CSE 344 Final Examination

Monday, December 11, 2017, 2:30-4:20

Name: ______________________________

Question	Points	Score
1	30	
2	20	
3	30	
4	40	
5	40	
6	40	
Total:	200	

- This exam is CLOSED book and CLOSED devices.
- You are allowed TWO letter-size sheets with notes (both sides).
- You have 110 minutes;
- Answer the easy questions before you spend too much time on the more difficult ones.
- Good luck!

1 Relational Data Model

1. (30 points)

   A university maintains a database of students, the courses they took, and their summer internships, with the following schema:

   ```
   Student(sid, name, zip)
   Takes(sid, cid, year)
   Course(cid, title, dept)
   Intern(sid, companyName)
   ```

   The keys are underlined; `Takes.sid` and `Intern.sid` are foreign keys to `Student`, while `Takes.cid` is a foreign key to `Course`.

   Solution:

   ```
   drop table if exists takes;
   drop table if exists student;
   drop table if exists course;
   drop table if exists intern;

   create table Student(sid int primary key,name text,zip int);
   create table Course(cid int primary key, title text, dept text);
   create table Takes(sid int references Student,
                      cid int references Course,
                      year int);
   create table Intern(sid int references Student, companyName text);

   insert into Student values(1, 'Alice', 98195);
   insert into Student values(2, 'Bob', 98196);
   insert into Student values(3, 'Carol', 98197);
   insert into Student values(4, 'David', 98198);
   insert into Course  values(344, 'Databases', 'CSE');
   insert into Takes   values(1, 344, 2017);
   insert into Takes   values(2, 344, 2017);
   insert into Takes   values(4, 344, 2016);
   insert into Intern values(2,'MS');
   ```

   (a) (10 points) If two students took the same course in the same year then we call them *pals*. Write a SQL query that computes, for each student, the number of his/her pals. Your query should return all students indicating their student ID and name, and the number of their pals, ordered lexicographically by their names. (Hint: students who never took any course have zero pals; you need to return these too.

Students who took at least one course have at least one pal, naturally (why?); you should not exclude himself/herself when you count their number of pals.)

Solution:

```
select s.sid, s.name, count(t2.sid)
from Student s
     left outer join Takes t on s.sid = t.sid
     left outer join Takes t2 on t.cid = t2.cid and t.year = t2.year
group by s.sid, s.name
order by s.name;
```

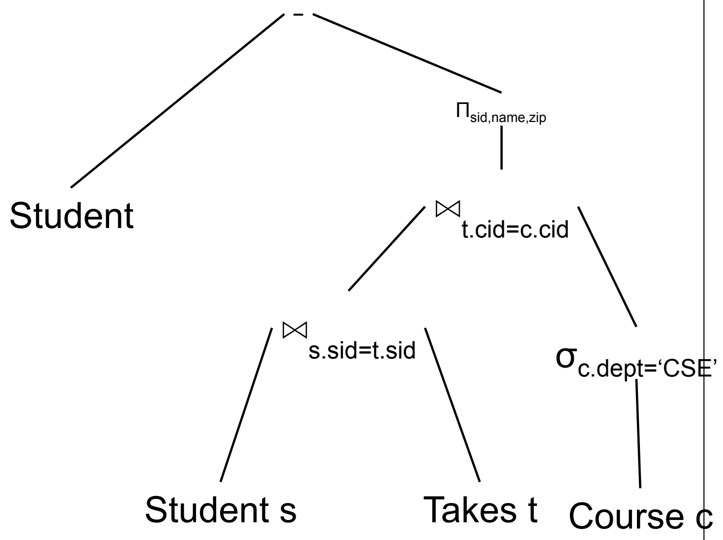
2 points for missing left outer join
2 points for `cont(*)`
2 points for unnecessary subquery
0-1 points for incomplete/missing group by, order by, on

```
Student(sid, name, zip)
Takes(sid, cid, year)
Course(cid, title, dept)
Intern(sid, companyName)
```

(b) Consider the following SQL query:

```
    select *
    from Student s
    where not exists
           (select *
            from Takes t, Course c
            where s.sid = t.sid
              and t.cid = c.cid
              and c.dept = 'CSE');
```

i. (5 points) Write this query in relational algebra.



Solution:
1 point off for missing $\pi_{\texttt{sid,name,zip}}$

ii. (5 points) Write this query in datalog.

