Table of Contents

Preface ................................................................. 7
  Audience ......................................................... 7
  Overview of This Manual ....................................... 7
  Action Request System Documents .......................... 8

Chapter 1 Using Relational Databases with AR System .......... 11
  The Database Structure and AR System ..................... 12
  The AR System Database ...................................... 12
  Using Relational Databases with AR System ............... 13
    Using IBM DB2 Universal Database with AR System ...... 14
    Using Informix with AR System ............................ 16
    Using Microsoft SQL Server with AR System ............. 17
    Using Oracle with AR System ............................. 18
    Using Sybase with AR System ............................. 19
  The AR System Data Dictionary .............................. 20
    Initial Table .................................................. 24
    Tables for Forms ............................................. 24
    Tables for Fields ............................................ 25
    Tables for Menus ............................................. 25
    Tables for Filters ........................................... 26
    Tables for Escalations ..................................... 26
    Tables for Active Links ................................... 26
    Tables for Mapping Workflow .............................. 27
    Tables for Containers ...................................... 27
Creating Tables for Forms ........................................ 28
   DB2 Data Types ............................................... 28
   Informix Data Types ........................................... 29
   Microsoft SQL Data Types ................................. 29
   Oracle Data Types ............................................. 30
   Sybase Data Types ............................................ 31
   Main Data Table .............................................. 31
   Status History Table ........................................ 32
   Attachment Tables ........................................... 33
   Currency Table ................................................ 34
   Indexing .......................................................... 34
   SQL Views ........................................................ 36
Updating Tables When AR System Forms Change .............. 37
   Adding Fields ................................................... 37
   Deleting Fields ................................................ 38
   Changing Character Field Lengths ....................... 39
Defining Rollback Segments in an Oracle Database ............ 42
   Initial Rollback Segment .................................. 43
   Adding the AR System Rollback Segment ................. 43
Working with Earlier AR System Versions That Use Oracle .... 44
Related Information ............................................. 45
   DB2 .............................................................. 45
   Informix .......................................................... 45
   Oracle .............................................................. 45
   Sybase ............................................................ 45
   MS SQL .......................................................... 45
Chapter 2  Technical Essays ........................................ 47
Converting AR System Dates to Database Dates ............... 48
Changing the Next Available ID for New Requests ............ 50
   The Request ID Field ....................................... 50
   Changing the Next Available ID for SQL Databases ...... 51
Changing the Request ID Field Length or Prefix .............. 53
   The Request ID Field ....................................... 53
   Preserving Existing Request ID Field Values ............. 54
   Changing Existing Request ID Field Values to the New Format .... 55
<table>
<thead>
<tr>
<th>Chapter 3</th>
<th>SQL Definitions of the Data Dictionary Tables</th>
<th>67</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DB2 Universal</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Informix</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td>Sybase and Microsoft SQL</td>
<td>140</td>
</tr>
</tbody>
</table>

| Index     |                                              | 169|
Preface

Audience

This guide is written for database administrators who are responsible for overseeing the interaction between the 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 manual assumes knowledge of database administration and familiarity with current Microsoft Windows platforms.

Note: You should be familiar with AR System Administrator before you begin.

Overview of This Manual

- Chapter 1, Using Relational Databases with AR System
- Chapter 2, Technical Essays
- Chapter 3, SQL Definitions of the Data Dictionary Tables
# Action Request System Documents

<table>
<thead>
<tr>
<th>Title and Part Number</th>
<th>Description</th>
<th>Audience</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AR System Concepts Guide</strong>&lt;br&gt;AR-510-CG-01</td>
<td>Overview of AR System 5.1 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>
<td>Print and PDF</td>
</tr>
<tr>
<td><strong>Developing AR System Applications: Basic</strong>&lt;br&gt;AR-510-DABG-01</td>
<td>Basic procedures for creating and modifying an AR System 5.1 application for tracking data and processes.</td>
<td>Administrators</td>
<td>Print and PDF</td>
</tr>
<tr>
<td><strong>Developing AR System Applications: Advanced</strong>&lt;br&gt;AR-510-DAAG-01</td>
<td>Advanced procedures for extending and customizing AR System 5.1 applications.</td>
<td>Administrators</td>
<td>Print and PDF</td>
</tr>
<tr>
<td><strong>Configuring AR System</strong>&lt;br&gt;AR-510-CFG-01</td>
<td>Server administration topics on configuring servers and the mid tier, and maintaining AR System 5.1.</td>
<td>Administrators</td>
<td>Print and PDF</td>
</tr>
<tr>
<td><strong>Optimizing and Troubleshooting AR System</strong>&lt;br&gt;AR-510-OTG-01</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>
<td>Print and PDF</td>
</tr>
<tr>
<td><strong>AR System Database Reference Guide</strong>&lt;br&gt;AR-510-DRG-01</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>
<td>Print and PDF</td>
</tr>
<tr>
<td><strong>AR System Distributed Server Option Administrator’s Guide</strong>&lt;br&gt;AR-510-DSOG-01</td>
<td>Server administration and procedures for implementing a distributed AR System 5.1 server environment with the Distributed Server Option.</td>
<td>Administrators</td>
<td>Print by special order and PDF</td>
</tr>
<tr>
<td><strong>AR System C API Reference Guide</strong>&lt;br&gt;AR-510-CAPI-01</td>
<td>Information about AR System 5.1 data structures, API function calls, and OLE support.</td>
<td>Administrators and Programmers</td>
<td>Print by special order and PDF</td>
</tr>
<tr>
<td><strong>AR System Java API</strong></td>
<td>Information about Java classes, methods, and variables that integrate with AR System.</td>
<td>Administrators and Programmers</td>
<td>HTML</td>
</tr>
<tr>
<td><strong>Installing AR System</strong>&lt;br&gt;AR-510-IG-01</td>
<td>Procedures for installing and licensing AR System 5.1.</td>
<td>Administrators</td>
<td>Print and PDF</td>
</tr>
<tr>
<td>Title and Part Number</td>
<td>Description</td>
<td>Audience</td>
<td>Format</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td><strong>AR System Email Engine Guide</strong>&lt;br&gt;AR-510-EEG-01</td>
<td>Procedures for installing, configuring, and using the AR System email engine.</td>
<td>Administrators</td>
<td>Print by special order and PDF</td>
</tr>
<tr>
<td><strong>AR System Error Messages Guide</strong>&lt;br&gt;AR-510-EMG-01</td>
<td>List and expanded descriptions of AR System 5.1 error messages.</td>
<td>Administrators and Programmers</td>
<td>Print by special order and PDF</td>
</tr>
<tr>
<td><strong>AR System Master Index</strong>&lt;br&gt;AR-510-MI-01</td>
<td>Combined index of all books.</td>
<td>Everyone</td>
<td>Print only</td>
</tr>
<tr>
<td><strong>AR System Release Notes</strong>&lt;br&gt;AR-510-RN-01</td>
<td>New features list, compatibility lists, international issues, open and fixed issues.</td>
<td>Everyone</td>
<td>Print and PDF</td>
</tr>
<tr>
<td><strong>AR System Windows User Tool Help</strong>&lt;br&gt;Help</td>
<td>Procedures for using AR System Windows User Tool.</td>
<td>Everyone</td>
<td>Help menu</td>
</tr>
<tr>
<td><strong>AR System Import Help</strong>&lt;br&gt;Help</td>
<td>Procedures for using AR System Import.</td>
<td>Administrators</td>
<td>Help menu</td>
</tr>
<tr>
<td><strong>AR System Administrator Help</strong>&lt;br&gt;Help</td>
<td>Procedures for creating and modifying an AR System 5.1 application for tracking data and processes.</td>
<td>Administrators</td>
<td>Help menu</td>
</tr>
<tr>
<td><strong>AR System Alert Help</strong>&lt;br&gt;Help</td>
<td>Procedures for using AR System Alert.</td>
<td>Everyone</td>
<td>Help Menu</td>
</tr>
</tbody>
</table>

Unless otherwise noted, online documentation is available in Adobe Acrobat (PDF) format on AR System product installation CDs and/or on the Customer Support web site.
This chapter details how AR System 5.1 interacts with the following database systems:

- DB2 Universal
- Informix
- Oracle
- Sybase
- Microsoft SQL Server

Topics include:

- Administration tasks performed at installation, with an overview of the data dictionary tables created and used by AR System
- Special considerations for each database, such as limits on queries
- Rules used to create and modify tables and columns in the system as the AR System administrator defines and changes the structure of forms

**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. For Oracle upgrades, refer to *Working with Earlier AR System Versions That Use Oracle* on page 44.
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.

Remedy 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.

Remedy reserves the right to change the structure of the 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 does 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 Developing AR System Applications: Basic guide for more information about this functionality of the system.

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 name and user for ARSystem and ARAdmin.

The characteristics of the AR System database vary depending on the type of relational database. Refer to the Installing AR System guide for information on the AR System database.
Using Relational Databases with AR System

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.

The AR System database user (ARAdmin by default) and password (ARAdmin# by default) are set during AR System server installation. To change the ARAdmin-database user name, stop the AR System server, update the database user name in the database, update the Db-user option in the ar.conf (ar.cfg) file, and restart the server. To change the password, use the AR System Administrator Server Information dialog box, or use the ARSetServerInfo API call. Do not change this password directly in the database.

Note: See Using IBM DB2 Universal Database with AR System on page 14 for special considerations for database user name and password with DB2.

After this AR System database is created, AR System creates a series of tables that form its data dictionary. Refer to The AR System Data Dictionary on page 20 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, refer to the Installing AR System guide.

For information about configuration options and parameters associated with specific databases, refer to the ar.conf or ar.cfg file documentation in the Configuring AR System guide.
Using IBM DB2 Universal Database with AR System

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

User Name and Password

- When DB2 database resides on the same machine as AR System server, the 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 DB2 database resides on a different machine than 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. The password must be AR#Admin#. Once AR System server is installed, you can change the password.
- 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 AR System Administrator.

Form and Field Limits

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

Creating a Tablespace with a Larger Page Size for a Particular Form

1. Stop the AR System server.
2. Create a tablespace with 32K page size. (You may want to name the tablespace something like TBS32K.)
3. Start the AR System server.
4. In AR System Administrator, open the Server Information dialog box.
5. From the Database tab, add the following options to the database configuration file.
   - Form: <form_name>
   - Clause: IN TBS32K
This will cause the table for the form \(<form\_name>\) to be created in the tablespace of 32K.

You can also specify the clause as follows:

**Form:**
Clause: IN TBS32K

This will cause the table for *all* the forms to be created in the tablespace of 32K.

6 Click OK to save this server information.

7 Create the form.

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

There are limits that 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 will return an error message because DB2 does not support using a field ID on the right of the LIKE predicate.

"Demo" LIKE 'Submitter'

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

Informix behaviors that you should 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 the percent symbol (%) or the underscore (_) characters, include a backslash (\) (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: `$INFORMIXDIR/lib:$INFORMIXDIR/lib/esql:$SHLIB_PATH`
- Solaris: `$INFORMIXDIR/lib:$INFORMIXDIR/lib/esql:$LD_LIBRARY_PATH`
- AIX: `$INFORMIXDIR/lib:$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 by default with log options and non-ANSI options.

Using Microsoft SQL Server with AR System
Microsoft SQL Server behaviors that you should 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, refer to your Microsoft SQL Server documentation.

International Character Sets
If you are using AR System 5.x with a Microsoft SQL server, when you submit a ticket with uppercase accented European characters (diacritical marks), for example Ü or Ä, the symbols become corrupted in the database. You will not be able to search for the original letters.
If you already have such letters in the database (for example, data from AR System 4.0.3) and try to read the database, you will see corrupted characters in AR System Windows User Tool 5.x.

