Kirjoitathan jokaiseen palauttamaasi paperin ylälaitaan selvästi 'T-106.5241, 19.12.2013', koko nimesi, opiskelijanumerosi, koulutusohjelmasi sekä montako paperia palautit yhteensä. Laskin on sallittu mutta sitä ei tarvita.
Please write on top of every answer sheet clearly 'T-106.5241, 19.12.2013', your full name, your student nr, your study program, and the total nr of sheets that you returned. Calculators are allowed but are not needed.

1. Määritä lyhyesti ja selkeästi seuraavat käsitteet a-d. *Explain briefly and clearly the following terms a-d.*
   (a) hajautettu tietokanta *distributed database*
   (b) keskinäinen johdonmukaisuus laiskassa toisintamisessa *mutual consistency in lazy replication*
   (c) johdettu vaakasuora ositus *derived horizontal partitioning*
   (d) yhdestä tietoalkiosta johtuva lukkiuma *one item deadlock*
   (e) Vertaile lyhyesti operaatioiden välistä rinnakkaisuutta ja operaationsisäistä rinnakkaisuutta ja anna kummastakin esimerkki.
   Briefly contrast interoperation parallelism and intraoperation parallelism giving an example of each one.

2. Osa tietokantakaaviostamme koostuu kahdesta relaatiosta, avaimet alleviivattuina:
   Part of our database schema consists of two relations, with keys underlined:
   `ORDERS (OrderId, DateOrdered, CustomerId, ProductId)`
   `PRODUCTS3 (ProductId, ProductName, ProductPrice, ProductColorId)`
   Relaatiossa ORDERS, pisteessä S_1, on 90 000 monikkoa. Relaatiossa PRODUCTS3, pisteessä S_2, on 100 monikkoa. Vain 10% relaation ORDERS monikoista viittaavat relaation PRODUCTS3 monikkoihin (tuotteisiin). *Relation ORDERS, at site S_1, has 90 000 tuples, and relation PRODUCTS3, at site S_2, has 100 tuples. Only* 10% *of the tuples in relation ORDERS refer to tuples (products) in PRODUCTS3.*
   Halutaan suorittaa pisteessä S_3 seuraava SQL kysely puoliliitoksen avulla:
   We want to run the following SQL-query at site S_3 using semi-join:

```
SELECT P.ProductId, P.ProductName, O.DateOrdered, O.CustomerId
FROM Products3 P, Orders O WHERE O.ProductId = P.ProductId
```

   (a) Halutaan laskea ensin puoliliitos pisteessä S_1. Mitä tulisi ensin lähettää pisteeseen S_1? Anna SQL-kysely. *We want to compute first the semi-join at Site S_1. What should be first shipped to S_1? Give the SQL-query.*
   (b) Olkoon F' monikkojoukko, joka lähetettiin pisteeseen S_1. Kirjoita SQL-Select lauseke puoliliitokselle, joka käyttää saatua F'. Mihin pisteeseen näin saadut tulokset lähetetään?
   Let F' be the tuples sent to S_1. Write the SQL-Select clause for a semi-join that makes use of F'. To what site do we send the result set thus obtained?
   (c) Olkoon F" edellisen puoliliitoksen tulosjoukko. Kirjoita lopullinen Select-lause, joka suoritetaan seuraavaksi, ja jossa hyödynnetään F". Lopputulos voidaan näin lähettää pisteeseen S_3. Montako monikkoa kaiken kaikkiaan siirrettiin yhteensä? *Let F" denote the result set obtained from the previous semi-join. Write a Select-clause that should be performed next, making use of F". The final result can now be shipped to S_3. All in all, how many tuples were transferred in total?*
   (d) Miten seuraavaa yhtälöä hyödynnettiin, eli mihin S ja R viittaavat ja miten laskettiin $S \ltimes R$?
   How was the following equation used, i.e. to what do S and R refer to and how was $S \ltimes R$ computed?
   $R \bowtie S = R \bowtie (S \ltimes R)$

3. Kysymys liittyy kaksivaiheiseen sitoutumiskäytäntöön (2PC), keskeytysoletukseen,'Presumed Abort'. Olkoon transaktion T:n koordinaattorina C ja kolmen alitransaktioiden osallistujina P_1, P_2 sekä P_3.
   This relates to the two-phased commitment protocol (2PC), specifically its 'Presumed Abort' version. Let the coordinator of the transaction T be C and denote its three participants P_1, P_2 and P_3 for the three subtransactions.

