CENTERS for MEDICARE AND MEDICAID SERVICES

DATABASE ADMINISTRATION
DB2 STANDARDS

9/1/2020

http://www.cms.hhs.gov/DBAdmin/downloads/DB2StandardsandGuidelines.pdf

Overview

This document describes the physical database design and naming standards for DB2 z/OS databases at the Centers for Medicare and Medicaid Services CMS Data Center.

While developing the physical database design all standards must be followed, it is imperative that care and consideration be given to the standards noted throughout this DB2 Standards and Guidelines document with regard to database object naming conventions, appropriate database object usage, and required object parameter settings. Although exceptions to the standards may be permitted, any deviation from the standards must be reviewed with and approved by the Central DBA staff prior to implementing into development.

DB2 Database Design Standards

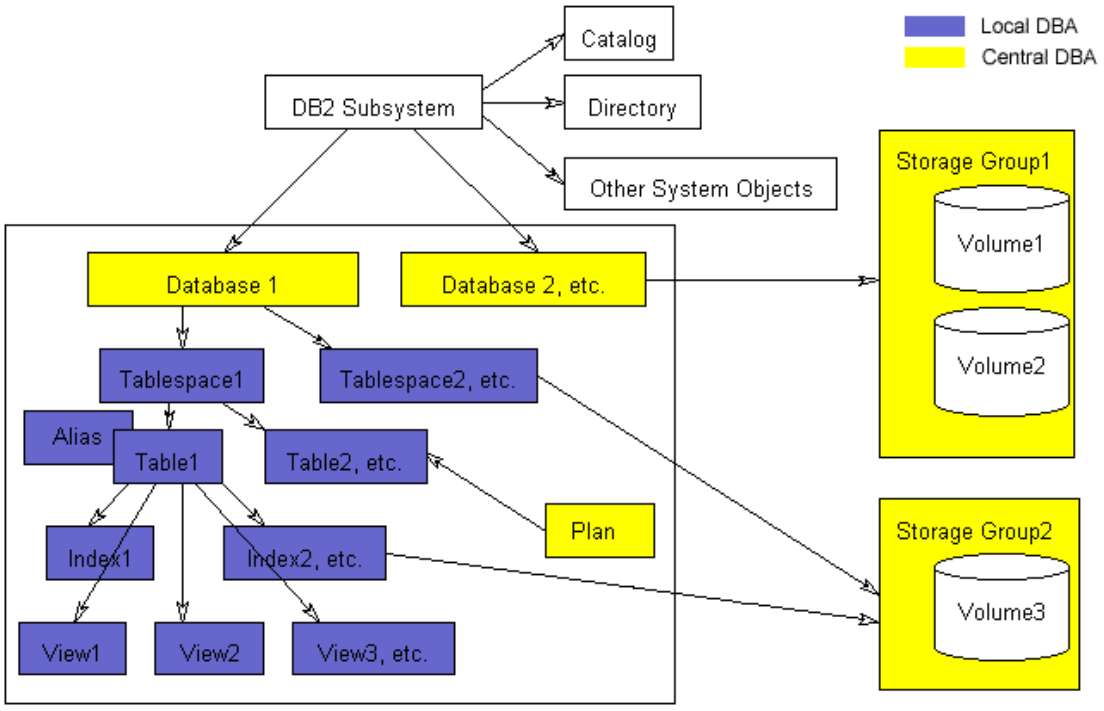
1.1 DB2 Design Overview

Creating and maintaining objects in DB2 is a shared responsibility. Systems Programmers are responsible for all system related objects including the catalogs, directories, buffer pools, and other DB2 system resources. Data Administrators create and maintain the logical models. In the development environments, Central DBAs create and maintain objects in preparation for subsequent use by the local DBAs. These objects include storage groups, databases, and application plans. In the test environments, Local DBAs are responsible for all database objects that will be utilized by their corresponding application. Tablespaces, tables, aliases, indexes, and views are all created and maintained by the local DBA. In integration, validation, and production Central DBAs maintain all database objects. Data is managed and maintained by the application team.

CA AllFusion ERwin should be used to translate the approved logical data model to a physical model and BMC Change Manager should be used to implement data definitions generated from ERwin to DB2. The CMS standards must be applied even when the physical data model is developed and implemented using case tools, such as ERwin and BMC, rather than using the tool's defaults.

Implementing a new database or a change to an existing database starts with a database service request to the central DBA and the central DA teams. The change requester completes the *CMS DB2 DB Service Request Form*. If required, the completed form is forwarded to the Central DA team for column naming approval. After Central DA approval, the form is sent to the Central DBA team for approval and implementation.

An overview and instructions for the *CMS DB2 DB Service Request and CMS DB2 DB Data Refresh* forms may be found in the *Overview of the CMS DB2 DB Service Request/Data Refresh Request Forms* document.



1.2 Databases

A Database in DB2 is an object which is created to logically group other DB2 objects such as tablespaces, tables, and indexes, within the DB2 catalog. Databases assist with the administration of DB2 objects by allowing for many operational functions to occur at the database level. DB2 commands such as -START DATABASE and -STOP DATABASE, for example, allow entire groups of DB2 objects to be started and stopped using one command.

See Standard Naming Convention.

1.2.1 Object Usage

STANDARD

Please refer to the roles and responsibilities documentation.

1.2.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining an application database. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
BUFFERPOOL bpname	Provide a valid bufferpool designation for tablespaces. The default value BP0 is used for the DB2 Catalog only. Refer to the bufferpool section of the doc.
STOGROUP STOGRPT1	Specify a valid storage group. STOGRPT1 is the standard application storage group in all subsystems. The storage group specified will indicate the default storage group that will be assigned to any tablespace or indexspace in the database only when that object was created without a storage group specification. Do not specify the default DB2 storage group (SYSDEFLT). Unauthorized tablespaces residing in the SYSDEFAULT storage group will be dropped without warning
INDEXBP bpname	Provide a valid bufferpool designation for indexes. The default value BP0 is used for the DB2 Catalog only. Refer to the bufferpool section of the doc.
CCSID encoding scheme	Specify the default encoding scheme, typically this will be CCSID EBCDIC.

1.3 Tablespaces

A tablespace is the DB2 object that holds data for one or more DB2 tables. It is a physical object that is managed in DB2.
See Standard Naming Convention.

1.3.1 Object Usage

STANDARD

- Assign one table per tablespace.
- Use UTS tablespaces for new tablespaces
- Define as PBR for all tables with a partitioning key unless they meet PBG criteria below
- Define as PBG-
    - for small tables that will always be contained in a single 4GB or less partition (growth past 4GB will require alter to PBR if an appropriate key can be determined)

    - or where no partitioning key can be determined.
    - CDBA concurrence must be obtained for PBG usage
- Create tablespaces explicitly using the CREATE TABLESPACE command rather than implicitly by creating a table without specifying a tablespace name.
- Use DB2 storage groups to manage storage allocation for all application tablespaces.
- Specify CLOSE YES for all tablespaces.
- Specify LOCKSIZE PAGE or ANY for all tablespaces that are not identified as 'read-only'. LOCKSIZE PAGE is recommended and should be used for new databases. An exception to this standard is when it is determined that all updates to the table are performed in batch by only by single threaded jobs, where LOCKSIZE TABLE is recommended. Due to the additional overhead associated with maintaining row-level locks, use of LOCKSIZE ROW is permitted only when a high level of application concurrency is necessary. Additionally, the Central DBA staff must review and approve the use of any LOCKSIZE option other than LOCKSIZE PAGE or LOCKSIZE ANY.
- Only specify COMPRESS YES when it has been determined that significant storage savings can be achieved by doing so. Typically, tablespaces containing primarily character data will benefit more than those containing mainly numeric data. Since data is compressed horizontally, tablespaces with longer average row lengths are also good candidates for compression. When determining the appropriateness of using DB2 compression, it is advisable to weigh the associated CPU overhead (especially when modifying data) against the savings in storage utilization as well as elapsed time for long running queries.
- For read-only tables, define tablespaces with PCTFREE, FREEPAGE of zero (0).
- Specify PRIQTY and SECQTY quantities that fall on a track or cylinder boundary. On a 3390 device a track equates to 48K and a cylinder equates to 720K. For tables over 100 cylinders always specify a PRIQTY and SECQTY to a number that is a multiple of 720. Be sure that PRIQTY and SECQTY allocations are not less than the size of one segment. For example, if SEGSIZE is 64, both PRIQTY and SECQTY should be no less than 256K (64 pages * 4K/page). Sliding scale allocation secondary, SECQTY=-1, is preferred, PRIQTY=-1 is useful in dev.
- Define SEGSIZE as an even multiple of four (4), where the number chosen is closest to the actual number of pages on a table stored in the tablespace. Specify SEGSIZE 64 for all tablespaces consisting of greater than 64 pages of data.
- For single partition tables use UTS PBR with MINVALUE/MAXVALUE in the Create Table partition range when the range is not predetermined.
- DBA review confirming that there is no appropriate partitioning key is required before defining Partition by Growth.
- Use multiple partition UTS PBR tablespaces when it is anticipated that the number of rows in tablespace will exceed one million rows, two gigabytes of storage or when it is determined that partitioning the tablespace will yield performance benefit (i.e. parallel query processing or independent partition utility processing).

Index Partitioned Tablespace

- Migrate existing index controlled partitioned tablespaces to UTS PBR when table changes are required by application development.

1.3.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining an application tablespace. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
IN database name	Specified database must be a valid application database. Application tablespaces must not be defined in the DB2 default database. **Unauthorized tablespaces residing in the DB2 default database, DSNDB04, will be dropped without warning.**
SEGSIZE	Specify the size of each segment of pages within the tablespace. For most tables, this value should be set to 64.
DSSIZE	Indicates the maximum data set size for each partition. A value in gigabytes that indicates the maximum size for each partition or, for LOB table spaces each data set.
NUMPARTS /MAXPARTITIONS	Specify the number of partitions (datasets) that will comprise the entire tablespace. MAXPARTITIONS requires DBA approval.
USING STOGROUP STOGRPT1	Indicate the DB2 storage group on which the application tablespaces will reside. STOGRPT1 is the standard application storage group. Do not specify the default DB2 storage group (SYSDEFLT). **Unauthorized tablespaces residing in the SYSDEFAULT storage group will be dropped without warning.**
PRIQTY	Primary quantity is specified in units of 1K bytes. Specify a primary quantity that will accommodate all of the data in the tablespace or partition. PRIQTY=-1 is allowed.
SECQTY	Secondary quantity is specified in units of 1K bytes. Specify a secondary quantity that is consistent with the anticipated growth of the table. The value specified should be large enough to prevent the tablespace from spanning more than three extents prior to the next scheduled REORG. SECQTY=-1 is preferred.

Parameter	Instructions
FREEPAGE	Indicate the frequency in which DB2 should reserve a page of free space on the tablespace or partition when it is reorganized or when data is initially loaded. Note: The maximum value for segmented tables is 1 less than the value specified for SEGSIZE.
PCTFREE	Indicate what percentage of each page on the tablespace or partition should be remain unused when it is reorganized or when data is initially loaded.
COMPRESS	Indicate whether DB2 should store data in the tablespace or partition in a compressed format
BUFFERPOOL bpname	Provide a valid bufferpool designation. Do not specify the default bufferpool of BP0, consult the bufferpool standards or the Central DBA staff.
LOCKSIZE	Specify the size of lock DB2 will acquire when data on the tablespace is accessed (see Object Usage standard above).
CLOSE	Indicate whether DB2 should close the corresponding VSAM dataset when no activity on tablespace is detected (see Object Usage standard above).

1.4 Tables

A table in DB2 is a named collection of columns and rows. The data represented in these rows can be accessed using SQL data manipulation language (select, insert, update, and delete). Generally, it is a table in DB2 that applications and end-users access to retrieve and manage business data.

See Standard Naming Convention.

1.4.1 Object Usage

STANDARD

- Assign one table per tablespace. Table and Clone are treated as one table.
- Default tablespaces and databases must not be used. Tables must be created in a tablespace which was explicitly created for the corresponding application.
- In conformance with relational theory, all rows of a table should be uniquely identified by a column or set of columns to avoid the advent of duplicate rows. It is therefore required that every table defined to DB2 includes a primary key. If a table does not have a viable group of columns that can be defined as a primary key, consult the Central DBA staff for direction.

1.4.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining an application table. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
(column definition, ...)	List column specifications for each column that is to be defined to the table (see Additional Table Definition Parameters below for more details).
IN database.tablespace	Specify the fully qualified tablespace name where data for the table should be stored. **Do not** reference the DB2 default database. **Unauthorized tablespaces residing in the DB2 Default database, DSNDB04, will be dropped without warning.**
PARTITION BY RANGE(...)	Required to create standard UTS PBR tablespace.

1.5 Columns

Columns in a DB2 table contain data that was loaded, inserted or updated by some process. All columns have a corresponding datatype to indicate the format of the data within. In addition to datatypes, other edits can be associated with columns in DB2 in order to enforce defined business rule. These rules include default values, check constraints, unique constraints, and referential integrity (foreign keys).

See Standard Naming Convention.

1.5.1 Object Usage

STANDARD

- Nulls should only be used when there is a need to know the difference between the absence of a value versus a default value.
- Nulls are used for missing dates and amounts. Character columns will generally be, Not Null with Default, to use the system default of blanks. instead of null for missing values. This saves the space and additional variable required by a null indicator. IE, Y/N/blank or A/B/C/blank, rather then Y/N/null. Nulls may be used for character columns to support RI such as on delete set null, or were blank is a valid non missing value.