**SQL OLEDB Provider**

If you have an SQL 7.0 provider (C:\Program Files\Common Files\System\Ole DB\sqloledb.dll) installed on the machine being used as your AR System server and you are not using Clustered SQL Server Module, install Microsoft Data Access Component 2.6 SP1 to improve the performance of the AR System server. (Microsoft Data Access Component 2.6 SP1 can be downloaded from http://www.microsoft.com/data/download_26sp1.htm.)

**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, refer to the Configuring AR System 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

Sybase behaviors that you should consider are described in the following sections.

**AIX Requirements**
If you are using Sybase 12.0, you must do the following to use AIX with a 5.x version of AR System server:

1. Install Sybase 12.5 client libraries.
2. Set the following environment variables to the 12.5 client path:
   ```
   SYBASE_OCS=<path to the 12.5 libraries>/OCS-12.5
   ```
3. To verify that the setup was successful, try to connect 12.5 ISQL to the 12.0 server.

**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 will be 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, refer to 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 may fail, and 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.

- Trailing spaces that you add to names, menu labels, and field labels in AR System Administrator are removed by the database.

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 21, Figure 1-2 on page 22, and Figure 1-3 on page 23).

All of these tables together contain the complete definition of the structures and workflow defined in your implementation of AR System. As you add new or alter existing structures in your system, appropriate updates are made to these tables to reflect the changes.
Figure 1-1: AR System Data Dictionary: Forms, Fields, VUIs, and Sample Forms
Figure 1-2: AR System Data Dictionary: Active Links, Filters, and Escalations
Figure 1-3: AR System Data Dictionary: Container
Initial Table

The first table is named control, and it contains one row. The columns contain information about the version of the database, \texttt{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 \texttt{schema} in the database tables). The \texttt{arschema} table contains information about the form definitions. The four main fields for this table are:

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

Another set of tables holds information associated with the form definition:

- \texttt{schema\_group\_ids}—Defines which groups have access to the form.
- \texttt{subadmin\_group}—Defines which groups have subadministrator access potential for this form.
- \texttt{schema\_list\_fields}—Defines which fields are returned in response to the \texttt{ARGetListEntry} and \texttt{ARGetListEntryWithFields} API calls.
- \texttt{schema\_vendor}—Defines how the form is attached to an ARDBC data source.
- \texttt{schema\_view}—Defines how the form is attached to a database table.
- \texttt{schema\_join}—Defines how the form is joined, if applicable.
- \texttt{schema\_index}—Defines the indexes for the form.
- \texttt{schema\_sort}—Defines the default sort order for this 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 \texttt{vui} table contains information about each VUI in each form. Every VUI is identified by the combination of the \texttt{schemald} that connects the VUI to a form, and the \texttt{vuid} that identifies that VUI within the form.
Tables for Fields

The field table contains 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 field table contains all the information for the field, except for the display information.

The vuid 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 will be displayed in the form. The field_permissions table contains information about the permissions of various groups to the individual fields. A 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 comprise 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 comprise 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 comprise 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 that can be defined: char_menu_list, char_menu_query, char_menu_file, char_menu_sql, and 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 that are 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, and 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 of 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 a containerId. Containers are used to define guides, applications, packing lists, workspaces, and web services. The three main fields for the table are:

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

Another 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 comprise 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 *Developing AR System Applications: Advanced* guide. For more information on the data structures used to define containers, see the *AR System C API Reference Guide*.

### Creating Tables for Forms

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

- The main data table holds all of the information for the form, with a column for each field except Attachments and Status History. For more information, refer to *Main Data Table* on page 31.
- The status history table contains all the information for the Status History field. For more information, refer to *Status History Table* on page 32.
- 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. For more information, refer to *Attachment Tables* on page 33.

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 the following data types:

- Integer, selection, and timestamp fields use `int`.
- Real fields use `float`. 

28  Chapter 1—Using Relational Databases with AR System
Decimal fields use **decimal**.

Character fields, with a defined maximum that is 255 bytes or less, use **varchar**.

Character fields, with a defined maximum from 256 bytes up to 32700 bytes, use **longvarchar**.

Diary fields and character fields, with no maximum or a maximum over 32700 bytes, use **clob** and hold up to 1 MB.

Attachment fields use **blob** and hold up to 1 GB.

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: **int**, **float**, **varchar**, and **byte**. AR System fields use the following data types:

- Integer, selection, and time stamp fields use **int**.
- Real fields use **float**.
- Character fields, with a defined maximum that is 255 bytes or less, use **varchar**.
- Diary fields and character fields, with no maximum or a maximum over 255 bytes, use **byte**.
- Attachment fields use **byte** and hold up to 2 GB.
- 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 the following data types:

- Integer, selection, and time stamp fields use **int**.
- Real fields use **float**.
- Character fields, with a defined maximum that is 8000 bytes or less, use **varchar**.
Diary fields and character fields, with no maximum or a maximum over 8000 bytes, use text.

Attachment fields use image and hold up to 2 GB.

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, long raw, and clob. AR System fields use these data types as follows:

- Integer, selection, and time stamp fields use number (15, 0).
- Real fields use float.
- Character fields, with a defined maximum that is 4000 bytes or less, use varchar.
- Diary fields and character fields, with a defined maximum that is more than 4000 bytes (or an unlimited length), use clob.

**Note:** If you are upgrading from a version of AR System prior to 4.5, and you have character and diary fields over 255 bytes that use the long data type, these fields will continue to use the long data type. After upgrading to AR System 5.x, new character fields 4000 bytes or less will use varchar, and diary fields will use clob. For more information, refer to Working with Earlier AR System Versions That Use Oracle on page 44.

Attachment fields use long raw and hold up to 2 GB. (For the Oracle RDBMS, the default maximum attachment size is 1 MB.) You can attach larger files by putting the Db-Max-Attach-Size configuration parameter in your ar.conf (ar.cfg) file. For more information about the ar.conf or ar.cfg file, refer to the Configuring AR System guide.

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.
Sybase Data Types

AR System uses five different Sybase data types for its columns: int, float, varchar, text, and image. AR System fields use the following data types:

- Integer, selection, and time stamp fields use int.
- Real fields use float.
- Character fields, with a defined maximum that is 255 bytes or less, use varchar.
- Diary fields and character fields, with no maximum or a maximum over 255 bytes, use text.
- Attachment fields use image and hold up to 2 GB.
- 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.

Main Data Table

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 21, 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, although it can be assigned by the creator of the field. Every ID is unique within a form, so there is never an issue with duplicate names. When assigned, the ID for a field cannot be changed, regardless of any changes to the field. For information on reserved and core IDs, refer to the Developing AR System Applications: Basic 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.
Because the 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. The 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 computed by concatenating the entry IDs of the regular forms (in the E columns) separated by vertical bars (|).

**Status History Table**

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 21, 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. Accordingly, 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 may 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, this table does not exist. If one 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, refer to the Developing AR System Applications: Basic guide.

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

### Attachment Tables

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 21, 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.

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 schemaID, and 536870920 is the attachment field ID. In Figure 1-1 on page 21, the attachment data table is labeled BnC<flID>.

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

The currency field has additional columns in the main data table and therefore a unique naming convention to distinguish the extra columns. Whereas typical fields follow the naming convention described in the “Main Data Table” section (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 below.

<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>USD, EUR, JPY, etc.</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 for 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. The system does not create additional indexes.

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

The indexing of a currency field has special considerations. Because a currency field is represented by multiple columns in the main data table, multiple columns will be indexed. Standard queries against a currency field will potentially use any of several different columns, depending on the currency type specified. In order to provide comprehensive coverage, indexing a currency field defines 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, carefully consider indexing a currency field 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, refer to the *AR System 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 (the field name of the field, 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. However, 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 either 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 each of these areas.

The discussion in 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.

**Note:** 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 data access to the system is not critical.

**Adding Fields**

When you add a new field, 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 28.

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 Modify All operation of AR System Windows User Tool to assign a default value for the field.
Deleting Fields

Deleting a field 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 a character’s field length.

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

**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 $\leq 255$ bytes, use the `Alter Table` command to change the columns. Do not recreate the table or the index.
- For any other change in length, create a new column with the new length restriction. Then copy all data from the original column to the new column, and delete the original column 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 less than or equal to 255 bytes and you decrease the length, no change is made to the table.
- If the original size is less than or equal to 255 bytes, you increase the length. If the new size is also less than or equal to 255 bytes, issue the `ALTER TABLE ... MODIFY...` command.
- If the original size is less than or equal to 255 bytes and the new size is greater than 255 bytes, create a new column with the new length restriction. Then copy all data from the original column to the new column and delete the original column. The data type of the column is changed from `varchar` to `byte`. 
If the original size is greater than 255 bytes and the new size is less than or equal to 255 bytes, create a new column with the new length restriction. Then copy all data from the original column to the new column and delete the original column. The data type of the column is changed from byte to varchar.

If the original size is greater than 255 bytes and the new size is also greater than 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, create a new column with the new length restriction. Then copy all data from the original column to the new column, and delete the original column from the main data table.

If the field is created in a previous version of AR System, 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, create a new column with the new length restriction. Then copy all data from the original column to the new column, and delete the original column 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 field length changes are handled differently, depending on the initial size of the field, and whether the field was created in the current, or in a previous version of AR System.

Table 1-1: Changing Character Field Lengths (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Administrator Action</th>
<th>AR System Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Decreases</strong> the length of a field that was created in the current version from &gt; 4000 bytes to &lt;= 4000 bytes.</td>
<td>Adds a new <code>varchar</code> column to the main data table; copies the data from the <code>clob</code> column to the new column; deletes the old column.</td>
</tr>
<tr>
<td><strong>Decreases</strong> the length of a field that was created in the current version from &lt;= 4000 bytes to &lt; 4000 bytes.</td>
<td>No restructuring performed.</td>
</tr>
<tr>
<td><strong>Decreases</strong> the length of a field that was created in a version prior to 4.5 from &gt;= 255 bytes to &lt;= 4000 bytes.</td>
<td>Adds a new <code>varchar</code> column to the main data table; copies the data from the <code>long</code> field to the new column; deletes the old <code>long</code> table.</td>
</tr>
<tr>
<td><strong>Decreases</strong> the length of a field that was created in a version prior to 4.5 from &lt;= 255 bytes to &lt; 255 bytes.</td>
<td>No restructuring performed.</td>
</tr>
<tr>
<td><strong>Increases</strong> the length of a field that was created in the current version from &lt;= 4000 bytes to &gt; 4000 bytes.</td>
<td>Adds a new <code>clob</code> column to the main data table; copies the data from the <code>varchar</code> column to the new column; deletes the old column.</td>
</tr>
<tr>
<td><strong>Increases</strong> the length of a field that was created in the version prior to 4.5 from &gt;= 255 bytes to &gt; 4000 bytes.</td>
<td>No restructuring performed.</td>
</tr>
<tr>
<td><strong>Increases</strong> the length of a field that was created in the version prior to 4.5 from &lt;= 255 bytes to &gt; 4000 bytes.</td>
<td>Adds a new <code>clob</code> column to the main data table; copies the data from the <code>varchar</code> column to the new column; deletes the old column.</td>
</tr>
<tr>
<td><strong>Increases</strong> the length of a field that was created in a version prior to 4.5 from &gt; 255 bytes to &gt; 4000 bytes.</td>
<td>Adds a new <code>clob</code> column to the main data table; copies the data from the <code>long</code> field to the new column; deletes the old <code>long</code> table.</td>
</tr>
</tbody>
</table>
Table 1-1: Changing Character Field Lengths (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Administrator Action</th>
<th>AR System Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increases the length of a field that was created in a version prior to 4.5 from (\leq 255) bytes to (\leq 4000) bytes.</td>
<td>No restructuring performed.</td>
</tr>
<tr>
<td>Increases the length of a field that was created in a version prior to 4.5 from (&gt; 255) bytes but (\leq 4000) bytes to (\leq 4000) bytes.</td>
<td>Adds a new <code>varchar</code> column to the main data table; copies the data from the <code>long</code> field to the new column; deletes the old <code>long</code> table.</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 \(\leq 255\) 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, create a new column with the new length restriction. Then copy all data from the original column to the new column, and delete the original column from the main data table.

