DB2 10.5 for Linux, UNIX, and Windows

DB2 native encryption

Last Updated: 2021-03-01

DB2® native encryption encrypts your DB2 database, requires no hardware, software, application, or schema changes, and provides transparent and secure key management. 30

Encryption is the process of transforming data into an unintelligible form in such a way that the original data either cannot be obtained or can be obtained only by using a decryption process. It is an effective way of protecting sensitive information that is stored on media or transmitted through untrusted communication channels. Encryption is mandatory for compliance with many government regulations and industry standards.

In an encryption scheme, the data requiring protection is transformed into an unreadable form by applying a cryptographic algorithm and an encryption key. A *cryptographic algorithm* is a mathematical function that is used in encryption and decryption processes. An *encryption key* is a sequence that controls the operation of a cryptographic algorithm and enables the reliable encryption and decryption of data.

Some data encryption solutions for protecting data at rest are suitable in cases of physical theft of disk devices, and some can protect against privileged user abuse. With *native database encryption*, the database system itself encrypts the data before it calls the underlying file system to write that data to disk. This means that not only your current data is protected, but also data in new table space containers or table spaces that you might add in the future. Native database encryption is suitable for protecting data in cases of either physical theft of disk devices or privileged user abuse.

A local or external key manager is typically used to manage the keys. A *database data encryption key* (DEK) is the encryption key with which actual user data is encrypted. A *master key* is a "key encrypting key": It is used to protect the DEK. Although the DEK is stored and managed by the database, the master key is stored and managed outside of the database.

These keys are shown in Figure 1, which provides an overview of DB2 native encryption.

Figure 1. An overview of DB2 native encryption

DATA_MASK scalar function

Last Updated: 2024-10-30

You can use the DATA_MASK built-in scalar function to do partial redaction, format preserving masking, strong cryptographic obfuscation, and integrity preserving tokenization. The function produces masked data that can be used by analytics tools to create insights without revealing sensitive data. You use the DATA_MASK function in the masking definition of a CREATE MASK statement.

Note: This scalar function is currently available on Db2® 11.5.9 and later for the following 64-bit Linux platforms:
- AMD (linuxamd64).
- Z Linux (linux390x64).
- Power PC (linuxppcle64).

The schema is SYSIBM.

DATA_MASK

The data type of the returned value is the same as the data type of the expression.
The DATA_MASK scalar function supports only Unicode databases.

Note: The masked output produced by the DATA_MASK scalar function might change as the masking algorithms evolve and the underlying **securehash** function is upgraded or changed.

expression

An expression that returns a value of any built-in data type that is not CLOB, BLOB, DBCLOB, VARGRAPHIC, GRAPHIC, and XML.

mask-type

An expression that specifies masking operation to be performed. See Table 1 for available masking operations. The expression must return a value that is a built-in INTEGER.

Note: If the value is not of type INTEGER, it is implicitly cast to INTEGER before evaluating the function.

Table 1. Valid mask types

INT value	Masking Name	Target Data-Type	Description
0	REDACT	VARCHAR & CHAR. All other data-types redacted to a default value	Performs full redaction on input with the string value supplied by the `mask-parameters` attribute.
1	REDACT PRESERVE LENGTH	VARCHAR & CHAR. All other data-types redacted to default value.	First character in `mask-parameters` is a redaction character. Masked output is a redaction character repeated for the length of the input value.
2	SUBSTITUTE	All supported data-types	For VARCHAR and CHAR, seeded secureHash(SHA-256) and following base64 encoding is done. Column length must be at least 43 bytes to avoid collisions. For all other data types, obfuscation is performed. If a `seed` value is not provided, the default redaction value for the data type is returned.
3	PARTIAL REDACT	VARCHAR & CHAR. All other data-types redacted to default corresponding value	Advance redaction using patterns that are supplied as regular expressions or a string slice expression by the `mask-parameters` attribute.