Solution:

```
nonAnswer(sid) :- Takes(sid, cid, -), Course(cid, -, 'CSE')
Answer(sid,name,zip) :- Student(sid,name,zip), !nonAnswer(sid)
```

2 points off for unsafe query

```
Student(sid, name, zip)
Takes(sid, cid, year)
Course(cid, title, dept)
Intern(sid, companyName)
```

(c) (10 points) We say that a student X *has a connection* to Y if either X,Y are pals, or they have a common connection. Alice wants to find a connection who was an intern at Microsoft. Write a datalog program to find all Alice's connections who were Microsoft interns. In other words, if Alice is a pal with someone, who is a pal with someone, who is a pal . . . , who is a pal with someone who interned at Microsoft, then you return the Microsoft intern; you should return the sid and name. Recall that *pal* means that the two students took a common course in a common year.

Solution:

```
ref(sid1,sid2) :- Takes(sid1,cid,y), Takes(sid2,cid,y)
ref(sid1,sid3) :- ref(sid1,sid2),ref(sid2,sid3)
answer(sid,name) :- Student(s,'Alice',-), ref(s,sid), Intern(sid,'Microsoft')
```

5 points off for non-recursive query

2 points off for wrong pals (self-join `Takes` on both `cid` and `year`)

2 NoSQL, JSon, SQL++

2. (20 points)

   (a) A NoSQL database stores one billion (10^9) key-value pairs. The system runs on 100 servers, and replicates every key-value pair on three randomly chosen servers, to increase reliability. Replicas may be out of sync, but they will eventually be reconciled by the system, i.e. if a value is updated on one server, eventually the other two replicas of that value will also be updated. Answer the following questions. In all cases, you may round your answer to the closest integer.

      i. (2 points) How many key-value pairs are stored on one server?

      i. **30 million (**$30 \cdot 10^6$**)**

      Number of key-value pairs:

      ii. (2 points) A transaction needs to access 5 different key-value pairs, and does not necessarily need to get the most up to date values. From how many servers does it need to read (in expectation)?

      ii. **5**

      Number of servers:

      iii. (2 points) A transaction needs to access 5 different key-value pairs, and must get the most up to date values. From how many servers does it need to read (in expectation)?

      iii. **15**

      Number of servers:

      iv. (4 points) One day three servers fail catastrophically, simultaneously. In expectation, how many key value pairs are lost after this failure? Your answer may be approximate. Note that the NoSQL database can recover a key-value pair if at least one replica survives; you need to compute the (approximate) number of key-value pairs that cannot be recovered after three servers fail.

      iv. **6000: each key is lost wi**

      Expected number:

(b) For each statement below indicate if it is true or false.

i. (2 points) The main reason why NoSQL database were introduced was because relational databases did not scale up to very large number of servers.

True or false?

i. **true**

ii. (2 points) In JSon one can represent nested collections.

True or false?

ii. **true**

iii. (2 points) The JSon data model is in First Normal Form.

True or false?

iii. **false**

iv. (2 points) It is easy to represent a many-to-one relationship in JSon, but it is difficult to represent a many-to-many relationship.

True or false?

iv. **true**

v. (2 points) SQL++ can express recursive queries, such as transitive closure.

True or false?

v. **false**

3 Query Evaluation

3. (30 points)

Consider the relations below:

```
Student(sid, name, zip)
Takes(sid, cid, year)
Course(cid, title, dept)
```

(a) We run the following SQL command:

```
CREATE INDEX newI on Student(zip,name);
```

For each of the following SQL queries indicate whether the index `newI` will speed it up, or will not affect its performance, or will slow it down. Assume that the optimizer is super-smart, and will always chose the absolute best possible plan.

i. (2 points)

```
select * from Student where name = 'Alice';
```

i. **The same**

Faster? The same? Or slower?

ii. (2 points)

```
select * from Student where zip = 98195;
```

ii. **Faster**

Faster? The same? Or slower?

iii. (2 points)

```
select Student.name, Student.zip
from Student, Takes, Course
where Student.sid=Takes.sid
  and Takes.cid = Course.cid
  and Course.title = 'Databases';
```

iii. **Faster**

Faster? The same? Or slower?

iv. (2 points)

```
insert into Student values(1234,'Alice',98195);
```

iv. **Slower**

Faster? The same? Or slower?

v. (2 points)

```
delete from Student where name = 'Alice';
```

v. **Slower**

Faster? The same? Or slower?

```
Student(sid, name, zip)
Takes(sid, cid, year)
Course(cid, title, dept)
```

(b) (5 points) Consider the following statistics on the relations and attributes:

$B(\texttt{Student}) = 2,000$	$B(\texttt{Takes}) = 1,000,000$	$B(\texttt{Course}) = 1,000$
$T(\texttt{Student}) = 200,000$	$T(\texttt{Takes}) = 10,000,000$	$T(\texttt{Course}) = 10,000$
$V(\texttt{Student}, \texttt{zip}) = 10,000$	$V(\texttt{Takes}, \texttt{sid}) = 200,000$	$V(\texttt{Course}, \texttt{dept}) = 100$
	$V(\texttt{Takes}, \texttt{cid}) = 5,000$	

