the primary database. (New logs from the primary database can be retrieved and applied to the copy of the database at any time.) The standby copy of the database can then be used in place of the primary database if, for some reason, the primary database goes down.

Thus, if you wanted to initialize a split mirror copy of a database named SAMPLE and make it a backup image that can be used to restore the primary database, you could do so by executing a db2inidb command that looks like this:

```
db2inidb SAMPLE AS MIRROR
```

Managing Log files With a User Exit Program

When a DB2 9 database is configured to support roll-forward recovery, DB2 can call a special program, known as a user exit program, to move archived log files from the active log directory/device to a specific storage location when they are no longer needed to support transaction logging. So just what is a user exit program? Essentially, a user exit program is a user-supplied application that is invoked by DB2 whenever an open log file is closed, a database is closed (i.e., all connections to the database are terminated), or the ROLLFORWARD DATABASE command/API is executed (to perform a roll-forward recovery operation on a DB2 9 database).

Earlier, we saw that changes made to a database are recorded in log files and that when archival logging is used, log files are allocated and filled as they are needed. Each time a full log file is closed and a new log file is allocated, DB2 calls the user exit program (provided one exists and the database has been configured to take advantage of it) and passes it an ARCHIVE request along with the name of the oldest filled log file that currently resides in the active log directory. The user exit program then copies the log file specified to a designated archive log file storage area (the user exit program runs as a background process) and returns control to DB2, which then removes the log file just copied from the active log file directory—provided it is no longer needed. At times, a log file may exist in both the active log file directory and the archive log file storage area. That's because DB2 does not remove a log file from the active log file directory as long as it contains records that are associated with a running transaction. (If a long-running transaction needs to be rolled back, information stored in the log file may be needed and user exit programs are not invoked by rollback operations.) When the database is closed, all log files in the active log file directory are closed, and in some cases, the user exit program is called again (with an ARCHIVE request) to copy all remaining log files to the designated archive log file storage area.

When a roll-forward recovery operation is to be performed on a database whose archive log files reside somewhere other than the active log file directory, the log files must either be copied to the active log file directory so that they are available for the roll-forward recovery operation or the location where the archived log files reside must be specified in the OVERFLOW LOG PATH option of the ROLLFORWARD DATABASE command (which is the command that is used to initiate a roll-forward recovery operation). If you are using a user exit program and if the database's transaction log files are not stored on a raw device, the user exit program will automatically move transaction log files to the active log directory for you. Here's how it works. Each time a log file that does not reside in the active log file directory is needed, DB2 calls the user exit program and passes it a RETRIEVE request along with the name of the log file the roll-forward recovery operation is looking for. The user exit program then copies the specified log file to the active log file directory. (To see which log files are actually needed, you can execute the ROLLFORWARD DATABASE command with the QUERY STATUS option specified). Log records stored in the log file copied will then be replayed against the database; once all records stored in the log file have been processed, DB2 removes the copy from the active log file directory.

Because archived log files can be moved from the active log directory to a specific storage location automatically by assigning the value DISK or TSM to the *logarcmeth1* or *logarcmeth2* database configuration parameter, user exits are rarely necessary.

Constructing a User Exit Program

Basically, there are two approaches you can take to develop a DB2 9 user exit program—you can modify one of the sample user exit programs provided with DB2 or you can construct your own. A number of sample user exit programs are provided with DB2 9 that demonstrate how user exit programs that interact with different devices and software interfaces are to be coded. These programs can be used “as is” or they can be modified to meet your specific needs. (When used “as is” you must provide values for some constants that are used by the program and you must compile it to create an executable application.) Comments within these programs provide technical information on how they are constructed.

On the other hand, if you understand the basic requirements for constructing a user exit program, you can create your own from the ground up. (If you decide to go this route, you may find it beneficial to use one of the sample user exit programs provided as a model to aid you in your development efforts.) By taking this approach, you can construct a user exit program that has been customized to work closely with your server environment and any hardware/software interfaces necessary. You can also construct a single user exit routine that works with multiple databases.

Four sample user exit programs are provided with DB2 9 for Linux, UNIX, and Windows. They are named as follows:

db2uext2.ctsm	Uses the Tivoli Storage Manager (TSM) utility to archive and retrieve database log files.
db2uext.ctape	Designed to archive and retrieve database log files to/from tape media.
db2uexit.cdisk	Uses the system copy command (COPY or cp) to archive and retrieve database log files to/from disk media.
db2uexit.cxbsa	Uses the XBSA Draft 0.8 published by the X/Open group, to archive and retrieve database log files. (This sample user exit program is only applicable for DB2 for AIX.)

All of these sample user exit programs are written in C and they can be found in the /sqllib/samples/c subdirectory of the current instance directory. However, whereas the sample user exit programs provided are written in C, user exit programs can be written in a wide variety of programming languages.

Regardless of which programming language is used, all user exit programs must be structured such that they perform three distinct tasks:

1. First, they must accept and parse the input parameters that will be sent to them when they are called by DB2. The calling format used is:

```
db2uext2
    -OS[OperatingSystem]
    -RL[DB2ReleaseLevel]
    -RQ[Request]
    -DB[DatabaseName]
    -NN[NodeNumber]
    -LP[LogFilePath]
    -LN[LogFileName]
    -AP[TSMPassword]
    -SP[LogExtentStartPage]
    -LS[LogExtentSize]
```

where:

OperatingSystem	Identifies the platform that the DB2 instance is running on (Linux, AIX, Solaris, HP-UX, or Windows).
B2ReleaseLevel	Identifies the DB2 product release level of DB2 that called the user exit program.
Request	Identifies the action being requested—ARCHIVE or RETRIEVE.
DatabaseName	Identifies the database, by name, that the log file is associated with.
NodeNumber	Identifies the local node number.
LogFilePath	Specifies the fully qualified path that identifies where active log files are located.
LogFileName	Identifies the name of a log file to be archived or retrieved.
TSMPassword	Specifies the password needed by the Tivoli Storage Manager utility.
LogExtentStartPage	Identifies the starting page where log information needed can be found (if the log file is stored on a raw device).
LogExtentSize	Identifies the size, in 4K pages, of the requested log extent (if the log file is stored on a raw device).

2. Then, they must perform whatever processing is needed in order to manipulate data according to the action/request received. In some cases, this involves copying a log file from one location to another using the system copy command. In other cases, this involves communicating with a special hardware component using a vendor-supplied interface.
3. Finally, they must exit with an appropriate return code that DB2 knows how to process. In order for DB2 to react properly to any error encountered during user exit program execution, all user exit programs must be coded so that they always exit with one of the following return code values:

   - 0 - Success
   - 4 - Resource error encountered
   - 8 - Operator intervention required
   - 12 - Hardware error
   - 16 - Software error
   - 20 - Error with one or more parameters passed
   - 24 - User Exit program not found
   - 28 - I/O failure
   - 32 - Operator terminated

If DB2 receives a return code of 4 or 8, it waits five minutes before attempting to perform the operation again. If DB2 receives a return code of 12 or greater, the request is suspended for five minutes and any additional requests are ignored. After five minutes have elapsed, an attempt to process the request is made again—if successful, all pending requests are also processed; if unsuccessful and a return code of 8 or greater is returned, the five minute suspensions continue until the problem is corrected or the database is stopped and restarted.

Whether you decide to modify an existing sample user exit program or create one from the ground up, you must convert it to a format that can be executed by DB2 on the database server. This means that unless the user exit program is written in REXX, it must be compiled and linked to produce an executable program. On

Linux and UNIX platforms, the file containing the source code for the user exit program should be compiled and linked to produce an executable program named `db2uext2`; on Windows platforms, the file containing the source code for the user exit program should be compiled and linked to produce a file named `db2uext2`.exe.

Configuring a DB2 Database to Use a User Exit Program

Once the source code file for a user exit program has been compiled and linked to produce an executable program, two actions must be performed before it can actually be used: the executable form of the user exit program must be copied to a specific location and the *logarcmeth1* or *logarcmeth2* database configuration parameter for the appropriate database(s) must be assigned the value `USEREXIT`.

