**Tehtävä 12 Komennot**

**Mitkä tietotyypit SQL:ssä on luvuille? Anna myös kunkin tietotyypin arvoalue.**

**TINYINT**

Käyttö: käytetään positiivisten ja negatiivisten kokonaislukujen tallentamiseen.  
Tavoite: -128–127  
Koko: 1 tavu (8 bittiä)

**SMALLINT**

Käyttö: käytetään positiivisten ja negatiivisten kokonaislukujen tallentamiseen.  
Kattavuus: -32 768–32 767  
Koko: 2 tavua (16 bittiä).

**MEDIUMINT**

Käyttö: käytetään positiivisten ja negatiivisten kokonaislukujen tallentamiseen.  
Kattavuus: -8 388 608–8 388 607  
Koko: 3 tavua (24 bittiä)

**INT**

Käyttö: käytetään positiivisten ja negatiivisten kokonaislukujen tallentamiseen.  
Kattavuus: -2.147.483.648–2.147.483.647  
Koko: 4 tavua (32 bittiä).

**bigint**

Käyttö: käytetään positiivisten ja negatiivisten kokonaislukujen tallentamiseen.  
Kattavuus: ± 9,22 x 1018  
Koko: 8 tavua (64 bittiä)

**FLOAT**

Käyttö: käytetään yksittäisten tarkkuuksien positiivisten ja negatiivisten fraktiotietojen tallentamiseen.  
Reach: 3.402823466E + 38 - -1.175494351E-38, 0 ja 1.175494351E-38 - 3.402823466E + 38.  
Koko: 4 tavua (32 bittiä).

**TUPLO / TODELLINEN**

Käyttö: käytetään kaksinkertaisen tarkkuuden positiivisten ja negatiivisten fraktiotietojen tallentamiseen.  
Reach: -1,79 ... E + 308 - -2,22 ... E-308, 0 ja 2,22 ... E-308 - 1,79 ... E + 308.  
Koko: 8 tavua (64 bittiä).

**DECIMAL / NUMERO**

Käyttö: käytetään positiivisten ja negatiivisten fraktiotietojen tallentamiseen.  
Reach: -1,79 ... E + 308 - -2,22 ... E-308, 0 ja 2,22 ... E-308 - 1,79 ... E + 308.  
Koko: 8 tavua (64 bittiä).

**Milloin SQL antaa virheilmoituksen ”out of range value”?**

Ylittää tietotyypin pituuden.

**Miksi UNSIGNED-luku voi olla huomattavasti suurempi kuin saman tietotyypin signed-luku?**

Samantyyppisillä UNSIGNED ja SINGNED varustetuilla muuttujilla (kuten int ja byte) on molemmilla sama alue (alue 65 536 ja 256 numeroa, vastaavasti), mutta UNSIGNED luvut voivat edustaa suurempaa suuruuslukua kuin vastaava etumerkillinen muuttuja.

Esimerkiksi UNSIGNED voi edustaa arvoja välillä 0–255, kun taas SIGNED voi edustaa -128–127.

**Miten varchar tietotyyppi eroaa char tyypistä?**

CHAR on kiinteäpituinen merkkijonotietotyyppi, joten kaikki kentässä jäljellä oleva tila on täytetty tyhjillä. CHAR vie 1 tavun per merkki. CHAR (100) -kenttä (tai muuttuja) vie siis 100 tavua levyltä riippumatta siinä olevasta merkkijonosta.

VARCHAR on muuttuvapituinen merkkijonotietotyyppi, joten se sisältää vain sille määrittämäsi merkit. VARCHAR vie 1 tavun merkkiä kohden, + 2 tavua pituustietojen säilyttämiseen. Jos esimerkiksi asetat VARCHAR (100) -tietotyypiksi = 'Jen', se vie 3 tavua (J, E ja N) plus 2 tavua tai yhteensä 5 tavua.

**Entä millainen tietotyyppi on varbinary?**

VARBINARY tallentaa arvot vaihtelevan pituisina. Arvoja ei ole täytetty millään tavulla.