- The Central DA team should be consulted for a list of standard domains and column definitions.
- Columns that represent the same data but are stored on different tables; must have the same name, datatype and length specification. An example of this is where tables are related referentially. If on a PROVIDER table, for example, the primary key is defined as PROV_NUM CHAR(08), then dependent tables must include PROV_NUM CHAR(08) as a foreign key column.
- Columns that contain data which is determined to have the same domain must be defined using identical datatype and length specifications. For example, columns LAST_CHG_USER_ID and CASE_ADMN_USER_ID serve different business purposes, however both should be defined as CHAR (08) in DB2. This standard applies to the entire enterprise and should not be enforced solely at an application level.
- All columns containing date information must be defined using the DATE/TIMESTAMP data types.
- All columns containing time information must be defined using the TIME/TIMESTAMP data types.
- Specify a datatype that most represents the data the column will contain. For numeric data, use one of the supported numeric data types taking into account the minimum and maximum value limits as well as storage requirements for each.
- For character data which may exceed 30 characters in length, consider use of the VARCHAR datatype. This could provide substantial savings in storage requirements. When weighing the benefits of defining a column to be variable in length, consider the average length of data that will be stored in this column. If the average length is less than 80% of the total column width, a variable length column may be appropriate. If data compression is used on the tablespace the central DBA should be contacted to determine if VARCHAR, should still be used.
- Consider sequence of the column definitions to improve database performance. Use the following as a guideline for sequencing columns on DB2 tables.
- Primary Key columns (for reference purposes only)
- Frequently read columns
- Infrequently read columns
- Infrequently updated columns
- Variable length columns
- Frequently updated columns
- For columns defined as DECIMAL be sure to use an odd number as the precision specification to ensure efficient storage utilization. Precision represents the entire length of the column, so in the definition DECIMAL (9,2), the precision is 9.
- For columns with whole numbers SMALLINT and INTEGER should be used.

1.5.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining columns in an application table. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
Column name	Provide a name for the column which conforms to DB2 Standard Naming Conventions.
Datatype specification	Provide a datatype specification consistent with the characteristics of the data the column will hold.
NOT NULL/ NOT NULL WITH DEFAULT [default value]	Indicate that a value or default value is required for the column. Under some circumstances, this clause may not be required (see Object Usage Standard above).

For additional information see IBM's Reference manuals.

1.6 Referential Constraints (Foreign Keys)

A referential constraint is a rule that defines the relationship between two tables. It is implemented by creating a foreign key on a dependent table that relates to the primary key (or unique constraint) of a parent table.

See Standard Naming Convention.

1.6.1 Object Usage

STANDARD

- Names for all foreign key constraints must be explicitly defined using data definition language (DDL). Do not allow DB2 to generate default names.
- DDL syntax allows for foreign key constraints to be defined along side of the corresponding column definition, as a separate clause at the end of the table definition, or in an ALTER table statement. Each of these methods are acceptable at CMS provided all of the required parameters noted below are included in the definition.
- Avoid establishing referential constraints for read-only tables.
- Do not define referential constraints on tables residing in multi-table tablespaces where the tables in the tablespace are not part of the same referential structure.
- When possible, limit the number of levels in a referential structure (all tables which have a relationship to one another) to three.
- Use column constraints instead of referential constraints to a code table for small groups of unchanging codes, such as sex, or y/n values and reason codes.

1.6.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining a foreign key constraint in an application table. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
constraint name	Specify a name for the foreign key constraint based on the see Standard Naming Convention.
(column name...)	Indicate the column(s) that make up the foreign key constraint. These columns must correspond to a primary key or unique constraint in the parent table.
REFERENCES table name	Specify the name of the table to which the foreign key constraint refers. If the foreign key constraint is based on a unique constraint of a parent table, also provide the column names that make up the corresponding unique constraint.
ON DELETE delete rule	Specify the appropriate delete rule which should be applied whenever an attempt is made to delete a corresponding parent row. Valid values include RESTRICT, NO ACTION, SET NULL, and CASCADE. (Warning: Use of the CASCADE delete rule can result in the mass deletion of numerous rows from dependent tables when one row is deleted from the corresponding parent. No response is returned to the deleting application indicating the mass delete occurred. Therefore, strong consideration should be given as to the appropriateness of implementing this rule in the physical design.)

For additional information see IBM's Reference manuals.

1.7 Table Check Constraints

A check constraint, (also known as a table or column constraint), is a rule defined to a table which dictates how edits will be applied against data that is either inserted or updated. The constraint rule is implemented using data definition language (DDL) and can apply to a specific column or multiple columns on a table. Constraint rules are always checked against one row of data at any point in time.

Check Constraints are preferred to RI on code tables for performance. Static codes with under a dozen values are more efficient using check constraints for enforcement, with descriptions kept locally in UI code rather accessing a code table with a select.

See Standard Naming Convention.

1.7.1 Object Usage

STANDARD

- Names for all check constraints must be explicitly defined using data definition language (DDL). Do not allow DB2 to generate default names.
- DDL syntax allows for check constraints to be defined along side of the corresponding column definition, as a separate clause at the end of the table definition, or in an ALTER table statement. Each of these methods are acceptable at CMS provided all of the required parameters noted below are included in the definition.
- Several restrictions apply when defining check constraints on a table or column. Refer to the DB2 SQL Reference manual for more details.
- Use caution when defining check constraints. Although DB2 will verify the syntax of SQL commands, it will not verify the meaning. It is possible to implement a check constraint that conflicts with other rules defined for the table. For example, the syntax listed below would be accepted by DB2 even though the column could never be inserted with a default value of 5.

```
CREATE table XYZ_TAB_A
(COL1_CD SMALLINT NOT NULL WITH DEFAULT 5
CONSTRAINT XYZ013_COL1_CD
CHECK (COL_1 BETWEEN 10 AND 20))
```

- Verify that check constraint rules do not conflict with any defined referential rules. For example, if a foreign key is defined as ON DELETE SET NULL and a check constraint is defined on the foreign key column as a rule that would prohibit nulls, DB2 will allow both rules to exist. However, any attempt to delete a parent row will fail due to the check constraint on the dependent table.
- Code tables with the ‘Not Enforced’ Foreign key parameter, may be used along with check constraints when there are many values which require descriptions, or when multiple application environments require consistent descriptions. This identifies the RI for the code table to Erwin while retaining the efficiency of using a check constraint.

1.7.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining a table check constraint in an application table. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
CONSTRAINT constraint name	Specify a name for the check constraint based on the Standard Naming Convention.
CHECK (condition)	Indicate the business rule using valid SQL syntax. This rule will look similar to conditions expressed in an SQL WHERE clause.

For additional information see IBM's Reference manuals.

1.8 Unique Constraints

A unique constraint is a rule defined to a table which indicates that any occurrence (row) in the table contains a distinct value in the column or combination of columns that make up the constraint. The constraint rule is implemented using data definition language (DDL) and can apply to a specific column or multiple columns on a table. A unique constraint is similar to a primary key where both implement entity integrity rules which state that each row in a table is uniquely identified by a non-null value or set of values. The major difference between primary keys and unique constraints is the fact that although only one primary key can be defined for one table, multiple unique constraints can exist.

1.8.1 Object Usage

STANDARD

- Names for all unique constraints must be explicitly defined using data definition language (DDL). Do not allow DB2 to generate default names.
- DDL syntax allows for unique constraints to be defined along side of the corresponding column definition, as a separate clause at the end of the table definition, or in an ALTER table statement. Each of these methods are acceptable at CMS provided all of the required parameters noted below are included in the definition.
- DB2 requires that a unique index be created to support each unique constraint defined for a table. DB2 will mark a table as unusable until such an index is created.

1.8.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining a unique constraint in an application table.

Parameter	Instructions
Constraint-name	Specify a name for the unique constraint based on the Standard Naming Convention.
(column name, ...)	Specify the name of the column or list of columns which make up the unique constraint.

1.9 ROWID

ROWID is a column data type that generates a unique value for each row in a table.

Using the ROWID column as the partitioning key may allow for random distribution across partitions. DB2 will always generate the value for a ROWID column if "GENERATED ALWAYS" is specified. ROWID has an internal representation of 19 bytes which never changes. The first two bytes are the length field followed by 17 bytes for the ROWID. The external representation of the ROWID is 40 bytes with the RID appended(this changes with a REORG). ROWID values can only be assigned to a ROWID column or ROWID variable. When using a host variable to receive a ROWID it is necessary to declare the host variable as a ROWID for processing by the precompiler:

```
EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 ABC-ROWID SQL TYPE IS ROWID,
EXEC SQL END DECLARE SECTION END-EXEC.
```

Assigning a character string to a ROWID you must first cast it to the ROWID data type. Two ways in which to cast a CHAR, VARCHAR, or HEX literal:

- CAST (expression AS ROWID)
- CAST (X'hex_literal' AS ROWID)

1.9.1 Object Usage

STANDARD

- Table cannot have more than one ROWID column.
- Trigger cannot modify a ROWID column.
- ROWID column cannot be declared a primary or foreign key.
- ROWID column cannot be updated.
- ROWID column cannot be defined as nullable.
- Cannot have a ROWID column on a temporary table.
- ROWID column cannot have field procedures.
- ROWID column cannot have check constraints.
- Tables with a ROWID column cannot have an EDITPROC
- ROWID data type cannot be used with DB2 private protocol, can only be used with DRDA.
- ROWID column is required for implementing LOBs.

1.9.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining a ROWID column. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
GENERATED BY DEFAULT	DB2 accepts valid row ID's as inserted values for a row. DB2 will generate a value if none are specified. Must define a unique single column index on the ROWID column (cannot INSERT until created). Recommended for tables populated from another table.
GENERATED ALWAYS	DB2 will always generate value. An index is not required if generated with this approach. LOAD utility cannot be used to load ROWID values if the ROWID column was created with the GENERATED ALWAYS clause. This is the recommended approach.

For additional information see IBM's Reference manuals.

1.10 Identity Columns

A column type for generating unique sequential value for a column when a row is added to the table. Sequential incrementing is not guaranteed if more than one transaction is updating the table. For a unit of work that is rolled back, the allocated numbers that have already been used are not reissued.

1.10.1 Object Usage

STANDARD

- Table cannot have more than one Identity column.
- Identity columns cannot be defined as nullable.
- Cannot have a FIELDPROC.
- Table(s) with an Identity column cannot have an EDITPROC.
- Use of "WITH DEFAULT" is not allowed.
- The data type of an identity column can be INTEGER, SMALLINT, or DECIMAL with a scale of 0. The column can also be a distinct type based on one of these data types.
- Issuing a recovery to a prior point in time will cause DB2 not to reissue those values already used.
- CREATE table ... LIKE ... INCLUDING IDENTITY causes the newly created table to inherit the identity column of the old table.
- The ALTER table statement can be used to add an identity column. If the table is populated when the ALTER is issued, the table is placed in the REORG pending state.

1.10.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining an Identity column. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
GENERATED BY DEFAULT	An explicit value can be provided, if a value is not provided then DB2 generates a value. Uniqueness is not guaranteed except for previously DB2 generated values. You must create an unique index to guarantee unique values. Developers need to check for -803 SQLCODE indicating an attempt to insert a duplicate value in a column with an unique index.

Parameter	Instructions
GENERATED ALWAYS	DB2 will always generate a value, an explicit value cannot be specified with an INSERT and UPDATE statement. RI to columns using GENERATED ALWAYS is not allowed as new values are assigned if the table is reloaded.
AS IDENTITY	This designates the column as an identity column. The column is implicitly NOT NULL.
START WITH	Only need to specify if the number is not "1" furthermore, must be a valid number for the chosen data type. This number can be a positive or negative number. If it is a positive number, it is an ascending number. If it is a negative number, it is a descending number.
INCREMENTED BY	Value by which the Identity column will be incremented (1 is the default). The value must be valid for the chosen data type.
CACHE 20	Preallocated values that are kept in memory by DB2 (the default is 20). The minimum value that can be specified is 2, and the maximum is the largest value that can be represented as an integer. During a system failure, all cached Identity column values that are yet to be assigned are lost, and thus, will never be used. Therefore, the value specified for CACHE also represents the maximum number of values for the identity column that could be lost during a system failure.
NO CACHE	No preallocated values are kept in memory by DB2. This is useful if it is necessary to assign numbers in sequence as the rows are inserted in a sysplex data sharing environment.

For additional information see IBM's Reference manuals.

1.11 Sequence Objects

Sequence Objects generate unique sequential numbers independently of a table. "Many users can access and increment the sequence concurrently without waiting. DB2 does not wait for a transaction that has incremented a sequence to commit before allowing another transaction to increment the sequence again."

Sequence Objects are preferred over identity columns when used as an id with RI on multiple tables. The Sequence object is known before inserting the parent record, removing the need to insert and select the parent, before inserting the child records.

1.11.1 Object Usage

Sequence Objects may not be defined with NO CACHE or ORDER, unless an exception is given by the Central DBA's. This does mean that there may be gaps in the numbering between records.

1.12 Views

A View in DB2 is an alternative representation of data from one or more tables or views. Although they can be accessed using data manipulation language (SELECT, INSERT, DELETE, UPDATE), views do not contain data. Actual data is stored with the underlying base table(s). Views are generally created to solve business requirements related to security or ease of data access. A view may be required to limit the columns or rows of a particular table a class of business users are permitted to see. Views are also created to simplify user access to data by resolving complicated SQL calls in the view definition.

See Standard Naming Convention.

1.12.1 Object Usage

STANDARD

- View definitions must reference base tables only.
- Do not create views which reference other views.

Use views sparingly. Create views only when it is determined that direct access to the actual table does not adequately serve a particular business need. In general, views should not be required if the data on a table is not sensitive and is only accessed using predefined SQL such as static embedded SQL.

1.12.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining a view. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
(column list, ...)	List column names for each column that is to be defined to the view. These names must conform to the DB2 Standard Naming Convention for column names.
AS select statement	Specify SQL select statement that defines this view. Do not include existing views as part of the view definition. Do not include a SELECT * ... as a select statement.

1.13 Indexes

An index is an ordered set of data values with corresponding record identifiers (RIDs) which identify the location of data rows on a tablespace. An index is created for one table and consists of data values from one or many columns on the table. When created, one or more VSAM linear datasets used to hold the data and RID values is also created. This physical object (dataset), known as an indexspace, can be managed in DB2 using storage groups (DB2 Managed).
See Standard Naming Convention.

