Tabel Fungsi Enkripsi

| **Name** | **Description** |
| --- | --- |
| [AES\_DECRYPT()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_aes-decrypt) | Decrypt using AES |
| [AES\_ENCRYPT()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_aes-encrypt) | Encrypt using AES |
| [COMPRESS()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_compress) | Return result as a binary string |
| [DECODE()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_decode) | Decodes a string encrypted using ENCODE() |
| [DES\_DECRYPT()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_des-decrypt) | Decrypt a string |
| [DES\_ENCRYPT()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_des-encrypt) | Encrypt a string |
| [ENCODE()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_encode) | Encode a string |
| [ENCRYPT()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_encrypt) | Encrypt a string |
| [MD5()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_md5) | Calculate MD5 checksum |
| [OLD\_PASSWORD()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_old-password) | Return the value of the pre-4.1 implementation of PASSWORD |
| [PASSWORD()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_password) | Calculate and return a password string |
| [SHA1(), SHA()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_sha1) | Calculate an SHA-1 160-bit checksum |
| [SHA2()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_sha2) | Calculate an SHA-2 checksum |
| [UNCOMPRESS()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_uncompress) | Uncompress a string compressed |
| [UNCOMPRESSED\_LENGTH()](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_uncompressed-length) | Return the length of a string before compression |

Banyak fungsi enkripsi dan kompresi mengembalikan string yang hasilnya mungkin berisi nilai byte yang berubah-ubah. Jika ingin menyimpan hasilnya, gunakan kolom dengan tipe data string biner VARBINARY atau BLOB. Ini akan menghindari masalah potensial dengan penghapusan spasi atau konversi kumpulan karakter yang akan mengubah nilai data, seperti dapat terjadi jika menggunakan tipe data string nonbinary (CHAR, VARCHAR, TEXT).

Beberapa fungsi enkripsi mengembalikan string karakter ASCII: MD5 (), OLD\_PASSWORD (), PASSWORD (), SHA (), SHA1 (). Nilai kembalian dari fungsi tersebut adalah string yang memiliki set karakter dan pengumpulan ditentukan oleh variabel character\_set\_connection dan collation\_connection sistem. Ini adalah string non-biner kecuali set karakternya biner.

Jika sebuah aplikasi menyimpan nilai dari fungsi seperti MD5 () atau SHA1 () yang mengembalikan string digit hex, penyimpanan dan perbandingan yang lebih efisien dapat diperoleh dengan mengonversi representasi hex ke biner menggunakan UNHEX () dan menyimpan hasilnya dalam BINARY (N) kolom. Setiap pasangan digit heksadesimal membutuhkan satu byte dalam bentuk biner, sehingga nilai N tergantung pada panjang string hex. Nis 16 untuk nilai MD5 () dan 20 untuk nilai SHA1 (). Untuk SHA2 (), N berkisar 28 hingga 32 tergantung pada argumen yang menentukan panjang bit yang diinginkan.

Ukuran untuk menyimpan string hex dalam kolom CHAR setidaknya dua kali, hingga delapan kali jika nilai disimpan dalam kolom yang menggunakan set karakter utf8 (di mana setiap karakter menggunakan 4 byte). Menyimpan string juga menghasilkan perbandingan yang lebih lambat karena nilai-nilai yang lebih besar dan kebutuhan untuk mengambil aturan kumpulan karakter ke dalam akun.

**Latihan 1:**

1. Buatlah sebuah database.
2. Misalkan suatu aplikasi menyimpan nilai string MD5 () dalam kolom CHAR (32):

CREATE TABLE akun (password CHAR(32));

INSERT INTO akun (password) VALUES(MD5('abcdef'));

1. Buatlah tabel baru. Untuk mengkonversi string hex ke bentuk yang lebih ringkas, ubah aplikasi untuk menggunakan UNHEX () dan BINARY (16) sebagai gantinya:

CREATE TABLE akun1 (password BINARY(16));

INSERT INTO akun1 (password) VALUES(UNHEX(MD5('abcdef')));

Apa perbedaan dari kedua tabel berikut?

Password atau nilai sensitif lainnya yang disediakan sebagai argumen untuk fungsi enkripsi dikirim dalam cleartext ke server MySQL kecuali koneksi SSL digunakan. Nilai-nilai tersebut akan muncul di setiap log MySQL yang ditulis. Untuk menghindari hal ini, mysql dapat mengenkripsi nilai sensitif di sisi klien sebelum mengirimnya ke server. Pertimbangan yang sama berlaku untuk kunci enkripsi. Untuk menghindari agar kunci tidak dapat dibaca, mysql dapat menggunakan prosedur yang tersimpan untuk mengenkripsi dan mendekripsi nilai pada sisi server.