**Defining Rollback Segments in an Oracle Database**

This discussion provides a basic introduction to rollback segments, which enable configuration of AR System within the Oracle 8i compatible database. If you are using Oracle 9i in 9i compatible mode and the Undo Management is set to AUTO, the Undo Tablespace will be used, in which case you can skip this section on Oracle rollback segments.

For a complete discussion of rollback segments, refer to the *Oracle RDBMS Database Administrator’s Guide*.

If you do not supply an alternate rollback segment during installation, AR System creates a rollback segment for the `ARSystem` tablespace named `ARSystemRoll`. This rollback segment is available to `ARSystem` to provide transaction logging when necessary. For a rollback segment to be created for a specific tablespace, a general rollback segment must be created and registered for the system.
Initial Rollback Segment

Rollback segments must be created in the system and then registered with the Oracle database in the startup file. To see if there is an existing rollback segment registered, check the system startup file for the Oracle database instance you are running. The file is named init<SID>.ora, where <SID> is the Oracle database system ID, which is located in the dbs directory under the Oracle database home directory (UNIX), or in the database directory under the Oracle database directory (Windows).

In this file, rollback segments are registered with a line starting with rollback_segments. If you have rollback segments r1 and r2, the line would look like:

```
rollback_segments=(r1, r2)
```

If you include this line in the system startup file, when the database is started, these two rollback segments will be online and available for transactions.

To add a rollback segment, refer to the next section.

Adding an Additional Rollback Segment

1 Run the sqlplus program as the system user and issue the following command:

   CREATE ROLLBACK SEGMENT r1;
   ALTER ROLLBACK SEGMENT r1 ONLINE;

   This creates the rollback segment and makes it available.

2 To make the rollback segment available to the database when you stop and start it, add the following line to the end of the system startup file:

   rollback_segments=(r1)

3 Save your changes.

Adding the AR System Rollback Segment

During installation, the rollback segment ARSystemRoll is created. All logging is performed in the system rollback segment. To use the new rollback segment, you must activate it.
Activating AR SystemRoll with the Oracle Database

1. Open the Oracle database system startup file.
2. Add a reference for ARSystemRoll to the rollback_segment line within the parentheses with a comma between the last segment listed and the new segment.
   
   For example, the new line might look like this:
   
   rollback_segments=(r1, ARSystemRoll)

3. Restart the Oracle database to have the new definition recognized and use the new rollback segment.

Working with Earlier AR System Versions That Use Oracle

In the current release of AR System, diary fields and all character fields that allow over 4000 bytes of text must use the Oracle data type clob. Character fields that allow 4000 bytes of text or less use the data type varchar. Fields created as clob and varchar are stored in the main data table.

In AR System versions prior to 4.5, character fields allowing over 255 bytes of text used the long data type. If you are upgrading from a previous version, note how the long character and diary fields work in the current version:

- The fields will continue to use the long data type, and a separate table will hold the values for these fields. During the upgrade, a new column called isLong is added to the field_char and diary_char tables to distinguish between long and clob fields. The isLong column is set to 1 if the data type is long, and 0 if the data type is clob or varchar. For information about converting long fields to clob or varchar, see Changing Character Field Lengths on page 39.

- The SQL view joins the base table and all of the associated long character tables. The resulting view has a single table that consists of all data (except the status history data) that is associated with the form. The database allows multiple long fields to be in a view, but not in a base table. All character and diary fields created in the current version use the main SQL view.
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**


**Informix**

- Informix Guide to SQL: Tutorial
- Informix Guide to SQL: Reference
- Informix Administrators Guide

For a discussion of the structure used by previous versions of AR System for the Informix database, see technical notes available at the Remedy web site ([www.remedy.com](http://www.remedy.com)).

**Oracle**

- SQL Reference Manual
- Oracle Administrator’s Guide

**Sybase**

- Sybase Commands Reference Manual
- Sybase Administration Guide

**MS SQL**

- Transact-SQL Reference Manual
- Microsoft SQL Server Administrator’s Companion
The technical essays in this chapter describe procedures for improving performance or enhancing security of your AR System environment. The following topics are covered:

- *Converting AR System Dates to Database Dates* on page 48
- *Changing the Next Available ID for New Requests* on page 50
- *Changing the Request ID Field Length or Prefix* on page 53

These essays address some of the most commonly requested AR System technical information. For access to the complete set of AR System technical essays, refer to the Technical Support section of the Remedy web site (http://www.remedy.com).
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_#>`. Alternatively, you can provide the SQL view name of the column (the database name of the field as displayed in AR System Administrator).

**Converting the Date and Time Format for a DB2 Universal Database**

Refer to the DB2 SQL Reference for information about dateline arithmetic.

**Converting 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 Where `<column_#>` is the number of the column for the date and time field, `<table_#>` 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, 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_#> units second)) from T<table_#>
```
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:

```
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>
```

Refer to the Informix Guide to SQL for information about the datetime, extend, and interval functions.

**Converting 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 who has write access to the AR System tables.

2. Where `<column_#>` is the number of the column for the date and time field, `<table_#>` 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, type the following command:

```
% SELECT TO_CHAR(TO_DATE('01/01/1970 00:00:00', 'MM/DD/YYYY HH24:MI:SS') + ((C<column_#> + <offset>)/(60*60*24)),
   'MM/SS/DD YYYY HH24:MI:SS') FROM T<table_#>;
```

Refer to the Oracle SQL Language Manual for information about the TO_DATE and TO_CHAR functions.

**Converting the Date and Time Format for a Sybase or MS SQL Server Database**

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

2. Where `<column_#>` is the number of the column for the date and time field, `<table_#>` 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, type the following command:

```
% select dateadd(second, C<column_#> + <offset>,
   'Jan 1, 1970') from T<table_#>
```

3. If desired, format the date field by using the convert function.

There are 12 different formats from which you can choose. All are explained in the Sybase Reference Manual.
Changing the Next Available ID for New Requests

You can change the next available ID when creating new AR System requests. This ID is used to automatically generate the unique index number that is attached to each AR System request. Under some conditions, you might need to reset the next available ID. For example, you might need to establish different ranges for a similar form on two different servers, or you might need to reserve a range of numbers for later use.

**Note:** Do not change the next available ID to a number lower than the greatest existing ID. The Request ID field value must be unique within AR System, and resetting the ID to a lower number could conflict with existing Request ID field values. If you try to submit a request with an existing ID, AR System will return an error and prevent the request from being submitted until the conflict is resolved.

If you must change the next available ID, make the change when the system is not in use to avoid conflicts with users who are submitting new requests.

The Request ID Field

Every form defined in AR System contains a set of core fields. The Request ID field is the core field with a unique field ID of 1. You can change the label of this field to something other than Request ID, but the field ID will always remain 1.

The Request ID field contains a character string that holds a unique index for each request. The form of this string is an optional prefix, which can consist of any alphanumeric characters, followed by a 0-padded numeral (for example, HD0000000000001). The field can be no fewer than 5 characters and no more than 15 characters in length. The prefix can be as long as the total length of the field less 5 characters.

When new requests are submitted, AR System generates a new ID for the request by appending the next available ID to the prefix, if a prefix is specified. The next available ID then increments in preparation for the next request to be submitted.
Changing the Next Available ID for New Requests

The Request ID field contains a unique number sequence. Create other fields to contain information that is specific to your site instead of using the Request ID field. Overloading the Request ID field with other information can restrict your ability to control this data and limits the flexibility of searches on the data.

**Changing the Next Available ID for SQL Databases**

Changing the Next Available ID for a Form in an SQL Database:

1. Stop ARServer.
2. Using any front-end tool that allows direct access to an SQL database, log in as a user who has write access to the AR System tables.
3. Connect to the AR System table area.
4. Find the Request ID field for the form you want to alter.
5. Update the next available ID.
6. Restart ARServer.

**Example SQL Database Procedures**

The following sections are examples of how to change the next available ID for DB2 Universal, Informix, Oracle, and MS SQL and Sybase databases. In these examples, the next available ID for a form named `ZZZ` is changed from the current value of 1291 to a new value of 25000.

**DB2 Universal Database Example**

```
> open DB2 command center
Connect to AR System.
> select name, nextId from arschema where name = 'ZZZ';
NAME   NEXTID
--- -------
ZZZ    1291
1 row(s) retrieved.
> update arschema set nextId = 25000 where name = 'ZZZ';
1 row(s) updated.
```

**Informix Database Example**