Because DB2 calls a user exit program on behalf of a database, and because a database can reside in a wide variety of locations, DB2 looks for a user exit program in one specific location. Therefore, it is your responsibility to ensure that the executable form of your user exit program resides in that location. On UNIX platforms, user exit programs should be stored in either the `/sqllib/bin` or the `/sqllib/adm` subdirectory of the current instance directory (which is where the other DB2 executable programs are stored). On Windows platforms, user exit programs must be stored in the `\sqllib\bin` subdirectory of the current instance directory (again, where the other DB2 executable programs are stored).

Storing an executable form of the user exit program in the appropriate location is just the first step. Whether or not a user exit program will actually be invoked on behalf of a database is determined by the value of that database's *logarcmeth1* or *logarcmeth2* database configuration parameter: if either parameter is set to `USEREXIT`, archival logging is performed and DB2 interacts with the user exit program as described earlier.

User Exit Program Considerations

When a user exit program is used to archive and retrieve transaction log files, the following items need to be taken into consideration:

- Only one user exit program can be called by a DB2 instance. Therefore, each program used must be designed to handle all actions/requests it may need to perform for each database within the instance.

- The log file size used has an impact on how well a user exit program executes; if a small log file size is used, the user exit routine will be called more often but may execute faster.
- In some cases, if a database is closed before a user exit program processing an archive request returns a positive response, another archive request will be sent as soon as the database is reopened. Thus a log file may be archived more than once.
- If two or more databases are using the same log storage device at the same time and that device uses changeable media (for example, a tape drive), and if one of the databases needs to perform a roll-forward recovery operation, a log file needed may not exist on the medium currently being used by the device.
- A user exit program does not guarantee that a database can be rolled forward to the point in time that a failure occurred; instead, it attempts to make the recovery window smaller. Should a disk containing online archive log files fail before the active log file is filled or before all log files are archived to a new location, those log files will be lost.
- User exit programs are not called as part of a rollback operation or when a database is restarted as part of crash recovery.
- If a user exit program receives a request to archive a log file that does not exist (because multiple requests to archive the file were made and the file has already been moved, copied, and deleted) or to retrieve a log file that does not exist (because it is located in another directory or the end of the logs has been reached), it should ignore the request and return a successful return code.
- A user exit program should allow for the existence of different log files that have the same name after a database recovery operation has been performed. In addition, a user exit program should be written to preserve both log files and to associate those log files with the correct database.

High Availability Disaster Recovery (HADR)

High availability disaster recovery (HADR) is a DB2 database replication feature that provides a high availability solution for both partial and complete site failures. HADR protects against data loss by replicating data changes from a source database, called the primary, to a target database, called the standby. In an HADR environment, applications can access the current primary database—synchronization with the standby database occurs by rolling forward transaction log data that is generated on the primary database and shipped to the standby database. And with HADR, you can choose the level of protection you want from potential loss of data by specifying one of three synchronization modes: synchronous, near synchronous, or asynchronous.

HADR is designed to minimize the impact to a database system when a partial or a complete site failure occurs. A partial site failure can be caused by a hardware, network, or software (DB2 or operating system) malfunction. Without HADR, a partial site failure requires restarting the server and the instance where one or more DB2 databases reside. The length of time it takes to restart the server and the instance is unpredictable. If the transaction load was heavy at the time of the partial site failure, it can take several minutes before a database is returned to a consistent state and made available for use. With HADR, the standby database can take over in seconds. Furthermore, you can redirect the clients that were using the original primary database to the standby database (which is now the new primary database) by using Automatic Client Reroute or retry logic in the applications that interact with the database. After the failed original primary server is repaired, it can rejoin the HADR pair as a standby database if both copies of the database can be made consistent. Once the original primary database is reintegrated into the HADR pair as the standby database, you can switch the roles so that the original primary database once again functions as the primary database. (This is known as failback operation).

A complete site failure can occur when a disaster, such as a fire, causes the entire site to be destroyed. Because HADR uses TCP/IP to communicate between a primary and a standby database, the databases can reside in two different locations. For example, your primary database might be located at your head office in one city, whereas your standby database is located at your sales office in another city. If a disaster occurs at the primary site, data availability is maintained by having the remote standby database take over as the primary database.

Requirements for HADR Environments

To achieve optimal performance with HADR, the system hosting the standby database should consist of the same hardware and software as the system where the primary database resides. If the system hosting the standby database has fewer resources than the system hosting the primary database, the standby database may not be able to keep up with the transaction load generated by the primary database. This can cause the standby database to fall behind or the performance of the primary database to suffer. More importantly, if a failover situation occurs, the new primary database may not have the resources needed to adequately service the client applications. And because buffer pool operations performed on the primary database are replayed on the standby database, it is important that the primary and standby database servers have the same amount of memory.

IBM recommends that you use similar host computers for the HADR primary and standby databases. (If possible, they should be from the same vendor and have the same architecture. This will ensure consistent performance on the two servers.) Furthermore, the operating system on the primary and standby database servers should be the same version, including patch level. You can violate this rule for a short time during a rolling upgrade, but use extreme caution when doing so. A TCP/IP interface must also be available between the HADR host machines, and a high-speed, high-capacity network should be used to connect the two.

The DB2 software installed on both the primary and the standby database server must have the same bit size (32 or 64), and the version of DB2 used for the primary and standby databases must be identical; for example, both must be either version 8 or version 9. During rolling upgrades, the modification level (for example, the fix pack level) of the database system for the standby database can be later than that of the primary database for a short while. However, you should not keep this configuration for an extended period of time. The primary and standby databases will not connect to each other if the modification level of the database system for the primary database is later than that of the standby database. Therefore, fix packs must always be applied to the standby database system first.

Both the primary and the standby database must be a single-partition database, and they both must have the same database name; however, they do not have to be stored on the same database path. The amount of storage space allocated for transaction log files should also be the same on both the primary and the standby database server; the use of raw devices for transaction logging is not supported. (Archival logging is performed only by the current primary database.)

Table space properties such as table space name, table space type (DMS, SMS, or Automatic Storage), table space page size, table space size, container path, container size, and container type (raw device, file, or directory) must be identical on the primary and standby databases. When you issue a table space statement such as CREATE TABLESPACE, ALTER TABLESPACE, or DROP TABLESPACE on the primary database, it is replayed on the standby database. Therefore, you must ensure that the table space containers involved with such statements exist on both systems before you issue the table space statement on the primary database. (If you create a table space on the primary database, and log replay fails on the standby database because the containers are not available, the primary database does not receive an error message stating that the log replay failed.) Automatic storage databases are fully supported, including replication of ALTER DATABASE statements. Similar to table space containers, the storage paths specified must exist on both the primary and the standby server.

Additionally, once an HADR environment has been established, the following restrictions apply:

- Clients cannot connect to the standby database.
- Self Tuning Memory Manager (STMM) can be run only on the current primary database.
- Backup operations cannot be performed on the standby database.
- Redirected restore is not supported. That is, HADR does not support redirecting table space containers. However, database directory and log directory changes are supported.
- Load operations with the COPY NO option specified are not supported.

Setting Up an HADR Environment

The process of setting up an HADR environment is fairly straightforward. After ensuring that the systems to be used as primary and secondary server are identical and that a TCP/IP connection exists between them, you simply perform the following tasks in the order shown:

1. Determine the host name, host IP address, and the service name or port number for both the primary and the secondary database server.

If a server has multiple network interfaces, ensure that the HADR host name or IP address maps to the intended interface. You will need to allocate separate HADR ports for each protected database—these cannot be the same as the ports that have been allocated to the instance. The host name can map to only one IP address.

2. Create the standby database by restoring a backup image or initializing a split mirror copy of the database that is to serve as the primary database.

   It is recommended that you do not issue the ROLLFORWARD DATABASE command on the standby database after the restore operation or split mirror initialization. The results of performing a roll-forward recovery operation might differ slightly from replaying the logs on the standby database using HADR. If the primary and standby databases are not identical when HADR is started, an error will occur.