* [AES\_DECRYPT(***crypt\_str***,***key\_str***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_aes-decrypt)

Fungsi ini mendekripsi data menggunakan algoritma AES (Advanced Encryption Standard) resmi. Untuk informasi lebih lanjut, lihat uraian AES\_ENCRYPT ().

* [AES\_ENCRYPT(***str***,***key\_str***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_aes-encrypt)

AES\_ENCRYPT () dan AES\_DECRYPT () mengimplementasikan enkripsi dan dekripsi data menggunakan algoritma AES resmi (Advanced Encryption Standard), yang sebelumnya dikenal sebagai "Rijndael." Standar AES memungkinkan berbagai key length. Fungsi-fungsi ini menerapkan AES dengan key length 128-bit, tetapi dapat diperpanjang hingga 256 bit dengan memodifikasi sumber. Key length adalah trade off antara kinerja dan keamanan.

AES\_ENCRYPT () mengenkripsi string str menggunakan key string key\_str dan mengembalikan string biner yang berisi output terenkripsi. AES\_DECRYPT () mendekripsi crypt\_str string yang dienkripsi menggunakan key string key\_str dan mengembalikan string cleartext asli. Jika salah satu argumen fungsi NULL, fungsi mengembalikan NULL.

Argumen str dan crypt\_str dapat berupa panjang, dan secara otomatis ditambahkan ke str sehingga merupakan kelipatan dari blok seperti yang dipersyaratkan oleh algoritma berbasis blok seperti AES. Lalu secara otomatis dihapus oleh fungsi AES\_DECRYPT (). Panjang crypt\_str dapat dihitung menggunakan rumus ini:

16 \* (trunc(string\_length / 16) + 1)

**Latihan 2:**

Untuk key length 128 bit, cara paling aman untuk melewatkan kunci ke argumen key\_str adalah membuat nilai 128-bit yang benar-benar acak dan menyebarkannya sebagai nilai biner.

1. Buatlah tabel coba dengan field kata yang bertipe data text, lalu ketikkan kode berikut:

INSERT INTO coba

VALUES (AES\_ENCRYPT('text',UNHEX('F3229A0B371ED2D9441B830D21A390C3')));

1. Sebuah frasa kata dapat digunakan untuk menghasilkan kunci AES dengan mencampur frasa kata seperti:

INSERT INTO coba

VALUES (AES\_ENCRYPT('text', UNHEX(SHA2('My secret passphrase',512))));

Jika AES\_DECRYPT () mendeteksi data yang tidak valid atau padding yang salah, ia mengembalikan NULL. Namun, dimungkinkan untuk AES\_DECRYPT () mengembalikan nilai non-NULL jika data input atau kunci tidak valid.

* [COMPRESS(***string\_to\_compress***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_compress)

Mengompresi string dan mengembalikan hasilnya sebagai string biner. Fungsi ini membutuhkan MySQL untuk dikompilasi dengan pustaka kompresi seperti zlib. Jika tidak, nilai kembaliannya selalu NULL. String yang dikompresi dapat dikompresi dengan UNCOMPRESS ().

**Latihan 3:**

Ketikkan perintah berikut

1. SELECT LENGTH(COMPRESS(REPEAT('a',1000)));
2. SELECT LENGTH(COMPRESS(''));
3. SELECT LENGTH(COMPRESS('a'));
4. SELECT LENGTH(COMPRESS(REPEAT('a',16)));

Bagaimana hasilnya?

Isi string yang dikompresi disimpan dengan cara berikut:

1. String kosong disimpan sebagai string kosong.
2. String yang tidak kosong disimpan sebagai 4-byte panjang string yang tidak dikompresi (byte rendah pertama), diikuti oleh string yang dikompresi. Jika string berakhir dengan ruang, ekstra. karakter ditambahkan untuk menghindari masalah dengan pemangkasan akhir agar hasilnya disimpan dalam kolom CHAR atau VARCHAR. (Namun, penggunaan tipe data string nonbinary seperti CHAR atau VARCHAR untuk menyimpan string yang dikompresi tidak disarankan karena konversi set karakter dapat terjadi. Gunakan kolom string VARBINARY atau BLOBbinary sebagai gantinya.)

* [DECODE(***crypt\_str***,***pass\_str***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_decode)

Mendekripsi crypt\_str string terenkripsi menggunakan pass\_str sebagai kata sandi. crypt\_str harus berupa string yang dikembalikan dari ENCODE ().