```
% dbaccess -.
> database ARSystem;
Database selected.
> select name, nextId from arschema where name = 'ZZZ';
```

Changing the Next Available ID for New Requests 51
name nextId
ZZZ   1291
1 row(s) retrieved.
>update arschema set nextId = 25000 where name = 'ZZZ';
1 row(s) updated.
><Control-C>

**Oracle Database Example**

% sqlplus
Enter user-name: ARAdmin
Enter password: <password> (AR#Admin# by default.)
SQL>select name, nextId from ARAdmin.arschema where name = 'ZZZ';
NAME   NEXTID
------------------------------ ----------
ZZZ   1291
SQL>update ARAdmin.arschema set nextId = 25000 where name = 'ZZZ';
1 row updated.
SQL>exit

**MS SQL Server and Sybase Database Example**

% isql -Usa
Password: <password>
1>use ARSystem
2>go
1>select name, nextId from arschema where name = 'ZZZ'
2>go
name   nextId
ZZZ   1291
(1 row affected)
1>update arschema set nextId = 25000 where name = 'ZZZ'
2>go
(1 row affected)
1>exit
Changing the Request ID Field Length or Prefix

After using a form for a while, you might need to change the prefix or length of the Request ID field, the key field in a form. Often, this change can be made and existing requests can retain the format used previously. However, you might need to convert existing Request ID field values to match the new prefix or length. This section offers background information and procedures to help you make changes to the Request ID.

The Request ID Field

Every form defined in AR System contains a set of core fields. The Request ID field is the core field with a unique field ID of 1. You can change the label of this field to something other than Request ID, but the field ID will always remain 1.

The Request ID field contains a character string that holds a unique index for each request. The form of this string is an optional prefix, which can consist of any alphanumeric characters, followed by a 0-padded numeral (for example, \texttt{HD0000000000001}). The field can be no fewer than 5 characters and no more than 15 characters in length. The prefix can be as long as the total length of the field less five characters.

When new requests are submitted, AR System generates a new ID for the request by appending the next available ID to the prefix, if a prefix is specified. The next available ID increments in preparation for the next request to be submitted.

The Request ID field is a unique number sequence. Create other fields to contain information that is specific to your site instead of using the Request ID field. Overloading the Request ID field with other information can restrict your ability to control this data and limits the flexibility of searches on the data.
### Changing the Length of the Request ID Field

1. Log in to AR System Administrator as a user with administrator access.
2. Open the form you want to alter.
3. Double-click the Request ID field. The Field Properties dialog box appears.
4. Select the Database tab and specify the desired length in the Input Length field.
   
   **Note:** The length of the Request ID field must be between 5 and 15 characters. If you specify a prefix for the Request ID field, the field must be at least five characters greater than the prefix.
5. Save the changes to the form.

### Changing the Prefix of the Request ID Field

1. Log in to AR System Administrator as a user with administrator access.
2. Open the form you want to alter.
3. Double-click the Request ID field. The Field Properties dialog box appears.
4. Click the Attributes tab.
5. Specify the desired prefix in the Default Value field.
   
   **Note:** The Request ID field must be between 5 and 15 characters in length. If you specify a prefix for the Request ID field, the field length must be at least five characters greater than the prefix.
6. Save the changes to the form.

### Preserving Existing Request ID Field Values

You might want to preserve the existing Request ID field values of your AR System requests for the following reasons:

- **Backward compatibility**—You might have cross-references that refer to requests by the Request ID field value.
- **History**—The Request ID field values were created with the old format, and there is no need for change.
Design—The design of your AR System calls for periodic change to the Request ID field. For example, you might use the current year as a prefix for the Request ID field.

No data—No requests have been submitted, so there are no Request ID fields to be converted.

Changing Existing Request ID Field Values to the New Format

You might want to change the existing Request ID field values of your AR System requests for any of the following reasons:

Consistency—All of the Request ID field values for a form follow the same format. If the format changes, all of the requests change to match the format.

Design—The design of your AR System has changed, and this design references the new format of the Request ID field. This is usually a change of the length of the field from a default setting of 15 to something shorter, and you need to eliminate the extra leading zeros.

Two methods of updating existing Request ID field values are explained in this essay: editing the AR Export file, and using commands that access the SQL database directly. You can edit AR Export files regardless of the database underlying your AR System. If you are using a flat-file database, editing the .arx file is the only available update strategy. If you are using an SQL database, you can access an SQL database directly. After implementing one of the strategies in this essay, read Status History Table and Attachment Tables on page 66 and Long Character and Diary Tables in Oracle on page 66.

Warning: Back up your database before performing the actions described in this section to ensure your original data is saved if there is a failure during the update.

Using an AR Export (.arx) File

You can edit AR Export (.arx) files regardless of the database underlying your AR System. If you are using a flat-file database, editing the .arx file is the only available update strategy. If you are using an SQL database, you can use the .arx strategy or one of the other strategies that bypass AR System to operate directly in the database.
Changing the Existing Request ID Field Format Through an AR Export File

1. In AR System Windows User Tool, open the form you want to change.
2. Choose Tools > Reporting.
   The Report dialog box appears.
   The Properties - << New Style >> dialog box opens.
4. Click Add All to add all the fields to the report style.
5. Select the Request ID field under Selected Fields and move it to the top of the list.
6. Choose Report > Export To > File and use .arx format to save all the data for the form to a file.
7. Close the Report dialog box.
8. Edit the file to change the format of the Request ID field (see Editing the .arx File, next).
9. In AR System Windows User Tool, delete all requests in the form.
11. Open AR System Import.
    The Open Import File dialog box appears.
13. Select the file you edited, and click Open.
14. Choose File > Open Form.
    The Open Form dialog box appears.
15. Select the form you want to change and click OK.
16. Click Add All.
17. Choose File > Preferences.
    The Preferences dialog box appears.
18. Select the Data tab. Disable fields’ pattern matching and make required fields optional during import by selecting the check boxes. Then select the Duplicate Request ID tab and select the Reject Duplicate Record option button.
19. Click OK to close the Preferences dialog box.
20. Choose Import to start the import process.
Changing the Request ID Field Length or Prefix

After the first few header lines, the remaining lines in the .arx file have a format as in the following example, where <other_data> is data from the form:

```
DATA "000000000000001" "<other_data>" 1 "<other_data>"
```

The Request ID field always follows the keyword DATA. In this example, the Request ID field has no prefix and is 15 characters in length. Use a text editor, such as WordPad, to convert the format of the Request ID field.

The commands in the following procedures change all lines that start with DATA "00000000 to DATA "ABC. The commands contain eight zeros. Five of these represent the difference between the original length of 15 characters and the new length of 10 characters. The other three zeros represent the spaces to be replaced by ABC.

The following procedures show how you can shorten a Request ID field to 10 characters and add a prefix of ABC.

**Editing the .arx File in Windows**

1. Open the .arx file in a text editor that has a Find/Replace command with a feature for matching case (for example, WordPad).
2. Use the Find/Replace command to search for DATA "00000000.
3. Use the match case feature.
4. Replace all instances of DATA "00000000 with DATA "ABC.
5. Save the changes to the file.

**Editing the .arx File in UNIX**

1. Open the file in a text editor.
2. Type the following command:
   ```
g /^DATA "00000000/s//DATA "ABC/
g
```
3. Save and close the file.
Using SQL Commands to Shorten the Request ID Field

Only administrators running AR System with an SQL database can update existing request ID field values by directly accessing the SQL database. The syntax for direct access is different for each SQL database that AR System supports. These commands are described in the following examples.

To use the methods described in this section, you must be familiar with basic commands in the SQL command interface. SQL commands bypass AR System completely. If you bypass AR System, verify that all data is valid when you are finished. Create a practice table in your database and practice the commands you will issue to ensure that you are issuing the correct commands. Be sure to back up your database or all the relevant tables.

To shorten the request ID field value, find the table holding the form being changed. The name of the table is the letter T added to the beginning of the form ID (also called schema ID). The request ID column of the table is always named C1.

Note: All related database tables, such as status history tables (H Tables), and Attachment tables (B Tables and BC Tables) must also be updated.

Database tables include:

- **T<schema_ID>**—A table that contains the data in your form. A table named T43 indicates that 43 is the schema ID.

- **T<schema_ID>C<field_ID>**—(Oracle only) Used for backward compatibility with forms created with ARServer versions prior to 4.5. It is a table that contains long text and diary data. A table named T43C536870924 indicates that 43 is the schema ID and 536870924 is the field ID. In this example, the field ID for the diary or long text field is 536870924. In many cases, there will be more than one long text or diary field on the form.

- **H<schema_ID>**—A table that contains the Status History information for your form. A table named H43 indicates that 43 is the schema ID.
Changing the Request ID Field Length or Prefix

- **B<schema_ID>**—A table that contains a list of all the attachments and related information for each record in your form. A table named B43 indicates that 43 is the schema ID.

- **B<schema_ID>C<field_ID>**—A table that contains the actual Binary objects for attachment fields in your form. A table named B43C536870924 indicates that 43 is the schema ID and 536870924 is the field ID. In this example, the field ID for the attachment field is 536870924. In some cases, there will be more than one attachment field on the form.

Note that in H<schema_ID> tables, T<schema_ID>C<field_ID> tables, and B<schema_ID>C<field_ID> tables, the Entry ID column is equivalent to the C1 column in the T<schema_ID> and B<schema_ID> tables.

**Finding the Schema ID and Field ID**

- To find the correct schema ID for your form, perform the following query:
  
  ```sql
  Select SchemaId, name from arschema order by 2
  ```

- To find the correct field ID (after you know the schema ID), perform the following SQL statement. This example assumes that the schema ID is 43:
  
  ```sql
  Select FieldId, FieldName from field where SchemaId = 43
  ```

**Changing Existing Request ID Field Values to the New Format When the Request ID Has a Prefix**

The following examples assume that the table is named T43, that the prefix is HD, and that the field size (including the prefix) will be 8 characters. The 6 represents the number of characters to keep, starting from the right side of C1. C1 is originally 15 characters long. Make sure that the number of characters in your prefix plus the second parameter in the `RIGHT` function is equal to the new size of the C1 field.

**DB2 Database Examples**

- To add a prefix to the T<schema_ID> table, use the following syntax:
  
  ```sql
  update T43 set C1 = 'HD' || RIGHT(C1, 6)
  ```

- To add a prefix to the B<schema_ID> table, use the following syntax:
  
  ```sql
  update B43 set C1 = 'HD' || RIGHT(C1, 6)
  ```
For the H<schema_ID> table, use the following syntax:

```
update H43 set entryId = 'HD' || RIGHT(entryId, 6)
```

For the B<schema_ID>C<field_ID> tables, use the following syntax:

```
update B43C536870924 set entryId = 'HD' || RIGHT(entryId, 6)
```

**Informix Database Examples**

In the following examples, the request ID is being shortened from 15 to 8 characters. The prefix HD is concatenated to the last 6 characters in the string, consisting of positions 10 through 15.

Note that for Informix databases, you must log in as the root user.

To add a prefix to the T<schema_ID> table, use the following syntax:

```
update T43 set C1 = 'HD'||C1[10,15]
```

To add a prefix to the B<schema_ID> table, use the following syntax:

```
update B43 set C1 = 'HD'||C1[10,15]
```

For the H<schema_ID> table, use the following syntax:

```
update H43 set entryId = 'HD'||entryId[10,15]
```

For the B<schema_ID>C<field_ID> tables, use the following syntax:

```
update B43C536870924 set entryId = 'HD'||entryId[10,15]
```

**Note:** In the functions C1[10,15] and entryId[10,15], the 10 represents the starting position of the characters to keep and 15 represents the ending position.
Oracle Database Examples
To add a prefix to the T<schema_ID> table, use the following syntax:

```
update T43 set C1 = 'HD'||substr(C1,10,6);
```

To add a prefix to the B<schema_ID> table, use the following syntax:

```
update B43 set C1 = 'HD'||substr(C1,10,6);
```

For the H<schema_ID> table, use the following syntax:

```
update H43 set entryId = 'HD'||substr(entryId,10,6);
```

For the B<schema_ID>C<field_ID> tables, use the following syntax:

```
update B43C536870924 set entryId = 'HD'||substr(entryId,10,6);
```

For the T<schema_ID>C<field_ID> tables, use the following syntax:

```
update T43C536870924 set entryId = 'HD'||substr(entryId,10,6);
```

**Note:** In the functions `substr(C1,10,6)` and `substr(entryId,10,6)`, the 10 represents the starting position of the characters to keep and the 6 is the number of characters to keep.

**MS SQL Server and Sybase Database Examples**
To add a prefix to the T<schema_ID> table, use the following syntax:

```
update T43 set C1 = "HD"+ RIGHT(C1, 6)
```

To add a prefix to the B<schema_ID> table, use the following syntax:

```
update B43 set C1 = "HD" + RIGHT(C1, 6)
```

For the H<schema_ID> table, use the following syntax:

```
update H43 set entryId = "HD" + RIGHT(entryId, 6)
```

For the B<schema_ID>C<field_ID> tables, use the following syntax:

```
update B43C536870924 set entryId = "HD"+ RIGHT (entryId, 6)
```

Changing the Request ID Field Length or Prefix  
61
Changing Existing Request ID Field Values to the New Format When the Request ID Does Not Have a Prefix

The following examples assume that the table is named T43 and that the field size will be 8 characters. The 8 represents the number of characters to keep, starting from the right side of C1. C1 is originally 15 characters long. Make sure that the number of characters in the second parameter in the RIGHT function is equal to the new size of the C1 field and that the sum of the two numeric values in the SUBSTR function is 16 (1 greater than the original length of C1).

DB2 Database Examples

- To add a prefix to the T<schema_ID> table, use the following syntax:
  update T43 set C1 = RIGHT(C1, 8)

- To add a prefix to the B<schema_ID> table, use the following syntax:
  update B43 set C1 = RIGHT(C1, 8)

- For the H<schema_ID> table, use the following syntax:
  update H43 set entryId = RIGHT(entryId, 8)

- For the B<schema_ID>C<field_ID> tables, use the following syntax:
  update B43C536870924 set entryId = RIGHT(entryId, 8)
Informix Database Examples
For Informix databases, you must log in as the root user.

In the following examples, the Request ID is being shortened from 15 to 8 characters. The request ID will consist of the last 8 characters in the string, consisting of positions 8 through 15.

▸ To add a prefix to the T<schema_ID> table, use the following syntax:
  update T43 set C1 = C1[8,15]

▸ To add a prefix to the B<schema_ID> table, use the following syntax:
  update B43 set C1 = C1[8,15]

▸ For the H<schema_ID> table, use the following syntax:
  update H43 set entryId = entryId[8,15]

▸ For the B<schema_ID>C<field_ID> tables, use the following syntax:
  update B43C536870924 set entryId = entryId[8,15]

Note: In the functions C1[8,15] and entryId[8,15], the 8 represents the starting position of the characters to keep and 15 represents the ending position.

Oracle Database Examples
▸ To add a prefix to the T<schema_ID> table, use the following syntax:
  update T43 set C1 = substr(C1,8,8);

▸ To add a prefix to the B<schema_ID> table, use the following syntax:
  update B43 set C1 = substr(C1,8,8);

▸ For the H<schema_ID> table, use the following syntax:
  update H43 set entryId = substr(entryId,8,8);
For the B<schema_ID>C<field_ID> tables, use the following syntax:
update B43C536870924 set entryId = substr(entryId,8,8);

For the T<Schema_ID>C<field_ID> tables, use the following syntax:
update T43C536870924 set entryId = substr(entryId,8,8);

Note: In the functions substr(C1,8,8) and substr(entryId,8,8), the first 8 represents the starting position of the characters to keep and the second 8 is the number of characters to keep.

MS SQL Server and Sybase Database Examples

To add a prefix to the T<schema_ID> table, use the following syntax:
update T43 set C1 = RIGHT(C1, 8)

To add a prefix to the B<schema_ID> table, use the following syntax:
update B43 set C1 = RIGHT(C1, 8)

For the H<schema_ID> table, use the following syntax:
update H43 set entryId = RIGHT(entryId, 8)

For the B<schema_ID>C<field_ID> tables, use the following syntax:
update B43C536870924 set entryId = RIGHT (entryId, 8)

Using SQL Commands to Lengthen the Request ID Field Value

The format for all of the supported databases is the same for lengthening the Request ID field format as with shortening the Request ID field format. Refer to Using SQL Commands to Shorten the Request ID Field on page 58 for hints about how to run the SQL interface, how to find the name of the table to be changed, and how to exit the SQL interface.

Note: Remember that the maximum length allowed for the Request ID field is 15 bytes.
In the following example, the length of the field is restored to 15 characters from the current 10 characters. When you have determined the name of the table (T43 in the example), issue the one of the following commands at the prompt:

- **DB2**
  ```
  % update T43 set C1 = '00000' || C1
  ```

- **Informix**
  ```
  % update T43 set C1 = '00000' || C1
  ```

- **Oracle**
  ```
  % update T43 set C1 = '00000' || C1
  ```

- **Sybase and MS SQL**
  ```
  % update T43 set C1 = '00000' + C1
  ```

This command updates all requests in the table by adding 5 leading zeros to the existing value of the Request ID field and assigning the resulting 15 character string to the Request ID field.

If you want to add a prefix, specify the prefix as part of the string to be added. For example, if you want to expand to 15 characters and add a prefix of ABC, use 'ABC00' instead of '00000' in the preceding example.

**Status History Table**

Status History information is stored in a separate table. This table uses the Request ID field as the link to the main table. Accordingly, the same procedure must be followed for the Request ID field values in the status history table.

The status history table follows a naming convention similar to the base table, except the letter H is used instead of the letter T. In the previous examples, the status history table would be H43. The column holding the Request ID field is named `entryId`. To update the status history table, use the commands described in the previous examples, substituting H43 for T43 and `entryId` for C1.
Attachment Tables

Attachment information is stored in two tables, the Attachment Details table and the Attachment Data table. The Attachment Details table holds attachment characteristics, such as the name and size of the attachment, and the Attachment Data table holds the actual attachment. These tables also use the Request ID field as the link to the main table, so the same procedure must be followed for the Request ID field values.

The Attachment Details table is named with a B followed by the schema ID (for example, B3). The Attachment Data table is named with a B followed by the schema ID, 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, 536870920 is the attachment field ID.

The column holding the Request ID in the Attachment Details table is named C1, and in the Attachment Data table, it is named entryId. To update the Request ID field in the attachment tables, use the commands described in the previous examples, substituting the appropriate table name, and using either C1 or entryId for the Request ID.

Long Character and Diary Tables in Oracle

For Oracle, you might need to update additional tables. All long character and diary fields created in a previous version of the AR System database are stored in separate tables. These tables are linked to the main table using the Request ID field as the reference. The procedure applied to the main table must also be applied to these tables. Long character and diary fields created in an Oracle database using the current version of AR System are stored in the main table, so no additional tables need to be updated.

Update one table for each character or diary field created in a previous version that uses the long data type. The tables are specified by the name of the main table followed by a C and the ID of the long character or diary field. A field with an ID of 10 in the example form would be listed as T43C10. The column holding the Request ID field is named entryId. To update each of the tables, use the commands described in the previous examples, substituting the appropriate table name for T43 and entryId for C1.
This chapter includes sets of SQL commands that define the AR System data dictionary for the following databases:

- DB2 Universal
- Informix
- Oracle
- Sybase and Microsoft SQL
The following includes a set of SQL commands that define the AR System data dictionary for DB2 Universal. For an explanation of these commands, see *A Complete Guide to DB2 Universal Database*.

```
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(30) not null,
lastChanged varchar(30) not null,
coreVersion int not null,
umFields int not null,
umVuis int not null,
defaultVui varchar(254) not null,
nextId int not null,
nextFieldId int not null,
maxStatEnums int not null,
upgrdVersion int ,
changeDiary clob(1M),
helpText clob(1M),
objProp 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 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_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 varchar(255) not null,
     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 varchar(255) not null);
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(30) not null,
     lastChanged varchar(30) not null,
     datatype int not null,
     fOption int not null,
     createMode int not null,
     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(30) not null,
     lastChanged varchar(30) not null,
     changeDiary clob(1M) ,
     helpText clob(1M) );
CREATE UNIQUE INDEX vui_ind
    ON vui (schemaId, vuiId) CLUSTER ;
Chapter 3—SQL Definitions of the Data Dictionary Tables

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_dispprop_ind
ON field_dispprop (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 ,
gbeMatchOp 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,
chapter 3—sql definitions of the data dictionary tables

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);
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) );
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 )
create unique index field_column_ind
on field_column (schemaId, fieldId)
cluster;
create table field_dec
(schemaId int not null,
fieldId int not null,
CREATE TABLE field_dec
(schemaId int not null,
fieldId int not null,
rangelow varchar(64),
rangehigh 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),
rangehigh 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 );
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(30) not null,
     lastChanged varchar(30) not null,
     refreshCode int not null,
     menuType int not null,
     changeDiary clob(1M) ,
     helpText clob(1M) ,
     objProp 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) );
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  varchar(254) not null,
   containerType int    not null,
   timestamp    int    not null,
   owner        varchar(30)  not null,
   lastChanged  varchar(30)  not null,
CREATE TABLE arcontainer
(containerId    int          not null,
label           varchar(255)         ,
description     clob(1M)             ,
changeDiary     clob(1M)             ,
helpText        clob(1M)             ,
objProp         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(30) not null,
lastChanged varchar(30) 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,
queryShort varchar(255),
queryLong clob(1M),
changeDiary clob(1M),
helpText clob(1M),
objProp clob(1M));
CREATE UNIQUE INDEX filter_ind
ON filter (name) CLUSTER;
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 clob(1M)            );
CREATE  INDEX filter_notify_ind
    ON filter_notify (filterId)
    CLUSTER ;
CREATE TABLE filter_notify_ids
    (filterId       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
    (filterId       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) );
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) );
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);
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) );
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)   );
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            );
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
    (name varchar(254) not null,
     escalationId int not null,
     timestamp int not null,
     owner varchar(30) not null,
     lastChanged varchar(30) not null,
     wkConnType varchar(30) not null,
     numActions int not null,
     numElses int not null,
     firetmType int not null,
     tminterval int not null,
     monthday int not null,
     weekday int not null,
     minute int not null,
     enable int not null,
     queryShort varchar(255) ,
     queryLong clob(1M) ,
     changeDiary clob(1M) ,
     helpText clob(1M) ,
     objProp clob(1M) );
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(30)  not null,
lastChanged   varchar(30)  not null,
wkConnType    int          not null,
alOrder       int          not null,
executeMask   int          not null,
controlfieldId int          ,
fieldId       int          not null,
enable        int          not null,
umActions     int          not null,
umElses       int          not null,
queryShort    varchar(255)         ,
queryLong     clob(1M)            ,
changeDiary   clob(1M)            ,
helpText      clob(1M)            ,
objProp       clob(1M)            );

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) ) ;
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 act_linkProcessInd
  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 ) ;

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,
     actionShort varchar(255) ,
     actionLong clob(1M) ,
     COMShort varchar(255) ,
     COMLong clob(1M) ) ;
CREATE  INDEX actlink_auto_ind
    ON actlink_auto (actlinkId)
    CLUSTER ;
CREATE TABLE actlink_push
    (actlinkId int not null,
    actionIndex int not null,
    fieldId int not null,
    assignShort varchar(255) ,
    assignLong clob(1M)    );
CREATE  INDEX actlink_push_ind
    ON actlink_push (actlinkId)
    CLUSTER ;
CREATE TABLE actlink_sql
    (actlinkId int not null,
    actionIndex int not null,
    assignShort varchar(255) ,
    assignLong clob(1M)    ,
    keywordList clob(1M)    ,
    parameterList clob(1M)  );
CREATE  INDEX actlink_sql_ind
    ON actlink_sql (actlinkId)
    CLUSTER ;
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 ,
    sortlst varchar(255) ,
    queryshort varchar(255) ,
    querylong clob(1M)    ,
    msgType int ,
    ...
msgNum        int                  ,
msgText       clob(1M)             ,
msgPane       char                 ,
reportstr     clob(1M)             ,
supresEptyLst 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);
CREATE  INDEX actlink_commit_ind
   ON actlink_commit (actlinkId)
   CLUSTER ;
CREATE TABLE actlink_close
     (actlinkId     int          not null,
      actionIndex   int          not null,
      closeAll      char                );
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);
CREATE  INDEX actlink_call_ind
   ON actlink_call (actlinkId)
   CLUSTER ;
CREATE TABLE actlink_exit
     (actlinkId     int          not null,
      actionIndex   int          not null,
      closeAll      char                );
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);

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_maping_ind
ON actlink_mapping (schemaId, actlinkId);

