CSE 344 Final Examination

March 18, 2014, 2:30pm-4:20pm

Name: ____________________

Question	Points	Score
1	20	
2	20	
3	20	
4	20	
5	20	
Total:	100	

- This exam is open book and open notes but NO laptops or other portable devices.
- You have 1h:50 minutes; budget time carefully.
- Please read all questions carefully before answering them.
- Some questions are easier, others harder; if a question sounds hard, skip it and return later.
- Even if you cannot fully answer a question, write partial solution for partial credit.
- Good luck!

Reference for SQL Syntax

Common aggregation functions in SQL

```
-- Relation R(A, B)

COUNT(*)
COUNT(A)
COUNT(DISTINCT A)
SUM(A)
AVG(A) == SUM(A)/COUNT(A)
MAX(A)
MIN(A)
```

Common keywords for sub-queries in SQL

```
IN
EXISTS
ANY
ALL
EXCEPT
(NOT) IN/EXISTS/...
```

The WITH Statement

```
WITH T AS (SELECT * FROM R WHERE R.K > 10),
     S AS (SELECT * FROM R WHERE R.a > 50)
SELECT * FROM T, S WHERE T.K<20 AND S.a < 20
```

Reference for the Relational Algebra

Name	Symbol
Selection	σ
Projection	Π
Join	$\bowtie$
Group By	γ
Set difference	$-$

SQL and Relational Languages

1. (20 points)

   Suppose a company stores information about its employees and their hobbies using the following schema:

   - `Emp(eid, name, city)`
   - `Hobbies(eid, hobby)`; `Hobbies.eid` references to `Emp.eid`

   (a) (5 points) Write an SQL query to output the `name` of the employees who live in `city = 'Seattle'` and have at least five (≥ 5) hobbies.

   Solution:
   Solution 1.

```
SELECT name
FROM Emp E, Hobbies H
WHERE E.eid = H.eid
    AND city = 'Seattle'
GROUP BY E.eid, name
HAVING count(hobby) >= 5
```

   Solution 2.

```
SELECT name
FROM Emp E
WHERE city = 'Seattle'
  AND 5 <= (SELECT count(hobby)
            FROM Hobbies H
            WHERE E.eid = H.eid)
```

- `Emp(eid, name, city)`
- `Hobbies(eid, hobby)`

(b) The following Relational Calculus (RC) expression finds all cities such that any employee living in such a city has no other hobby than White Water Rafting (`hobby = 'WWR'`) or Philately (`hobby = 'PT'`).

$$\boxed{Ans(c) = \forall e\, \forall n\, \forall h\, ((Emp(e,n,c) \wedge Hobbies(e,h)) \Rightarrow ((h =' WWR') \vee (h =' PT')))}$$

Convert this RC expression into equivalent non-recursive Datalog with negation, Relational Algebra, and SQL.

i. (5 points) Write an equivalent **non-recursive Datalog program with negation**.

Solution:

$$\begin{aligned}
& Ans(c) \\
= \; & \forall e\, \forall n\, \forall h\, (\neg(Emp(e,n,c) \wedge Hobbies(e,h)) \vee ((h =' WWR') \vee (h =' PT'))) \\
= \; & \neg\exists\, e\, \exists n\, \exists h\, \neg[\neg(Emp(e,n,c) \wedge Hobbies(e,h)) \vee ((h =' WWR') \vee (h =' PT'))] \\
= \; & \neg\exists\, e\, \exists n\, \exists h\, [(Emp(e,n,c) \wedge Hobbies(e,h)) \wedge \neg((h =' WWR') \vee (h =' PT'))] \\
= \; & \neg\exists\, e\, \exists n\, \exists h\, [(Emp(e,n,c) \wedge Hobbies(e,h)) \wedge ((h \neq' WWR') \wedge (h \neq' PT'))]
\end{aligned}$$

Datalog + negation program:

```
NonAns(c) :- Emp(e, n, c), Hobbies(e, h), h != 'WWR', h!= 'PT'
Ans(c) :- Emp(_, _, c), NOT NonAns(c)
```