   When setting up the standby database using the RESTORE DATABASE command, it is recommended that the REPLACE HISTORY FILE option be used; use of the following options must be avoided: TABLESPACE, INTO, REDIRECT, and WITHOUT ROLLING FORWARD.

   When setting up the standby database using the db2inidb utility, do not use the SNAPSHOT or MIRROR options. You can specify the RELOCATE USING option to change one or more of the following configuration attributes: instance name, log path, and database path. However, you must not change the database name or the table space container paths.

3. Set the HADR configuration parameters on both the primary and the standby databases.

   After the standby database has been created, but before HADR is started, the HADR configuration parameters shown in Table 4.2 need to be set.

Table 4.2 HADR-Specific Configuration Parameters

Parameter	Value Range / Default	Description
hadr_db_role	N/A	Read-only. Indicates the current role of the database, if it is part of a high availability disaster recovery (HADR) environment. Valid values are STANDARD, PRIMARY, or STANDBY.
hadr_local_host	Any valid character string Default: NULL	Specifies the local host for high availability disaster recovery (HADR) TCP communication. Either a host name or an IP address can be used.
hadr_local_svc	Any valid character string Default: NULL	Specifies the TCP service name or port number for which the local high availability disaster recovery (HADR) process accepts connections.
hadr_remote_ host	Any valid character string Default: NULL	Specifies the TCP/IP host name or IP address of the remote HADR node.
hadr_remote_ inst	Any valid character string Default: NULL	Specifies the instance name of the remote server. Administration tools, such as the Control Center, use this parameter to contact the remote server. HADR also checks whether a remote database requesting a connection belongs to the declared remote instance.
hadr_remote_svc	Any valid character string Default: NULL	Specifies the TCP service name or port number that will be used by the remote HADR node.
hadr_remote_ svc	SYNC, NEARSYNC, ASYNC Default: NEARSYNC	Specifies the synchronization mode to use for HADR. This determines how primary log writes are synchronized with the standby database when the systems are in peer state. Valid values for this configuration parameter are: ● SYNC (Synchronous) This mode provides the greatest protection against transaction loss, and using it results in the longest transaction response time among the three modes. In this mode, log writes are considered successful only when log records have been written to log files on the primary database and when the primary database has received acknowledgement from the standby database that the log records have also been written to log files on the standby database. The log data is guaranteed to be stored at both sites. ● NEARSYNC (Near Synchronous) Although this mode has a shorter transaction response time than synchronous mode, it also provides slightly less protection against transaction loss. In this mode, log writes are considered successful only when log records have been written to the log files on the primary database and when the primary database has received acknowledgement from the standby system that the log records have also been written to main memory on the standby system. Loss of data occurs only if both sites fail simultaneously and if the target site has not transferred all of the log data that it has received to nonvolatile storage.

Table 4.2 HADR-Specific Configuration Parameters

Parameter	Value Range / Default	Description
		● ASYNC (Asynchronous) This mode has the highest chance of transaction loss if the primary system fails. It also has the shortest transaction response time among the three modes. In this mode, log writes are considered successful only when log records have been written to the log files on the primary database and have been delivered to the TCP layer of the primary system's host machine. Because the primary system does not wait for acknowledgement from the standby system, transactions might be considered committed when they are still on their way to the standby system.
hadr_timeout	1–4,294,967,295 Default: 120	Specifies the time (in seconds) that the HADR process waits before considering a communication attempt to have failed. (The value assigned to this configuration parameter must be the same for both the primary and the standby database.)

4. Connect to the standby instance and start HADR on the standby database.

   HADR is started by executing the START HADR command. The basic syntax for this command is:

```
START HADR ON [DATABASE | DB] [DatabaseAlias]
<USER [UserName] <USING [Password]>>
AS [PRIMARY <BY FORCE> | SECONDARY]
```

where:

DatabaseAlias	Identifies the alias assigned to the database for which HADR is to be started.
DatabaseAlias	Identifies the alias assigned to the database for which HADR is to be started.
UserName	Identifies the name assigned to a specific user under whom HADR is to be started.
Password	Identifies the password that corresponds to the name of the user under whom HADR is to be started.

Thus, if you wanted to start HADR on a database named SAMPLE and indicate that it is to act as a standby database, you could do so by executing a START HADR command that looks something like this:

```
START HADR ON DATABASE sample AS STANDBY
```

5. Connect to the primary instance and start HADR on the primary database.

   In this case, you would execute a START HADR command that looks something like this:

   ```
   START HADR ON DATABASE sample AS PRIMARY
   ```

   You can also set up an HADR environment using the Set Up HADR Databases Wizard, which can be activated by selecting the High Availability Disaster Recovery action from the Databases menu found in the Control Center. Figure 4–16 shows the Control Center menu items that must be selected to activate the Set Up HADR Databases Wizard; Figure 4–17 shows how the first page of the Set Up HADR Databases Wizard might look immediately after it is activated.

Once an HADR environment has been established, the following operations will be replicated automatically in the standby database whenever they are performed on the primary database:

- Execution of Data Definition Language (DDL) statements (CREATE, ALTER, DROP)
- Execution of Data Manipulation Language (DML) statements (INSERT, UPDATE, DELETE)
- Buffer pool operations
- Table space operations (as well as storage-related operations performed on automatic storage databases)
- Online reorganization
- Offline reorganization
- Changes to metadata (system catalog information) for stored procedures and user-defined functions (UDFs)

HADR does not replicate stored procedure and UDF object and library files. If this type of replication is needed, you must physically create the files on identical paths on both the primary and standby databases. (If the standby database cannot find the