1.13.1 Object Usage

STANDARD

- Use DB2 storage groups to manage storage allocation for all indexes.
- Specify CLOSE YES for all indexes.
- Specify COPY YES for all indexes that will be Image Copied.
- For read-only tables, define indexes with PCTFREE 0, FREEPAGE 0.
- When creating an index on a table that already contains over 5,000,000 rows, use the DEFER option then use a REBUILD INDEX utility to build the index. By stating DEFER YES on the CREATE INDEX statement, DB2 will simply add the index definition to the DB2 catalog; the actual index entries will not be built. Using this method can significantly reduce the amount of resources needed to create an index on a large table.
- One index on every table must be explicitly defined as the clustering index.
- Specify PRIQTY and SECQTY quantities that fall on a track or cylinder boundary. On a 3390 device a track equates to 48K and a cylinder equates to 720K.
- For indices over 100 cylinders always specify a PRIQTY and SECQTY on a cylinder boundary, or a number that is a multiple of 720.

1.13.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining an index. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
CLUSTER	One index for each table must be designated as the clustering index. Not doing so will cause DB2 to assume the first index created on the table as the clustering index.
USING STOGROUP stogroup name	Indicate the DB2 storage group on which the indexspace will reside. Do not specify the default DB2 storage group (SYSDEFLT). Unauthorized objects defined in the default storage group will be dropped without warning. See Standard Naming Convention.
PRIQTY	Primary quantity is specified in units of 1K bytes. Specify a primary quantity that will accommodate all of the index entries in the table or partition (for partitioned indexes).
SECQTY	Secondary quantity is specified in units of 1K bytes. Specify a secondary quantity that is consistent with the anticipated growth of the table or partition (for partitioned indexes). The value specified should be large enough to prevent the indexspace from spanning more that three extents prior to the next scheduled REORG.
FREEPAGE	Indicate the frequency in which DB2 should reserve a page of free space on the indexspace when data is initially loaded to the table or when the index or index partition is rebuilt or reorganized.
PCTFREE	Indicate what percentage of each page on the indexspace should be remain unused when data is initially loaded to the table or when the index or index partition is rebuilt or reorganized.
BUFFERPOOL bpname	Provide a valid bufferpool designation. The default value is BP0 which should never be used - it is reserved for the DB2 catalog. Consult with the central DBA to determine the appropriate bufferpool setting for your database objects.
CLOSE YES	Indicate whether DB2 should close the corresponding VSAM dataset when no activity on indexspace is detected (see Object Usage standard above).
COPY	Indicates Whether the COPY utility is allowed for this index. Valid values are NO or YES.

1.14 Table Alias

A DB2 Alias provides an alternate name for a table or view that resides on the local or remote database server. At CMS, aliases will be maintained to allow application programs use to reference unqualified table and view names of different owners with the single owner of their packages.
See Standard Naming Convention.

1.14.1 Object Usage

STANDARD

Aliases are utilized in the development environments to access another application's tables with different qualifiers. A common owner is used in production instead of aliases.

1.14.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining an alias. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
CREATE ALIAS alias name	Specify an alias name consistent with the Standard Naming Convention.
FOR owner.tablename	Provide a fully qualified table name (creator.tablename) for the corresponding table.

1.15 Synonyms

A DB2 Synonym is an alternate name an individual can assign to a table or a view. After creating a synonym, the individual can refer to the unqualified synonym name (name without a creator prefix). DB2 will recognize the unqualified name as a synonym and will translate the synonym name to the actual fully qualified table or view name (creator.name). With current releases of DB2, synonyms no longer provide a significant benefit in a development environment and will not be supported at CMS.

1.15.1 Object Usage

STANDARD

Applications developed and maintained at CMS must not reference DB2 synonyms. Synonyms will not be supported in the validation and production environments.

1.16 Stored Procedures

A Stored Procedure is a compiled program defined at a local or remote DB2 server that is invoked using the SQL CALL statement.
See Standard Naming Convention.

1.16.1 Object Usage

STANDARD

- Stored Procedures will be maintained, migrated and compiled through Endevor.
- Close and commit statements must be executed in the invoking modules to release held DB2 resources.
- To plan the configuration of the WLM environment, the application is responsible for notifying the Central DBA of the intent to use External Stored Procedures and what purpose the Stored Procedures will have, prior to developing and testing. This will include but is not limited to, business requirements, type of access, and performance considerations.
- Nested External Stored Procedures are not permitted at CMS. After invoking the initial stored procedure subsequent procedures should be invoked using application language Call/Link statements. Nested Stored Procedure calls have been proven to be inefficient.
- External Stored Procedures will be defined as STAYRESIDENT YES and PROGRAM TYPE SUB.

1.16.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining a Stored Procedure. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
IN, OUT, INOUT	Identifies the parameter as an input, output or input and output parameter to the Stored Procedure.
DYNAMIC RESULT SETS integer	Specifies the number of query result sets that can be returned. A value of zero indicates no result sets will be returned.
EXTERNAL NAME procedure-name	Identifies the user-written program/code that implements the Stored Procedure.

Parameter	Instructions
LANGUAGE	Identifies the application programming language that the Stored Procedure is coded in.
PARAMETER STYLE	Specifies the linkage convention used to pass parameters to the Stored Procedure.
DBINFO	Specifies whether specific information known by DB2 is passed to the Stored Procedure when it is called.
COLLID	Identifies the package collection that is used when the Stored Procedure is called.
WLM ENVIRONMENT	Identifies the MVS Workload Manager (WLM)environment that the Stored Procedure is to run in.
STAY RESIDENT YES	Specifies whether the Stored Procedure load module stays resident in memory when the stored procedure terminates.
PROGRAM TYPE SUB	Identifies if the Stored Procedure runs as a Main or a Subroutine.
SECURITY	Identifies how the Stored Procedure interacts with RACF to control access to non-DB2 resources. Only USER and DB2 are allowed.
COMMIT ON RETURN	Specifies if DB2 commits the transaction immediately on return from the Stored Procedure.

1.17 User Defined Functions

A User Defined Function (or UDF) is similar to a host language subprogram or function that is invoked in an SQL statement.

UDF's will follow the rules used for Stored Procedures.

Two common CMS function to convert Railroad Retirement board numbers to SSA HICN format have been defined:

CMS.CMS_RRBTOSSA (rrb CHAR(12)) RETURNS CHAR(11)
CMS.CMS_SSATORRB (ssa CHAR(11) RETURNS CHAR(12)

1.18 User Defined Types

A Distinct Type (or User Defined Data Type) is based on a built-in database data type, but is considered to be a separate and incompatible type for semantic purposes. Example: One could define a US_Dollar and Mexican Peso data types, although they may both be defined as DECIMAL(10,2), the business may want to prevent these columns from being compared to one another.

There are currently no CMS standards for UDTs. If there is a need for a UDT please contact the Central DBA group for guidance.

1.19 Triggers

A Trigger is a defined set of SQL defined for a table that executes when a specified SQL event occurs.
See Standard Naming Convention.

1.19.1 Object Usage

STANDARD

- Single triggers combining multiple actions for a table should be used rather than multiple triggers to reduce the trigger calling overhead.
- Triggers firing other triggers should be avoided.

1.19.2 Required Parameters (DDL Syntax)

STANDARD

The parameters listed below must be included in the DB2 data definition language (DDL) when defining a Trigger. DB2 default settings must not be assumed for any of the following.

Parameter	Instructions
ON table table-name	Identifies the triggering table that the trigger is associated with.
FOR EACH	ROW - Specifies that DB2 executes the trigger for each row of the triggering table that the triggering SQL operation changes or Inserts. STATEMENT - Specifies that DB2 executes the triggered action only once for the triggering SQL operation.
WHEN	Identifies a condition that evaluates to true, false, or unknown.
BEGIN AUTOMIC	Specifies the SQL that is executed for the triggered action.

1.20 LOBs

A large object is a data type used by DB2 to manage unstructured data. DB2 provides three built-in data types for storing large objects:

- Binary Large Objects, also known as BLOBs can contain up to 2GB of binary data. Typical uses for BLOB data include photographs and pictures, audio and sound clips, and video clips.
- Character Large Objects, also known as CLOBs can contain up to 2GB of single byte character data. CLOBs are ideal for storing large text documents in a DB2 database.
- Double Byte Character Large Objects, also known as DBCLOBs can contain up to 1GB of double byte character data, for a total of 2GB. DBCLOBs are useful for storing text documents in languages that require double byte characters, such as Kanji.

BLOBs, CLOBs, and DBCLOBs are collectively referred to as LOBs. The actual data storage limit for BLOB, CLOBs, and DBCLOBs, is 1 byte less than 2 gigabytes of data.
ROWID column is required for implementing LOBs.
There are currently no written standards for LOBs. The CMS standard for unstructured data is to use Content Manager. If there is a need for LOBs please contact the Central DBA group for guidance.

1.21 Buffer Pools

Buffer pools are areas of virtual storage where DB2 temporarily stores pages of table spaces or indexes. When an application program requests a row of a table, the page containing the row is retrieved by DB2 and placed in a buffer. If the requested page is already in a buffer, DB2 retrieves the page from the buffer, significantly reducing the cost of retrieving the page.

1.21.1 Object Usage

STANDARD

Bufferpools are assigned and created by the Software Support Group with Central DBA input.

The Bufferpool assignments are listed below:

BP1	Large Tables (> 48000K or > 1000 tracks)
BP2	Large Indexes (> 48000K or > 1000 tracks)
BP3	Medium Tables ((> 240K and < 48000K) or (> 5 tracks and < 1000 tracks))
BP4	Medium Indexes ((> 240K and < 48000K) or (> 5 tracks and < 1000 tracks))
BP5	Small Tables (< than 240K or < than 5 tracks)
BP6	Small Indexes (< than 240K or < than 5 tracks)

1.22 Capacity Planning

It is the responsibility of the Project GTL to schedule a meeting with EDCG, Lockheed Martin and the Central DBA team to discuss and plan for impacts a new system or major modifications to an existing system will have on the CMS computer environments. This includes but not limited to Development, Test, Performance Testing, Production, etc.

1.23 Space Requests

It is the responsibility of the Project GTL to request necessary space from EDCG. The space required in each subsystem (all test environments and production) must be delivered as part of the pre-development walk-through.

Application Programming

2.1 Data Access (SQL)

STANDARD

The following is a list of SQL usage standards to which all application programs developed and maintained at CMS must adhere.

General SQL Usage

- Application programs must not include data definition language (DDL) or data control language (DCL) SQL statements.
- SQL statements must include unqualified table and view names only. At no time should a table name be referenced from an application program using the table creator as a prefix.
- SQL columns must be accessed with elementary data items. At no time should host structures be referenced in an SQL statement. The use of group level data items, or structures are only permitted to manipulate variable length data or null indicators, (i.e., VARCHAR).

SQL Error Handling

- All application programs must check the value of SQLCODE immediately after each executable SQL command is issued, to determine the outcome of the SQL request. Depending on the requirements of the application program, appropriate logic to handle all possible values of SQLCODE must be performed
- All application programs must utilize a CMS Standard SQL Error Handling Routine calling DSNTIAR and formatting output to SYSOUT to handle all unexpected DB2 error conditions that are not accommodated within the application.
- Standard Error Handling Routines
- A complete list of SQLCODEs can be located in the DB2 Messages and Codes or DB2 Reference Summary manuals.

DECLARE TABLE

- Table Declarations for all tables and views accessed must be generated using the DB2 DCLGEN command.
- Table declarations must be stored individually as members in a Endevor library. Table declarations will be expanded in application source code at program preparation time using the EXEC SQL, INCLUDE command.

HOST VARIABLES

- Host Variable Declarations must be defined as elementary data items in the Working-Storage section in Cobol programs. Group level data items as host variables are only permitted to manipulate variable length data or null indicators, (i.e., VARCHAR).

DECLARE CURSOR

- DECLARE CURSOR statements must be defined in the Working-Storage section in Cobol programs.
- Since cursor definitions include a valid SQL SELECT statement, DECLARE CURSOR statements must also adhere to the SELECT statement standards.
- Cursor declarations for read-only cursors must include the FOR FETCH ONLY, to make use of block fetching (improved I/O).
- Cursor declarations for cursors which will retrieve a small number of rows, must include the OPTIMIZE FOR 1 ROWS clause to avoid sequential prefetch processing (increased I/O).

SELECT

- SELECT statements must include an explicit list of column names and expressions, which are being retrieved from DB2. Asterisks (*) are only permitted in SELECT statements under the following circumstances:
- When using the SQL COUNT column function to determine a number of rows (SELECT COUNT(*) FROM table-name...). (SELECT 1 INTO :WS-COUNT FROM table-name WHERE …) with a -811 check for duplicates, or similar correlated select, is preferred for existence checking when the number of rows isn't important
- Old code with a subselect to determine the existence of a condition (... WHERE EXISTS (SELECT * FROM table-name ...)). Using (… WHERE EXISTS (SELECT 1 FROM table-name)) is preferred.

INSERT

- INSERT statements must use the format of a column list with corresponding host variables for each column. INSERT INTO tablename (col1, col2, col3,...) VALUES (:hvar1, :hvar2, :hvar3...)

LOCK table

- Due to the adverse affect on concurrent usage of DB2 resources, the use of the LOCK table command is limited. Use of the LOCK table statement must be approved by the Central DBA staff when it is required to achieve improved performance or data integrity.

CLOSE CURSOR

- All application programs designed to process SQL cursors must execute a CLOSE CURSOR statement for every corresponding OPEN CURSOR.

2.2 Application Recovery

To achieve a high level of data recoverability and application concurrency, all DB2 application programs updating data, (INSERT, DELETE, UPDATE), will incorporate logical unit of work processing. Application logic will clearly identify begin and end points for sets of work which must be processed as a whole. Commit and rollback commands will be incorporated where appropriate.

All batch DB2 application programs will include abend/restart logic which will minimize the impact to DB2 resources as well as the allotted batch processing window in the event of an application failure.

Quickstart / MVS from BMC is the standard development tool that must be used at CMS in order to comply with this standard.