CREATE TABLE alert_user
(username varchar(30) NOT NULL,
password varchar(30),
clientIPAddr varchar(16) NOT NULL,
actualIPAddr varchar(16) NOT NULL,
serverIPAddr varchar(16) NOT NULL,
clientPort int NOT NULL,
CREATE TABLE alert_user (
    username     varchar(30)  not null,
    password     varchar(255)         ,
    clientIPAddr varchar(255)         ,
    clientPort   int                  ,
    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, password, clientIPAddr, clientPort);
CREATE TABLE alert_time
    (username      varchar(30)  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 server_cache
    (serverId     int          not null,
     server       varchar(255) not null);
CREATE UNIQUE INDEX server_cache_ind
    ON server_cache (serverId) CLUSTER ;
CREATE TABLE user_cache
    (serverId     int          not null,
     entryId      varchar(15)  not null,
     userName     varchar(30)  not null,
     password     varchar(255)         ,
     authUserName varchar(30)          ,
     authString   varchar(255)         ,
     email        varchar(255)         ,
     notifyMech   int                  ,
     licType      int                  ,
     ...
licTypeFText int ,
licTypeReserv1 int ,
timestamp int ,
validateKey varchar(30) ,
shortGroup varchar(255) ,
longGroup clob(1M) );
CREATE INDEX user_cache_ind2
ON user_cache (userName)
CLUSTER ;
CREATE UNIQUE INDEX user_cache_ind
ON user_cache (serverId, entryId);
CREATE TABLE group_cache
(serverId int not null,
entryId varchar(15) not null,
groupId int not null,
groupName varchar(30) not null,
groupType int not null,
floatLic int ,
floatLicFTS int ,
timestamp int );
CREATE INDEX group_cache_ind2
ON group_cache (groupId) CLUSTER ;
CREATE UNIQUE INDEX group_cache_ind
ON group_cache (serverId, entryId);
INSERT INTO control VALUES (18, 1, 1, 2, 1, 1, 1, 1);
INSERT INTO group_cache VALUES (0,
'000000000000001', 1, 'Administrator', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0,
'000000000000002', 2, 'Customize', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0,
'000000000000003', 3, 'Submitter', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0,
'000000000000004', 4, 'Assignee', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0,
'000000000000005', 0, 'Public', 1, 0, 0, 0);
INSERT INTO group_cache VALUES (0,
'000000000000006', 5, 'Sub Administrator', 2, 0, 0, 0);
The following includes a set of SQL commands that define the AR System data dictionary for Informix databases. For an explanation of the commands, see the Informix Guide to SQL: Reference.

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(30)  not null,
   lastChanged  varchar(30)  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                  ,
   changeDiary  byte                 ,
   PRIMARY KEY (name, schemaId));

INSERT INTO group_cache VALUES (0,
  '0000000000000007', 7, 'Assignee Group', 2, 0, 0, 0);
INSERT INTO user_cache VALUES (0,
  '0000000000000001', 'Demo', '', NULL, NULL, NULL, 1,
  1, 0, 0, 0, NULL, '1;', NULL);
helpText byte ,
objProp 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_index
(schemaId int not null,
listIndex int not null,
numFields int not null,
uniqueFlag int not null,
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    varchar(255) not null,
 keyField     varchar(254) not null,
 queryShort   varchar(255)         ,
 queryLong    byte                 );
CREATE UNIQUE INDEX schema_view_ind 
ON schema_view (schemaId);
CREATE TABLE schema_vendor
(schemaId int not null,
 vendorName varchar(254) not null,
 tableName varchar(255) not null);
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(30) not null,
 lastChanged varchar(30) not null,
 datatype int not null,
 fOption int not null,
 createMode int not null,
 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
(schemaId int not null,
 vuiId int not null,
 vuiName varchar(254) not null,
 locale varchar(30) ,
 vuiType int ,
 timestamp int not null,
 owner varchar(30) not null,
 lastChanged varchar(30) not null,
 changeDiary byte ,
 helpText byte );
CREATE UNIQUE CLUSTER INDEX vui_ind
ON vui (schemaId, vuiId);
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 byte);
CREATE UNIQUE INDEX field_dispprop_ind
ON field_dispprop (schemaId, fieldId, listIndex, vuiId);
CREATE TABLE field_int
(schemaId int not null,
fieldId int not null,
rangLow int,
rangHigh int);
CREATE UNIQUE CLUSTER INDEX field_int_ind
ON field_int (schemaId, fieldId);
CREATE TABLE field_real
(schemaId int not null,
fieldId int not null,
rangLow float,
rangHigh float,
arprecision int);
CREATE UNIQUE CLUSTER INDEX field_real_ind
ON field_real (schemaId, fieldId);
CREATE TABLE field_diary
(schemaId int not null,
fieldId int not null,
fullTextOptions int);
CREATE UNIQUE CLUSTER INDEX field_diary_ind
ON field_diary (schemaId, fieldId);
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 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);
```sql
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 byte);

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

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);

CREATE UNIQUE CLUSTER INDEX field_column_ind
ON field_column (schemaId, fieldId);

CREATE TABLE field_dec
(schemaId int not null,
 fieldId int not null,
 rangeLow varchar(64),
 rangeHigh varchar(64),
 arprecision int);

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

CREATE TABLE field_curr
(schemaId int not null,
 fieldId int not null,
 rangeLow varchar(64),
 rangeHigh varchar(64),
 arprecision int);"
funcCurr byte,
allowCurr byte);

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

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

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

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

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

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

CREATE UNIQUE CLUSTER INDEX field_date_ind
ON field_date (schemaId, fieldId);

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

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(30)  not null,
 lastChanged  varchar(30)  not null,
 refreshCode  int          not null,
 menuType     int          not null,
 changeDiary  byte                 ,
 helpText     byte                 ,
 objProp      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(30)  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                        ,

CREATE TABLE char_menu_query
(labelField5 int,
valueField int not null,
sortOnLabel int not null,
queryShort varchar(255),
queryLong byte,
keywordList lvarchar,
parameterList lvarchar,
externList lvarchar
);
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,
fileLocation int 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,
labelIndex2 int,
labelIndex3 int,
labelIndex4 int,
labelIndex5 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(30) not null,
   lastChanged varchar(30) not null,
   numReferences int not null,
   label varchar(255),
   description byte,
   changeDiary byte,
   helpText byte,
   objProp 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);
CREATE TABLE filter
  (name         varchar(254) not null,
   filterId     int          not null,
   timestamp    int          not null,
   owner        varchar(30)  not null,
   lastChanged  varchar(30)  not null,
   wkConnType   varchar(30)  not null,
CREATE TABLE filter
(fOrder int not null,
opSet int not null,
enable int not null,
numActions int not null,
numElses int not null,
queryShort varchar(255),
queryLong byte,
changeDiary byte,
helpText byte,
objProp 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__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                );
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 );
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
(filterId int not null,
actionIndex int not null,
serverName varchar(64) not null,
guideName varchar(254) not null,
guideMode int not null,
guideTableId int );
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,
 escalationId int          not null,
 timestamp    int          not null,
 owner        varchar(30)  not null,
 lastChanged  varchar(30)  not null,
 wkConnType   int          not null,
 numActions   int          not null,
 numElses     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,
 queryShort   varchar(255)         ,
 queryLong    byte                 ,
 changeDiary  byte                 ,
 helpText     byte                 ,
 objProp      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(30) not null,
lastChanged varchar(30) 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,
queryShort varchar(255) ,
queryLong byte ,
changeDiary byte ,
helpText byte ,
objProp 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 );
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,
accessOpt 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);
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                  ,
  sortlst       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 );
CREATE CLUSTER INDEX actlink_call_ind
  ON actlink_call (actlinkId);

CREATE TABLE actlink_exit
  (actlinkId int not null,
   actionIndex int not null,
   closeAll char );
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(30)  not null,
   password      varchar(30)          ,
   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, password, clientIPAddr, clientPort);

CREATE TABLE alert_time
   (username      varchar(30)  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 server_cache
  (serverId   int          not null,
   server     varchar(255) not null);
CREATE UNIQUE CLUSTER INDEX server_cache_ind
  ON server_cache (serverId);

CREATE TABLE user_cache
  (serverId       int          not null,
   entryId        varchar(15)  not null,
   userName       varchar(30)  not null,
   password       varchar(255)         ,
   authUserName   varchar(30)          ,
   authString     varchar(255)         ,
   licensePool    varchar(30)          ,
   email          varchar(255)         ,
   notifyMech     int                  ,
   licType        int                  ,
   licTypeFText   int                  ,
   licTypeReserv1 int                  ,
   timestamp      int                  ,
   validateKey    varchar(255)         ,
   shortGroup     varchar(255)         ,
   longGroup      byte                 );
CREATE CLUSTER INDEX user_cache_ind2
  ON user_cache (userName);
CREATE UNIQUE INDEX user_cache_ind
  ON user_cache (serverId, entryId);

CREATE TABLE group_cache
  (serverId    int          not null,
   entryId     varchar(15)  not null,
   groupId     int          not null,
   groupName   varchar(30)  not null,
   groupType   int          not null,
   floatLic    int                  ,
   floatLicFTS int                  ,
   timestamp   int                  );
CREATE CLUSTER INDEX group_cache_ind2
  ON group_cache (groupId);
CREATE UNIQUE INDEX group_cache_ind
    ON group_cache (serverId, entryId);
INSERT INTO control VALUES (18, 1, 1, 2, 1, 1, 1, 1);
INSERT INTO group_cache VALUES (0, '0000000000000001', 1, 'Administrator', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0, '0000000000000002', 2, 'Customize', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0, '0000000000000003', 3, 'Submitter', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0, '0000000000000004', 4, 'Assignee', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0, '0000000000000005', 0, 'Public', 1, 0, 0, 0);
INSERT INTO group_cache VALUES (0, '0000000000000006', 5, 'Sub Administrator', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0, '0000000000000007', 7, 'Assignee Group', 2, 0, 0, 0);