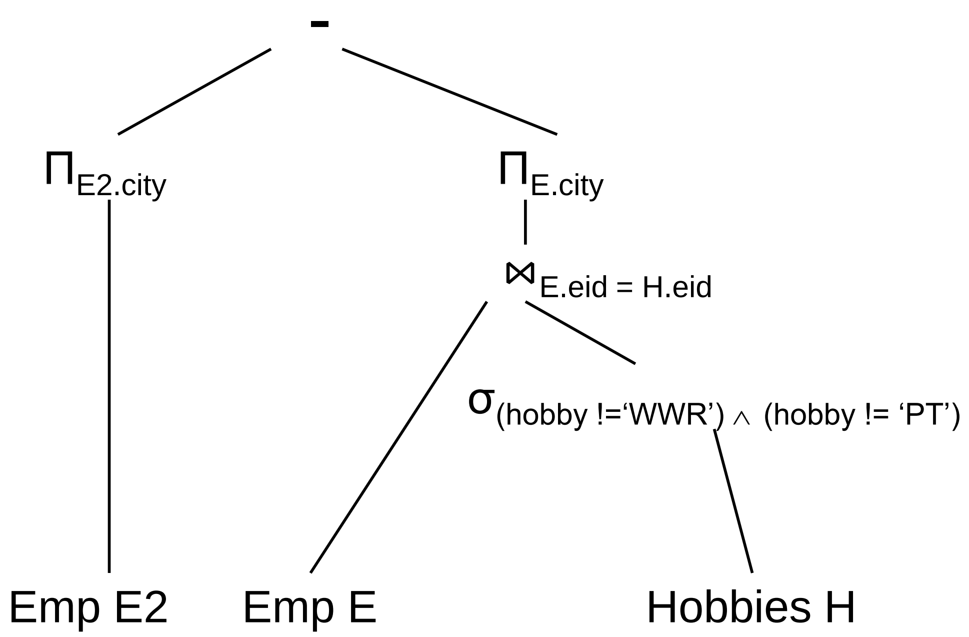
`Emp(_, _, c)` is added to make the second datalog rule safe.

- `Emp(eid, name, city)`
- `Hobbies(eid, hobby)`

ii. (4 points) Write an equivalent **Relational Algebra (RA) expression or logical query plan**.

$Ans(c) = \forall e \; \forall n \; \forall h \; ((Emp(e,n,c) \; \wedge \; Hobbies(e,h)) \; \Rightarrow \; ((h =' WWR') \; \vee \; (h =' PT')))$

Solution:


-
$\Pi_{E2.city}$
$\Pi_{E.city}$
$\bowtie_{E.eid = H.eid}$
$\sigma_{(hobby\ != 'WWR') \wedge (hobby\ != 'PT')}$
Emp E2
Emp E
Hobbies H

Note that the difference '-' operator needs the same schema on its both sides.

(**This page is intentionally left blank**)

- `Emp(eid, name, city)`
- `Hobbies(eid, hobby)`

iii. (4 points) Write an equivalent **SQL query**.

$Ans(c) = \forall e\ \forall n\ \forall h\ ((Emp(e, n, c)\ \wedge\ Hobbies(e, h))\ \Rightarrow\ ((h =' WWR')\ \vee\ (h =' PT')))$

Solution:

```
  SELECT E2.city
  FROM Emp E2
  WHERE E2.city NOT IN
         (SELECT E.city
          FROM Emp E, Hobbies H
          WHERE  E.eid = H.eid
          AND H.hobby <> 'WWR'
          AND H.hobby <> 'PT')
```

Another similar solution can use `EXCEPT`.

(c) (2 points) Write a **Relational Calculus (RC)** expression to output the names of employees who are not interested in skiing (i.e. do not have any `hobby = 'ski'`).

Solution:

$$\begin{aligned} & Ans(n) \\ = \ & \exists\ e \exists\ c\ Emp(e, n, c)\ \wedge\ \forall\ h\ (Hobbies(e, h) \Rightarrow (h\ != 'ski')) \end{aligned}$$

Database Design, Views

2. (20 points)

   (a) (**Design theory/normalization**)

   i. (3 points) Consider Relation $R(ABCD)$.
   and functional dependencies (FDs): $BD \rightarrow AC$; $AB \rightarrow D$; $AC \rightarrow B$

   - Is this relation in Boyce-Codd Normal Form (BCNF)? Write YES/NO.

   i. **YES**

   .