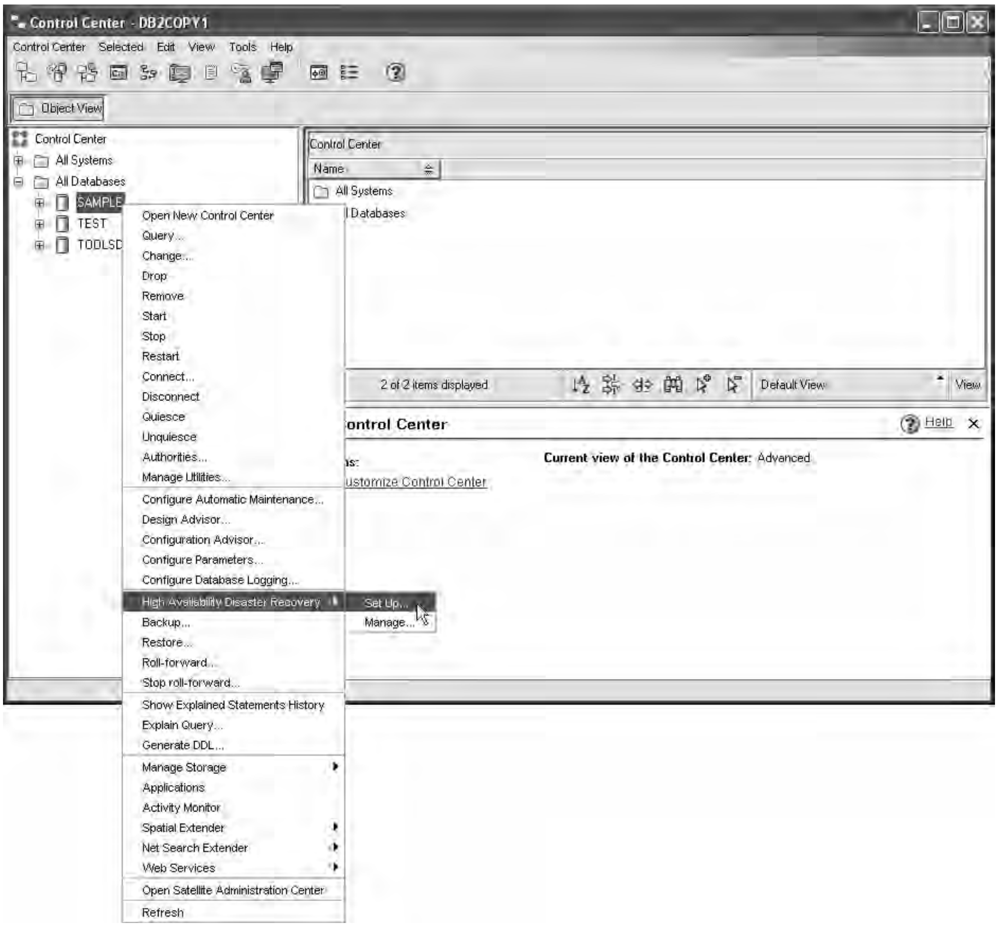


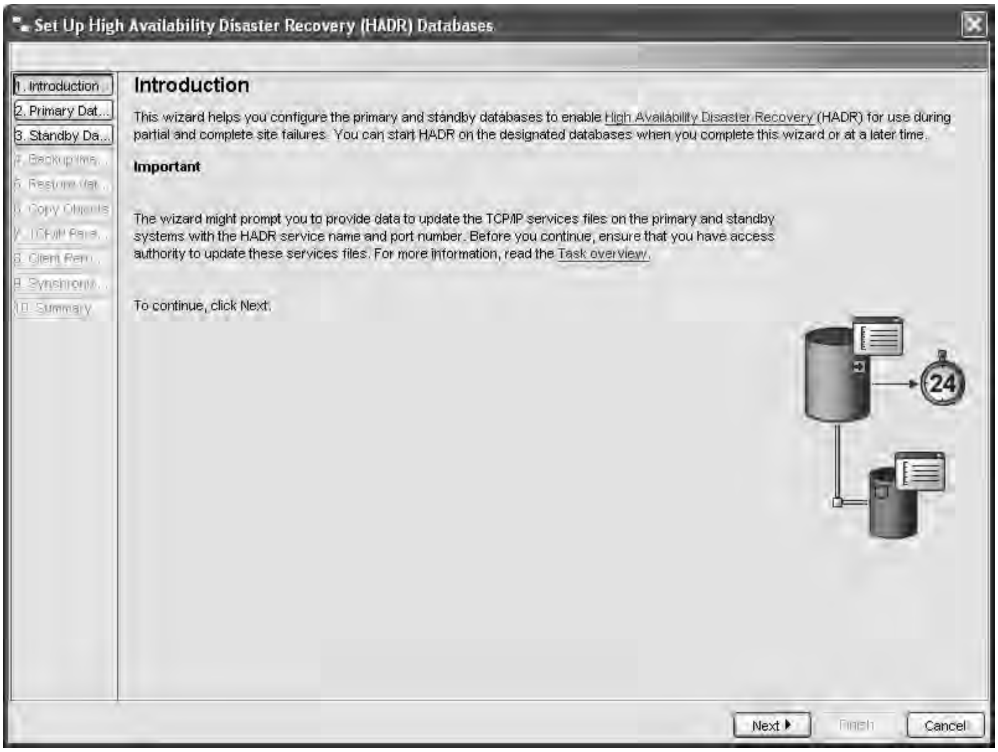
Figure 4–16: Invoking the Set Up HADR Databases Wizard from the Control Center.

referenced object or library file, the stored procedure or UDF invocation will fail on the standby database.)

Nonlogged operations, such as changes to database configuration parameters and to the recovery history file, are not replicated to the standby database.

Automatic Client Reroute and HADR

Automatic Client Reroute is a DB2 feature that allows client applications to recover from a loss of communication with the server so that the application can continue



Introduction

This wizard helps you configure the primary and standby databases to enable High Availability Disaster Recovery (HADR) for use during partial and complete site failures. You can start HADR on the designated databases when you complete this wizard or at a later time.

Important

The wizard might prompt you to provide data to update the TCP/IP services files on the primary and standby systems with the HADR service name and port number. Before you continue, ensure that you have access authority to update these services files. For more information, read the Task overview.

To continue, click Next.

Figure 4–17: The first page of the Set Up HADR Databases Wizard.

its work with minimal interruption. (If automatic client reroute is not enabled, client applications will receive an error message indicating that a connect attempt has failed due to a timeout and no further attempts will be made to establish a connection with the server.) However, rerouting is only possible when an alternate database location has been specified at the server and the TCP/IP protocol is used.

The automatic client reroute feature can be used with HADR to make client applications connect to the new primary database immediately after a takeover operation. In fact, if you set up HADR using the Set Up High Availability Disaster Recovery (HADR) Databases Wizard, automatic client reroute is enabled by default. If you set up HADR manually, you can enable the automatic client reroute feature by executing the UPDATE ALTERNATE SERVER FOR DATABASE command; automatic client reroute does not use the values stored in the *hadr_remote_host* and *hadr_remote_svc* database configuration parameters.

For example, suppose you have cataloged a database named SALES on a client workstation as being located at host named SVR1. Database SALES is the primary database in an HADR environment and its corresponding standby database, also named SALES, resides on a host named SVR2 and listens on port number 456. To enable automatic client reroute, you simply specify an alternate server for the SALES database stored on host SVR1 by executing the following command:

```
UPDATE ALTERNATE SERVER FOR DATABASE sales
USING HOSTNAME svr2 PORT 456
```

Once this command is executed, the client must connect to host SVR1 to obtain the alternate server information. Then, if a communication error occurs between the client and the SALES database at host SVR1, the client will first attempt to reconnect to the SALES database on host SVR1. If this fails, the client will then attempt to establish a connection with the standby SALES database located on host SVR2.

Load Operations and HADR

Load operations present a special problem for HADR. Because load operations are not recorded in a database's transaction log files, whether a load operation can be duplicated is dependent on whether a copy of the loaded data was saved as part of the load process (which is the case if the COPY YES option of the LOAD command is specified). If a load operation is performed on the primary database with the COPY YES option specified, the command will execute on the primary database, and the data will be replicated to the standby database—provided the load copy image created can be accessed by the standby database via the path or device provided with the LOAD command. If the standby database cannot access the load copy image, the table space in which the table is stored is marked invalid on the standby database, and the standby database will stop replaying log records that pertain to this table space. To ensure that the load operation can access the copy on the standby database, it is recommended that you use a shared location for the load copy image output file location specified with the COPY YES option. Alternatively, you can deactivate the standby database while the load operation is performed, perform the load operation on the primary database, place a copy of the load copy image produced in the standby path, and then activate the standby database.

If a load operation is executed on the primary database with the NONRECOVERABLE option specified, data will be loaded into the appropriate table in the primary database, the corresponding table on the standby database will be marked invalid, and the standby database will stop replaying log records that pertain to this table. You can reissue the LOAD command with the COPY YES and REPLACE options specified to restore the table on the standby database, or you can drop the table and recover the space.

Because executing a load operation with the COPY NO option specified is not supported by HADR, an attempt to perform such an operation is automatically converted to a load operation that behaves as if the NONRECOVERABLE option was specified. To prevent this behavior, you can set the DB2_LOAD_COPY_NO_OVERRIDE registry variable on the primary database to COPY YES, in which case all load operations performed will behave as if the COPY YES option were specified. When setting this variable, make sure that the device or directory specified on the primary database can be accessed by the standby database using the same path, device, or load library.

Load operations against primary databases in an HADR environment can have an impact on indexes as well as tables and table spaces. If a load operation is performed on the primary database with the COPY YES option specified, affected indexes will be replicated as follows:

- If the indexing mode is set to REBUILD, and the table being loaded has been assigned the LOG INDEX BUILD attribute, or if the table being loaded has been assigned the DEFAULT attribute, and the *logindexbuild* database configuration parameter on the primary database is set to ON, the primary database will include the rebuilt index object in the copy file so that the standby database can replicate the index object. If the index object on the standby database is marked invalid before the load operation is performed, it will become usable again after the load operation as a result of the index rebuild.
- If the indexing mode is set to INCREMENTAL and the table being loaded has been assigned the LOG INDEX BUILD attribute, or if the table being loaded has been assigned the NULL attribute and the *logindexbuild* database configuration parameter on the primary database is set to ON, the index object on the standby database is updated only if it is not marked invalid before the load operation. Otherwise, the index is marked invalid on the standby database.