2.3 Program Preparation

2.3.1 DB2 Package

All DB2 application programs developed and maintained at CMS are bound to DB2 as packages and are included in DB2 plans using the PKLIST parameter in the BIND PLAN statement. Application programs are normally bound as part of the Endevor procedures.

All DB2 packages must be bound using an ISOLATION level of 'CS' (cursor stability). Specifying any other isolation level, such as repeatable read (RR) or uncommitted read (UR) requires approval of the Central DBA staff. EXPLAIN(YES) is required.

The following is an example of BIND PACKAGE syntax which should be used to create DB2 packages at CMS:

```
BIND PACKAGE(XYZ0D0)    -
   MEMBER(XYZPROG1)     -
   LIBRARY('ASCM.#ENDVOUT.XYZ.XYZP0.CODDBRM')-
   OWNER(DBXYZ0D0)      -
   QUALIFIER(DBXYZ0D0)  -
   VALIDATE(BIND)       -
   EXPLAIN(YES)         -
   RELEASE(COMMIT)      -
   ISOLATION(CS)        -
   ACTION(REPLACE)
```

2.3.2 DB2 Plan

All DB2 applications developed and maintained at CMS will implement DB2 plans which contain a list of DB2 packages only. Use of the MEMBER parameter in the BIND PLAN statement to include DBRM members is prohibited.

All DB2 plans must be bound using an ISOLATION level of 'CS' (cursor stability). Specifying any other isolation level, such as repeatable read (RR) or uncommitted read (UR) requires approval of the Central DBA staff.

2.3.3 Explains

Aliases to plan tables for each database qualifier (EX DBXYZ0D1) will be created by the Central DBA group. EXPLAIN(YES) is required on all binds to allow access paths to be reviewed for performance.

2.4 DB2 Development Tools

SPUFI queries may not use RR repeatable read. This has been restricted by not allowing access to the RR plan.

Adhoc access should be limited to Warehouse tables. Adhoc access to Operational tables requires special approval from the owning GTL.

The DB2 sample programs should be called with non-version plan names: DSNTEP2 DSNTEP4, DSNTIAD, and DSNTIAUL.

Naming Standards

- **3.1 Conventions for Objects and Datasets**

This section discusses the standard naming conventions for DB2 objects and datasets. These conventions were designed to meet the following objectives:

- Guarantee uniqueness of DB2 object names within a DB2 subsystem.
- Provide a uniform naming structure for DB2 objects of similar types.
- Simplify physical database design decisions regarding naming strategies.
- Provide ability to visually group DB2 Objects by the application and/or subject area for which the objects were designed to support.

- **The following naming standards apply to any DB2 object (database, tablespace, table, view, package, etc.) used to hold or maintain user data, as well as external user objects, (partitioned datasets, production job names, etc.) that will be used in conjunction with DB2 as part of standard application development and system operation procedures.**

3.1.1 Standard Naming Format for DB2 Objects

All DB2 objects will be named according to the standard formats listed in the following table. All object names must begin with an alphabetic character and cannot begin with DSN, SQL, or DSQ.

Object	Format	Example
• Required Attributes		
Alias	creator.aliasname	DBXYZOT1.NPS_PROV_INFO
• Alias names will always match the name of the table to which the alias pertains		
Auxillary Index	DBAAAxEn.AISSSNNN	DBNLR0D0.AI001001
• AAAxEn and DBAAAxEn follow the standard of naming convention of Table Owner and Tablespace respectively • AI for Auxiliary index • SSS the Tablespace numeral from characters 4,5,6 • NNN is a running number for that particular table. For example a partition Tablespace having 200 parts and 2 CLOB columns can have can have NNN from 1-200 for column 1 and 401 to 600 for column 2.		
Auxillary Table	DBAAAxEn.ATSSSNNN	DBNLR0D0.AT001001
• AAAxEn and DBAAAxEn follow the standard of naming convention of Table Owner and Tablespace respectively		

	• AT for Auxiliary index • SSS the Tablespace numeral from characters 4,5,6 • NNN is a running number for that particular table. For example a partition Tablespace having 200 parts and 2 CLOB columns can have can have NNN from 1-200 for column 1 and 401 to 600 for column 2.	
Auxillary Tablespace	AAAxEn.ASSSSNNN	NLR0D0.AS001001
	• AAAxEn and DBAAAxEn follow the standard of naming convention of Table Owner and Tablespace respectively • AI for Auxiliary index • SSS the Tablespace numeral from characters 4,5,6 • NNN is a running number for that particular table. For example a partition Tablespace having 200 parts and 2 CLOB columns can have can have NNN from 1-200 for column 1 and 401 to 600 for column 2.	
Column	bb..bb	PROV_ID
	• 30 character name (maximum), • Applications creating DCLGENs should use a shorter limit to keep the generated variable length with in 30 when the DCLGEN prefix and indicator suffix are added. • The name should be derived from the business name identified during the business/data analysis process; • The name must include acceptable class and modifying words as defined by the data administration group.	
Collection	AAAxEn	XYZ0D0
	• 6 character name • The first three-characters (AAA) is the Application Identifier • The fourth char numbers multiple collections • The last chars are the environment and version • Plans are always created by the Central DBA	
Database	AAAxEn	XYZ0T0
	• 6 character name • the first three characters indicate the application responsible for the data; this abbreviation must be approved by APCSS and the Central DBAs • The fourth char numbers multiple databases • The last chars are the environment and version • All Databases are defined by the Central DBA	
DBRM	AAAbbbbb	NPSPROG1
	• 8 character name (maximum), (three-character Application Identifier & five-char description);	

	• DBRM name must be identical to the source module name of the corresponding application program.	
DCLGEN	AAAttt for tables AAAVttt for views	NPS001 NPSV001
	• 6-7 character library member name • for tables, the DCLGEN member name will correspond to the name of the corresponding tablespace. • for approved views, the DCLGEN member name should begin with a three-character Application Identifier, a one-position alphabetic character ('V' for 'view'), and a three-position sequential number uniquely identifying the view member name	
Foreign Key	AAAtttFn	NPS001F1
	• 8 character name; • three-character Application Identifier, a three-position sequential number indicating the corresponding tablespace number, a one-position fixed character ("F"), and a one-position sequential number indicating the specific foreign key for the corresponding table/tablespace; • must be unique within the corresponding database	
Index	AAAtttnn	NPS00201
	• 8 character name; • three-character Application Identifier, a three-position sequential number indicating the corresponding tablespace number, and a two-position sequential number indicating the specific index for the corresponding table/tablespace; • must be unique within database	
Package	AAAbbbbb	NPSPROG1
	• 8 character name (maximum), (three-character Application Identifier & five-char description); package name must be identical to the source module name of the corresponding application program.	
Plan	AAACxEn	MSIB0V0
	• CMS utilizes two plans for each application one for online programs and one for Batch programs • 7 character name • The first three-characters (AAA) is the Application Identifier • The fourth is either a ‘C’ for CICS or a ‘B’ for non-CICS • The fifth char numbers multiple plans • The last chars are the environment and version • Plans are always created by the Central DBA	

Primary Key	AAAtttPK	NPS001PK
	• 8 character name; • three-character Application Identifier, a three-position sequential number indicating the corresponding tablespace number, a two-position fixed character ("PK");	
Program	AAAbbbbb	NPSPROG1
	• 8 character name (maximum), • three-character Application Identifier & five-char descriptive name • use existing standards (refer to CMS Data Center Users' Guide	
Schema	See Collection	
	• The name is the same as the collection used for the application	
Sequence Object	AAA_bbbbbb_SQNC	NPS_TRANS_ID_SQNC
	• 30 character name (maximum), • Prefixed with three character Application identifier • Suffixed with _SQNC • bbbbbb descripitive name following DA standards. This will usually be the column name the sequence is being used to populate.	
Storage Group	STOGRPbb	STOGRPT1
	• STOGRPT1 is the standard storage group for all applications in all the subsystems. • Storage groups are defined by the Central DBA staff	
Stored Procedure	AAASPbbb_aaaaaaaaa	MBDSP001_BENE_MATCH
	• 30 character name (maximum), • Three-character application identifier, • a two position fixed character “SP”, • a three position alphanumeric program id, • for external procedures AAASPbbb will match the external program name, for internal SQL procedures this will match the member name the definition is stored as in Endevor • a functional description, Abbreviations used will match the CMS DA dictionary. The description is optional for existing stored procedures.	
Table	AAA_bb.bbb	NPS_PROV_INFO

• 30 char (max); 18 char names fit best in reporting tools. • Three-character application identifier followed by an underscore (_) and up to 26 alphanumeric characters describing the contents of the table. • Clone table names will be the base table name with the suffix _COPY. • Table names must be created in the same manner as column names and must follow the same rules. Refer to the data administration standards for more information. • Table description should include prime word as described in the CMS Information Systems Development Guide. • $ may replace the underscore in the development environment for experimental or utility tables, that will not be used in integration, validation, or production. • Note: table name is qualified by an 8 character Table owner. • CHECK DATA discard exception tables will match the original table's name with the addition of _DSCRD as a suffix.		
Table Check Contraint	AAAttt_bb…bb	NPS001_CLASS_CODE
• 18 character name (maximum); • three-character Application Identifier with a three-position sequential number indicating the corresponding tablespace number, • an underscore (_), • and up to 11 alphanumeric characters describing the constraint.		
Table Owner	DBAAAxEn	DBMED0D0 DBPVM0P0
• 8 character name • First two characters are "DB" • The next three s (AAA) is the Application Identifier • The sixth char numbers multiple owners within the application • The last chars are the environment and version		
Tablespace	AAAnnn	NPS001
• 6 character name; • three-character Application Identifier and a three-position sequential number indicating the specific tablespace number; • must be unique within database		
Trigger	AAAtnnnx	DSYB0210
• 8 character name • 1-3 (AAA) is the Application Id • 4 (t) describes the trigger type (B-efore or A-fter) • 5-7 (nnn) Tablespace number • 8 (x) Sequence number - 0-9 A-Z		

User Defined Function	AAAFNbbb_aaaaaaaa	RASFN001_COUNTRY_RATE
• 30 character name (maximum), • Three-character application identifier, • a two position fixed character “FN”, • a three position alphanumeric program id, • AAAFNbbb will match the external program name, or the member name the udf definition is stored as under Endevor. • a functional description. Abbreviations used will match the CMS DA dictionary. The description is optional for existing stored procedures.		
View	AAA_bb…bb_VW	CME_RPS_BENE_VW
• 30 character name (maximum), • three-character Application Identifier followed by an underscore (_) and up to 21 alphanumeric characters describing the contents of the view ending with “_VW” • “_VW” is optional if the view is replacing a table name used by existing code. • Use of views must be approved by the Central DBA group • Note: view name is qualified by an 8 character table owner		

3.1.2 Application Identifiers

Standard names for most DB2 objects and datasets include a three-character code that indicates to which application the object belongs. Exceptions to this naming convention are subsystem IDs, storage groups, and columns.
The Application Identifier identifies a CMS-specific system or application. The APCSS, Central DBA Group and application development teams jointly determine and assign the three-character code. Contact APCSS for a list of assigned application identifiers.

3.1.3 Environment Identifiers

An environment, and version, En is added to database, owner, collection, and plan.

- E – life cycle stage.
- D – development/code test
- T – Functional test
- I – Integration
- V – Validation
- P – Production
- E – Emergency
- X,Y,Z – temporary copies for analysis outside life cycles.

n = relative development version within application. N will match up to an endeavor path.

- 0 – production like version
- 1 – prod+1 development
- 2 – prod+2 development as needed

The production environment will only exist as P0.

3.1.4 Obsolete Object names

To support multiple development versions the DB2 objects standard was updated in May 2006. Project defined prior to this continue to use the old names until they require the new life cycle. Old production objects for applications using the new life cycle may also continue using the old versions.

Object	Format	Example
	• Required Attributes	
Collection	AAAttt	NPS001
	• 6 character name • The first three-characters (AAA) is the Application Identifier • The last three-positions are always '001'	
Database	sssAAAnnn	PRVNPS01
	• 8 characters name; • First three characters represent Subject Area under which the data contained within the database is categorized • the next three characters indicate the application responsible for the data; this abbreviation must be approved by Central DBA and APCSS • the last two positions are sequential numbers used to uniquely identify the database within subject and application	
DCLGEN (views)	VAAAttt for views	VNPS001
	• 6-7 character library member name • for approved views, the DCLGEN member name should begin with a one-position alphabetic character ('V' for 'view'), a three-character Application Identifier, and a three-position sequential number uniquely identifying the view member name	
Plan	AAAcnnn	MSIB001
	• CMS utilizes two plans for each application one for online programs and one for Batch programs • 7 character name • The first three-characters (AAA) is the Application Identifier • The fourth is either a 'C' for CICS or a 'B' for non-CICS • The last three are always '001'	
Schema	See Collection	
Table Owner	DBA$AAA	DBAMED, DBADB1P
	• 7 character name using APCSS id • 8 character name using production subsystem id	

- First four characters are “DBA$”
- Last three characters is the Application Identifier in test or the DB2 subsystem Id in production.

- DBA$DB1P production pattern was used from 2006 to 2013 alongside development using the environment naming convention.

3.2 Image Copy Dataset Names

The naming standard system image copy datasets as follows:

&ssid.LLL.&db.&ts.[P&part.][&ictype].G0000V00

Local Copy Example: DB2T.LOC.NPS0P1.NPS001.P000.G0708V00
Offsite Copy Example: DB1P.OFS.MBD0P1.MBD023.P001.G0102V00

Full copies, for point in time retention or copies for maintenance, outside of a normal GDG cycle may use:

&ssid.LLL.&db.&ts.[P&part.]D&jdate.T&hour&min

Example: DB2P.LOC.HIT0D0.HIT023.D06357.T1234

Where:

&ssid	DB2 Subsystem ID (e.g., DB2T, DB1V, DB2P, etc)
LLL	location of imagecopy dataset (LOC = local, OFS = offsite vault, or DRV = hotsite disaster recovery vault). The storage medium (Tape, DASD) will be directed by media management based on a DSN pattern.
&db	eight-character database name
&ts	six-character tablespace name
P&part	partition number (optional for non partitioned copies, 000 if present)
&ictype	type of copy (F = full, I = Incremental) if both types are used. (optional, No type implies full copy.)
GxxxxVxx	generation dataset group number
&idate	Julian date of image copy
T&hour&min	Time of image copy

- At least two production copies will be made, one local and one to an offsite vault.
- Test only requires one copy locally. When the test file is not easily rebuilt from production, an offsite copy of test data should be sent to the archive vault. Test copies should not be in the disaster recovery vault, unless permission is given by the central DBA and media management group.
- For full image copies, at least 3 generations should be kept.