   - Identify a key (not a superkey).

   i. BD **or** AB **or** AC

   Solution:
   $BD^+ = \{B, D, A, C\}$, $AB^+ = \{A, B, D, C\}$, $AC^+ = \{A, C, B, D\}$.

   ii. (3 points) Consider Relation $R(ABCDE)$.
   and functional dependencies (FDs): $A \rightarrow C$; $B \rightarrow AE$; $E \rightarrow D$.

   - Is this relation in Boyce-Codd Normal Form (BCNF)? Write YES/NO.

   ii. **NO**

   .

   - Identify a key (not a superkey).

   ii. B

   Solution: $A^+ = AC \neq A$ and $\neq ABCDE$).

   $A^+ = \{A, C\}$, $B^+ = \{B, A, E, C, D\}$, $E^+ = \{ED\}$.

   Keys = B.

   iii. (3 points) If any of the above relations is not in BCNF, decompose it into BCNF. Also underline the keys in the final relations after decomposition. Otherwise write "BOTH IN BCNF".

Solution:

1. In R, $A^+ = AC$. So partition R into $R1(A, C)$ and $R2(A, B, D, E)$.
2. In $R2$, $E^+ = DE$. So partition $R2$ into $R3(D, E)$ and $R4(E, A, B)$.

No other violation.

Final relations: $R1(\underline{A}, C)$, $R3(\underline{E}, D)$, $R4(\underline{B}, A, E)$.

Alternative decomposition

1. In R, $E^+ = DE$. So partition R into $R1(E, D)$ and $R2(A, B, C, E)$.
2. In $R2$, $A^+ = AC$. So partition $R2$ into $R3(A, C)$ and $R4(A, B, E)$.

No other violation.

Final relations: $R3(\underline{A}, C)$, $R1(\underline{E}, D)$, $R4(\underline{B}, A, E)$.

(b) (**Views**) Given three relations $R(A, B)$, $S(B, C)$, $T(C, D)$, someone has created two views $V1, V2$ as follows:

```
CREATE VIEW V1 AS
  SELECT A, C, sum(B) as bs, count(B) as bc
  FROM R, S
  WHERE R.B = S.B
  GROUP BY A, C
```

```
CREATE VIEW V2 AS
  SELECT DISTINCT B, S.C, D
  FROM S, T
  WHERE S.C = T.C
```

For each of the following SQL queries, write equivalent SQL queries that will only access the views $V1, V2$. without using the original relations R, S, T.

i. (3 points) SQL Query:

```
  SELECT DISTINCT R.A, T.D
  FROM R, S, T
  WHERE R.B = S.B AND S.C = T.C
```

Solution:

```
  SELECT DISTINCT V1.A, V2.D
  FROM V1, V2
  WHERE V1.C = V2.C
```

ii. (3 points) SQL Query:

```
  SELECT A, avg(B) as myavg
  FROM R, S
  WHERE R.B = S.B
  GROUP BY A
```

Solution:

```
  SELECT V1.A, ((sum(bs) * 1.0)/sum(bc)) as myavg
  FROM V1
  GROUP BY V1.A
```

iii. (1 point) Your answer in the above question (part b.ii) will not hold if we omit the relation S and the join condition "`WHERE R.B = S.B`". Explain why.

```
(i.e. for the query )
  SELECT A, avg(B) as myavg
  FROM R
  GROUP BY A
```

Solution: Using the view $V1$ we won't consider those R tuples that do not join with any S tuples.

(c) i. (1 point) Is this statement correct?

"Materialized views are precomputed offline, and therefore are fast at runtime."

i. **TRUE**

Write TRUE/FALSE.

ii. (1 point) Is this statement correct?