IBM recommends you set the *logindexbuild* database configuration parameter to ON for HADR databases to ensure that complete information is logged for index creation, re creation, and reorganization. Although this means that index builds might take longer on the primary system and that more log space may be required, the indexes will be rebuilt on the standby system during HADR log replay and will be available when a failover takes place. If index operations on the primary system are not logged and a failover occurs, any invalid indexes that remain after the failover is complete will have to be rebuilt before they can be accessed—while indexes are being recreated, they cannot be accessed by any application.

Problem Determination Tools

Along with creating database backup images on a regular basis, database monitoring is a vital activity that, when performed, provides continuous feedback on the health of a database system. And because database monitoring is such an integral part of database administration, DB2 comes equipped with a built-in monitoring utility known as the Database System Monitor. Although the name "Database System Monitor" suggests that only one monitoring tool is available, in reality the Database System Monitor is composed of two distinct tools (a snapshot monitor and one or more event monitors) that can be used to capture and return system monitor information.

The snapshot monitor is designed to collect information about the state of a DB2 instance and the databases it controls at a specific point in time (i.e., at the time the snapshot is taken). Additionally, the snapshot monitor can be tailored to retrieve specific types of monitoring data (for example, it could be configured to collect just information about buffer pools). Snapshots can be taken by executing the GET SNAPSHOT command from the DB2 Command Line Processor (CLP) by using the snapshot administrative views and/or snapshot table functions, or by embedding the snapshot monitor APIs in a C or C++ application. Snapshots are useful for determining the status of a database system and, when taken at regular intervals, can provide valuable information that can be used to observe trends and identify potential problem areas.

Although the snapshot monitor provides a method for recording information about the state of database activity at a given point in time, an event monitor can be used to record information about database activity when an event or transition in database activity occurs. (Such a transition takes place when a connection is established, a deadlock cycle is encountered, an SQL statement is executed, and a transaction is started or stopped.) Thus, event monitors provide a way to collect monitor data when events or activities that cannot be monitored using the snapshot monitor occur. Unlike the snapshot monitor, which resides in the background and is always available, event monitors are special objects that must be created. Event monitors are created by executing the CREATE EVENT MONITOR SQL statement and event monitors are activated (or deactivated) by executing the SET EVENT MONITOR SQL statement. When an event monitor is activated (started), it sits quietly in the background and waits for one of the events it is associated with to occur. Immediately after an event being monitored takes place, the event monitor collects monitor data associated with the event that triggered it and writes all data collected to the event monitor's target location. Thus, the event itself controls when monitor data is collected—unlike with the snapshot monitor, no special steps are required to capture the monitor data.

Along with the database system monitor, several other tools are available to help a database administrator isolate and identify problems with a system, database, or application. Some of the more popular problem determination tools are the DB2 memory tracker utility and the DB2 Problem Determination tool.

The DB2 Memory Tracker

The DB2 memory tracker utility is used to produce a complete report of memory status for instances, databases, and agents. This utility provides the following information about memory pool allocation:

- Current size
- Maximum size (hard limit)
- Largest size (high water mark)
- Type (identifier indicating function for which memory will be used)
- Agent who allocated pool (only if the pool is private)

(This information is also available from the snapshot monitor.)

The DB2 memory tracker is invoked by executing the `db2mtrk` command. The syntax for this command is:

```
db2mtrk
<-i>
<-d>
<-p
<-m | -w>
<-r [Interval] <Count>>
<-v>
<-h>
```

where:

Interval	Identifies the number of seconds to wait between subsequent calls to the DB2 memory tracker.
Count	Identifies the number of times to repeat calls to the DB2 memory tracker.

All other options shown with this command are described in Table 4.2.

Table 4.2 db2mtrk Command Options

Option	Meaning
-i	Specifies that information about instance level memory is to be collected and displayed.
-d	Specifies that information about database level memory is to be collected and displayed.
-p	Specifies that information about private memory is to be collected and displayed.
-m	Specifies that maximum values for each memory pool is to be collected and displayed.
-w	Specifies that high water mark values for each memory pool are to be collected and displayed.
-v	Indicates that verbose output is to be returned.
-h	Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.

Thus, if you wanted to see how memory is utilized by the active databases on a system, and you wanted to capture and view this information every two minutes, you could do so by executing a `db2mtrk` command that looks something like this:

```
db2mtrk -d -r 120
```

Assuming a database named SAMPLE is active at the time the db2mtrk command is issued, the results produced should look something like this:

```
Tracking Memory on: 2008/05/21 at 14:00:38

Memory for database: SAMPLE

   utilh     pckcacheh catcacheh bph (1)   bph (S32K) bph (S16K) bph (S8K)
   64.0K     128.0K    64.0K     1.2M      704.0K     448.0K     320.0K

   bph (S4K) shsorth   lockh     dbh       other
   256.0K    0         320.0K    4.3M      128.0K
```

The DB2 Problem Determination Tool

The DB2 Problem Determination tool is used to obtain quick and immediate information from the DB2 database system memory sets without acquiring any latches. Two benefits to collecting information without latching include faster data retrieval and no competition for engine resources. However, because the DB2 Problem Determination tool works directly with memory, it is possible to retrieve information that is changing as it is being collected; hence, the data retrieved might not be completely accurate. A signal handler is used to prevent the DB2 Problem Determination tool from aborting abnormally when changing memory pointers are encountered. However, this can result in messages such as "Changing data structure forced command termination" appearing in the output produced. Nonetheless, this tool can be extremely helpful for problem determination.

The DB2 Problem Determination tool is invoked by executing the db2pd command. The basic syntax for this command is:

```
db2pd
<- version | -v >
<-inst>
<[-database | -db] [DatabaseName] ,...>
<-alldatabases | -alldbs>
<-full>
<-everything>
<-hadr [-db [DatabaseName] | -alldbs]>
<-utilities>
<-applications [-db [DatabaseName] | -alldbs]>
```

```
<-agents>
<-transactions [-db [DatabaseName] | -alldbs]>
<-bufferpools [-db [DatabaseName] | -alldbs]>
<-logs [-db [DatabaseName] | -alldbs]>
<-tablespaces [-db [DatabaseName] | -alldbs]>
<-dynamic [-db [DatabaseName] | -alldbs]>
<-static [-db [DatabaseName] | -alldbs]>
<-fcm>
<-memsets>
<-mempools>
<-memblocks>
<-dbmcfg>
<-dbcfg [-db [DatabaseName] | -alldbs]>
<-catalogcache [-db [DatabaseName] | -alldbs]>
<-tcbstats [-db [DatabaseName] | -alldbs]>
<-reorg [-db [DatabaseName] | -alldbs]>
<-recovery [-db [DatabaseName] | -alldbs]>
<-reopt [-db [DatabaseName] | -alldbs]>
<-osinfo>
<-storagepaths [-db [DatabaseName] | -alldbs]>
<-pages [-db [DatabaseName] | -alldbs]>
<-stack [all | [ProcessID]]>
<-repeat [Interval] <[Count]>>
<-command [CmdFileName]>
<-file [OutFileName]>
<-interactive>
<-h | -help>
```

where:

DatabaseName	Identifies, by name, the database with which the DB2 Problem Determination tool is to interact.
ProcessID	Identifies the process, by ID, for which a stack trace file is to be produced.
Interval	Identifies the number of seconds to wait between subsequent calls to the DB2 Problem Determination tool.
Count	Identifies the number of time to repeat calls to the DB2 Problem Determination tool.
CmdFileName	Identifies the name assigned to an ASCII format file that contains DB2 Problem Determination tool command options that are to be used.

OutFile	Identifies the name of the file to which information returned by the DB2 Problem Determination tool is to be written.