**Anna esimerkkejä teksti – ja binäärityyppisistä tiedoista.**

BINARY, VARBINARY & VARBINARY (max) ovat binäärimerkkijonotietotyyppejä SQL Serverissä. Näitä tietotyyppejä käytetään binääriraakatietojen tallentamiseen enintään (32 kt – 1) tavun pituiseksi. Kuvatiedostojen (BMP-, TIFF-, GIF- tai JPEG-muotoiset tiedostot), Word-tiedostot, tekstitiedostot jne. sisältö ovat esimerkkejä binääritiedoista.

**Mitkä kaksi komentotapaa on luoda perusavain? Kirjoita syntaksit.**

Jos taulua ei ole vielä luotu:

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName)  
);

Taulu on jo olemassa:

ALTER TABLE Persons  
ADD PRIMARY KEY (ID);

**Mikä on perusavaimen merkitys?**

Pääavaimen tarkoituksena on olla uniikki arvo taulussa olevien rivien erottamiseksi toisistaan. Yhdessä taulussa ei voi olla useita samanarvon sisältäviä pääavaimia. Jokaisella taululla on oltava pääavain ja niitä voi olla vain yksi per taulu. Pääavaimen arvo ei voi olla NULL eli tyhjä.

**Entä mikä on viiteavaimen (foreign key) tehtävä?**

Kun taulujen välisiä suhteita kuvataan niin tauluihin pitää pystyä viittaamaan. Päätarkoitus viiteavaimella on toimia tietokannan näkökulmasta tiedon eheyden varmistuksessa. Tämä tarkoittaa sitä, että kun tietoja poistetaan, lisätään tai päivitetään, tulisi viittausten pysyä toimivina.

**Miten viiteavain määritetään SQL-komennolla? Kirjoita myös parametrina annettavat viite-eheyssäännöt ja niiden merkitys.**

CONSTRAINT [viiteavaimen\_nimi] FOREIGN KEY ([viiteavain\_sarake]) REFERENCES [viitattava\_taulu] ([viitattavan\_taulun\_pääavain])

**Millä toisella tavalla voit määrittää kentän yksilölliseksi PRIMARY KEYn lisäksi?**

UNIQUE-rajoite varmistaa, että kaikki sarakkeen arvot ovat erilaisia.

**Mitä hyötyä on kentän indeksoinnista? Millainen kenttä kannattaa indeksoida?**

* Se auttaa sinua vähentämään tietojen hakemiseen tarvittavien I/O-toimintojen kokonaismäärää, joten sinun ei tarvitse käyttää tietokannan riviä hakemistorakenteesta.
* Tarjoaa käyttäjille nopeamman tiedonhaun ja -haun.
* Indeksointi auttaa myös vähentämään taulukkotilaa, koska sinun ei tarvitse linkittää taulukon riville, koska ROWID-tunnusta ei tarvitse tallentaa hakemistoon. Näin voit vähentää pöytätilaa.
* Et voi lajitella tietoja johtosolmuissa, koska ensisijaisen avaimen arvo luokittelee sen.

**Mitä tarkoittavat seuraavat kentän määritteet**

**Not Null:** Tämä pakottaa kentän sisältämään aina arvon, mikä tarkoittaa, että et voi lisätä uutta tietuetta tai päivittää tietuetta lisäämättä arvoa tähän kenttään.

**Auto\_increment:** Mahdollistaa yksilöllisen numeron luomisen automaattisesti, kun uusi tietue lisätään taulukkoon.

**Default:** Käytetään sarakkeen oletusarvon asettamiseen. Oletusarvo lisätään kaikkiin uusiin tietueisiin, jos muuta arvoa ei ole määritetty.

**Zerofill:** Täyttää kentän näytettävän arvon nollilla sarakemäärityksessä asetettuun näytön leveyteen asti.

+--------------------+-----------+

| First | Second |

+--------------------+-----------+

| 000000000000000001 | 1 |

| 000000000000000012 | 12 |

| 000000000000000123 | 123 |

| 000000000123456789 | 123456789 |

+--------------------+-----------+