Tehtävä 12 Komennot

Selvitä netin sql-sivustoja hyödyntäen

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

BIT (arvo 1 – 64)

TINYINT (arvo -128 – 127 tai 0 – 255)

BOOL (arvo 0 = false, ei nolla = true)

BOOLEAN (arvo vastaa BOOL arvoa)

SMALLINT (arvo -32768 – 32767 tai 0 – 65535)

MEDIUMINT (arvo -8388608 – 8388607 tai 0 – 16777215)

INT (arvo -2147483648 – 2147483647 tai 0 – 4294967295)

BIGINT (arvo -9223372036854775808 – 9223372036854775807 tai 0 – 18446744073709551615)

FLOAT (arvo 0 – 24 numeroa ennen desimaalipilkkua)

DOUBLE (arvo 25 – 53 numeroa ennen desimaalipilkkua)

DECIMAL (arvo max 65 numeroa ennen desimaalipilkkua ja max 30 numeroa pilkun jälkeen)

Milloin SQL antaa virheilmoituksen ”out of range value”?

Kun syötetään johonkin sarakkeeseen arvo, joka ei pysy annetun datatyypin sisällä.

Esimerkiksi TINYINT määritettyyn sarakkeeseen syötetään 655 luku.

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

Koska unsigned- luvussa ei ole miinusmerkkisiä lukuja. Esimerkkinä menköön SMALLINT (arvo -32768 – 32767 tai 0 – 65535).

Miten varchar tietotyyppi eroaa char tyypistä?

CHAR, on kiinteämittainen merkkijono, joka on täsmälleen annetun pituuden mittainen. Jos **CHAR**-tyyppiseen kenttään syötetään vähemmän tietoa, kuin sille on varattu tilaa, niin jäljelle jäänyt tila kentästä täytetään välilyönneillä.

Esimerkiksi haluttaessa määritellä kenttä, johon voidaan täyttää henkilön henkilötunnus niin tietotyypiksi kannattaa valita **CHAR**, koska henkilötunnushan on aina tietyn mittainen merkkijono (11 merkkiä).

VARCHAR, on vaihtuvamittainen merkkijono, joka on maksimissaan annetun pituuden mittainen.

Esimerkiksi tallennettaessa henkilön etunimi ja sukunimi niin kenttien tyypiksi sopii erinomaisesti **VARCHAR**. Etunimien ja sukunimien pituushan vaihtelee eri ihmisillä.

Entä millainen tietotyyppi on varbinary?

Tallentaa binaariset tavujonot, joiden sarakkeen pituus määritetään kokoparametrilla. Sarakkeen kokoa ei ole vahvistettu. Jos koko on määritelty 10 tavuksi ja tallennettu data on 5 tavua, sarake vie vain 5 tavua muistiin. Suurin mahdollinen koko on 8000 tavua.

Mitkä kaksi komentotapaa on luoda perusavain? Kirjoita syntaksit.

TAPA 1:

```
CREATE TABLE [kaupungit](
 [Id] INT IDENTITY,
 [nimi] VARCHAR (30) DEFAULT NULL,
 [asukasluku] INT DEFAULT NULL,
[pinta-ala] DECIMAL (10,2) DEFAULT NULL,
PRIMARY KEY ([Id])
 );
```

TAPA 2:

```
CREATE TABLE product (
[id] INT PRIMARY KEY,
[name] VARCHAR(100) NOT NULL,
```

[producer] VARCHAR(100) NOT NULL,

[price] DECIMAL(7,2)

);

Mikä on perusavaimen merkitys?

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

Pääavain voi koostua yhdestä tai useammasta taulun sisältämistä sarakkeista. Arvon ei siis tarvitse olla pelkkä numero vaan voi sisältää muitakin arvoja. Usein voidaan nähdä nousevasti laskettu numeroarvo (1,2,3,4,5...), joka toimii pääavaimena. Suositus on kuitenkin, että pääavain olisi mahdollisimman luonnollinen. Esimerkiksi henkilötunnus voisi toimia pääavaimena taulussa, jossa on kuvattu henkilöitä.

Entä mikä on viiteavaimen (foreign key) tehtävä?