"Queries that use Virtual views get data that are not always up-to-date."

ii. **FALSE**

Write TRUE/FALSE.

iii. (2 points) Insert attribute-level and/or tuple-level constraint(s) to the following `CREATE TABLE` statement to ensure that

1. the value of attribute B is always > 10 and < 20.
2. the value of attribute A is not null.

```
CREATE TABLE R (
  A INT,
  B INT

)
```

Solution:

```
CREATE TABLE R (
  A INT NOT NULL,
  B INT CHECK (B > 10 AND B < 20)
)
```

Transactions

3. (20 points)

   (a) Consider the following schedule.

```
r1(X), r2(X), r3(X), r1(Y), w2(Z), r3(Y), w3(Z), w3(Y)
```

Recall that a schedule $S1$ is conflict-equivalent to another schedule $S2$ if $S2$ can be obtained from $S1$ by swapping adjacent non-conflicting actions.

i. (1 point) Is this schedule serial?

Write YES/NO.

i. **NO**

ii. (2 points) Draw the precedence graph.

Solution: There will be an edge from T_2 to T_3 due to the conflict $w2(Z), w3(Z)$. There will be an edge from T_1 to T_3 due to the conflict $r1(Y), w3(Y)$.

iii. (1 point) Is this schedule conflict-equivalent to (T_2, T_3, T_1)?

Write YES/NO.

iii. **NO**

Solution: In any conflict-equivalent schedule, both T_1 and T_2 must precede T_3.

iv. (1 point) Is this schedule conflict-equivalent to (T_2, T_1, T_3)?

iv. **YES**

Write YES/NO.

Solution: In any conflict-equivalent schedule, both T_1 and T_2 must precede T_3.

(b) Consider the following isolation levels in SQL Server:
`READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ.`

For each of these isolation levels, complete the tables with three transactions T_1, T_2, T_3 as follows:

1. Insert lock/unlock on the element B. Use
   - $L_i(B), U_i(B)$ for long-term read or write locks by T_i in strict 2PL, and
   - $SL_i(B)$, $SU_i(B)$ for short-term read locks by T_i.
   - Also assume that short-term read locks $SL_i(B)$ can be requested by T_i and granted while another transaction T_j is holding the long-term lock $L_j(B)$.

   (note that this is a simplified locking scenario, e.g., SQL Server uses more complex locking scheme in practice.)
2. You also have extra empty rows in the tables to write any action that will be deferred by the system. Write "$L_i(B)$ `REQUESTED`" and "$L_i(B)$ `GRANTED`" in appropriate cells for any deferred lock requests, and strike out the original action that is deferred.
3. Mention the new value of B on disk if it changes at any point.
4. Mention the value of B read by the transactions in the read statements $r_i(B)$.

Note: $w_i(B, 300)$ means write $B = 300$. A_i means abort/rollback T_i, and C_i means commit T_i.

For example, the cells may look like this (this is not a real answer):

1	$\mathbf{L_1(B)}$ r_1(B) $= \mathbf{50}$			
2		$\mathbf{L_2(B)}$ `REQUESTED` ~~$w_2(B)$~~		
3	w_1(B, 80)			
4	$\mathbf{U_1(B)}$ A_1			

i. (3 points) **READ UNCOMMITTED**

Time	T_1	T_2	T_3	B **on disk**
0				20
1	r_1(B)			
2		w_2(B, 30)		
3	r_1(B)			
4		A_2		
5			w_3(B, 40)	
6			C_3	
7	r_1(B)			
8	C_1			
9				
10				
11				
12				

Solution:

Time	T_1	T_2	T_3	B **on disk**
0				20
1	r_1(B) = **20**			
2		$L_2(B)$ w_2(B, 30)		
3	r_1(B) = **30**			
4		$U_2(B)$ A_2		
5			$L_3(B)$ w_3(B, 40)	
6			$U_3(B)$ C_3	**40**
7	r_1(B) = **40**			
8	C_1			
9				
10				
11				
12				

ii. (3 points) **READ COMMITTED**

Time	T_1	T_2	T_3	B **on disk**
0				20
1	r_1(B)			
2		w_2(B, 30)		
3	r_1(B)			
4		A_2		
5			w_3(B, 40)	
6			C_3	
7	r_1(B)			
8	C_1			
9				
10				
11				
12				

