Tietokantojen perusteet (Avoin yliopisto), uusintakoe 17.9.2014

Vastaa tehtävät 1, 2, 3 ja 4 **erillisille** konsepteille. Kirjoita jokaiseen konseptiin kurssin nimi, kokeen päivämäärä, henkilötunnus, oma nimi ja nimikirjoitus.

Vaikka jättäisit johonkin tehtävään vastaamatta, tulee vastauspaperi siinäkin tapauksessa palauttaa.

1. Relaatiotietokannat pohjautuvat relaatiomalliin. Mitä relaatiomallilla ja tiedon relaatiolla tarkoitetaan? Mitä yleisiä relaatioyhteyksiä tietokannassa voidaan esittää ja miten nämä yhteydet toteutetaan tietokantaan? Anna esimerkkejä. (4p)

2. (5p) *Kumpulan omenapuutarha Oy* on päättänyt toteuttaa sähköisen järjestelmän tukemaan jokapäiväistä toimintaansa.

   > Kumpulan omenapuutarha on Kumpulan ytimessä sijaitseva suuri puutarha joka kasvattaa pääsääntöisesti omenoita yhdellä tiluksella. Omenapuita on tiluksella jo 10 000, ja lajikkeita on lukuisia.
   >
   > Omenapuulajikkeeseen liittyy nimi, tunnuspiirteet ja hoito-ohjeet. Omenapuulla on nimi, istutusaika, hedelmätuotannon alkamispäivä ja kasvupaikka. Tilus on jaettu alueisiin, joissa jokaisella puulla on neliön muotoinen paikka (esimerkiksi alue A, paikka 1). Kaikilla puilla ei välttämättä ole paikkaa. Jokaiseen alueeseen liittyy vähintään yksi vastuutarhaaja. Tarhalla on kerrallaan yksi päätarhaaja.
   >
   > Tarhassa työskentelee useita tarhaajia. Tarhaajista halutaan tallentaa vähintään henkilötunnus, etunimi, sukunimi, sähköposti, puhelinnumero ja tilinumero. Sen lisäksi tarhaajien työsuhteet tulee tallentaa järjestelmään. Oleellisimmat tiedot ovat työsuhteen alku- ja loppupäivä, työaika prosentuaalisesti sekä tuntipalkka.
   >
   > Tarhaajat kirjaavat järjestelmään omenapuille suoritettavista hoitotoimenpiteistä toimenpideraportteja. Toimenpideraportin kirjaa yksi tarhaaja, se liittyy yhteen puuhun ja sisältää kuvauksen toimenpiteestä. Toimenpiteestä on tärkeää tallentaa myös ajankohta.
   >
   > Sadonkorjuuta varten tarhalla on lista tilauksista. Tilauksista tallennetaan tilaaja, tilauspäivä, omenalajike, toivottu määrä sekä kilohinta. Sadonkorjuussa lasketaan jokaisen omenapuun vuotuinen tuottavuus ja nämä tiedot tallennetaan kyseisen puun sadonkorjuutilastoihin.

   Tee kuvaukselle tietosisältökartoitus. Luo kartoituksen pohjalta **UML-käsitekaavio** tietotarpeelle. Yhteyksien ja osallistumisrajoitteiden merkintä on oleellista, attribuuteista riittää oleellisimmat. **Pelkkä kaavio riittää vastaukseksi**.

3. *Kesälaitumien niittokoneet Ky* myy ruohonleikkureita. Alla on suunnitelma yrityksen tietokannasta.

   Tyyppi(id, nimi)
   Voimanlähde(id, nimi)
   Malli(id, tyyppi_id → Tyyppi, voimanlähde_id → Voimanlähde, mallinumero, nimi, hevosvoimat)
   Ruohonleikkuri(id, malli_id → Malli, sarjanumero, hyllypaikka)
   Hinta(id, malli_id → Malli, myyntihinta, ostohinta)
   Myynti(id, ruohonleikkuri_id → Ruohonleikkuri, myyntihinta, päivämäärä)

   Muodosta SQL-kyselyt, joilla seuraavat käyttötapaukset voidaan toteuttaa. *Sivulla kolme on lyhyt SQL-syntaksin referenssi.*

   (a) Hae kaikki ruohonleikkurit, joiden mallin nimi on ”SuperTurboCutter 3000” ja jotka sijaitsevat hyllypaikassa ”P4”. (0,5p)

   (b) Poista kaikki vialliset ruohonleikkurit, joiden sarjanumero alkaa merkkijonolla ”22”. (1p)