Kun taulujen välisiä suhteita kuvataan niin tauluihin pitää pystyä viittaamaan. Pääavain toimii aina uniikkina avaimena ja siksi voi toimia toisessa taulussa viiteavaimena. Viiteavaimella voidaan sanoa olevan seuraavia ominaisuuksia:

Taulu voi sisältää yhden tai useamman viiteavaimen toisiin tauluihin. Viiteavain voi myös viitata tauluun itseensä eikä erilliseen tauluun. Viiteavaimet ovat myös indeksoituja, jotta tietojen hakeminen nopeutuu. Viiteavaimena toimivan kentän ei tarvitse olla uniikki.

Päätarkoitus viiteavaimella on toimia tietokannan näkökulmasta tiedon eheyden varmistuksessa. Tämä tarkoittaa sitä, että kun tietoja poistetaan, lisätään tai päivitetään, tulisi viittausten pysyä toimivina. Esimerkiksi, jos poistamme tuoteryhmän niin sitä ei tulisi pystyä poistamaan, jos siihen on viitattu tuotteissa. Muutoin tuotteella ei olisi enää tuoteryhmää.

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

CONSTRAINT [FK_Suoritetut_tehtavat_Astronautit] FOREIGN KEY ([Astronautti_Id]) REFERENCES [dbo].[Astronautit] ([Astronautti_Id]),

CONSTRAINT [FK_Suoritetut_tehtavat_Tehtava] FOREIGN KEY ([Tehtava_Id]) REFERENCES [dbo].[Tehtava] ([Id])

Yllä olevat esimerkit ovat astronauttitehtävästä ja ylemmässä esimerkissä suoritetut_tehtävät- taulussa FOREIGN KEY- komennolla annetaan Astronautti_Id- sarakkeeseen viiteavain, jolla viitataan Astronautit-tauluun ja sen Astronautti_Id sarakkeeseen.

Millä toisella tavalla voit määrittää kentän yksilölliseksi PRIMARY KEYn lisäksi?

UNIQUE-määritteellä voidaan estää kaksoisarvojen esiintyminen jossakin kentässä. Jos lisäämme työntekijän tietoihin kentän sosiaaliturvatunnusta varten niin voimme **UNIQUE**-määritteellä varmistaa ettei ole mahdollista antaa kahdelle työntekijälle samaa henkilötunnusta.

Esimerkki UNIQUEN käytöstä:

```
CREATE TABLE Tyontekija (TyontekijaID INTEGER NOT NULL,
Etunimi VARCHAR(32) NOT NULL,
Sukunimi VARCHAR(64) NOT NULL,
Hetu CHAR(11) NOT NULL,
Palkka NUMERIC(7,2) NOT NULL DEFAULT 2000,
Syntymaaika DATE NOT NULL ,
Osasto INTEGER NOT NULL DEFAULT 1,
CONSTRAINT Tyontekija_PK
PRIMARY KEY (TyontekijaID),
CONSTRAINT Tyontekija_Hetu_U
UNIQUE (Hetu)
)
```

Mitä hyötyä on kentän indeksoinnista? Millainen kenttä kannattaa indeksoida?

Indeksi on tietorakenne, jonka avulla tietokannan hallintajärjestelmän on mahdollista hakea tiettyjä tietoja taulusta nopeammin, ja siten nopeuttaa vasteaikaa käyttäjän kyselyille (Connolly & Begg 2005). Indeksi määritellään yleensä niille tietokannan kentille, jotka toistuvat usein haku- tai lajitteluehdoissa. Tämä mahdollistaa sekä yksittäisten hakujen nopeuttamisen että suurtenkin tietomäärien järjestämisen. Määrittely voidaan tehdä käyttäen yhtä tai useampaa saraketta, ja relaatiotietokannoissa indeksi onkin usein kopio osasta tietokantataulua.

Mitä tarkoittavat seuraavat kentän määritteet

Not Null: Kenttään täytyy syöttää jokin arvo, eli se on pakollinen kenttä.

Auto_increment: Tarkoittaa, että aina kun tauluun lisätään uusi rivi, tähän kenttään tulee seuraava vapaana oleva tunnusnumero.

Default: Kentän oletusarvon määritys. Esimerkiksi DEFAULT NULL, oletusarvona tällöin nolla.

Zerofill: ZEROFILL täydentää kentän näytettävän arvon nollilla sarakemäärityksessä asetettuun näytön leveyteen asti. Esim arvo 2 voitaisiin näyttää 0000000002 jos tilaa on kymmenelle numerolle.