Solution:

Time	T_1	T_2	T_3	B **on disk**
0				20
1	$SL_1(B)$ r_1(B) = **20** $SU_1(B)$			
2		$L_2(B)$ w_2(B, 30)		
3	$SL_1(B)$ r_1(B) = **20** $SU_1(B)$			
4		$U_2(B)$ A_2		
5			$L_3(B)$ w_3(B, 40)	
6			$U_3(B)$ C_3	**40**
7	$SL_1(B)$ r_1(B) = **40** $SU_1(B)$			
8	C_1			
9				
10				
11				
12				

iii. (3 points) **REPEATABLE READ**

Time	T_1	T_2	T_3	B **on disk**
0				20
1	r_1(B)			
2		w_2(B, 30)		
3	r_1(B)			
4		A_2		
5			w_3(B, 40)	
6			C_3	
7	r_1(B)			
8	C_1			
9				
10				
11				
12				

Solution:

Time	T_1	T_2	T_3	B **on disk**
0				20
1	$L_1(B)$ r_1(B) = **20**			
2		$L_2(B)$ REQUESTED ~~w_2(B, 30)~~		
3	r_1(B) = **20**			
4		~~A_2~~		
5			$L_3(B)$ REQUESTED ~~w_3(B, 40)~~	
6			~~C_3~~	
7	r_1(B) = **20**			
8	$U_1(B)$ C_1			
9		$L_2(B)$ GRANTED w_2**(B, 30)**		
10		$U_2(B)$ A_2		
11			$L_3(B)$ GRANTED w_3**(B, 40)**	
12			$U_3(B)$ C_3	**40**

(c) i. (1 point) What does the isolation level `SERALIZABLE` achieve that is not guaranteed in `REPEATABLE READ`?

Solution: Takes care of phantom problem.

ii. (1 point) In SQLite, can a READ lock co-exist with a PENDING lock? Write YES/NO.

ii. **YES**

iii. (1 point) In SQLite, can a READ lock co-exist with an EXCLUSIVE lock? Write YES/NO.

iii. **NO**

(d) (3 points) Can the following schedule be produced by a scheduler following two-phase locking protocol (2PL)? Explain briefly why or why not.

```
r1(X), w1(X), r2(X), w2(X), r3(Y), w3(Y), r1(Y), w1(Y)
```

Solution: NO. It cannot be produced by a two-phase-locking scheduler.

In order for T_2 to have access to X, T_1 must have unlocked X. But then T_1 must have all its locks, including the lock on Y, before T_2 accesses X. Then T_1 also had the lock on Y before T_3 accessed Y. We know that T_1 held its lock on Y to the end, because the last thing that happens is an access of Y by T_1. Therefore, it appears that both T_1 and T_3 hold a lock on Y at the same time, which is not possible.

Parallel Databases and MapReduce

4. (20 points)

Consider two relations `R(a,b)` and `S(b,c)`. Both are horizontally partitioned across $N = 3$ nodes as shown in the diagram below. Each node locally stores approximately $\frac{1}{3}$ of the tuples in `R` and $\frac{1}{3}$ of the tuples in `S`.

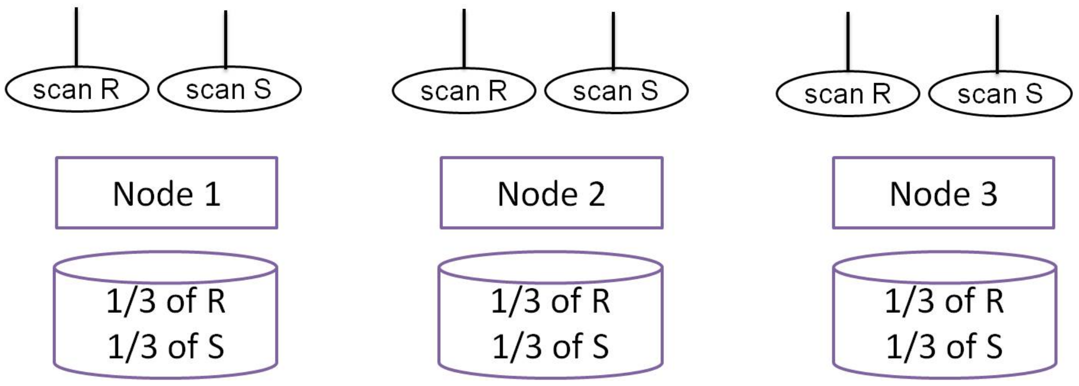
- R is **block-partitioned**.
- S is **hash-partitioned on S.b**.