All other options shown with this command are described in Table 4.3.

Table 4.3 db2pd Command Options

Option	Meaning
-version | -v	Specifies that the current version and service level of the installed DB2 product is to be collected and displayed.
-inst	Specifies that all instance level information available is to be collected and displayed.
-alldatabases | -alldbs	Specifies that the utility is to attach to all memory sets of all available databases.
-full	Specifies that all output is to be expanded to its maximum length. (If this option is not specified, output is truncated to save space on the display.)
-everything	Specifies that all options are to be used and that information is to be collected and displayed for all databases on all database partition servers that are local to the server.
-hadr	Specifies that information about HADR is to be collected and displayed.
-utilities	Specifies that information about utilities is to be collected and displayed.
-applications	Specifies that information about applications is to be collected and displayed.
-agents	Specifies that information about agents is to be collected and displayed.
-transactions	Specifies that information about active transactions is to be collected and displayed.
-bufferpools	Specifies that information about buffer pools is to be collected and displayed.
-logs	Specifies that information about transaction log files is to be collected and displayed.
-locks	Specifies that information about locks is to be collected and displayed.
-tablespaces	Specifies that information about table spaces is to be collected and displayed.
-dynamic	Specifies that information about the execution of dynamic SQL statements is to be collected and displayed.
-static	Specifies that information about the execution of static SQL and packages is to be collected and displayed.
-fcm	Specifies that information about the fast communication manager is to be collected and displayed.

Table 4.3 db2pd Command Options (Continued)

Option	Meaning
-memsets	Specifies that information about memory sets is to be collected and displayed.
-mempools	Specifies that information about memory pools is to be collected and displayed.
-memblocks	Specifies that information about memory blocks is to be collected and displayed.
-dbmcfg	Specifies that information about current DB2 Database Manager configuration parameter settings is to be collected and displayed.
-dbcfg	Specifies that information about current database configuration parameter settings is to be collected and displayed.
-catalogcache	Specifies that information about the catalog cache is to be collected and displayed.
-tcbstats	Specifies that information about tables and indexes is to be collected and displayed.
-reorg	Specifies that information about table and data partition reorganization is to be collected and displayed.
-recovery	Specifies that information about recovery activity is to be collected and displayed.
-reopt	Specifies that information about cached SQL statements that were reoptimized using the REOPT ONCE option is to be collected and displayed.
-osinfo	Specifies that operating system information is to be collected and displayed.
-storagepaths	Specifies that information about the automatic storage paths defined for the database is to be collected and displayed.
-pages	Specifies that information about buffer pool pages is to be collected and displayed.
-stack	Specifies that stack trace information is to be collected and displayed.
-repeat	Specifies that the command is to be repeated after the specified number of seconds for the specified number of times.
-command	Specifies that db2pd commands that are stored in the specified file are to be executed.
-file	Specifies that all information collected is to be written to the specified file.
-interactive	Indicates that values specified for the DB2PDOPT environment variable are to be overridden when running the db2pd command.
-help | -h	Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.

So if you wanted to determine which indexes are actually being used for accessing data in a database named SAMPLE, you could do so by executing a db2pd command that looks something like this:

```
db2pd -tcbstats -db sample
```

On the other hand, if you wanted to obtain a list of applications that are currently connected to a database named SAMPLE, you could do so by executing the LIST APPLICATIONS command from the DB2 Command Line Processor, or by executing a db2pd command that looks something like this:

```
db2pd -applications -db sample
```

There is no minimum connection requirement for executing the db2pd command. However, if a database-level option is specified, that database must be active before the requested information can be returned.

Practice Questions

Question 1

Which of the following commands must be executed in order to configure a database named SAMPLE for infinite logging?

❍ A. UPDATE DB CFG FOR sample USING LOGARCHMETH1 LOGRETAIN LOGARCHMETH2 -1

❍ B. UPDATE DB CFG FOR sample USING LOGARCHMETH1 LOGRETAIN LOGARCHMETH2 INFINITE

❍ C. UPDATE DB CFG FOR sample USING LOGARCHMETH1 LOGRETAIN LOGSECOND -1

❍ D. UPDATE DB CFG FOR sample USING USING LOGARCHMETH1 LOGRETAIN LOGSECOND INFINITE

Question 2

A database administrator executes the following command:

```
GET DB CFG FOR sample
```

Assuming the database named SAMPLE has been configured to use archival logging, which of the following information can be obtained from output produced?

❍ A. The RID of the last record written to an active log file.

❍ B. The name of the first active log file.

❍ C. The transaction ID of the last transaction written to an active log file.

❍ D. The number of archived log files created.

Question 3

Which two of the following commands can be used to enable dual logging for a database named SAMPLE? (Select two)

❑ A. db2set DB2_NEWLOGPATH=D:\logs_copy

❑ B. db2set DB2_NEWLOGPATH=1

❑ C. UPDATE DB CFG FOR sample USING failarchpath D:\ logs_copy

❑ D. UPDATE DB CFG FOR sample USING mirrorlogpath D:\ logs_copy

❑ E. UPDATE DB CFG FOR sample USING logarchmeth2 MIRRORPATH: D:\ logs_copy

Question 4

A database administrator would like to configure a database named PAYROLL such that whenever a log file becomes full, DB2 will attempt to archive it automatically. Which of the following commands will achieve this objective?

❍ A. `UPDATE DB CFG FOR payroll USING LOGARCHMETH1 USEREXIT`
❍ B. `UPDATE DB CFG FOR payroll USING LOGARCHMETH1 LOGRETAIN`
❍ C. `UPDATE DB CFG FOR payroll USING LOGARCHMETH2 OFF`
❍ D. `UPDATE DB CFG FOR payroll USING LOGARCHMETH2 AUTOARCHIVE`

Question 5

If the following commands are executed:

```
db2set –g DB2_FORCE_APP_ON_MAX_LOG=FALSE;
UPDATE DB CFG FOR sample USING LOGFILSIZ 10000;
UPDATE DB CFG FOR sample USING MAX_LOG 10;
```

What will happen if an application with a long running transaction consumes more than 100,000 log pages?

❍ A. The transaction will be rolled back and the application will be forced to terminate.
❍ B. The transaction will be rolled back, but the application will be allowed to continue.
❍ C. The work performed by the current statement will be rolled back; the application will be allowed to commit or roll back the work performed by other statements in the transaction.
❍ D. The work performed by the current statement and all previous statements in the transaction will be rolled back; the application will be allowed to commit or rollback any remaining statements in the transaction.

Question 6

Which of the following commands will display only the media header information for an existing backup image?

❍ A. `db2 inspect –h`
❍ B. `db2 inspect –H`
❍ C. `db2ckbkp -h`
❍ D. `db2ckbkp -H`

Question 7

Which command will back up a database named ACCOUNTING to a Tivoli Storage Manager (TSM) server using 4 concurrent TSM client sessions and 8 buffers?

- ❍ A. BACKUP DATABASE accounting USE TSM WITH 4 SESSIONS OPEN 8 BUFFERS
- ❍ B. BACKUP DATABASE accounting USE TSM WITH 4 SESSIONS 8 BUFFERS
- ❍ C. BACKUP DATABASE accounting USE TSM OPEN 4 SESSIONS WITH 8 BUFFERS
- ❍ D. BACKUP DATABASE accounting USE TSM OPTIONS 4 SESSIONS WITH 8 BUFFERS

Question 8

A database named TEST was backed up to disk and a database administrator needs to determine if this backup was taken with using the INCLUDE LOGS option of the BACKUP DATABASE command.

How can the database administrator accomplish this?

- ❍ A. Examine the output from the command LIST HISTORY BACKUP ALL FOR test.
- ❍ B. Check the value of the BKUPLOGS database configuration parameter.
- ❍ C. Examine the header information stored in the backup image using the db2ckbkp command.
- ❍ E. Check the value of the DB2_BKUP_INCLLOGS registry variable.

Question 9

A database named SALES was created across 8 database partitions. Then, 12 tables were created and populated using the default table spaces provided – no additional table spaces were created.