- For incremental copies, all generations after the previous full copy should be kept.
- The same number of generations is not required for onsite and offsite copies. Disaster recovery, DRV, will have 3 full copy generations even when additional copies are kept locally.
- Production files, even if read only, require an offsite copy at least once a year, unless provision for rebuilding the file from source data is provided to APCSS.
- When hotsite recovery is required, image copies must be made at least every six months. This insures that at least two backups exist at the hotsite in the event that a tape error prevents the first tape from being read.
- Image copy files may only be kept for three years. Unloaded data, and tape refreshing is needed for longer data retention.
- Catalog control, EXPDT=99000, will be used to control GDG expiration. A expiration date, like EXPDT=98030, is used be for non GDG image copies.

The naming standard for user image copy datasets as follows:

P#appid.&ssid.&db.&ts.[P&part.]D&jdate.T&hour&min
D#appid.&ssid.&db.&ts.[P&part.]D&jdate.T&hour&min

Example: P#MCS.DB1I.MCS1I1.HIT023.P001.D06357.T1234

Where:

P/D	P=Prod; D=Dev
appid	3 character application identifier
&ssid	DB2 Subsystem ID (e.g., DB2T, DB1V, DB2P, etc)
&db	eight-character database name
&ts	six-character tablespace name
P&part	partition number (optional for non partitioned copies, 000 if present)
&idate	Julian date of image copy
T&hour&min	Time of image copy

3.3 DB2 Subsystem Names

The Subsystem ID (SSID) is a unique four-character name used to identify a DB2 address space to the MVS operating system. Within each DB2 subsystem, there is a directory and catalog containing all of the DB2 objects (i.e., tablespaces, tables, indexes, application plans, etc.) defined within.

DB2 Subsystems

DB1T	MMA Test and development
DB1I	MMA Integration and Stress testing
DB1V	MMA Validation
DB1P	MMA Production
DB2T	Non-MMA Test and development
DB2I	Non-MMA Integration and Stress testing
DB2V	Non-MMA Validation
DB2P	Non-MMA Production
DB2W	Data Warehouse
DB3T	System maintenance testing

3.4 Production Library Names

The following is a list of standard library names to be used by applications developers and local and central DBAs.

Library Name	Description
PROD.DBRM.LIB	This library contains the database request modules produced by the DB2 precompiler, which are input to the BIND process.
PROD.COPY.LIB	This library contains the output from DCLGENs. The member name should be the name of the table or view. Each member contains the DB2 object columns and its COBOL copybook representation.
PROD.JCL.LIB	This library stores the batch JCL used to submit standard Production DB2 utility jobs.
PROD.PROC.LIB	This library holds the cataloged JCL procedures (i.e., application) jobstreams for all DB2 applications.

3.5 Test Library Names

The following is a list of standard library names used by Endevor for development.

Library Name	Description
ASCM.#ENDVOUT.xxx.yyyyyy.DEVDBRM Where xxx is the application ID and yyyyyy is the Endevor subsystem	This library contains the database request modules produced by the DB2 precompiler, which become input to the BIND process.
ASCM.#ENDVOUT.xxx.COPYLIB.DEVCOPY	This library contains the output from DCLGENs. The member name should be the name of the table or view. Each member contains the DB2 object columns and its COBOL copybook representation. This is a PANVALET structure.

3.6 Utility Job Names

A utility ID is a name used to uniquely identify a utility in the DB2 subsystem, only one instance of a unique utility ID may be active within DB2 at any given time. The standard for naming utilities will be to use the jobname. For multi-step and/or multi-utility jobs, the first seven characters of the jobname and 1, incremented by 1, for each successive step will be used.

Utility ID Format: Example: Jobname MSI#01IC (image copy with three steps running one utility per step)

Utilids: MSI#01I1, MSI#01I2, MSI#01I3

DB2 Utility Type Codes

Code	Description
IC	Image Copy
RG	Reorg
RC	Recover
RI	Recover Index
RS	Runstats
QU	Quiesce
MD	Modify
RB	Rebuild Index

3.7 DB2/ORACLE Naming Issues

Objects that are replicated in both databases must be named the same. Please refer to the EDG/DSS/DA naming standards.

Security

4.1 RACF Groups

4.1.1 DB2 Owner Groups

Central DBA's will create RACF Groups for the table owner names, DBxyz0en, which will be used to provide dbadm privileges for the application. Central DBA's will connect the user ids of the local DBA's, assigned to the application, to these groups in development and test. GTL approval will be required to designate a user as a local DBA.

4.1.2 Role Based Access Groups

Non dbadm privileges will be assigned to Role base RACF groups. These groups will be setup by RACF support and the application's GTL or security maintainer. A Central DBA and RACF support will permit these Role groups to the database privileges needed by the application. The GTL or the application's security maintainer will have connect authority for the Role groups to maintain the users accessing the application. The role groups are not limited to providing DB2 security, but should also provide dataset, and CICS accesses needed by users in that role.
The basic Roles will be created for users, developers, and local dba's. Additional Roles may be created as needed by the application to identify separate user or developer roles. For example instead of a single user group, there could be multiple roles for analysts reporting on the data, updaters correcting beneficiary information, and providers viewing their submitted enrollments.

Basic Role Based RACF Groups

xyzRBAG	Parent group for roles. The security maintainer and GTL have group special access in this group, allowing them to add or remove users in the role groups.
PxyzUSR	RACF group for Application users in production
DxyzUSR	RACF group for Application users in development, and testing stages.
DxyzDEV	RACF Group for Application Developers in development and testing.
DxyzDBA	RACF Group to Local DBAs in development and test. This will be used for privileges other than DB2 dbadm.

Where xyz is the assigned application id.

4.1.3 DB2 specific RACF Groups

DB2 specific RACF groups may exist for older applications, or for support databases maintained by the Central DBA's. These groups are defined and maintained by the Central DBA's under SYS$ADM. They should be replaced by role based groups when possible.

Old DB2 Specific RACF Groups

DBA$xyz	RACF Group to be used by the local DBA for non DBADM privileges
DEV$xyz	RACF Group to be used by the Application Developer to manipulate DB2 data, bind application packages, execute specific database utilities.
USR$xyz	RACF Group to be used by the Application User in the development environment. This RACF Group should facilitate user testing activities.

Where xyz is the assigned application identifier.

The central DBA may grant the local DBA connect authority within RACF so that the local DBA may add or remove user ids to/from the DEV$xxx and USR$xxx RACF Groups. The central DBA will add or remove individual RACF ids to/from the DBA$xxx RACF Group as requested by the designated local DBA and the Application's management.

4.2 DB2 Security Administration

STANDARD

DB2 Security at CMS is administered using RACF groups. Individual user ids are associated with one or more RACF Groups, each having a different set of authorities and privileges in DB2. All central DBAs, local DBAs, application developers, and application users must be assigned to and use RACF Groups within DB2. DB2 privileges and authorities are not granted to individual userids.

The following sections describe the DB2 security procedures for each of the DB2 processing environments.

4.2.1 Development

The DB2 development environment includes Code test (D) and Function test (T). In Development, security will be administered at four levels (see DBA Roles and Responsibilities for more details). They are:

- Central DBA
- Local DBA
- Application Developer
- Application User

4.2.1.1 Central DBA Privileges

Central DBA's have DB2 SYSADM authority. Central DBA's work with all applications databases to provide consistency across CMS. In development, they work with the RACF maintainers to setup the DB2 RACF resources, and they define the database and aliases for the application.

4.2.1.2 Local DBA Privileges

Local DBA's have DB2 DBADM authority. Local DBA's concentrate on one application, and are closely involved with the developers. They create and maintain all database objects needed for the application. The local DBA does not have authority to create databases or aliases, and will request these from the Centrals.

The local DBA will have select, update, insert, and delete privileges on the tables, and the following database level privileges.

DISPLAYDB	issue the DB2 -DISPLAY DATABASE command
IMAGECOPY	execute the COPY utility to create a backup of a tablespace
LOAD	execute the LOAD utility
STATS	execute the RUNSTATS utility to update catalog statistics
STARTDB	issue the DB2 -START DATABASE command
STOPDB	issue the DB2 –STOP DATABASE command
BIND	Execute binds and rebinds to the applications collections

The above privileges may be augmented or removed by the central DBA as experience dictates.

4.2.1.3 Application User and Developer Privileges

Both the user and developer roles, DxyzUSR, DxyzDEV, will have Select, Update, Insert, and Delete to the application's tables, and Execute on the plans.

The developer role DxyzDEV will be given system-level privileges, BINDADD and CREATE IN COLLECTION 'xyz0En', to the Application's Developer RACF Group (DxyzDEV). All packages created by the application development team must be created using the assigned collection ids.

The following authorities are set up for each plan:

Online	EXECUTE ON PLAN xxxC0En TO DxyzUSR, DxyzDEV
Batch	EXECUTE ON PLAN xxxB0En TO DxyzDEV

Where En is the environment.

The above privileges may be augmented or removed by the central DBA as experience dictates.

4.2.1.4 Catalog Access

All DB2 users in the test DB2 subsystems will have SELECT authority on most DB2 catalog tables.

4.2.2 Integration

The DB2 Integration environment support inter-application system testing, and stress testing. Integration security will be administered at the same four levels as development (see DBA Roles and Responsibilities for more details). They are:

- Central DBA
- Local DBA
- Application Developer
- Application User

4.2.2.1 Central DBA Privileges

Central DBA's have DB2 SYSADM authority. The Central DBA's create and maintain all database objects in integration.

4.2.2.2 Local DBA Privileges

Local DBA's have access to maintain the data in integration. The local DBA does not have authority to create or change objects with ddl.

The local DBA will have select, update, insert, and delete privileges on the tables, and the following database level privileges.

DISPLAYDB	issue the DB2 -DISPLAY DATABASE command
IMAGECOPY	execute the COPY utility to create a backup of a tablespace
LOAD	execute the LOAD utility
STATS	execute the RUNSTATS utility to update catalog statistics
STARTDB	issue the DB2 -START DATABASE command
STOPDB	issue the DB2 –STOP DATABASE command
BIND	Execute binds and rebinds to the applications collections

4.2.2.3 Application User and Developer Privileges

Both the user and developer roles, DxyzUSR, DxyzDEV, will have Select, Update, Insert, and Delete to the application's tables, and Execute on the plans.

Packages will be bound by the developers using Endevor.

The following authorities are set up for each plan:

Online EXECUTE ON PLAN xxxC0In TO DxyzUSR, DxyzDEV
Batch EXECUTE ON PLAN xxxB0In TO DxyzDEV

The above privileges may be augmented or removed by the central DBA as experience dictates.

4.2.2.4 Catalog Access

All DB2 users in the test DB2 subsystems will have SELECT authority on most DB2 catalog tables.

4.2.3 Validation

The DB2 Validation environments support user and system testing. Validation security will be administered similarly to production using four levels. (see DBA Roles and Responsibilities for more details). They are:

- Production Control
- Central DBA
- Local DBA
- Application User

4.2.3.1 Production Control

A surrogate user id xyzVALDU will be setup to match the production surrogate priviledges. The local DBA, and application developers may request batch work, and database utilities such as data loads to be run under the scheduler.

4.2.3.2 Central DBA Privileges

Central DBA's have DB2 SYSADM authority. The Central DBA's create and maintain all database objects in validation. The Centrals will also support loading data and reorganizing tablespaces.

4.2.3.3 Local DBA Privileges

Local DBA's will only have DISPLAYDB and table Select privileges to review the validation data. Data loads and changes, and batch work including reorgs will be submitting with the production control scheduler.

4.2.3.4 Application User and Developer Privileges

The user role, VxyzUSR, privileges will match production security. This generally will be limited to Execute on plans, but may include Select, Update, Insert, or Delete to the application's tables, as needed in production.

Developer's may have select access for reviewing testing at the discretion of the GTL.

Packages will be bound by the developers using Endevor.

The following authorities are set up for each plan:

Online	EXECUTE ON PLAN xxxC0Vn TO VxyzUSR
Batch	EXECUTE ON PLAN xxxB0Vn TO xyzVALDU

4.2.3.5 Catalog Access

All Local DBAs in the Validation DB2 subsystems will have SELECT authority on most DB2 catalog tables.

4.2.4 Production

In the DB2 Production environment, security is administered at four levels (see DBA Roles and Responsibilities for more details). They are:

- Production Control
- Application Coordinator
- Central DBA
- Application User

4.2.4.1 Production Control

A surrogate user id xyzPRODU will be setup with utility access, select, update, insert, and delete to the tables, and execute on the batch plans. All scheduled and planned maintenance will be run under APCSS using the production surrogate id.

4.2.4.2 Application Coordinator

The Application Coordinator is the person responsible for monitoring operational issues regarding the application. They are the first contact for issues in production by APCSS, and triage the problem to determine if developer or DBA support is required. They may have select, and display access to production for analyzing problem.

4.2.4.3 Central DBA Privileges

The Central DBA's create and maintain all database objects in production. The Centrals can support loading data and reorganizing tablespaces, but this should normally be done with prepared APCSS jobs using the production surrogate id.

4.2.4.4 Application User

Application users, PxyzUSR, will generally be limited to Execute on plans. Most host-based applications (COBOL) developed at CMS, will not require direct access DB2 tables directly. For Application Users who must access DB2 using Dynamic SQL, (QMF, SAS, Visual Basic), additional privileges may be needed.