* [DES\_DECRYPT(***crypt\_str***[,***key\_str***])](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_des-decrypt)

Mendekripsi string yang dienkripsi dengan DES\_ENCRYPT (). Jika kesalahan terjadi, fungsi ini mengembalikan NULL.

Fungsi ini berfungsi hanya jika MySQL telah dikonfigurasi dengan dukungan SSL. Jika tidak ada argumen key\_str yang diberikan, DES\_DECRYPT () memeriksa byte pertama dari string terenkripsi untuk menentukan nomor kunci DES yang digunakan untuk mengenkripsi string asli, dan kemudian membaca kunci dari file kunci DES untuk mendekripsi pesan. Agar ini berfungsi, pengguna harus memiliki hak istimewa SUPER. File kunci dapat ditentukan dengan opsi server -des-key-file.

Jika meneruskan fungsi ini argumen key\_str, string itu digunakan sebagai kunci untuk mendekripsi pesan.

Jika argumen crypt\_str, tidak menjadi string terenkripsi, MySQL mengembalikan crypt\_str yang diberikan.

* [DES\_ENCRYPT(***str***[,{***key\_num***|***key\_str***}])](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_des-encrypt)

Kunci enkripsi yang digunakan akan dipilih berdasarkan argumen kedua DES\_ENCRYPT (). Tanpa argumen, kunci pertama dari file kunci DES akan digunakan. Dengan argumen key\_num, nomor kunci yang diberikan (0 hingga 9) dari file kunci DES yang digunakan. Dengan argumen key\_str, string kunci yang diberikan digunakan untuk mengenkripsi str.

File kunci dapat ditentukan dengan opsi server -des-key-file.

Return string adalah string biner di mana karakter pertama adalah CHAR (128 | key\_num). Jika kesalahan terjadi, DES\_ENCRYPT () mengembalikan nilai NULL.

Setiap nilai key\_num harus berupa angka dalam rentang dari 0 hingga 9. Baris dalam file mungkin dalam urutan apa pun. des\_key\_str adalah string yang digunakan untuk mengenkripsi pesan. Harus ada setidaknya satu spasi antara nomor dan kuncinya. Kunci pertama adalah kunci default yang digunakan jika tidak menentukan argumen kunci apa pun ke DES\_ENCRYPT ().

Dapat memberi tahu MySQL untuk membaca nilai kunci baru dari file kunci dengan pernyataan FLUSH DES\_KEY\_FILE. Ini membutuhkan RELOADPrivilege.

Salah satu manfaat memiliki satu set kunci default adalah memberikan cara untuk memeriksa keberadaan nilai kolom terenkripsi, tanpa memberikan hak kepada pengguna akhir untuk mendekripsi nilai-nilai tersebut.

Contoh:

SELECT customer\_address FROM customer\_table WHERE crypted\_credit\_card = DES\_ENCRYPT('credit\_card\_number');

* [ENCODE(***str***,***pass\_str***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_encode)

Enkripsi str menggunakan pass\_str sebagai kata sandi. Hasilnya adalah string biner dengan panjang yang sama dengan str. Untuk mendekripsi hasilnya, gunakan DECODE ().

Fungsi ENCODE () seharusnya tidak lagi digunakan. Jika masih perlu menggunakan ENCODE (), nilai harus digunakan bersamaan untuk mengurangi risiko.

Contoh:

ENCODE('cleartext', CONCAT('my\_random\_salt','my\_secret\_password'))

* ENCRYPT(str[,salt])

Mengenkripsi str menggunakan crypt crypt Unix () system call dan mengembalikan string biner. Argumen harus berupa string dengan setidaknya dua karakter atau hasilnya akan menjadi NULL. Jika tidak ada argumen yang diberikan, nilai acak digunakan.

ENCRYPT () mengabaikan semua kecuali delapan karakter pertama dari str, setidaknya pada beberapa sistem. Perilaku ini ditentukan oleh implementasi panggilan sistem crypt () yang mendasarinya.

Penggunaan ENCRYPT () dengan set karakter multibyte ucs2, utf16, atau utf32 tidak disarankan karena panggilan sistem mengharapkan string diakhiri oleh nol byte.

Jika crypt () tidak tersedia di sistem Anda (seperti halnya dengan Windows), ENCRYPT () selalu mengembalikan NULL

* [MD5(***str***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_md5)

Menghitung MD5 128-bit untuk string. Nilai dikembalikan sebagai string 32 digit heksadesimal, atau NULL jika argumennya NULL. Nilai kembalian dapat digunakan, misalnya, digunakan sebagai kunci hash. Nilai kembalian adalah string dalam set karakter koneksi.

