Kirjoita jokaiseen palauttamaasi paperin ylälaitaan selvästi 'T-106.5241, 13.12.2011', koko nimesi, opiskelijanumerosi, koulutusohjelmasi sekä montako paperia palautit yhteensä. Laskin on sallittu mutta sitä ei tarvita.

Write on top of every answer sheet clearly 'T-106.5241, 13.12.2011', your full name, your Student Id, your study program, and the total number of sheets that you returned. Calculators are allowed but are not needed.

[1] (6p) Määritä lyhyesti mutta selkeästi seuraavat käsitteet a-f: *Explain briefly but clearly the following key-terms* a-f:

[a] hajautettu tietokanta *Distributed database*
[b] johdettu vaakasuora ositus Derived *horizontal partitioning*
[c] hajautettu lukkiuma ja sen havaitseminen *Distributed deadlock and its detection*
[d] yhteislevyjärjestelmä *Shared disk system*
[e] Operaation sisäinen rinnakkaisuus (anna esimerkki) *Intraoperation parallelism (given an example)*
[f] Vahva keskinäinen johdonmukaisuus *Strong mutual consistency*

[2] Tietokantakaaviomme koostuu kahdesta relaatiosta, avaimet alleviivattuina. *Our database schema consists of two relations, with keys underlined:*

SHIPMENTS (ShipId, OrderId, ShipCost, DestinId) DESTINATIONS (DestinId, Destination, Commission)

Relaatio SHIPMENTS, 100 000 riviä, sijaitsee pisteessä S_1 ja relaatio DESTINATIONS, 500 riviä, pisteessä S_2.
Relation SHIPMENTS, *with 100 000 rows, is located at site* S_1 *and relation* DESTINATIONS, *with 500 rows at site* S_2.

Halutaan suorittaa pisteessä S_3 seuraava SQL kysely: *We want to run the following SQL-query at Site* S_3:

```
SELECT S.ShipId, D.Commission* S.ShipCost, D.Destination FROM Shipments S, Destinations D
WHERE S.DestinId = D.DestinId
```

[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?(1p) *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, jossa hyödynnetään F''. ja jonka tulos lähetetään pisteeseen S_3. *Let F'' denote the result set obtained from the previous semi-join. Write the final Select-clause that makes use of F'' and the result of which is shipped to* S_3.

[d] Mikä (Mitkä) seuraavista lauseista (i-iv) on(ovat) tosi(a)? *Which of the following statements (i-iv) is (are) true?*
Relaatio R:n attribuutteja merkitään joukolla A. *The attributes for relation R are denoted by the set A.*
[i] Relaation R ja relaation S:n puoliliitoksessa on aina vähemmän monikkoja kuin R:n ja S:n luonnollisessa liitoksessa.
The semi-join of relation R and relation S always contains less tuples than the natural-join of R and S.
[ii] $R \ltimes_F S = \Pi_A (R \bowtie_F S)$ [iii] $R \ltimes_F S = S \ltimes_F R$ [iv] $R \bowtie_F S = (R \ltimes_F S) \bowtie_F S$

[e] Kumpi kyselyistä Q1 tai Q2 hyötyisi eniten kiertovuoro-osituksesta? Mikä ositusmenetelmä (kiertovuoro, hajautusositus, osaväliositus) osittaa SHIPMENTS taulun monikot jonkin muun perusteen kuin attribuutin arvon perusteella?
Which of the two queries Q1 or Q2 would benefit the most from round-robin? Which partitioning technique (round-robin, hash partitioning, range partitioning) partitions the tuples in table SHIPMENTS according to some other property besides an attribute value?
[Q1] `SELECT * FROM Shipments` [Q2] `SELECT DestinId FROM Shipments WHERE ShipId >1301`

[3] (6p) Kurssilla esitettiin kaksivaiheinen sitoutumiskäytäntö (2PC), josta 'Presumed on Abort' muunnelma.
In the course, you studied the two-phase commitment protocol (2PC), specifically the 'Presumed on Abort' version.
[a] Miten 2PC takaa atomisen sitouttamisen? Entä mikä on mielestäsi 2PC:n heikoin kohta?
How does 2PC guarantee atomic commitment? What would you consider to be the weakest point of 2PC?

[b] Osallistuja on juuri kirjannut lokiinsa valmiuskirjauksen $<T_i, L, P>$ ja pakottanut sen levylle. Minkä viestin se lähettää heti tämän kirjauksen jälkeen? Anna esimerkki mitä voisi mennä vikaan, jos osallistuja ei pakottaisi valmiuskirjausta levyllensä.
The participant has just written on its a prepared log record $<T_i, L, P>$ *and forced it to disk. What message will it send immediately following this logging? Give an example of what could go wrong if the participant did not force-write this prepare log to disk.*

[c] Jos kohdassa [b] osallistuja romahtaisi heti sen jälkeen kun sen lokikirjaus on mennyt levylle, mitä se tekisi elpyessään? *If in question [b] the participant were to crash right after its log record was written to disk, what would it do on recovery?*
[d] Eräs osallistuja on äänestänyt Vote(T,abort). Olettaen, että romahduksia ja häiriöitä ei esiinny, onko tämä viimeinen viesti, joka vaihdetaan kyseisen osallistujan ja Koordinaattorin välillä? Miksi tai miksi ei? *A given participant has voted Vote(T,abort). Assuming there are no crashes or failures, is this the last message exchanged between this participant and the Coordinator? Why or why not?*

[4] (6p)Päätösvaltakäytännössä p=lukuvallan koko ja q=kirjoitusvallan koko ja n=toisinnettujen pisteiden lukumäärä.
In the Quorum Consensus protocol, p=size of Read Quorum, q=size of Write Quorum and n=number of replica sites.