INSERT INTO user_cache VALUES (0, '0000000000000001', 'Demo', '', NULL, NULL, NULL, 1, 1, 0, 0, NULL, '1;', NULL);

Oracle

The following includes a set of SQL commands that define the AR System data dictionary for Oracle databases. For an explanation of these commands, see the Oracle SQL Language 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(30) not null,
lastChanged varchar(30) not null,
coreVersion number(15,0) not null,
umFields number(15,0) not null,
umVuis 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) null,
helpText clob null,
changeDiary clob null,
objProp 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_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,
     f10          number(15,0)     null,
     f11          number(15,0)     null,
     f12          number(15,0)     null,
     f13          number(15,0)     null,
     f14          number(15,0)     null,
     f15          number(15,0)     null,
     f16          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,
CREATE TABLE schema_join
(schemaId number(15,0) 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 varchar(255) not 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 varchar(255) not 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(30) not null,
lastChanged varchar(30) not null,
datatype number(15,0) not null,
ofOption number(15,0) not null,
createMode number(15,0) not 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
  (schemaId number(15,0) not null,
   vuiId 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(30) not null,
   lastChanged varchar(30) not null,
   helpText clob null,
   changeDiary clob null);
CREATE UNIQUE INDEX vui_ind
  ON vui (schemaId, vuiId);
CREATE INDEX vui_schema_ind
  ON vui (schemaId);
CREATE TABLE field_dispprop
  (schemaId number(15,0) not null,
   fieldId number(15,0) null,
   listIndex number(15,0) not null,
   vuiId number(15,0) null,
   propShort varchar(255) null,
   propLong clob null,
   propLong clob null);
CREATE UNIQUE INDEX field_dispprop_ind
  ON field_dispprop (schemaId, fieldId, listIndex, vuiId);
CREATE TABLE field_int
  (schemaId 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 (schemaId, fieldId);
CREATE TABLE field_real
  (schemaId 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 (schemaId, fieldId);

CREATE TABLE field_diary
  (schemaId     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 (schemaId, fieldId);

CREATE TABLE field_char
  (schemaId     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,
attchType    number(15,0) not 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,
numColumns   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);
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(30) not null,
   lastChanged varchar(30) not null,
   refreshCode number(15,0) not null,
   menuType number(15,0) not null,
   helpText clob null,
   changeDiary clob null,
   objProp 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 varchar(255) null,
     queryLong clob null,
     keywordList clob null,
     parameterList clob null,
     externList clob null);
CREATE INDEX char_menu_qry_ind
    ON char_menu_query (charMenuId);
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 INDEX char_menu_file_ind
    ON char_menu_file (charMenuId);
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,
     labelIndex2 number(15,0) null,
     labelIndex3 number(15,0) null,
     labelIndex4 number(15,0) null,
CREATE INDEX char_menu_sql_ind
ON char_menu_sql (charMenuId);

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 INDEX char_menu_dd_ind
ON char_menu_dd (charMenuId);

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(30)  not null,
 lastChanged   varchar(30)  not null,
 numReferences number(15,0) not null,
 label         varchar(255)     null,
 description   varchar(2000)    null,
 helpText      clob             null,
 changeDiary   clob             null,
 objProp       clob             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(30)  not null,
     lastChanged varchar(30)  not null,
     wkConnType number(15,0) not null,
     fOrder    number(15,0) not null,
     opSet     number(15,0) not null,
     enable    number(15,0) not null,
     numActions number(15,0) not null,
     numElses  number(15,0) not null,
     queryShort varchar(255)     null,
     queryLong clob             null,
     changeDiary clob             null,
     helpText  clob             null,
     objProp   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,
CREATE TABLE filter_notify
(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);
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);
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);
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(30) not null,
  lastChanged varchar(30) 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,
queryShort varchar(255) null,
queryLong clob null,
helpText clob null,
changeDiary clob null,
objProp 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(30) not null,
   lastChanged varchar(30) not null,
   wkConnType number(15,0) not null,
   alOrder number(15,0) not null,
   executeMask number(15,0) not null,
   controlfieldId number(15,0) 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,
   queryShort varchar(255) null,
   queryLong clob null,
   helpText clob null,
   changeDiary clob null,
   objProp clob null);
CREATE UNIQUE INDEX actlink_ind
ON actlink (name);
CREATE UNIQUE INDEX actlink_id_ind
ON actlink (actlinkId);
CREATE TABLE actlink_group_ids
  (actlinkId   number(15,0) not null,
   groupId     number(15,0) not null);
CREATE INDEX actlink_group_ids_ind
  ON actlink_group_ids (actlinkId);
CREATE TABLE actlink_macro
  (actlinkId   number(15,0) not null,
   actionIndex number(15,0) not null,
   macroName   varchar(254) not null,
   shortText   varchar(255)   null,
   longText    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,
   actionIndex number(15,0) not null,
   fieldId     number(15,0) not null,
   assignShort varchar(255)   null,
   assignLong  clob             null,
   keywordList clob             null,
   parameterList clob           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_process_ind
  ON 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)     null);
CREATE INDEX actlink_schar_ind
    ON actlink_set_char (actlinkId);
CREATE TABLE actlink_dde
    (actlinkId   number(15,0) not null,
     actionIndex number(15,0) not null,
     serviceName varchar(64)  not null,
     topic       varchar(64)  not null,
     action      number(15,0) not null,
     path        varchar(255) not null,
     command     varchar(255) not null,
     item        clob             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,
     clsId          varchar(128) not null,
     isVisible      char         not null,
     actionShort    varchar(255)     null,
CREATE TABLE actlink_auto
(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);
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,
noMatchCtnu    char         null,
pollIntval     number(15,0) null,
sortlst        varchar(255) null,
queryshort     varchar(255) null,
querylong      clob         null,
msgType        number(15,0) null,
msgNum number(15,0) null,
msgText clob null,
msgPane char null,
reportstr clob null,
supresEptyLst 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);
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_wait
  (actlinkId number(15,0) not null,
   actionIndex number(15,0) not null,
   buttonTitle varchar(64) default 'Continue' null);

CREATE INDEX actlink_wait_ind
ON actlink_wait (actlinkId);

CREATE TABLE actlink_gotoaction
  (actlinkId 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 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(30) not null,
   password varchar(30) 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, password, clientIPAddr, clientPort);
CREATE TABLE alert_time
    (username     varchar(30)   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   long raw        null);
CREATE UNIQUE INDEX support_file_ind
    ON support_file (fileType, id, id2, fileId);
CREATE TABLE server_cache
    (serverId      number(15,0) not null,
     server        varchar(255) not null);
CREATE UNIQUE INDEX server_cache_ind
    ON server_cache (serverId);
CREATE TABLE user_cache
    (serverId       number(15,0) not null,
     entryId       varchar(15)  not null,
     userName      varchar(30)  not null,
     password      varchar(255)     null,
     authUserName  varchar(30)      null,
     authString    varchar(255)     null,
     licensePool   varchar(30)      null,
     email         varchar(255)     null,
     notifyMech    number(15,0)     null,
     licType       number(15,0)     null,
     licTypeFText  number(15,0)     null,
     licTypeReserv1 number(15,0)     null,
     timestamp     number(15,0)     null,
     validateKey   varchar(30)      null,
     shortGroup    varchar(255)     null,
     longGroup     varchar(4000)    null);
CREATE INDEX user_cache_ind2
    ON user_cache (userName);
CREATE UNIQUE INDEX user_cache_ind  
ON user_cache (serverId, entryId);
CREATE TABLE group_cache  
(serverId number(15,0) not null,
 entryId varchar(15)  not null,
 groupId number(15,0) not null,
 groupName varchar(30)  not null,
 groupType number(15,0) not null,
 floatLic number(15,0) null,
 floatLicFTS number(15,0) null,
 timestamp number(15,0) null);
CREATE INDEX group_cache_ind2  
ON group_cache (groupId);
CREATE UNIQUE INDEX group_cache_ind  
ON group_cache (serverId, entryId);
commit;
INSERT INTO control VALUES (18, 1, 2, 1, 1, 1, 1, 1);
INSERT INTO group_cache VALUES (0,  
'000000000000001', 1, 'Administrator', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0,  
'000000000000002', 2, 'Customize', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0,  
'000000000000003', 3, 'Submitter', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0,  
'000000000000004', 4, 'Assignee', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0,  
'000000000000005', 0, 'Public', 1, 0, 0, 0);
INSERT INTO group_cache VALUES (0,  
'000000000000006', 5, 'Sub Administrator', 2, 0, 0, 0);
INSERT INTO group_cache VALUES (0,  
'000000000000007', 7, 'Assignee Group', 2, 0, 0, 0);
INSERT INTO user_cache VALUES (0,  
'000000000000001', 'Demo', '', NULL, NULL, NULL, NULL, 1, 1, 0, 1, 1, '1;', NULL);
commit;
Sybase and Microsoft SQL

The following includes a set of SQL commands that define the AR System data dictionary. For an explanation of these commands, see the Sybase Commands Reference Manual or the Transaction SQL Reference Manual.

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,
   actlinkId int not null,
   adminExtId int not null,
   charMenuId 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(30) not null,
   lastChanged varchar(30) not null,
   coreVersion int not null,
   numFields int not null,
   numVuis int not null,
   defaultVui varchar(254) not null,
   nextId int not null,
   maxStatEnums int not null,
   nextFieldId int not null,
   upgrdVersion int null,
   changeDiary text null,
   helpText text null,
   objProp text null)
go
CREATE UNIQUE INDEX schema_ind
  ON arschema (name)
CREATE UNIQUE CLUSTERED INDEX schema_id_ind
  ON arschema (schemaId)

goto
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_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
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,
     options      int          null,
     queryShort   varchar(255)  null,
     queryLong    text          null)
```sql
CREATE UNIQUE INDEX schema_join_ind
ON schema_join (schemaId)

