int         not null)

CREATE UNIQUE INDEX alert_user_ind
    ON alert_user (username, clientIPAddr, clientPort)

CREATE TABLE alert_time
    (username      varchar(254)  not null,
     checkpointTime int         not null)

CREATE UNIQUE INDEX alert_time_ind
    ON alert_time (username)

CREATE TABLE support_file
    (fileType    int          not null,
     id          int          not null,
     id2         int          not null,
     fileId      int          not null,
     timestamp   int          not null,
     fileContent image            null)

CREATE UNIQUE CLUSTERED INDEX support_file_ind
    ON support_file (fileType, id, id2, fileId)

CREATE TABLE servgrp_config
    (name          varchar(64)      null,
     checkInterval int          not null)

CREATE TABLE servgrp_op_mstr
    (operation      varchar(255) not null,
     opNum          int          not null,
     configLabel    varchar(255)     null,
     configCommand  varchar(50)      null,
     categoryStrs   varchar(255)     null)

CREATE TABLE ft_pending
    (serverName    varchar(64)  not null,
     schemaId      int          not null,
     fieldId       int          not null,
     fileId        int          not null,
     fieldId2      int          not null,
     fileId3       int          not null,
     fileContent   image            null)
CREATE CLUSTERED INDEX ft_pending_ind
ON ft_pending (seqNum)

go
Appendix A

Database user names, passwords, and dates

The procedures in this appendix help improve the performance or enhance the security of your AR System environment.

The following topics are provided:

- Changing the AR System database user name and password (page 130)
- Converting AR System dates to database dates (page 131)

Note: These procedures address the most commonly requested AR System technical information. For access to the complete set of AR System technical information and procedures, visit the Customer Support website at http://www.remedy.com.
Changing the AR System database user name and password

The AR System database user (ARAdmin by default) and password (ARAdmin by default) are set during AR System server installation. You can, however, change them to suit your needs.

► To change the AR System database user name

1. Stop the AR System server.
2. Update the database user name in the database. See your database documentation for more information.
3. Update the Db-user option in the ar.conf (ar.cfg) file.
   a. Open the ar.conf (ar.cfg) file with a text editor.
   b. Change the Db-user entry to match the value you specified in step 2.
   c. Save the ar.conf (ar.cfg) file.
4. Restart the AR System server.

► To change the AR System database password

Use the BMC Remedy Administrator Server Information dialog box, or use the ARSetServerInfo API call. See Configuring AR System for more information.

**WARNING:** Do not change this password directly in the database.

**Note:** See “Using IBM DB2 Universal Database with AR System” on page 13 for special considerations for database user name and password with DB2.
Converting AR System dates to database dates

AR System keeps track of the date and time to run escalations, stamps requests with the date and time they were submitted, and informs you when alerts were sent. To track the date and time, AR System uses a format that measures the number of seconds from January 1, 1970, 12:00 a.m. Greenwich Mean Time (GMT). While accurate, this format can be an awkward format to read. You might want to translate it to a format that your database can easily read.

Each database requires different commands for the date and time conversion. The following procedures describe how you can use your database to convert the AR System date and time format.

**Note:** In the SQL commands in the following procedures, the column number is referenced by `<column_number>`. Alternatively, you can provide the SQL view name of the column (the database name of the field as displayed in BMC Remedy Administrator).

---

**To convert the date and time format for a DB2 Universal database**

- See your DB2 documentation for information about dateline arithmetic.

**To convert the date and time format for an Informix database**

1. Using any front-end tool that allows direct access to an Informix-SQL database, log in as the root user.

2. Type the following command:

   ```sql
   % select (extend((extend(datetime(1970-1-1) year to day, year to hour) - interval(<offset_hours> hour to hour), year to second) + C<column_number> units second) from T<table_number>
   
   where <column_number> is the number of the column for the date and time field, <table_number> is the number of the form table, and <offset_hours> is a positive or negative number representing the number of hours later or earlier than GMT.
   ```
If the date is greater than 09/10/2001, you will receive an error. To avoid an error, you can display minutes instead of seconds by using the following command:

```sql
% select (extend((extend(datetime(1970-1-1) year to day, year to hour) - interval(<offset_hours> hour to hour), year to minute) +(<column_number>/60) units minute) from T<table_number>
```

See the Informix Guide to SQL: Reference and Syntax manual for information about the `datetime`, `extend`, and `interval` functions.

▶ **To convert the date and time format for an Oracle database**

1. Using any front-end tool that enables direct access to an Oracle SQL database, log in as a user with write access to the AR System tables.

2. Type the following command:

```sql
% SELECT TO_CHAR(TO_DATE('01/01/1970 00:00:00', 'MM/DD/YYYY HH24:MI:SS') + ((<column_number> + <offset>)/(60*60*24)), 'MM/DD/YYYY HH24:MI:SS') FROM T<table_number>
```

   where `<column_number>` is the number of the column for the date and time field, `<table_number>` is the number of the form table, and `<offset>` is a positive or negative number representing the number of seconds later or earlier than GMT. See your Oracle documentation for information about the `TO_DATE` and `TO_CHAR` functions.