[a] Jos n=12 koostuen pisteistä $\{S_1, S_2,..S_{12}\}$ ja lukuvalta koostuu pisteistä $\{S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$, miksei Kirjoitusvalta voi koostua pisteistä $\{S_4, S_8, S_9, S_{10}, S_{11}, S_{12}\}$ vaikka Luku- ja Kirjoitusvalta leikkaavat? Määritä yksi annetun lukuvallan kanssa toimiva kirjoitusvalta (luettele sen pisteet). *If n=12, consisting of sites* $\{S_1, S_2,..S_{12}\}$ *and a Read Quorum consists of sites* $\{S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$ *why can't a Write Quorum consist of sites* $\{S_4, S_8, S_9, S_{10}, S_{11}, S_{12}\}$ even *though the Read and Write quorums intersect? Define one Write Quorum (list its sites) that will work with the given Read Quorum.*

[b] Jos nyt p =5 ja edelleen n=12, osoita, että tasan yksi kolmasosa pisteistä voi romahtaa jollekin q:n arvolle. Mikä on tällöin tuo q:n arvo?
If now p =5 and n still=12 show that exactly one third of the sites can fail for some value of q. What is q in this case?

[c] Halutaan, että q=p+1. Mitä tällöin *n* tulee olla, kun yksi piste tulee voida romahtaa? Anna pienimmät toimivat arvot p:lle ja n:lle (q siis = p+1) kun päätösvalta voi toimia vaikka yksi piste romahtaakin.
We want q=p+1. What should n be so that one site is allowed to fail? Give the minimum working values for p and n (q thus =p+1) when the Quorum Consensus can operate even if one site fails.

[d] Päätösvaltakäytäntö:Tosi vai epätosi (i-v)? Jos väittämä on epätosi, perustele lyhyesti miksi se on epätosi.
Quorum Consensus statements: True or False (i-v)*? If the statement is false, briefly state why it is false.*
[i] Arvo q on aina vähintään yhtä suuri kuin p. *The value for q is always at least equal to p.*
[ii] Luettaessa tuoreinta arvoa jostain lukuvallasta, luetaan enintään p/2 toisinnetta. *When reading the most up-to date value from some Read Quorum, at most p/2 replicas will be read.*
[iii] Mitä enemmän p:tä pienennetään sitä helpommaksi kirjoitusoperaatio tulee. *The more p is reduced, the easier a write-operation becomes.*
[iv] Olkoon n=5 ja p=2. Todetaan, että jokin piste S_i on muita alttiimpi romahtamaan. Todennäköisyys sille, että piste S_i esiintyy jossakin toimivassa lukuvallassa on 2/5. *Let n=5 and p=2. We notice that a given site* S_i *is more prone to failure than others. The probability that site* S_i *will appear in any valid Read quorum is 2/5.*
[v] Luku operaation suorituskyky päätösvallassa on vähintään yhtä hyvä kuin lukeminen laiskassa toisinnassa. *The performance of a Read operation with Quorum Consensus is at least as good as Reading with lazy replication.*

[5] (6p) Käytössä on asiakaspalvelin, joka toimii sivulähetyksiin perustuen (sivupalvelin), jossa pysyvä loki on palvelimella.
Consider a client database server, based on page shipping (page-server) with the stable log at the server.
[a] Mitä tarkoittaa asiakkaan WAL-käytäntö? Miten määritellään sivun *p* nykyversio ?
What is meant by the client's WAL policy ? How do we define the current version of a page p ?
[b] Mitä on asiakkaalla transaktion välinen puskurointi? *What is meant by inter-transaction caching at the client?*

[c] Miten voi syntyä tilanne, että transaktio on sitoutunut, ja siihen liittyvä päivitetty sivu on likainen asiakkaan puolella mutta ei Palvelimen puolella? Mitä haittaa tästä voisi olla tapauksessa, jossa Palvelin romahtaa ja käynnistyy uudelleen ARIES elvytyksellä? Vihje: mitä voi tapahtua toistovaiheessa?. How *can a situation occur where a transaction has committed but the related update page is dirty at the client side but not at the Server? What harm could this cause in a situation where the Server crashes and recovery is done using ARIES ? Hint: What can happen in the Redo-phase?*

[d] Oletetaan yhteislevyjärjestelmä, jossa sivu sisältää päivityksiä vain yhdeltä solmulta. Mitä solmu A tekee sen päivittämälle sivulle ennen kuin lähettää sen sitä pyytävälle solmulle B? Jos solmu A nyt kaatuisi, miksi samanlaista ongelmaa elpymisessä ei tule kuten edellisessä kohdassa *[c]*?
Consider a shared disk environment, where one page contains updates from only node. What does node A do to the page it updated before shipping it to node B which requested it? If node A were now to crash, why won't we run into a similar problem with recovery as in the previous case [c]?