CREATE TABLE schema_view
(schemaId int not null,
tableName varchar(255) not null,
keyField varchar(254) not null,
queryShort varchar(255) null,
queryLong text null)

CREATE UNIQUE INDEX schema_view_ind
ON schema_view (schemaId)

CREATE TABLE schema_vendor
(schemaId int not null,
vendorName varchar(254) not null,
tableName varchar(255) not null)

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(30) not null,
lastChanged varchar(30) not null,
datatype int not null,
fOption int not null,
createMode int not null,
defaultValue varchar(255) null,
changeDiary text null,
helpText text null)
```
CREATE UNIQUE CLUSTERED INDEX field_ind
ON field (schemaId, fieldId)
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(30) not null,
lastChanged varchar(30) not null,
changeDiary text null,
helpText text null)

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

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)

CREATE UNIQUE INDEX field_dispprop_ind
ON field_dispprop (schemaId, fieldId, listIndex, vuiId)

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

CREATE UNIQUE CLUSTERED INDEX field_int_ind
    ON field_int (schemaId, fieldId)

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

CREATE UNIQUE CLUSTERED INDEX field_real_ind
    ON field_real (schemaId, fieldId)

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

CREATE UNIQUE CLUSTERED INDEX field_diary_ind
    ON field_diary (schemaId, fieldId)

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)

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,
attachType int not null)
go
CREATE UNIQUE CLUSTERED INDEX field_attach_ind
ON field_attach (schemaId, fieldId)
go
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)
go
CREATE UNIQUE CLUSTERED INDEX field_table_ind
ON field_table (schemaId, fieldId)
go
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 null)
go
CREATE UNIQUE CLUSTERED INDEX field_column_ind
ON field_column (schemaId, fieldId)
go
CREATE TABLE field_dec
(schemaId int not null,
fieldId int not null,
rangepLow varchar(64) null,
rangepHigh 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 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(30)  not null,
   lastChanged  varchar(30)  not null,
   refreshCode  int          not null,
   menuType     int          not null,
   changeDiary  text             null,
CREATE UNIQUE CLUSTERED 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(30)  not null,
     label        varchar(254) not null,
     childType    int          not null,
     value        varchar(255)     null)

CREATE CLUSTERED 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              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)

CREATE CLUSTERED INDEX char_menu_qry_ind
    ON char_menu_query (charMenuId)
CREATE TABLE char_menu_file
  (charMenuId int not null,
   path varchar(30) not null,
   fileLocation int not null,
   filename varchar(255) not null)

CREATE CLUSTERED 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,
   labelIndex2 int null,
   labelIndex3 int null,
   labelIndex4 int null,
   labelIndex5 int null,
   valueIndex int not null,
   sqlCmdShort varchar(255) null,
   sqlCmdLong text null,
   keywordList text null,
   parameterList text null,
   externList text null)

CREATE CLUSTERED 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 null,
arschema varchar(254) null,
hiddenToo int null)
go
CREATE CLUSTERED INDEX char_menu_dd_ind
ON char_menu_dd (charMenuId)
go
CREATE TABLE arcontainer
(name varchar(254) not null,
containerId int not null,
containerType int not null,
timestamp int not null,
owner varchar(30) not null,
lastChanged varchar(30) not null,
numReferences int not null,
label varchar(255) null,
description text null,
changeDiary text null,
helpText text null,
objProp text null)
go
CREATE UNIQUE CLUSTERED INDEX arctr_ind
ON arcontainer (name)
CREATE UNIQUE INDEX arctr_id_ind
ON arcontainer (containerId)
go
CREATE TABLE arctr_group_ids
(containerId int not null,
groupId int not null,
permission int not null)
go
CREATE CLUSTERED INDEX arctr_group_ind
ON arctr_group_ids (containerId)
go
CREATE TABLE arctr_subadmin
(containerId int not null,
groupId int not null)
go
CREATE CLUSTERED INDEX arctr_subadmin_ind
    ON arctr_subadmin (containerId)
go
CREATE TABLE cntnr_ownr_obj
    (containerId   int         not null,
     ownerObjType int         not null,
     ownerObjId   int         not null,
     objIndex     int         not null)
go
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)
go
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)
go
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(30) not null,
lastChanged varchar(30) 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,
queryShort varchar(255) null,
queryLong text null,
changeDiary text null,
helpText text null,
objProp 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,
fieldIdCode int not null,
subjectText varchar(255) null,
behavior int null,
go
CREATE CLUSTERED INDEX filter_notify_ind
    ON filter_notify (filterId)
go
CREATE TABLE filter_notify_ids
    (filterId    int          not null,
     actionIndex int          not null,
     fieldId     int          not null)
go
CREATE CLUSTERED INDEX filter_notify_ids_ind
    ON filter_notify_ids (filterId, actionIndex)
go
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)
go
CREATE CLUSTERED INDEX filter_message_ind
    ON filter_message (filterId)
go
CREATE TABLE filter_log
    (filterId    int          not null,
     actionIndex int          not null,
     logFile     varchar(255)     null)
go
CREATE CLUSTERED INDEX filter_log_ind
    ON filter_log (filterId)
go
CREATE TABLE filter_set
    (filterId    int         not null,
     actionIndex int         not null,
     fieldId     int         not null,
     assignShort varchar(255)    null,
     assignLong  text            null)
go
CREATE CLUSTERED INDEX filter_set_ind
    ON filter_set (filterId)
go
CREATE TABLE filter_process
    (filterId    int          not null,
     actionIndex int          not null,
     command     varchar(255) not null)
go
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)
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)
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 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)
go
create clustered index filter_exit_ind
    on filter_exit (filterId)
go
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(30) not null,
lastChanged varchar(30) not null,
wkConnType int not null,
numActions int not null,
numElses 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,
queryShort varchar(255) null,
queryLong text null,
changeDiary text null,
helpText text null,
objProp text null)

CREATE UNIQUE CLUSTERED INDEX escalation_ind
ON escalation (name)
CREATE UNIQUE INDEX escalation_id_ind
ON escalation (escalationId)
go
CREATE TABLE escal_mapping
  (schemaId     int          not null,
   objIndex     int          not null,
   escalationId int          not null)
go
CREATE UNIQUE INDEX escal_mapping_ind
  ON escal_mapping (schemaId, escalationId)
go
CREATE TABLE actlink
  (name         varchar(254) not null,
   actlinkId    int          not null,
   timestamp    int          not null,
   owner        varchar(30)  not null,
   lastChanged  varchar(30)  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,
   queryShort   varchar(255)     null,
   queryLong    text             null,
   changeDiary  text             null,
   helpText     text             null,
   objProp      text             null)
go
CREATE UNIQUE CLUSTERED INDEX actlink_ind
  ON actlink (name)
CREATE UNIQUE INDEX actlink_id_ind
  ON actlink (actlinkId)
go
CREATE TABLE actlink_group_ids
  (actlinkId   int         not null,
   groupId     int         not null)
go
CREATE CLUSTERED INDEX actlink_group_ids_ind
    ON actlink_group_ids (actlinkId)
go
CREATE TABLE actlink_macro
    (actlinkId int not null,
     actionIndex int not null,
     macroName varchar(254) not null,
     shortText varchar(255) null,
     longText text null)
go
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)
go
CREATE CLUSTERED INDEX actlink_set_ind
    ON actlink_set (actlinkId)
go
CREATE TABLE actlink_process
    (actlinkId int not null,
     actionIndex int not null,
CREATE TABLE actlink_process
(command     varchar(255) not null,
keywordList varchar(255)    null,
parameterList varchar(255)   null)

CREATE CLUSTERED INDEX actlink_process_ind
    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     text         not null,
msgPane     char  default '0' null)

CREATE CLUSTERED INDEX actlink_message_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)     null,
propShort   varchar(255)     null,
propLong    text             null,
focus       int              null,
accessOpt   int              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,
CREATE CLUSTERED INDEX actlink_dde_ind ON actlink_dde (actlinkId)
go
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)     null,
 actionLong     text             null,
 COMShort       varchar(255)     null,
 COMLong        text             null)
go
CREATE CLUSTERED INDEX actlink_auto_ind ON actlink_auto (actlinkId)
go
CREATE TABLE actlink_push
(actlinkId   int          not null,
 actionIndex int          not null,
 fieldId     int          not null,
 assignShort varchar(255)     null,
 assignLong  text             null)
go
CREATE CLUSTERED INDEX actlink_push_ind ON actlink_push (actlinkId)
go
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 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(255) null,
assignLong text null,
windowMode int null,
noMatchCtnu 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,
supresEptyLst char null,
targetLocation varchar(255) null)

CREATE CLUSTERED INDEX actlink_open_ind
ON actlink_open (actlinkId)

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

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,
     guideTableId    int          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)
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)
    go
CREATE TABLE alert_user
    (username      varchar(30)  not null,
     password      varchar(30)      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, password, clientIPAddr, clientPort)
go
CREATE TABLE alert_time
    (username      varchar(30)  not null,
     checkpointTime int         not null)
go
CREATE UNIQUE INDEX alert_time_ind
    ON alert_time (username)
go
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)
go
CREATE UNIQUE CLUSTERED INDEX support_file_ind
    ON support_file (fileType, id, id2, fileId)
go
CREATE TABLE server_cache
    (serverId   int          not null,
     server     varchar(255) not null)
go
CREATE UNIQUE CLUSTERED INDEX server_cache_ind
    ON server_cache (serverId)
go
CREATE TABLE user_cache
    (serverId       int          not null,
     entryId        varchar(15)  not null,
     userName       varchar(30)  not null,
     password       varchar(255)     null,
     authUserName   varchar(30)      null,
     authString     varchar(255)     null,
     licensePool    varchar(30)      null,
CREATE CLUSTERED INDEX user_cache_ind2
    ON user_cache (userName)
CREATE UNIQUE INDEX user_cache_ind
    ON user_cache (serverId, entryId)
go
CREATE TABLE group_cache
    (serverId    int          not null,
     entryId     varchar(15)  not null,
     groupId     int          not null,
     groupName   varchar(30)  not null,
     groupType   int          not null,
     floatLic    int              null,
     floatLicFTS int              null,
     timestamp   int              null)
go
CREATE CLUSTERED INDEX group_cache_ind2
    ON group_cache (groupId)
CREATE UNIQUE INDEX group_cache_ind
    ON group_cache (serverId, entryId)
go
INSERT INTO control VALUES (18, 1, 1, 2, 1, 1, 1, 1)
INSERT INTO group_cache VALUES (0,
    '000000000000001', 1, 'Administrator', 2, 0, 0, 0)
INSERT INTO group_cache VALUES (0,
    '000000000000002', 2, 'Customize', 2, 0, 0, 0)
INSERT INTO group_cache VALUES (0,
    '000000000000003', 3, 'Submitter', 2, 0, 0, 0)
INSERT INTO group_cache VALUES (0,
    '000000000000004', 4, 'Assignee', 2, 0, 0, 0)
INSERT INTO group_cache VALUES (0, '000000000000005', 0, 'Public', 1, 0, 0, 0)
INSERT INTO group_cache VALUES (0, '000000000000006', 5, 'Sub Administrator', 2, 0, 0, 0)
INSERT INTO group_cache VALUES (0, '000000000000007', 7, 'Assignee Group', 2, 0, 0, 0)
INSERT INTO user_cache VALUES (0, '000000000000001', 'Demo', ' ', NULL, NULL, NULL, NULL, 1, 1, 0, 0, 0, NULL, '1;', NULL)
go
Index

A
active links
database tables 26
table 22
AR Export file 55–57
AR System database user and password,
changing 13
attachment tables 33, 35

C
case sensitivity in databases
Microsoft SQL 17
Sybase 19
changing database user and password 13
character fields in databases
Informix 16
Microsoft SQL 17
connections, databases 16
containers
database tables 27
table 23
converting dates 48

D
data dictionary, AR System 20–28
data types
DB2 28
Informix 29
Microsoft SQL 29
Oracle 30
Sybase 31
database connections, Informix databases 16
database user and password, changing 13
databases
attachment tables 33
date conversion 48
DB2 68–93
field length 39, 44
fields 37, 38
forms 37
indexing 34
Informix 93–116
installing 12
main data table 31
Microsoft SQL 140–168
Oracle 116–139
Request ID and 51
rollback segments 42
SQL views 36
status history table 32
structure 12
Sybase 140–168
time conversion 48
view user interface (VUI) 24
date/time format, converting 48
DB2 database
  considerations 14
  data types 28
  date/time format 48
  field length 39
  field size limit 14
  Request ID and 51
  SQL commands 68–93
  user name and password 14
definitions, tables 20
diary fields in databases
  Informix 16
  Microsoft SQL 17
  Sybase 19
Direct SQL, external Informix databases 17
E
escalations
  database tables 26
table 22
export file 55
external Informix databases 17
F
field limits in databases
  DB2 14
  Informix 16
  Microsoft SQL 17
  Sybase 19
fields
  adding to database 37
  changing existing value 55
  changing length 39, 53
  database tables 25
  deleting from database 38
  join form views 25
  Oracle database and 44
  preserving existing values 54
  Request ID and 50
  table 21
filters
  database tables 26
table 22
forms
  database tables 24, 28
  databases 37
  performance tuning 37
  Request ID and 50
  table 21
I
indexing, tables 34
Informix database
  character string and diary field limits 16
  data types 29
  date/time format 48
  Direct SQL 17
  field length 39
  maximum database connections 16
  modulo operator 16
  Request ID and 51
  shared libraries 17
  SQL commands 93–116
  wildcards 16
international character sets, Microsoft SQL database 17
J
join forms
  main data view 31
  Sybase databases and 20
L
libraries, shared 17
M
main data table 31, 34
menus, database tables 25
Microsoft SQL database
  case sensitivity 17
  character string and diary field limits 17
  data types 29
  date/time format 49
  field length 40
  international character sets 17
  Request ID and 51
  SQL commands 140–168
  SQL OLEDB Provider 18
Microsoft SQL Server. See Microsoft SQL database
modulo operator, Informix database 16

N
next available Request ID 50

O
Oracle database
data types 30
date/time format 49
diary fields 44
field length 41
long character fields 44
Request ID and 51, 66
rollback segments 42
searches and 18
SQL commands 116–139

P
performance tuning, forms 37
prefixes
lengthening 64
shortening 59
primary key 34

R
Request ID field
changing length or prefix 53
changing value 55
number 50
preserving existing values 54
rollback segments, Oracle database 42

S
searches in databases
Informix 16
Microsoft SQL 17
Oracle 18
Sybase 19
shared libraries 17

SQL commands
DB2 68–93
Informix 93–116
lengthening prefix 64
Microsoft SQL 140–168
Oracle 116–139
shortening prefix 59
Sybase 140–168

SQL OLEDB Provider, Microsoft SQL database 18
SQL, views 36
status history table 32, 35
Sybase database
case sensitivity 19
character sets 20
character string and diary field limits 19
data types 31
date/time format 49
field length 42
joins 20
Request ID and 51
SQL commands 140–168

T
tables, database
active links 26
attachment details 33
containers 27
escalations 26
fields 25
filters 26
forms 24, 28
indexing 34
main 31
menus 25
status history 32
workflow mapping 27

technical essays
AR System dates, converting 48
next available Request ID 50
Request ID field 53
time conversion 48

Index ▶ 171
U
user name and password, DB2 database  14

V
view user interface (VUI)  24
views, SQL  36

W
wildcards in Informix  16
workflow mapping, database tables  27