**Latihan 4:**

Cobalah fungsi berikut dan lihat hasilnya:

1. SELECT MD5('testing');
2. SELECT MD5('hello');
3. SELECT MD5('hello world');

* [OLD\_PASSWORD(***str***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_old-password)

OLD\_PASSWORD () ditambahkan ketika implementasi PASSWORD () diubah di MySQL 4.1 untuk meningkatkan keamanan.OLD\_PASSWORD () mengembalikan nilai implementasi PASSWORD pra-4.1 () sebagai string, dan dimaksudkan untuk mengizinkan agar dapat diubah ulang kata sandi untuk klien pra-4.1 yang perlu terhubung ke server MySQL.

* [PASSWORD(***str***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_password)

Mengembalikan string kata sandi hash yang dihitung dari str kata sandi cleartext. Nilai kembalian adalah string dalam set karakter koneksi, atau NULL jika argumennya adalah NULL. Fungsi ini adalah antarmuka SQL ke algoritma yang digunakan oleh server untuk mengenkripsi password MySQL.

Variabel sistem old\_passwords mengontrol metode hashing kata sandi yang digunakan oleh fungsi PASSWORD (). Ini juga mempengaruhi hashing kata sandi yang dilakukan oleh CREATE USER dan GRANT statement yang menentukan kata sandi menggunakan klausa IDENTIFIKASI BY.

Tabel berikut menunjukkan, untuk setiap metode hashing kata sandi, nilai yang diizinkan dari kata\_password lama dan plugin otentikasi mana yang menggunakan metode hashing.

| **Password Hashing Method** | **old\_passwords Value** | **Associated Authentication Plugin** |
| --- | --- | --- |
| MySQL 4.1 native hashing | 0 or OFF | mysql\_native\_password |
| Pre-4.1 (“old”) hashing | 1 or ON | mysql\_old\_password |

**Latihan 5:**

1. Ketikkan perintah berikut lalu lihat hasilnya

SET old\_passwords = 0;

SELECT PASSWORD('mypass'), OLD\_PASSWORD('mypass');

1. Ketikkan perintah berikut lalu lihat hasilnya

SET old\_passwords = 1;

mysql> SELECT PASSWORD('mypass'), OLD\_PASSWORD('mypass');

Apakah perbedaan diantara perintah tersebut?

* [SHA1(***str***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_sha1), [SHA(***str***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_sha1)

Perhitungan SHA-1 160-bit untuk string, seperti yang dijelaskan dalam RFC 3174 (Secure Hash Algorithm). Nilai dikembalikan sebagai string 40 digit heksadesimal, atau NULL jika argumennya NULL. Salah satu kegunaan yang mungkin untuk fungsi ini adalah sebagai kunci hash Nilai kembalian adalah string dalam set karakter koneksi.

**Latihan 6:**

Cobalah perintah berikut:

1. SELECT SHA1('abc');
2. SELECT SHA1('hello');
3. SELECT SHA1('my password');

SHA1 () dapat dianggap setara dengan MD5 () yang secara kriptografi lebih aman.

* [SHA2(***str***, ***hash\_length***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_sha2)

Perhitungan SHA-2 fungsi hash (SHA-224, SHA-256, SHA-384, dan SHA-512). Argumen pertama adalah string cleartext yang akan di-hash. Argumen kedua menunjukkan panjang bit yang diinginkan dari hasil, yang harus memiliki nilai 224, 256, 384, 512, atau 0 (yang setara dengan 256). Jika salah satu argumen adalah NULL atau panjang hash bukan salah satu dari nilai yang diizinkan, nilai kembalian adalah NULL. Jika tidak, hasil fungsi adalah nilai hash yang berisi jumlah bit yang diinginkan. Nilai kembalian adalah string dalam set karakter koneksi.

**Latihan 7:**

Cobalah perintah berikut:

1. SELECT SHA2('abc', 224);
2. SELECT SHA2('hello', 256);
3. SELECT SHA2('my password', 384);

SHA2 () dapat dianggap lebih aman secara kriptografi daripada MD5 () atau SHA1 ().

* [UNCOMPRESS(***string\_to\_uncompress***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_uncompress)

Uncompresses string yang dikompresi oleh fungsi COMPRESS (). Jika argumennya bukan nilai terkompresi, hasilnya adalah NULL. Fungsi ini membutuhkan MySQL untuk dikompilasi dengan pustaka kompresi seperti zlib. Jika tidak, nilai kembaliannya selalu NULL.

* [UNCOMPRESSED\_LENGTH(***compressed\_string***)](https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html#function_uncompressed-length)

Mengembalikan panjang yang telah dikompresi sebelum kompresi.

Contoh:

SELECT UNCOMPRESSED\_LENGTH(COMPRESS(REPEAT('a',30)));