(a) **Parallel-DB**

i. (8 points) Draw a parallel relational algebra plan for the following SQL query and show how it will be executed across $N = 3$ machines.
You will get more credit for picking an efficient plan that leverages the parallelism as much as possible.
If you need to re-shuffle the data, include required operators and edges to re-shuffle the data across machines.

SQL Query:

```
SELECT R.a, avg(S.c) as myavg
FROM R, S
WHERE R.b = S.b
AND R.a <= 100 and S.c > 200
GROUP BY R.a
```



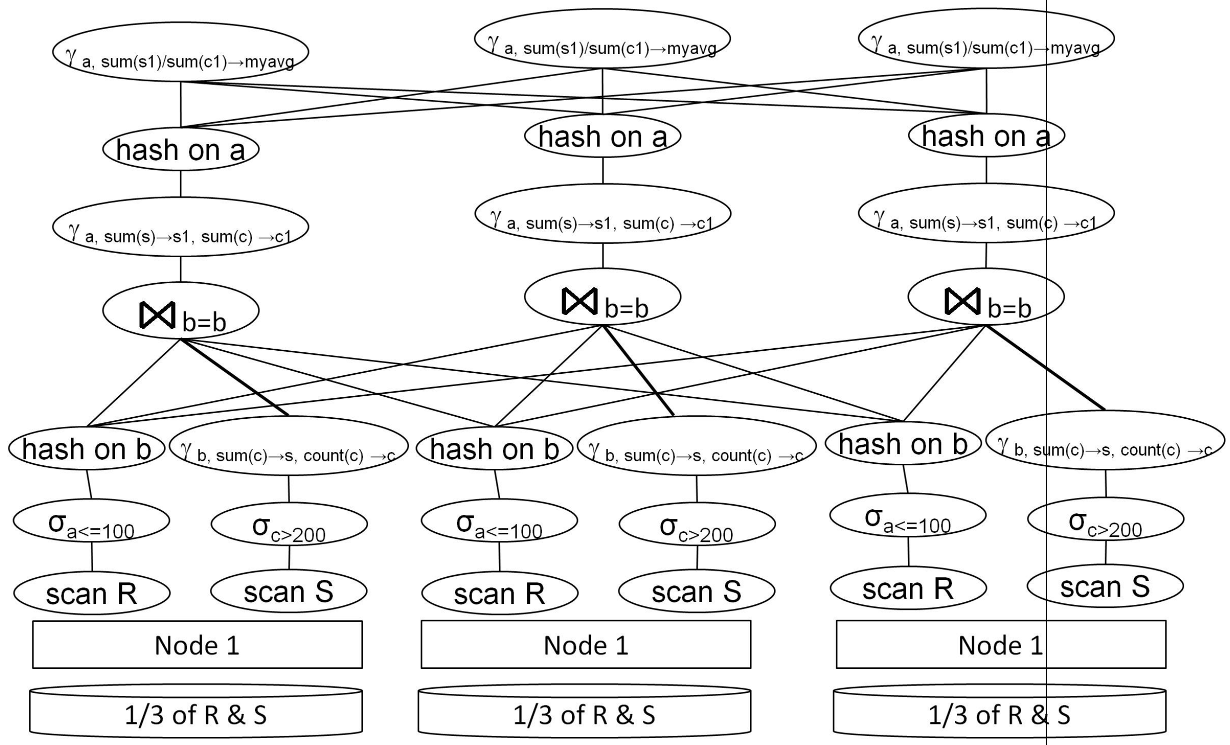
(**This page is intentionally left blank**)

- R is **block-partitioned**.
- S is **hash-partitioned on S.b**.

```
SELECT R.a, avg(S.c) as myavg
FROM R, S
WHERE R.b = S.b
AND R.a <= 100 and S.c > 200
GROUP BY R.a
```

Answer:

Solution:


γ a, sum(s1)/sum(c1)→myavg
hash on a
γ a, sum(s)→s1, sum(c) →c1
⋈ b=b
hash on b
γ b, sum(c)→s, count(c) →c
σ a<=100
σ c>200
scan R
scan S
Node 1
1/3 of R & S

ii. (1 point) Which steps can be removed/have to be added if R is hash-partitioned on R.b?