Which of the following commands can be used to backup just the user tables on database partition 5 while the database remains online?

- ❍ A. db2_all "<<+5< db2 BACKUP DATABASE sales TABLESPACE (*) ONLINE TO /backup_dir"
- ❍ B. db2_all "<<+5< db2 BACKUP DATABASE sales TABLESPACE (userspace1) ONLINE TO /backup_dir"
- ❍ C. db2_all "<<-5< db2 BACKUP DATABASE sales TABLESPACE (*) ONLINE TO /backup_dir"
- ❍ D. db2_all "<<-5< db2 BACKUP DATABASE sales TABLESPACE (userspace1) ONLINE TO /backup_dir"

Question 10

Which of the following statements about incremental (cumulative) backups is NOT true?

❍ A. The predecessor of an incremental backup image is always the most recent successful full backup image of the same object (database or table space).

❍ B. Database recovery involves restoring the database using the most recent full backup image available and applying each incremental backup image produced since the last full backup, in the order in which they were created.

❍ C. Before an incremental backup image can be created, a full backup image must already exist.

❍ D. Along with updated data and index pages, each incremental backup image also contains all of the initial database metadata that is normally found in a full database backup image.

Question 11

Which of the following statements about the DB2 Check Backup utility (`db2ckbkp`) is NOT true?

❍ A. Only users who have read permission on backup image files are allowed to use the `db2ckbkp` utility.

❍ B. Multiple backup images can be analyzed with a single `db2ckbkp` command.

❍ C. Only users who have System Administrator authority are allowed to use the db2ckbkp utility.

❍ D. If a backup image was compressed when it was created, the library used to compress the data must be accessible to the `db2ckbkp` utility.

Question 12

A database named PRODDB has a weekly full backup taken on Sunday, and non-cumulative (delta) backups taken daily. A database crash occurs on Wednesday.

What is the best way to return the database to the state it was in when the last backup image was made?

❍ A. RESTORE DATABASE proddb TAKEN AT (Sunday);
RESTORE DATABASE proddb INCREMENTAL TAKEN AT (Tuesday);

❍ B. RESTORE DATABASE proddb TAKEN AT (Sunday);
RESTORE DATABASE proddb INCREMENTAL AUTO TAKEN AT (Tuesday);

❍ C. RESTORE DATABASE proddb TAKEN AT (Sunday);
RESTORE DATABASE proddb INCREMENTAL AUTO TAKEN AT (Monday);
RESTORE DATABASE proddb INCREMENTAL AUTO TAKEN AT (Tuesday);

❍ D. RESTORE DATABASE proddb INCREMENTAL AUTO TAKEN AT (Tuesday);

Question 13

While cleaning up files stored on an older AIX server, a database administrator found a shell script that contained the following commands:

```
db2 “RESTORE DATABASE sample FROM C:\backups TO D:\DB_DIR INTO sample_2
REDIRECT”

db2 “SET TABLESPACE CONTAINERS FOR 0 USING
(PATH ‘D:\DB_DIR\SYSTEM’)”

db2 “SET TABLESPACE CONTAINERS FOR 1 USING
(PATH ‘D:\DB_DIR\TEMP’)”

db2 “SET TABLESPACE CONTAINERS FOR 2 USING
(PATH ‘D:\DB_DIR\USER’)”

db2 “RESTORE DATABASE sample CONTINUE”
```

What was this file designed to perform?

❍ A. A multiple table space recovery operation.
❍ B. A reverted restore operation.
❍ C. A partial table space reconstruction operation.
❍ D. A redirected restore operation.

Question 14

A database named PAYROLL was created using automatic storage, with the following storage paths: /path1, /path2, /path3. Later, the database was backed up using the following command:

```
BACKUP DATABASE payroll TO /backups
```

If the PAYROLL database needs to be cloned, which of the following commands will re-create it in a new location while maintaining the original storage paths?

❍ A. RESTORE DB payroll FROM /backups INTO /path1, /path3, /path3 DBPATH /payroll2
❍ B. RECOVER DB payroll FROM /backups INTO /path1, /path3, /path3 DBPATH /payroll2
❍ C. RESTORE DB payroll FROM /backups INTO payroll2
❍ D. RECOVER DB payroll FROM /backups INTO payroll2

Question 15

A database administrator attempted to recover a database named SAMPLE by executing the following command:

```
RECOVER DATABASE sample TO END OF LOGS;
```

During the roll-forward phase of recovery, an error occurred. Once the problem is corrected, which of the following commands should be executed to force the recovery process to start over at the beginning?

❍ A. RECOVER DATABASE sample RESTART;
❍ B. RECOVER DATABASE sample TO END OF LOGS RESTART;
❍ C. RECOVER DATABASE sample RESTART AND CONTINUE;
❍ D. RECOVER DATABASE sample START AT RESTORE AND CONTINUE;

Question 16

Which of the following is NOT a requirement for an HADR environment?

❍ A. The operating system on the primary server and the standby server must be the same (including fix pack level).
❍ B. The database path on the primary server and the standby server must be the same.
❍ C. The DB2 software version and bit size (32 or 64) used on the primary server and the standby server must be the same.
❍ D. Table spaces and table space containers on the primary server and the standby server must be identical.

Question 17

Which of the following is NOT replicated in an HADR environment?

❍ A. Data Definition Language (DDL) statements
❍ B. Buffer pool operations
❍ C. Database configuration parameter changes
❍ D. System catalog information for stored procedures

Question 18

A system administrator wants to configure an HADR environment so that whenever an application issues a COMMIT, data in the log buffers will be written to the log files on the primary server and then be sent to the standby server and written to the log files there, before a return code will be sent to the application.

Which of the following commands can be used to accomplish this?

❍ A. UPDATE DB CFG FOR sales USING HADR_SYNCMODE SYNC

❍ B. UPDATE DB CFG FOR sales USING HADR_SYNCMODE NEARSYNC

❍ C. UPDATE DB CFG FOR sales USING HADR_SYNCMODE ASYNC

❍ D. UPDATE DB CFG FOR sales USING HADR_SYNCMODE INSYNC

Question 19

Two database servers have been enabled for synchronous HADR. If the HADR_TIMEOUT configuration parameter on the primary server is set to 30, which two of the following statements are true? (Select two)

❑ A. When HADR is started on the secondary server, HADR must be started, within 30 seconds, on the primary server.

❑ B. The HADR_TIMEOUT configuration parameter on the secondary server must also be set to 30.

❑ C. If the primary server does not receive acknowledgement of a log buffer within 30 seconds it stops processing all requests.

❑ D. The HADR_TIMEOUT configuration parameter on the secondary server must be assigned a value less than 30.

❑ E. If the primary server does not receive acknowledgement of a log buffer write within 30 seconds it drops out of peer mode.

Question 20

While setting up an HADR environment, a system administrator executed the following commands at the primary server:

```
db2set DB2COMM=TCPIP;
UPDATE ALTERNATE SERVER FOR DATABASE sales
    USING HOSTNAME svr2 PORT 456;
```

What will happen if a failover event occurs while a client application is connected to the SALES database on the primary server?

❍ A. The client application will receive an error message indicating that a connect attempt has failed due to a timeout and no further attempts will be made to establish a connection with the server.

❍ B. The client application will receive an error message indicating that a connect attempt has failed due to a timeout and the system administrator will have to manually switch over to the standby server SVR2 before further attempts are made to establish a connection with the server.

❍ C. The client application will automatically connect to the standby database on SVR2 and that database will become the primary database.

❍ D. The client application will automatically connect to the standby database on SVR2 and that database will become the primary database – provided the HADR_REMOTE_HOST and HADR_REMOTE_SVC database configuration parameters have been set appropriately.