Application Users are not limited to one RACF group. Multiple groups providing different table access, or execute to different plans will be used as needed for application security.

The following authorities are set up for each plan:

Online	EXECUTE ON PLAN xxxC0Pn TO PxyzUSR
Batch	EXECUTE ON PLAN xxxB0Pn TO xyzPRODU

4.2.4.5 General Authorizations

Local DBAs in the Production DB2 subsystems will have SELECT authority on most DB2 catalog tables.

4.3 Accessing DB2 Resources

This section describes how to access DB2 resources from different executing environments. Specifically, the following explains how to associate a process with a set of authorization Ids. In general, access to a DB2 subsystem at CMS will be controlled through IBM RACF using RACF groups. The sections below provide information to access DB2 resources from different executing environments. Since DB2 security rules differ from subsystem to subsystem, these instructions may not apply to every user in each of the CMS DB2 environments.

4.3.1 Dynamic SQL Applications (SPUFI, SAS, QMF, DB2 Connect, etc.)

Dynamic SQL Applications include all processes (batch and online) that present SQL statements to DB2 at execution time. When DB2 receives dynamic SQL statements, it uses the value of the CURRENT SQLID special register to validate authorizations. Initially, CURRENT SQLID is set to the primary authorization ID of the individual issuing the SQL statement. The value of the primary authorization ID will differ depending on the execution environment.

The value of CURRENT SQLID can be changed to one of the RACF groups associated with the primary authorization ID using the SET CURRENT SQLID command.

Dynamic SQL should be coded with unqualified table names. The SET SCHEMA command should be used to set the owner for the database.

4.3.2 Static SQL Applications

Static SQL Applications includes all processes (batch and online) in which the SQL statements have been prepared prior to executing the application. In other words, the application was precompiled and bound to DB2 so that all authorization rules access path definitions could be determined prior to executing the applications. For these 'static' applications, DB2 generally verifies authorizations at bind time using the authorization ID of the owner of the DB2 application package or plan. The owner of the application must have authority to execute all of the SQL statements included in the application. At run time, simply DB2 verifies that the primary authorization ID of the individual running the application has the authority to execute the package or plan. The value of the primary authorization ID will differ depending on the execution environment.

4.3.3 Execution Environments

Batch

DB2 applications running in a batch environment (JCL jobstream) recognize a primary authorization ID set to the RACF userid associated with the batch job. This is determined by either the USER= parameter on the job card when specified, or the TSO user id which submitted the job. In either case, it is the primary authorization ID that must have EXECUTE authority on the application plan.

CICS

When executing CICS transactions, the primary ID and related secondary IDs are determined by the RACF user ID entered with the CESN signon transaction. Typically, EXECUTE authority for DB2 application packages and plans is granted to the same EUA controlled role-based access RACF group that also grants access to the CICS transaction that executes the plan .

TSO

DB2 applications running in TSO recognize the RACF logon ID that was used to log on to TSO as the primary authorization ID. It is this primary authorization ID that must have EXECUTE authority on the application plan

Database Migration Procedures

This section outlines procedures required to implement a DB2 application database at Centers for Medicare & Medicaid Services (CMS). It is intended to compliment the methodology and procedures described in the DBA Roles and Responsibilities document for implementing relational databases.

These Application Database Migration procedures are in place to promote communication and coordination among all affected functional areas (Applications Development, Data Administration, Database Administration, Capacity Planning, Computer Operations, Production Control, Media Management, etc.). The Project GTL is responsible for coordinating the review and participation of all participants listed. The procedures are designed to ensure the success of the application development efforts as well as the integrity of the overall database architecture by proactively identifying items of concern prior to production implementation. The end result of adherence to these procedures can be a significant reduction of time and effort required to implement DB2 applications.

The following procedures identify critical points within the application and physical database development process. Each milestone is substantiated by a formal review. Because this document focuses on the physical implementation of DB2 databases at CMS, the identified review points begin after the conceptual and logical design for an application has been completed and approved.

5.1 DB2 Database Migration Overview

DB2 database objects designed to support applications developed at CMS will follow a set migration path as defined by Project Team and ENDEVOR setup. At least three environments (development, validation, and production) are required. For audit purposes, all changes to CMS database environments require GTL approval. Physical database objects will initially be implemented and tested in the development DB2 subsystem. After development and initial testing, all application components as designated by a GTL approved Data Base Service Request (DBSR) are migrated to respective Integration and/or Validation DB2 subsystems. Pre-production verification of the application (including security rules, performance, etc.) may be initiated in Integration if part of the project migration path and final pre-production verification is performed in the validation environment. Once applications are verified in Validation, they can be migrated to Production for final implementation.

5.2 Preliminary Physical Database Design Review

Prior to implementing DB2 database objects in the Development DB2 subsystem, a Preliminary Physical Database Design review must take place. This review not only provides an opportunity to review and verify the application database architecture, but also serves as a means to coordinate scheduling and resource requirements.

Time Frame: Immediately preceding implementation of physical database into the development environment. [1][2]

Participants: Application Development [3], Local and Central DBAs, Local and Central DAs, Project GTL or Technical Lead

Purpose:

- Insure the physical database design and SQL will perform well and follow standards prior to coding. This review must occur prior to coding, to prevent the need for recoding due to database changes, or performance problems.
- Establish preliminary DASD and resource requirements, providing the lead time for to purchase additional resources that may be required to support the production release.
- Determine database environments and development stages to be implemented and the proposed time line for development, testing, and production release.
- Insure user roles and database access requirements required to establish security, have been documented.
- Discuss application project plan, preliminary application migration strategies & time frames as well as impact on existing target environment.

Input:

- The physical database design (physical and logical database model, DDL, preliminary space estimates for all database environments).
- Size of tables, record counts, and expected growth.
- Documentation of application architecture and execution environments, data flows, transaction throughput estimates, number of users and their locations, etc.
- SQL representative of frequently executed and/or more complex queries.

Output:

- Approved preliminary physical database design.
- Implementation schedule for development environment
- Request for additional DASD (if necessary).
- Modified Project Plan to include all necessary migration steps

Approval Criteria:

- Physical database design which conforms to design standards noted in the DB2 Standards and Guidelines document.

- The database design supports the application design for performance and usability.
- The DASD estimates are supported by the table design, row count, and expected growth.
- The user roles for development and testing have been determined.
- Consensus from all groups that the physical design as presented can be implemented.

Comments:
[1] Assumptions: Conceptual/Logical Review process completed and approved.
[2]Actual application coding efforts should not begin until the preliminary physical database design is approved.
[3] Application Development includes representatives from the application development group responsible for the new/changed application as well as representatives from any application that is in some way affected by the new/changed application.

5.2.1 Pre-Development Migration Review Checklist

Documentation for Pre-Development walkthrough should be presented to the Central DBA Group at least ten business days before the scheduled walkthrough date. **If this date is missed your project walkthrough date will be delayed.**

It is the responsibility of the project GTL to invite the lead application developers, business owner, Local and Central DA, and Local and Central DBA's.

The following topics are mandatory sections that should be included in your documentation:

- Introduction
- Project plan
- Database models: physical and approved logical
- DDL for all Database objects to be created
- Architecture (Diagrams depicting interaction with other applications)
- Security requirements (User Roles, Master ID's/RACF requirements/DB2 Alias usage)
- Expected DASD requirements, transaction throughput
- Sample SQL of the complicated and most frequently executing queries
- Plans for Validation migration (timeline)
- Contact Information

5.2.2 Development Setup

Upon the initiation of a DB2 application development project, the central DBA will perform several steps to prepare the development environment for the Application Development team. These activities include:

- Create DBADM RACF owner groups for the new application and connect local DBA's;
- Request DB2 RACF resources be setup giving access to the Role Based RACF Groups requested by the GTL.
- Create a DB2 database which will be used to group all DB2 objects created for the application;
- Create online and/or batch application plans;
- Create plan table aliases and quick start aliases;

The local DBA is primarily responsible for maintaining database objects for the application. Once the initial database environment is turned over to the local DBA by the central DBA, object creation and maintenance can begin.

Each of the tasks noted above are described in the sections that follow.

5.2.1.1 Create Table Qualifier RACF groups

For the development environment, the central DBA will create the table qualifier RACF groups and connect the local dba's to the groups for DBADM privileges. The naming convention for the group is as follows:DBxyz0en.

5.2.1.2 Request DB2 RACF resources

As part of the initial steps in setting up a DB2 application database in the Development environment, the central DBA will work with the RACF group to setup the DB2 RACF resources granting DB2 privileges to the role groups defined by the application. **The prerequisite for this is the setup of RACF Role based groups for the application, by the EUA RACF group at the request of the GTL.**

5.2.1.3 Application Database

The central DBA will create at least one database that will be used to manage the DB2 objects (tables, tablespaces, indexes, etc.) for the application. The local DBA will be responsible for the creation and maintenance of all DB2 objects within this database. The local DBA must use authorities associated with the table owner RACF group to accomplish this task.

To create or maintain an application database object using DDL (data definition language), the local DBA must first connect to the table owner using the SET CURRENT SQLID command. This will ensure that the local DBA has the appropriate authority to create or manage the object and that all objects created are owned by the RACF Owner group.

When an application duplicates tables of another application for isolated testing, these copied tables will be placed in a separate database.
Space estimates are required when setting up the development databases to plan for DASD requirements for all stages of the lifecycle including production. The development stages, development, function test, and application test, will be at most 10% of the production data.

5.2.1.4 Application Plans

The central DBA will create application plans in DB2 which can be used to associate application programs (DB2 packages). Plans will be created for the online and batch environments and will be named xyzC0En and xyzB0En respectively (where xyz is the assigned application identifier and En is an environment). Plans will contain a generic package list as part of their definitions which will include the packages created under the collection id(xyz0en) assigned for the application. More than one plan may be created if multiple collections are used to separate access to programs.

5.2.1.5 Plan Tables

A database exists for the purpose of storing Explain Tables in all DB2 subsystems. To simplify performance reporting, in integration, validation, and production, each subsystem will have a common set of tables with aliases.

The Central DBA is responsible for creating the Aliases or Tables required in each of the environments referenced by the Endevor path(s) defined for the Application. This should be done when RACF authority is requested for a NEW application, or as soon thereafter as practical.

The JCL to create the required Tables, as well as instructions for substitutions and execution may be found in TEST.JCL.LIB (EXPLTABL).

The Template used to support the Table creation process will be maintained at the current DB2 release level by the Central DBA Group. All tables created at prior release levels are upward compatible and no intervention is required for past releases at BIND time.

Note: Each application development group must have Tables PLAN_TABLE, PLAN_HST, DSN_STATEMNT_TABLE, and DSN_STATEMNT_HST defined in the environment referenced by the current path for the CA-Endevor stage in order to successfully BIND DB2 Packages using CA-Endevor program preparation procedures. The additional tables for Visual Explain and with V9, Optimization Service Center, should also be setup along with the basic explain tables.

5.2.1.6 Aliases

The Central DBA's will create an alias with the application's owner for QSTART.CKPTCNTL_TABLE. The Central DBA's will also create aliases for tables from other applications used by the new application, as the access is approved by the application owning the table.

5.2.1.7 Central DBA Policies

The central DBA Group maintains overall control of all DB2 subsystems. While there is no predetermined limit to the number of databases a local DBA may control within an application area, central DBA will require justification/explanation for the need.

It is the local DBA's responsibility to create and monitor the number of objects housed within a given database. In the event database definitions (DBDs) or any other resource begins to impact performance of the EDM pool and/or other subsystem activity, central DBA will initiate necessary corrective actions, to include a review of subsystem activity, any necessary resource maintenance, and investigation of all anomalies such as unusually large DBDs. These activities can and should be initiated by any local DBA that feels that their development efforts are being impeded.

5.3 Pre-Integration/Validation Migration Review

This review must be conducted prior to migrating database objects to integration and/or validation. Database objects are moved to integration/validation when it has been determined that the corresponding application is ready for production implementation. Due to restricted database authorities, the central DBA will be responsible for creating the physical objects (databases, tablespaces, tables, indexes, etc.) in the integration and validation environment. The Central DBA will migrate physical database components, as designated by the DBSR, from the respective project development and/or test environment using a CMS standard migration software. Actual data migration will be the responsibility of the local DBA for the application. Migration of all other components for the application (source code, copybooks, datasets, etc.) will be the responsibility of the application development team.

Time Frame: Prior to migration of application programs and database objects to integration environment [1], or validation if there is not an integration stage.

Participants: Application Development, Local and Central DBAs, Project GTL or Technical Lead

Purpose:

- Review the physical database design to be implemented, focusing on any changes made to the design since the preliminary database design review.
- Review results of the database application architecture review.
- Review database performance tests results from development environment (if any) and review database performance testing criteria for the validation environment.
- Review security requirements and service level agreements for validation and production environments.
- Review current space requirements for validation and production environments
- Review database utility job streams (Image Copy, Recover, Reorg, Runstat…)
- Discuss recovery/synchronization plans and restartability.
- Discuss archive strategy.
- Discuss disaster/recovery plans.
- Review Preliminary Migration Plan and make recommendations for modifications where appropriate
- Discuss application migration strategies & time frames, impact on existing target environment, and backout strategies.

Input:

- Current physical model
- DDL for all objects to be created (tables, views, indexes, tablespaces, etc)
- Documentation from Database Application Architecture Review (Explain Reports)
- Documented results of database performance tests in development environment (when applicable)
- Database performance test plan for integration environment

- List of user classifications with required authority levels
- Current space estimates for validation and production environments
- Database utility job streams
- Preliminary Migration Plan (provided by the Local DBA in conjunction with the Central DBA). See Sample Migration Plan
- Current Project Plan
- All documentation must be provided, via soft copy, to the Central DBA Group at least 2 weeks prior to the Pre-integration Migration Review meeting.

Output:

- Approved physical database design
- Accepted database performance test plan for integration environment
- Approved Migration Plan with detailed backout strategy
- Request for additional DASD for integration, validation and/or production (if necessary).
- Documented security requirements including RACF groups and associated authorizations for integration, validation and production environments.
- Modified Project Plan

Approval Criteria:

- Physical database and SQL conform to CMS design standards.
- The database design and SQL supports the application design for performance and usability.