Solution: Remove 'hash on R.b' and the corresponding re-shuffle.

iii. (1 point) Which steps can be removed/have to be added if S is range-partitioned on S.c?

Solution: Add 'hash on S.b' before join on R.b = S.b and re-shuffle S tuples.

(b) (2 points) Write one advantage and one disadvantage of map-reduce compared to parallel databases.

Solution: Several answers are possible. For instance,

- (advantage) (1) more scalable, easy to speedup, (2) light-weight (no overhead of transactions, logging etc.)
- (disadvantage) (1) file format requires more time in parsing, (2) user has to do most of the optimization while writing the queries.

(c) Consider two relations $R(a, b)$ and $S(b, c)$. Answer the following questions if the following SQL query is executed by a single **MapReduce job** (**not Pig**). You can describe your answers in English, no pseudocode is necessary.

```
SELECT R.b, max(S.c) as cmax
FROM R, S
WHERE R.b = S.b
AND R.a <= 100
GROUP BY R.b
```

i. (4 points) For the **Map function**, what are the computations performed, and what will be its outputs? Assume that the Map function reads a block of R or S relation as input.

Solution:

- If the map function processes a block of the R relation, it applies the selection predicate to each R tuple in that block ($R.a <= 100$), and if the tuple passes the selection, it outputs a record with `key`$= R.b$ and `value`$= ('R', R.a)$.
- If the map function processes a block of the S relation, it outputs a record with `key`$= S.b$ and `value`$= ('S', S.c)$.

ii. (4 points) For the **Reduce function**, what will be its inputs, what are the computations performed, and what will be its outputs?

Solution:

- **Input to the reducer**: The same b as the key and a list of R or S tuples $('R', R.a)$ or $('S', S.c)$.

- **Computation and Output**: The reducer performs the cross product of R and S tuples, and computes the $max(S.c)$ value as `cmax`. Note that the cross product is empty if there are no R tuple in the input to the reducer.
  A more efficient approach is to check if there is at least one R tuple in its input, then output $max(S.c)$ as `cmax`, i.e. output $key = b, value = cmax$, otherwise output nothing.

Miscellaneous

5. (20 points)

   (a) (1 point) Recall the 3-valued logic for evaluating NULL in SQL.
   Suppose $A = NULL$, $B = 5$, $C = 10$.
   What will be the value of

   $$((A < 7) \quad OR \quad (C > 2)) \quad AND \quad (B > 10)$$

   (a) **FALSE**

   Write TRUE/FALSE/UNKNOWN

   Solution: (UNKNOWN OR TRUE) AND FALSE = TRUE AND FALSE = FALSE.

   (b) Consider relations $R(A, B)$ and $S(C, D)$. There are two types of queries in the workload.

```
Type1:
     SELECT A, D
     FROM R, S
     WHERE R.B = S.C
```

```
Type2:
    SELECT *
    FROM R
    WHERE R.A < ? AND R.B > ?
```

   i. (1 point) Which index is more useful?
   (a) $Index\text{-}R(A, B)$, or (b) $Index\text{-}R(B, A)$

   i. **(b)**

   Write (a) or (b) or "both are same"

   ii. (1 point) Which queries can be answered by the covering index $Index\text{-}R(A, B)$

   ii. **Type2**

   Write Type1 or Type2 or "None"

(c) Consider the relations

1. `Buyer(bid, bname, city)`
2. `Buys(bid, pname, price)`; bid references `Buyer`