▶ **To convert the date and time format for a Sybase or Microsoft SQL Server database**

1. Using any front-end tool that enables direct access to a Sybase or Microsoft SQL Server database, log in as a user who has write access to the AR System tables.

2. Type the following command:

```sql
% select dateadd(second, C<column_number> + <offset>, 'Jan 1, 1970') from T<table_number>
```

   where `<column_number>` is the number of the column for the date and time field, `<table_number>` is the number of the form table, and `<offset>` is a positive or negative number representing the number of seconds later or earlier than GMT.

3. Optionally, you could format the date field by using the `convert` function. There are 12 different formats from which you can choose. See your Sybase documentation.
Index

A
active links table
described 30
illustrated 26
AR System data dictionary 24–32
AR System database user and password, changing 130
attachment tables
data 34, 36
details 34, 36

C
case sensitivity in databases
Microsoft SQL 17
Sybase 18
changing database user and password 130
character fields in databases
Informix 16
Microsoft SQL 17
CLOB storage 24
connections, Informix databases 16
containers table
illustrated 27
containers table, described 31
control table, described 27
converting dates 131
currency table, described 35

D
data dictionary, AR System 24–32
data types

DB2 20
Informix 21
Microsoft SQL 22
Oracle 23
Sybase 24
database connections, Informix databases 16
database user and password, changing 130
databases
attachment tables 34
currency table 35
date converting 131
field length 40
fields 39
forms 38
IBM DB2 SQL commands 50–70
indexing 36
Informix maximum connections 16
Informix SQL commands 71–88
installing 12
main data table 32
Microsoft SQL commands 106–127
Oracle 89–106
SQL views 37
status history table 33, 36
structure 12
Sybase SQL commands 106–127
time, converting 131
Unicode support 44–48
Unicode, creating 45
Unicode, migrating 46
view user interface (VUI) 28

Index ▲ 133
date/time format, converting 131
definitions, tables 24
diary fields in databases
  Informix 16
  Microsoft SQL 17
  Sybase 18
Direct SQL, external Informix databases 17
E
escalations table
  described 30
  illustrated 26
external Informix databases 17
F
field limits in databases
  DB2 14
  Informix 16
  Microsoft SQL 17
  Sybase 18
fields
  adding to forms 39
  changing length 40
  deleting from database 39
  join form views 29
  table 25
fields table, described 29
filters table
  described 30
  illustrated 26
forms
  database tables 32
  databases 38
  performance tuning 39
forms table
  described 28
  illustrated 25
I
IBM DB2 database
  data types 20
  date/time format 131
  field length 40
  field size limit 14
  SQL commands 50–70
IBM DB2 database (continued)
  user name and password 13
  using with AR System 13–15
indexing, tables 36
Informix database
  character string and diary field limits 16
  data types 21
  date/time format 131
  Direct SQL 17
  field length 40
  maximum database connections 16
  modulo operator 16
  shared libraries 17
  SQL commands 71–88
  using with AR System 16–17
  wildcards 16
J
join forms
  main data view 32
  Sybase databases and 19
L
libraries, shared 17
M
main data table
  index 36
  join form 32
menus table
  described 29
  illustrated 26
Microsoft SQL Server Database
  case sensitivity 17
  character diary field limits 17
  character string 17
  data types 22
  field length 41
Microsoft SQL Server database
  date/time format 132
  SQL commands 106–127
  using with AR System 17
Microsoft SQL Server. See Microsoft SQL Server Database
modulo operator, Informix database 16

O
Oracle database
CLOB storage 24
data types 23
date/time format 132
field length 42
searches and 18
SQL commands 89–106
using with AR System 18

P
password, changing 130
performance tuning, forms 39
primary key 36

S
searches in databases
Informix 16
Microsoft SQL 17
Oracle 18
Sybase 18
shared libraries 17
SQL commands
DB2 50–70
Informix 71–88
Microsoft SQL Server 106–127
Oracle 89–106
Sybase 106–127
SQL, views 37
status history table 33, 36
Sybase database
case sensitivity 18
character sets 19
character string 18
data types 24
date/time format 132
diary field limits 18
field length 42
joins 19
SQL commands 106–127
Sybase database (continued)
SQL statement length limit 19
using with AR System 18–19

T
table types, database
active links 30
attachment details 34
containers 31
currency 35
escalations 30
fields 29
filters 30
forms 28, 32
main 32
menus 29
status history 33
workflow mapping 31

U
Unicode
compliance 44
database, creating 45
database, requirements for installation 45
database, support 44–48
Informix considerations 48
Microsoft SQL considerations 47
migrating existing AR database 46
user name and password, DB2 database 13

V
view user interface (VUI) 28
views, SQL 37
VUI (view user interface) 28

W
wildcards in Informix 16
workflow mapping table
described 31
illustrated 26