Comments:
[1] Physical database changes identified after an application is moved to integration must first be implemented in the test environment. These changes are then subject to a Pre-integration review prior to migration to the integration environment

5.3.1 Pre-Integration/Validation Migration Review Checklist

Documentation for Pre-integration walkthrough should be presented to the Central DBA Group at least ten business days before the scheduled walkthrough date. If this date is missed your project walkthrough date will be delayed.

It is the responsibility of the project GTL to invite the lead application developers, business owner, and local DBA's.

The following topics are mandatory sections that should be included in your documentation:

- Introduction
- Project plan
- Physical Database model, DDL for all objects being created
- System architecture

- Test plans for integration/validation environment
- Security requirements (RACF requirements/list of user roles with required authority levels)
- Space requirements for integration and validation, expected space requirements for production environment(s)
- Database backup/recovery plans, resynchronization procedure (if dependent on other projects)
- Disaster/recovery schema
- Archiving schema
- Explain reports (representation of the busiest/most important transactions)
- Database utility job streams
- Pre-integration migration plan with back out contingencies
- Contact Information

5.4 Pre-Production Migration Review

This review serves as the final review point prior to moving a DB2 application to production. It is primarily intended to provide an opportunity to coordinate the efforts of all affected functional areas (Applications Development, Database Administration, Capacity Planning, Systems Operations, Production Control, etc.). All DB2 applications presented for Pre-Production Migration Review must first migrate to the DB2 Validation environment.

Time Frame: Prior to migration of application programs and database objects to production environment and after application has been verified in validation environment.

Participants: Application Development, Local and Central DBAs, Project GTL or Technical Lead.

Purpose:

- Review database performance test results from validation environment.
- Review security requirements and service level agreements for production environment.
- Verify space requirements for production environment.
- Review batch utility job streams and job scheduling requirements.
- Discuss application migration strategies & time frames, impact on existing target environment, and backout strategies.
- Discuss recovery/synchronization plans and restartability.
- Discuss archive strategy.
- Discuss disaster/recovery plans.

Input:

- Stress test results from validation environment
- Current space estimates for production environment and verification that required DASD is available should be approved by the Central DBA.
- Documented security requirements including RACF groups and associated authorizations for production environment (from Pre-Validation Review) with sign-off from data custodian
- Documented service level agreements
- Database utility job streams (Image Copy, Recover, Reorg, Runstat…) and scheduling requirements
- Current project plan
- Preliminary migration plan (provided by Local DBA in conjunction with the Central DBA)
- All documentation must be provided, via soft copy, to the Central DBA Group at least 2 weeks prior to the Pre-Production Migration Review meeting.

Output:

- Approval for the application and database objects to move to production

- Approved Migration Plan (with detailed backout strategy) for application programs and database objects
- Approved security requirements including RACF groups and associated authorizations for production environment.

Approval Criteria:

- The production environment has the resources in place to support the addition of the application
- The application will not adversely affect the current production processes.
- The application and database production security is in place
- Maintenance utilities and scheduling are prepared.
- Disaster recovery plans are in place.

5.4.1 Pre-Production Migration Review Checklist

Documentation for Pre-Production walkthrough should be presented to the Central DBA Group at least ten business days before the scheduled walkthrough date. If this date is missed your project walkthrough date will be delayed.

It is the responsibility of the project GTL to invite the lead application developers, business owner, and local DBA's.

The following topics are mandatory sections that should be included in your documentation:

- Introduction
- Project plan
- Physical Database model, DDL for all objects being created
- System architecture
- Documented service level agreements
- Stress/performance test results from validation
- Security requirements (RACF requirements/list of user classifications with required authority levels) with sign-off from data custodian
- Space requirements for production environment(s)
- Database backup/recovery procedure(s), resynchronization procedure (if applicable)
- Disaster/recovery procedure(s)
- Archiving procedure(s)
- Explain reports (representation of the busiest/most resource intensive transactions)
- Database utility job streams (production ready)
- Pre-Production migration plan with back out contingencies

- Contact Information

Database Utilities

A full suite of DB2 utilities are available at CMS that provide for overall administration of DB2 database objects. Primarily, these utilities are available to Corporate and Project DBAs. Project DBAs may, if appropriate, grant access to a limited number of these utilities to application developers as well.
CMS supports DB2 utilities from two primary vendors, IBM and BMC Software. Many products from both vendors overlap in functionality; however, there is a difference in the functions and performance of each. Therefore, CMS has established a standard set of utilities (with vendor designation) for DB2 utility jobs intended for production. The following documents these standards.

6.1 CMS Standard DB2 Utilities

The following matrix indicates the CMS's vendor selection for DB2 utilities. All DB2 utility jobs intended for migration to the CMS production environment must be developed using these designated tools. Please refer to the appropriate vendor reference manual for specific execution procedures.

Utility	Vendor	Product
Check Data	IBM	CHECK DATA*
Imagecopy	BMC	Copy Plus
Load	BMC	Load Plus
Modify	BMC	Copy Plus
Quiesce	BMC	QUIESCE
Rebuild Index	BMC	REBUILD INDEX
Recover	BMC	Recover Plus
Reorg	BMC	Reorg Plus
Runstats	BMC	RUNSTATS
Unload	BMC	Unload Plus

* These utilities are initiated from the IBM utility driver program DSNUTILB

6.2 Restrictions on Utilities with QREPed Tables

IBM Replication Server, aka QREP, replicates table data between database environments such as DB2 to Oracle using updates recorded in the DB2 log. DB2 'QREPed' tables are identified by having DATA CAPTURE CHANGES, in the DDL.

Standard

Tables that are replicated with QREP must have all data changes fully logged with commits every 500 updates.

Use of the LOAD utility, or REORG with discard, either in a single job, or in a production process, must be reviewed and approved by both the Central DBA group and the QREP group. Any unlogged updates require fully resynchronizing the table using QREP, before replication can be resumed. This requires manual triggering, and can require table updating and replication to be unavailable for several hours for large tables.

BMC Load can be used with SHRLEVEL CHANGE, and APCOMMIT 500 to compatibly load records in QREPed files using insert processing.

IBM Load with SHRLEVEL NONE LOG YES, does not create log records usable for QREP replication.

QREPed tables must have primary key for the QREP apply process to use.

Database Performance Monitoring

7.1 TOP 10 SQL Performance Measures

In an effort to proactively monitor the efficiency of the SQL statements executing in the production DB2 subsystems, the Central DB2 DBA staff has developed a set of reports for capturing high usage SQL statements. The reports come from the BMC APPTUNE product and will consist of one static SQL report and one dynamic SQL report per day for each production subsystem (DB2W will have only a static report). Each daily report will have the Top 10 SQL statements that used the most total CPU time.

Subsystem	Static Report	Dynamic Report
DB1P	Yes	Yes
DB2P	Yes	Yes
DB2W	Yes	NO*

Collecting SQL data for DB2W exhausted the available storage allocated to the APPTUNE product and had to be turned off.

The complete reports are stored on the mainframe for 90 days and have the following GDG names:

Subsystem	Static Report GDG Name	Dynamic Report GDG Name
DB1P	P@DBA.@DB2.DB1P.TOP10.SF301	P@DBA.@DB2.DB1P.TOP10.DF301
DB2P	P@DBA.@DB2.DB2P.TOP10.SF301	P@DBA.@DB2.DB2P.TOP10.DF301
DB3P	P@DBA@DB2.DB3P.TOP10.SF301	P@DBA.@DB2.DB3P.TOP10.DF301
DB2W	P@DBA@DB2.DB2W.TOP10.SF301	Not Collected

The data from each of these reports will be stored for 90 days in a DB2 table named SYS$ADM.DBA_TOP10_PERF. The table structure is as follows:

Column Name	Column Description
SSID	DB2 Subsystem ID – DB1P ,DB2P, DB3P, DB2W
INTERVAL_DATE	Date that SQL executed
RPT_TYPE	Report Type – SF301 for Static, DF301 for Dynamic
OWNER	Object Owner
COLLID	Collection ID
PACKAGE	Package Name
STMTNO	SQL Statement ID
SQL_CALLS	Number of Times SQL was executed
TOT_CPU_TIME	Total CPU Time Consumed by the SQL
AVG_CPU_TIME	Average CPU Time for One Execution of the SQL
TOT_ELAP_TIME	Total Elapsed Time Consumed by the SQL
AVG_ELAP_TIME	Average Elapsed Time for One Execution of the SQL

On a weekly basis each Central DBA will be responsible for analyzing an SQL that is at the top of the Top 10 report for their respective applications and make recommendations and take actions to attempt to improve the performance of the high usage SQL. **Please note that Central DBAs can only make recommendations and do not have the authority to require an application to make a change.** A spreadsheet named the WeeklyTop10Analysis.xls will be used to track the status of this analysis and will be stored on the CMS network (H:\DB2DBA\WeeklyTop10Analysis.xls).

In addition a weekly report will be emailed to those interested providing a list of the Top 45 SQL that used the most Total CPU Time for the week.

Glossary

active log: The portion of the DB2 log to which log records are written as they are generated. The active log always contains the most recent log records, whereas the archive log contains those records that are older and will no longer fit on the active log.

Alias: An alternate name that can be used in SQL statements to refer to a table or view in the same or a remote DB2 subsystem.
application plan - A control structure used by DB2 to process SQL statements during the execution of an application. The application plan is created during the BIND process.

Application package: An object containing a set of SQL statements that have been bound statically and that are available for processing.
archive log - The portion of the DB2 log that contains log records that have been moved (off-loaded) from the active log because there is no more space for them in the active log.

BIND: The DB2 statement that creates an application plan or package. A DB2 bind is to SQL what a compile and link edit is to host language source code. The BIND process determines which access paths to the data the application plan will utilize. BINDs can be done automatically and dynamically.

BSDS (Boot Strap DataSet): A VSAM KSDS dataset that contains name and status information for DB2, as well as relative byte address (see RBA below) range specifications for all active and archive log datasets. It also contains passwords for the DB2 Directory and Catalog, and lists of conditional restart and checkpoint records.

Bufferpool: The main storage reserved to satisfy the buffering requirements for one or more tablespaces or indexes.

CESN: The sign-on transaction used in CICS during which the User ID is entered.

CICS (Customer Information Control System): One of three (TSO, CICS, IMS) MVS host environments for which DB2 provides services to manage the interface between the host address space and DB2 address space.

Clustering index: The index that determines how rows are physically ordered in a tablespace.

Collection: An ID or qualifier which is assigned to DB2 packages. Used to manage packages in logical groups.

Column: The equivalent of a conventional data field that are the attributes of rows.

COMMIT: A SQL statement that terminates a unit of recovery. A COMMIT releases all locks. Data that was changed is made permanent within the database.

Commit point: A point in time when data is considered consistent. See point-of-consistency below.

Constraint: A rule that limits the values that can be inserted, deleted, or updated in a table.

DASD (Direct Access Storage Device): The physical volume where data is stored.

Database: A collection of DB2 objects. When you define a database, you give a name to an eventual collection of tables, indexes, and tablespaces in which they reside. A single database, for example, may contain all the data associated with beneficiaries (names, provider ID, Medicaid classification, etc.). Databases do not physically exist but are a logical group of DB2 objects that DB2 uses to assign certain authorities and that permit sensible management of data.

DB2 Catalog: DB2-maintained tables that contain descriptions of DB2 objects such as tables, views, and indexes.

DB2 Directory: The system database that contains internal objects such as database descriptors, skeleton cursor tables, and opening/closing log of relative byte addresses (RBA) for tablespaces.

DB2 command: An instruction to the DB2 subsystem allowing a user to start or stop DB2, to display information on current users, to start or stop databases, to display information on the status of databases and so on. These commands are generally entered using the DSN Command Processor in DB2 Interactive (DB2I).

DB2 Interactive (DB2I): The DB2 facility that provides for the execution of SQL statements, DB2 (operator) commands, programmer commands and utility invocation.

DB2 Objects: DB2 data objects are databases, storage groups, tablespaces, tables, indexes, indexspaces and views. A more detailed description of these objects can be found in the IBM DB2 Administration Guide. In general, an object is anything you can create or manipulate with SQL.

DB2 utility: A standard MVS batch job which requires that DB2 be running. Example DB2 utilities are COPY, LOAD, QUIESCE and REORG.

DBRM (Database Request Module): The module that contains information about SQL statements in an application program. The DBRM is created as output from the DB2's precompiler and used as input to the BIND process.

DCL (Data Control Language): One of the three components of SQL (the others being DDL and DML) comprised of SQL statements that control authorization to access and use the database, i.e. grant and revoke DB2 privileges. CMS uses RACF to control access rather the DB2 Grants.

DCLGEN (Declarations Generator): A subcomponent of DB2 which automatically generates host language declarations (e.g., COBOL copybooks) for SQL tables. In other words, the general copy library for tables.

DDL (Data Definition Language): One of the three components of SQL (the others being DCL and DML) comprised of SQL statements that define objects that make up the databases, i.e., create, alter and delete DB2 objects.

Distinct type: A user-defined data type that is internally represented as an existing type (its source type), but is considered to be a separate and incompatible type for semantic purposes.

DML (Data Manipulation Language): One of the three components of SQL (the others being DCL and DDL) comprised of SQL statements that retrieve and update the data, i.e., select, insert, update and delete data.

DMS (Disk Management Storage): A disk management tool used to manage DASD pools.

DRDA (Distributed Relational Database Architecture): The kind of database architecture using the relational data model and where some or all data is stored on a computer different from the computer used by programs or users that access the data.

Dynamic SQL: SQL statements are created, prepared, and executed while a program is executing. Therefore, it is possible with dynamic SQL to change the SQL statement during program execution and have many variations of a SQL statement at run time.

Embedded dynamic SQL: SQL statements that are not completely composed at the time the application in which the statement is embedded is prepared. Instead, the statements are prepared and executed while the program is executing. In dynamic SQL, the SQL source is contained in host language variables rather than being coded into the application program. The SQL statement can change several times during the application program's execution. SQL which is coded within an application program.