   (c) Laske kuinka monta ruohonleikkurimallia yrityksellä on myynnissä, joiden voimanlähde on ”vety” (voimanlähteen nimi). (0,5p)

   (d) Laske yrityksen tulos (myyntihintojen summa) heinäkuulle 2014 (1.7.–31.7.) kaikille ruohonleikkureille, joiden tyyppi on ”robotti” (tyypin nimi). Myyntihinta pitää tarkistaa aina Myynti-taulusta, sillä asiakas on saattanut saada alennusta. Päivämäärä on ilmaistu date-tyyppinä. (1p)

4. Selitä lyhyesti seuraavat tietokantoihin liittyvät käsitteet. Anna kustakin käsitteestä myös lyhyt esimerkki.

   (a) Tietokohteen attribuutti, arvo ja arvojoukko (1p)

   (b) Tietokantatransaktio (1p)

   (c) SQL-alikysely (1p)

   (d) SQL-injektio (1p)

Brief SQL reference

SELECT syntax:

```
SELECT [ALL | DISTINCT] select-list
   FROM table-reference-list
   [WHERE search-condition]
   [GROUP BY column-name [, column-name]... ]
   [HAVING search-condition]
   [[UNION | INTERSECT | EXCEPT] [ALL ] select-statement]...
   [ORDER BY {unsigned integer | column-name}
   [ASC|DESC]]

select-list ::= * | select-sublist [ {, select-sublist} ... ]
select-sublist ::=   derived-column | [table-name | table-identifier].*
derived-column ::=  expression [ [AS] column-alias] ]

table-reference-list ::= table-reference [, table-reference ... ]
table-reference ::=  table-name [[AS] correlation-name] |
   derived-table [[AS] correlation-name [(derived-column-list )]]  | joined-
table
table-name ::= table-identifier |   schema-name.table-identifier
derived-table ::=  subquery
derived-column-list ::= column-name-list
joined-table ::= cross-join | qualified-join | ( joined-table)
cross-join ::=  table-reference CROSS JOIN table-reference
qualified-join ::= table-reference [NATURAL] [join-type] JOIN
    table-reference [join-specification]
join-type ::=  INNER | outer-join-type [OUTER] | UNION
outer-join-type ::=  LEFT | RIGHT | FULL
join-specification ::= join-condition | named-columns-join
join-condition ::= ON search-condition
named-columns-join ::= USING (column-name-list)
column-name-list ::= column-identifier [ { , column-identifier} ...]

search-condition ::=  search-item | search-item { AND | OR }
    search-item
search-item ::= [NOT] { search-test | (search-condition) }
search-test ::= comparison-test | between-test | like-test |
    null-test | set-test | quantified-test | existence-test
comparison-test ::=
    expression { = | <> | < | <= | > | >= }  { expression | subquery }
between-test ::=  column-identifier [NOT] BETWEEN expression AND
expression
like-test ::=  column-identifier [NOT] LIKE value [ESCAPE value]
null-test ::=  column-identifier IS [NOT] NULL
set-test ::=  expression [NOT] IN ( {value [,value]... | subquery} )
quantified-test ::=
   expression { = | <> | < | <= | > | >= } [ALL | ANY | SOME]
      subquery
existence-test ::= EXISTS subquery
subquery ::= (query-specification)

expression ::=    expression-item | expression-item { + | - | * | / }
      expression-item
expression-item ::=    [ + | - ] { value | column-identifier | function |
    case-expression    | cast-expression | ( expression ) }
value ::= literal | USER | variable
function ::= set-function | string-function |
    numeric-function | datetime-function | system-function |
    datatypeconversion-function
set-function ::= COUNT (*) |
    { AVG | MAX | MIN | SUM | COUNT }( { ALL | DISTINCT }
     expression )
datatypeconversion-function ::=
   CONVERT_datatype(value-exp)   where datatype is SQL datatype
case-expression ::=   case-abbreviation | case-specification
case-abbreviation ::=
   NULLIF(value-exp, value-exp) | COALESCE(value-exp {, value-exp}...)
```

INSERT syntax:

```
insert into tablename [{column_list)]  { values ( value_list ) | subquery }
```

UPDATE syntax:

```
update tablename set assignment [, assignment ...]
[where search_condition]
assignment ::=   column_name = expression
```

DELETE syntax:

```
delete from tablename [where search_condition]

date_literal::= date ‘yyyy-mm-dd’
```