Question 21

Which two commands can be used to list all applications that are currently connected to a database named PAYROLL? (Select two)

❑ A. db2pd –connections –db payroll

❑ B. db2pd –list applications –db payroll

❑ C. LIST APPLICATIONS FOR payroll

❑ D. GET APPLICATION SNAPSHOT FOR payroll

❑ E. SELECT * FROM TABLE(SNAP_GET_APPLICATIONS(‘payroll’,-1)) AS applications

Question 22

Which command will take 10 different snapshots of how memory is used by all active databases in the DB2 instance every 5 minutes?

❍ A. db2mtrk –d –i 5 10

❍ B. db2mtrk –d –r 5 –c 10

❍ C. db2mtrk –d –i 300 –c 10

❍ D. db2mtrk –d –r 300 10

Question 23

A db2mtrk command was used to produce the results shown below:

```
Tracking Memory on: 2008/05/21 at 14:00:38

Memory for database: SAMPLE

   utilh    pckcacheh catcacheh bph (1)  bph (S32K) bph (S16K) bph (S8K)
   64.0K    128.0K    64.0K     1.2M      704.0K     448.0K     320.0K

   bph (S4K) shsorth   lockh     dbh       other
   256.0K    0         320.0K    4.3M      128.0K
```

Which option was specified when the db2mtrk command was executed?

❍ A. –d

❍ B. –i

❍ C. –p

❍ D. –w

Question 24

When the following command was executed from the DB2 Command Line Processor:

```
db2pd –tablespaces –db SAMPLE
```

The following information was displayed:

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 25 days 08:32:31

Tablespace Configuration:
Address    Id    Type Content PageSz ExtentSz Auto Prefetch BufID
BufIDDisk FSC NumCntrs MaxStripe  LastConsecPg Name
0x7F91C230 0     DMS  Regular 8192   4        Yes  4        1     1
Off 1        0          3            SYSCATSPACE
0x7F91CA20 1     SMS  SysTmp  8192   32       Yes  32       1     1
On  1        0          31           TEMPSPACE1
0x7F91F1E0 2     DMS  Large   8192   32       Yes  32       1     1
Off 1        0          31           USERSPACE1
0x7F91F9D0 3     DMS  Large   8192   32       Yes  32       1     1
Off 1        0          31           IBMDB2SAMPLEREL
0x7EBAD220 4     DMS  Large   8192   32       Yes  32       1     1
Off 1        0          31           TBSP1
```

```
Tablespace Statistics:
Address    Id   TotalPgs   UsablePgs  UsedPgs    PndFreePgs FreePgs  HWM
State      MinRecTime NQuiescers
0x7F91C230 0     8192        8188     6852       0          1336     6852
0x00000000 0          0
0x7F91CA20 1     1           1          1        0          0          0
0x00000000 0          0
0x7F91F1E0 2     4096        4064       1760     0          2304     1760
0x00000000 0          0
0x7F91F9D0 3     4096        4064       608        0           3456   608
0x00000000 0          0
0x7EBAD220 4     200         160        156        0           0      156
0x00000000 0          0

Tablespace Autoresize Statistics:
Address    Id    AS  AR  InitSize    IncSize     IIP MaxSize LastResize
LRF
0x7F91C230 0     Yes Yes 33554432    -1          No  None         None
                     No
0x7F91CA20 1     Yes No  0           0           No  0            None
                     No
0x7F91F1E0 2     Yes Yes 33554432    -1          No  None         None
                     No
0x7F91F9D0 3     Yes Yes 33554432    -1          No  None         None
                     No
0x7EBAD220 4     No  No  0           0           No  0            None
                     No

Containers:
Address   TspId ContainNum Type  TotalPgs UseablePgs StripeSet Container
0x7F91C8B0 0     0          File    8192       8188       0
C:\DB2\NODE0000\SAMPLE\T0000000\C0000000.CAT
0x7F91D050 1     0          Path    1          1          0
C:\DB2\NODE0000\SAMPLE\T0000001\C0000000.TMP
0x7F91F860 2     0          File    4096       4064       0
C:\DB2\NODE0000\SAMPLE\T0000002\C0000000.LRG
0x7EF220D0 3     0          File    4096       4064       0
C:\DB2\NODE0000\SAMPLE\T0000003\C0000000.LRG
0x7EBADF30 4     0          File    200        160        0
C:\DB295_ESE\tbsp1.dat
```

Based on this information, which of the following is a valid statement?

❍ A. Table space SYSCATSPACE has been resized 5 times

❍ B. Table space IBMDB2SAMPLEREL is an SMS table space

❍ C. Table space USERSPACE1 cannot be accessed until it has been rolled forward

❍ D. Table space TBSP1 is about to run out of storage space

Answers

Question 1

The correct answer is **C**. If you are concerned about running out of log space, and you want to avoid allocating a large number of secondary log files, you can configure a database to use what is known as infinite logging. To enable infinite logging for a database, you simply set the *logsecond* database configuration parameter to -1. However, in order to use infinite logging, a database must be configured to use archival logging. This is done by assigning the value LOGRETAIN to the *logarcmeth1* or *logarcmeth2* database configuration parameter.

Question 2

The correct answer is **B**. The name of the current active log file (also referred to as the first active log file) can be obtained by executing the GET DATABASE CONFIGURATION command and examining the contents of the *loghead* database configuration parameter. (This will appear as "First active log file" in the list of information returned by the GET DATABASE CONFIGURATION command.). Other log-related information returned by the GET DATABASE CONFIGURATION command includes:

● Log buffer size (4KB)	*logbufsz*
● Log file size (4KB)	*logfilsiz*
● Number of primary log files	*logprimary*
● Number of secondary log files	*logsecond*
● Path to log files	*logpath*
● Overflow log path	*overflowlogpath*
● Mirror log path	*mirrorlogpath*
● Block log on disk full	*blk_log_dsk_ful*
● Percent max primary log space by transaction	*max_log*
● Num. of active log files for 1 active UOW	*num_log_span*
● First log archive method	*logarchmeth1*
● Options for logarchmeth1	*logarchopt1*
● Second log archive method	*logarchmeth2*
● Options for logarchmeth2	*logarchopt2*
● Failover log archive path	*failarchpath*
● Number of log archive retries on error	*numarchretry*
● Log archive retry Delay (secs)	*archretrydelay*

Question 3

The correct answers are **B** and **D**. To enable log file mirroring, you simply assign the fully qualified name of the mirror log location (path) to the *mirrorlogpath* database configuration parameter. Alternately, on UNIX systems, you can assign the value 1 to the DB2_NEWLOGPATH registry variable—in this case, the name of the mirror log location is generated by appending the character "2" to the current value of the *logpath* database configuration parameter. Ideally, the mirror log path used should refer a physical location (disk) that does not see a large amount of disk I/O and that is separate from the physical location used to store primary log files.

Question 4

The correct answer is **A**. When a DB2 9 database is configured to support roll-forward recovery, DB2 can call a special program, known as a user exit program, to move archived log files from the active log directory/device to a specific storage location when they are no longer needed to support transaction logging. So just what is a user exit program? Essentially, a user exit program is a user-supplied application that is invoked by DB2 whenever an open log file is closed, a database is closed (i.e., all connections to the database are terminated), or the ROLLFORWARD DATABASE command/API is executed (to perform a roll-forward recovery operation on a DB2 9 database).

Once the source code file for a user exit program has been compiled and linked to produce an executable program, two actions must be performed before it can actually be used: the executable form of the user exit program must be copied to a specific location and the *logarcmeth1* or *logarcmeth2* database configuration parameter for the appropriate database(s) must be assigned the value USEREXIT.

It is important to note that IBM is recommending users move away from user exit programs. An easier method for moving log files from the active log directory to an archive log storage location is to assign the value DISK:, followed by the desired location path to the *logarchmeth1* or *logarcmeth2* database configuration parameter.

Question 5

The correct answer is **C**. The *max_log* configuration parameter is used to control the maximum percentage of the total active log space available that any one transaction can consume. This configuration parameter prevents transactions from consuming all of the available active log space, thereby preventing other applications from running. By default, if a running transaction exceeds this threshold, the offending transaction is rolled back and the application that the transaction is running under will be forced to terminate. (This behavior