Embedded static SQL: Also known as static SQL, these are SQL statements that are created and prepared before program execution. After the SQL statement is embedded in an application, the only variance allowed at execution is in the values of the host variables.

Entity: Any person, place, thing, or event about which the enterprise collects data.

Foreign key: A column, or combination of columns, whose values must match those of a primary key in another table in order to maintain a relationship between tables. A given foreign key value represents a reference from the row(s) containing it to the row containing the matching primary key value.

HDCUG: Abbreviation for CMS Data Center Users' Guide. Originally HCFA Data Center User’s guide.

Host structure: A group of host variables. A host structure is defined using host language statements.

Host variable: Any data element declared in a host language (COBOL, PL/1, etc) that is referenced in an embedded SQL statement. Host variables are used to transfer data to and from DB2, evaluate a WHERE or HAVING clause, receive or assign special register values such as CURRENT DATE or CURRENT SQLID, or set or detect the existence of NULL values.

Image copy: Image copies are backups used for database recovery. An exact reproduction of all or part of a tablespace. The DB2 COPY utility can make full image copies (to copy the entire tablespace) or incremental image copies (to copy only those pages modified since the last image copy).

Index: An index is an ordered set of pointers to data in a table and is stored separately from the table. Each index is a separate physical structure based on data values in one or more columns of the table. Once an index is created, it is maintained by DB2 so that DB2 decides when to use or not to use the index to access data in the table. Indexes can be used to enforce uniqueness of rows in a table and enhance the performance of data retrieval operations.

Indexspace: A page set used to physically store the entries of one index (or index partition); automatically assigned when an index is created.

Key: A column or an ordered collection of columns identified in the description of a table, index, or referential constraint.

KSDS (Key Sequenced Dataset): A type of data organization used by VSAM datasets that uses key sequences.

Leaf page: The lowest-level index page which contains the keys and record IDs (RIDs) that identify individual rows in the related table.

Lock: A control structure used to serialize data updates and thereby prevent access to inconsistent data during concurrent access by multiple users.

Log: A collection of records that describe the events that occur during DB2 execution and their sequence. The information recorded is used for recovery in the event of a failure during DB2 execution. Each log record is identifiable by the relative byte address (RBA) of the first byte of its header. The record's RBA (see log RBA below) is a similar timestamp in that it uniquely identifies records that start at a particular point in the continuing log.

Log RBA (Relative Byte Address): The address of each byte in the DB2 log obtained by its offset from the beginning of the log.

MVS (Multiple Virtual Storage): IBM application development platform using virtual storage where each job to be executed is assigned its own range of addresses (address space) between 0 and 16 megabytes.

Nonleaf page: A page in an index structure that contains keys and page numbers of subordinate nonleaf or leaf pages. The nonleaf page never points directly to data.

Null: A data state that indicates the absence of information.

Object: Anything that can be created or manipulated with SQL -- that is, databases, tablespaces, tables, views, or indexes. See DB2 Objects above.

ODBC (Open Data Base Connectivity)
Package: see application package.

Page: A unit of storage within a tablespace (4K or 32K) or indexspace (4K) that is the unit of I/O. In a tablespace, a page contains one or more rows of a table.

Page set: The total collection of pages that make up an entire table space or index space. Each page set is made from a collection of VSAM data sets.

Plan: See application plan.

Point-of-consistency: A point in time at which all data is static and consistent. There are three types of point-of-consistency: 1) an application point-of-consistency guarantees the integrity of the application data at a particular point in time; 2) a tablespace set point-of-consistency refers to those points at which all of the data in a set of tablespaces is static and consistent; 3) a system point-of-consistency refers to those points in a DB2 system at which all data -- both system and user -- within that system are static and consistent.

Primary key: A column, or combination of columns, within a table whose values together form the "principal unique identifier" of rows in that table. In other words, a table's primary key serves to uniquely identify each row in that table and consists of those columns required to ensure that no duplicate rows occur in the table. This non-duplication ensures that each instance of the entity (the table) is unique.

Quiesce point: An established point that corresponds to a point-of-consistency for all identified tablespaces in the control statement. The value of the quiesce point (log RBA) is stored in the catalog table SYSIBM.SYSCOPY.

RACF (Resource Access Control Facility): A security system that controls access to z/os resources by assigning privileges to users.

RBA (Relative Byte Address): See Log RBA above.

RCT (Resource Control Table): CICS System table containing parameter settings which control application connections between CICS and DB2. The RCT is used by

the CICS attachment facility to govern the way in which DB2 resources are accessed. Rules specified in the RCT include the method DB2 should use to allocate application plans, check authorizations, allocate resources, etc.

RDBMS (Relational Data Base Management System): A kind of database management system (DBMS) based on the relational data model in which the DBMS presents the complete information content of the database to the user as a collection of two-dimensional tables (columns and rows).

Recovery: The process of rebuilding databases after a system or application failure.

Recovery log: See log above.

Referential integrity: The condition that exists when all intended references from data in one column of a table to data in another column are valid. It maintains the consistency of relationships between tables by requiring every foreign key value in a table to have a matching primary key value in a corresponding table or else be labeled null.

RID (Record IDentifier): The internal record identifier used to locate a given row in a table. It is composed of the page number and record ID within the page.

Row: The smallest unit of data in a table that can be inserted or deleted; physically stored as records on a page. It is one instance of the entity that its table represents; conventionally called the record occurrence.

Schema: A logical grouping for user-defined functions, distinct types, triggers, and stored procedures. When an object of one of these types is created, it is assigned to one schema, which is determined by the name of the object. For example, the following statement creates a distinct type T in schema C:
CREATE DISTINCT TYPE C.T ...

SQL (Structured Query Language): A database language, originally developed by IBM, to support the definition, manipulation, and control of data in a relational database.

SQLCA (SQL Communication Area): The communication block of variables used by DB2 to inform an application program of the status of the system as a result of a prior SQL call.

SQLCODE: Variable passed from DB2 to an application program, tool, or utility which indicates the status of the most recently issued executable SQL call. SQLCODE can take on one of three values: 1) SQLCODE = 0 indicates that the last SQL command executed successfully; 2) SQLCODE >0 indicates the last SQL command executed successfully with some warnings; and 3) SQLCODE < 0 indicates that an error condition was encountered and the last SQL command did not execute successfully.

SQLDA (SQL Descriptor Area): The group of variables used in the execution of some SQL statements. It is used in application programs containing embedded dynamic SQL.

SPUFI (SQL Processor Using File Input): A subcomponent of DB2I that allows interactive access to data from TSO. It allows a user to execute SQL statements without embedding them in a program.

Storage group: A named set of direct access storage device (DASD) volumes from which DB2 automatically allocates storage space, defines the necessary VSAM datasets, and extends or deletes them as required.

Stored procedure: A user-written application program that can be invoked through the use of the SQL CALL statement.

Subject area: A high-level category of data that exists in an organization. Subject areas are based on broad grouping of entities that an enterprise is concerned with in performing its work.

Table: Collections of rows (also called tuples) having the same columns (attributes). It contains data about a given entity. The rows of a table are unordered and physically stored in a tablespace. For example, a beneficiary entity (table) would consist of a row for each beneficiary and columns such as beneficiary ID, beneficiary name, and beneficiary status. A table may be defined to have a primary key, a column or set of columns whose values uniquely identify each row (beneficiary ID in the beneficiary entity).

Table check constraint: A user-defined constraint that specifies the values that specific columns of a base table can contain.

Tablespace: A tablespace is a VSAM dataset in which one or more tables are stored. It is a physical object for which disk space is allocated using primary and secondary quantities. A tablespace is broken up into pages of four kilobytes each. Many DB2 utilities including RUNSTATS, REORG and COPY, run against tablespaces (not tables).

Tablespace set: The group of all tables related to each other through referential constraints, and which must be recovered to the same point-of-consistency.

Thread: The DB2 structure that defines a connection between an application and DB2 in order to control access. At any given time, the number of active threads equals the number of users (programs, utilities, interactive users, etc.) accessing DB2. If the maximum number of concurrent threads (set at installation) is exceeded, an application must wait until a thread becomes available.

Trigger: A set of SQL statements that are stored in a DB2 database and executed when a certain event occurs in a DB2 table.

User-defined function (UDF): A function that is defined to DB2 by using the CREATE FUNCTION statement and that can be referenced thereafter in SQL statements. A user-defined function can be an external function, a sourced function, or an SQL function. Contrast with built-in function.

View: A view is a logical table that is derived from combining tables and/or other views into a single, logical table. A view can also be a subset of columns from a single table or view. Like a table, a view consists of rows and columns, but unlike a table, the data in a view is not physically stored. A view is defined to DB2 by referencing other views or tables, and the definition of the view is the only thing that is physically stored in the DB2 Catalog. When a user references a view, DB2 assembles the data from the underlying tables and views according to the definition. It is essentially transparent to the user whether the base table or logical view is being used. Views are a powerful tool used in relational databases to simplify SQL coding and can be used as a security device to restrict which columns of a table a user can access.

Value: The intersection of a column and a row which is the smallest unit of data that can be retrieved or changed.

VSAM (Virtual Storage Access Method): A mass storage access method that contains logical rather than physical datasets.

Document Changes

Oct 2007
1.5.1 – Clarify usage of Not Null with Default for missing character values.
1.5.1 – Clarify TIMESTAMP permitted for date and time columns.
1.6.1 – Add reason codes as example of check constraint usage instead of code table.
1.7 – Dozen given as typical transition between check constraint and code table usage.
1.7.1 – Foreign Key Not Enforced allowed when using both check constraints and code tables.
3.1.1 – New Primary key constraint name pattern matching foreign key pattern. This had been undefined and was inconsistent.

Dec 2007
3.1 – Maximum Column name length reduced from 30 to 21 characters to support DCLGEN standard of prefixing 5 characters to identify tablespaces and –IND for null indicators. DB2 V7 18 character limit references removed.
5.2 – Dasd Estimates for all lifecycle changes listed as specific requirement, rather then being part of data and transaction load estimates.
5.2.1.3 Dasd Estimates repeated as requirement for development setup, and 10% maximum development size added from new lifecycle doc.

May 2008
Clean up for Adobe 8, and 508 compliance.
1.2.2, 1.3.2 STOGRPT1 is the standard storage group.
Remove 2.4 showing how to use SPUFI/QMF/SAS/Quickstart. These did not contain CMS specific standards and screen prints were not 508.
Image copy names moved up as section 3.2 from 3.5
3.3 Subsystem names MMA/Non-MMA idenfied.
5 Clarify purpose of review stages, and approval criteria.
6.1 Show BMC MODIFY as the primary modify utility, matching its current use as the more flexible version. Show IBM RUNSTATS matching its current use to provide full statistics.
Remove 6.2 How to Guide on BMC utilities. This did not contain any CMS specific standards and screen prints were not 508.
Add 6.2 Restrictions on Utilities with QREPed files.

3.1.1 add _DSCRD suffix as standard for check data discard tables.

June 2008

1.3.1 add Table controlled partitioning should be used with new partitioned tablespaces.

July 2008

2.3.1, 2.3.3, 5.2.1.5 Explicitly say EXPLAIN(YES) is required in addition to simply showing as part of the Bind sample. This has been enforced as part of the Endevor setup. Add that Aliases to common plan tables will be used in I,V,P stages.

Sept 2008

1.10.2 update Identity column with Generate Always should not be used as a key for other tables.
1.11 add section on Sequence Objects. Numbering of old sections 1.11 to 1.23 increased by insert.
3.1.1 add Sequence Object naming standard.

March 2009

1.16.1 Removed requirement of WLM for stored procedures. With V9, DB2 managed SPAS are obsolete, while Internal SQL procedures are allowed.
1.17 Set basic UDF standards to match Stored procedure standards pending further testing.
3.1.1 Updated Stored procedure naming standard, added UDF naming standard

Sept 2009

Cover Added CMS logo and full name to clarify the source of this doc when seen outside of CMS.
1., 1.1 Updated overview to apply to the document rather than the first section.

Mar 2010

3.1.1 Made Internal SQL Stored proc and UDF naming the same as other procs and UDFs to point back to the Endevor member where the definition is maintained.

Nov 2012

1.1 Update overview, clarify that data content is managed by application.
1.3 Remove simple tablespace restriction text, not supported as of V10.
1.4 Table and clone use same tablespace.
1.6 Update Function requirements for external/internal sql.
1.7 Add list of common CMS functions RRBTOSSA, SSATORRB.
1.20 Content Manager is CMS standard for unstructured data.
3.1.1 Add Naming convention for Auxillary object and clone tables. Limit columns names to 30 char unless shorter needed for DCLGEN for that app.
3.2 Doc standard names for adhoc application image copies.
5.1 Update migration overview including linkage of databases with Endevor stages, and GTL approval of database service requests for documenting changes.

Dec 2012
6.2 Primary key needed for QREP tables.
June 2013

3.1.1 3.1.4 Change production table owner to use the environment pattern from development rather than the single production owner for the subsystem. Ie DBPVM0P0 instead of DBA$DB1P. This will require adding aliases to share tables between applications. This makes production names consistant with the development lifecycle and will help with selecting tables in Data studio which uses Schema instead of database for listing tables. The use of the single production owner becomes grandfathered, no conversion of existing tables is planned. Table owners map to Racf groups so contain the DB prefix to avoid conflicts with other application group names.

June 2016
2.4 The DB2 sample programs should be called with non-version plan names ie DSNTEP2 rather than DSNTEP71

April 2017
1.3 Tablespaces should be defined as universal tables spaces (UTS) as this is the announce direction of IBMs future tablespace support, and provides updated support for large tables and features such as clone tables. Remove reference to LOCKPART YES as this no longer needed to trigger selective partition locking. Make PAGE locking preferred over ANY to highlight apps not doing locking.
1.4.2 Tables, Partition by Range required for UTS PBR.

Sept 2020
1.3 Clarify use of PBG for small tables not expected to grow beyond 4G, IE segmented TS replacement.