The following SQL query outputs the names of the buyers who did not buy any product with price < 10.

```
SELECT bname
FROM Buyer B1
WHERE 10 <= ALL(SELECT price
                FROM Buys B2
                WHERE B1.bid = B2.bid)
```

i. (2 points) Write an equivalent SQL query that uses `NOT IN` for the subquery. You need to complete the following query.

```
SELECT bname
FROM Buyer B1
WHERE B1.bid NOT IN (...
```

Solution:

```
SELECT bname
FROM Buyer B1
WHERE B1.bid NOT IN (SELECT B2.bid
                     FROM Buys B2
                     WHERE B1.bid = B2.bid
                       AND B2.price < 10)
```

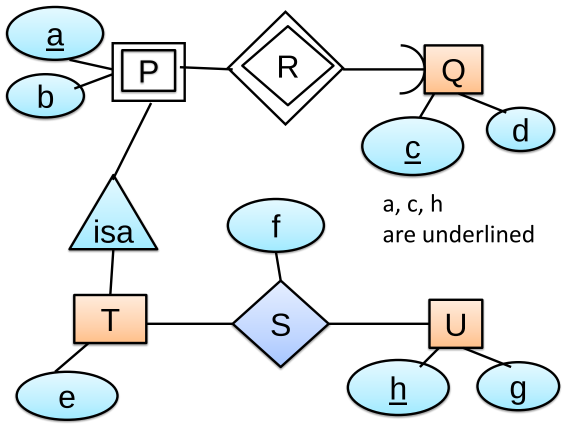
ii. (2 points) Write an equivalent SQL query that does not use a subquery.

Solution:

```
SELECT bname
```

```
FROM Buyer B1, Buys B2
WHERE B1.bid = B2.bid
GROUP BY bid, bname
HAVING MIN(price) >= 10
```

(d) (5 points) Consider the following E/R diagram.



Write the schema of the relations corresponding to all the entities and relationships in this diagram. You need to underline the keys of each relation, and mention if there is a foreign key referencing to another relation.

e.g. your answer should look like (this is not a real answer)

1. M1(m, n, o, p), where p references to attribute p of relation $M2$.
2. ....

Solution:

1. $P(\underline{a, c}, b)$, where c references to attribute c of relation Q.
2. $Q(\underline{c}, d)$.
3. $T(\underline{a, c}, e)$, where a, c reference to attributes a, c of relation P. (note: c cannot directly reference to $Q.c$).
4. $U(\underline{h}, g)$
5. $S(\underline{a, c, h}, f)$, where a, c reference to attributes a, c of relation T and h references to attribute h of relation U.

(e) i. (1 point) Write one difference between Relational data model and semi-structured data model (XML).

> **Solution:** Several answers are possible, for instance:
>
> Relational data model: fixed schema, flat structure, binary representation (good for performance, bad for exchange).
> Semi-structured data model/XML: Flexible schema, nested structure (trees), Text representation (good for exchange, bad for performance).

ii. (1 point) Is this statement correct?
"Elements in XML are unordered".

Write CORRECT/INCORRECT.

ii. **INCORRECT**

iii. (1 point) Is this statement correct?
"Attributes in XML are unordered".

Write CORRECT/INCORRECT.

iii. **CORRECT**

iv. (1 point) Is this statement correct?
"Conversion of XML to Relational databases is easier than conversion of Relational

databases to XML".

Write CORRECT/INCORRECT.

iv. **INCORRECT**

(f) (2 points) The following relation captures the XOR function X of two bits A, B. Identify all non-trivial functional dependencies in this relation.

A	**B**	**X**
0	0	0
0	1	1
1	0	1
1	1	0

Solution: $AB \to X$; $AX \to B$; $BX \to A$.

(g) (1 point) Suppose you are handling a data integration problem where you do not care much whether you have the most up-to-date data but the queries should run as fast as possible. Which approach would you prefer:

(a) Data Warehouse, (b) Mediator?

(g) **(a)**

Write (a) or (b) or "none"

(h) (1 point) Which of the following properties is not a key feature of a NoSQL system?

(a) Flexible schema, (b) Simpler interface than SQL, (c) Strict ACID concurrency model

(h) **(c)**

Write (a) or (b) or (c) or "none"