(a) Kun koordinaattorin lisäksi on kolme osallistujaa, montako viestiä liikkuu *yhteensä* kun oletetaan, että transaktio T tulee sitoutumaan eikä mitään häiriöitä esiinny? Mitä tietoa koordinaattori pakottaa lokilevyllensä ja missä tilanteissa? *In addition to the coordinator there are now three participants, so how many messages are exchanged in *total* assuming transaction T will commit and no site will fail? What information does the coordinator force to its log disk and at what instances?*

(b) Osallistuja P_1 romahtaa heti kun se on lähettänyt viestin $vote(T, ready)$ koordinaattorille. Mitä P_1 tekee elpyessään? Onko P_1 estynyt? Onko P_1 epävarmuustilassa? *Participant P_1 crashes just after sending $vote(T, ready)$ to the coordinator. What does it do upon recovery? Is P_1 blocked? Is P_1 in a state of uncertainty?*

(c) Oletetaan, että koordinaattori saakin nyt viestin $vote(T, abort)$ osallistujalta P_1. Voiko koordinaattori heti tehdä päätöksen peruuttaa transaktio ja lähettää osallistujille P_2 ja P_3 viestin $abort(T)$ ennen kuin se on edes saanutkaan näiden osallistujien äänestysviestit? Perustele lyhyesti.
Assume that the coordinator now gets the message $vote(T, abort)$ from P_1. Can it immediately decide to abort the transaction and send the message $abort(T)$ to P_2 and P_3 before even getting their votes? Justify briefly.

4. (a) Tehtävän 2 ORDERS taulu halutaan osittaa (hajauttaa useammalle levylle) huomioiden kaksi seuraavaa seikkaa:
(i) Taulukon rivimäärä vaihtelee ajan myötä. (ii) Taulukkoon tehdään runsaasti pistekyselyitä.
The table ORDERS in question 2 needs to be partitioned (declustered over several disks) with the following two facts taken into account:
(i) The number of tuples in the table changes with time. (ii) A lot of point-queries are made to the table.
Vertaa kiertovuoro-ositusta, hajautusositusta ja osaväliositusta ottaen huomioon edellä olevat rajoitukset. Mitä yhtä menetelmää näistä suosittelisit? Perustele vastauksesi lyhyesti.
Contrast round-robin, hash-partitioning and range partitioning with respect to the previous requirements. Which one of these techniques would you recommend? Justify your answer briefly.

(b) Käytetään sovittua päätösvaltakäytäntöä, jossa $n = 10$ ja lukuvallan koko $p = 6$. Osoita, että käytäntö toimii vaikka tasan 40% pisteistä romahtaisikin jollekin q:n arvolle. Mikä on tällöin tuo q:n arvo? Mitä etuja/haittoja tällaisesta päätösvallasta on verrattuna ROWAn?
We use the Quorum Consensus protocol where $n = 10$ and the size of the Read Quorum $p = 6$. Show that the protocol can function normally even when exactly 40% of the sites fail for some value of q. What is this value of q? What advantages/disadvantages does such a quorum have in relation to ROWA?

(c) Käytössäsi on yksityislevyjärjestelmä ja yhteislevyjärjestelmä. Kumman valitsisit jos kriteerinä on aineiston saatavuus vaikka jokin piste romahtaisikin? Kumpi järjestelmä tukee helpommin kuormantasausta? Perustele lyhyesti vastauksesi.
Given a shared nothing architecture and a shared- disk architecture, which one would you select if the criterion is data availability even in the case of a site failure? Which architecture more easily supports load balancing? Justify your answers briefly.

5. Kohdat (a) -(b) liittyvät asiakas-palvelin järjestelmään, joka perustuu sivupalvelimeen.
Items (a)-(b) relate to a Client-Server system based on a page server.

(a) Mistä löytyy sivun p nykyversio kun käytössä on laiska tarvehakuinen päivitysten levityskäytäntö?
Where do we find the current version of a page p when using lazy demand-driven update propagation?

(b) Palvelimen romahduksen jälkeen elvytys tapahtuu ARIESin avulla, kuitenkin päivitykset eräälle sivulle p, jota oli päivitetty asiakaalla menetetään. Löydätkö syitä tähän johtaneeseen virhetilanteeseen?
Following a crash at the server, recovery was done with ARIES, yet the updates for a page p updated at a client were lost. Can you think of reasons that could have led to such an error?

(c) Miten toimii keskinopea sivujen levityskäytäntö yhteislevyjärjestelmässä? Mikä on sen etu elvytyksessä verrattuna kaikkein nopeimpaan levityskäytäntöön?
How does the medium-speed page propagation work in the shared-disk environment? What is its advantage at recovery over the fastest propagation scheme?