Estimate the size of the answer of the following SQL query:

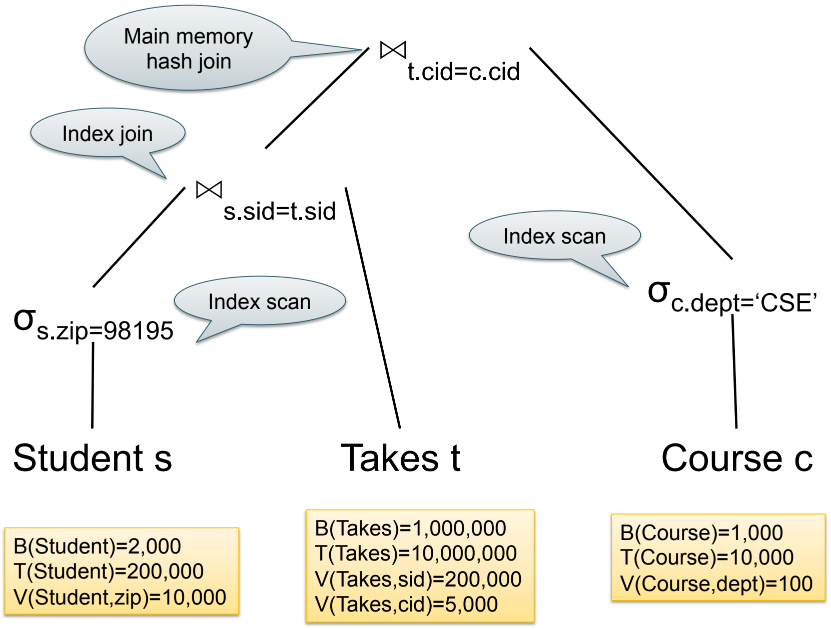
```
select *
from Student s, Takes t, Course c
where s.sid = t.sid and t.cid = c.cid
  and s.zip = 98195 and c.dept = 'CSE';
```

Solution:

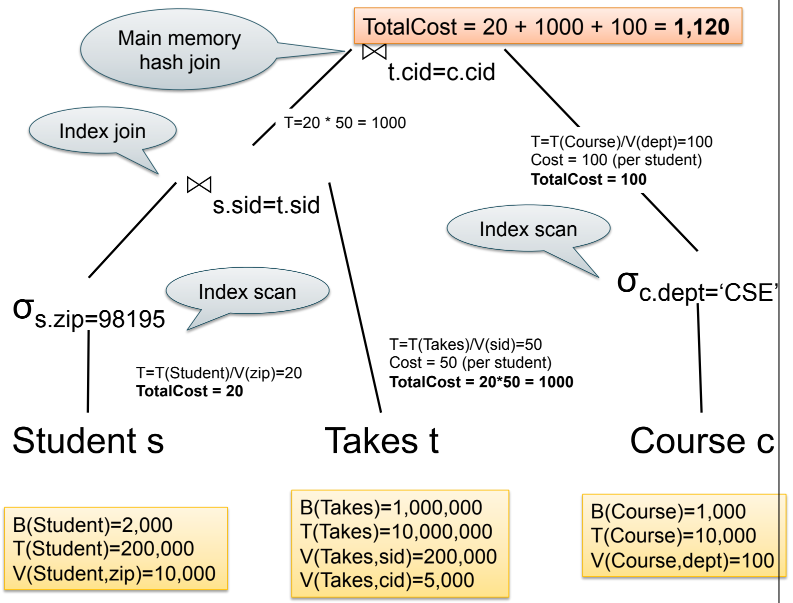
$$\frac{T(\texttt{Student}) \times T(\texttt{Takes}) \times T(\texttt{Course})}{V(\texttt{Student}, \texttt{zip}) \times V(\texttt{Course}, \texttt{dept}) \times \max(V(\texttt{Student}, \texttt{sid}), V(\texttt{Takes}, \texttt{sid})) \times \max(V(\texttt{Takes}, \texttt{cid}), V(\texttt{Course}, \texttt{cid}))}$$
$$= \frac{2 \cdot 10^5 \cdot 10^7 \cdot 10^4}{10^5 \cdot 10^2 \cdot (2 \cdot 10^5) \cdot 10^4} = 1$$

(c) For each of the physical plans below, estimate its cost. Assume the size of the main memory is $M = 2000$ pages and that all indexes are unclustered, and are stored in main memory (hence accessing them requires zero disk I/O's).

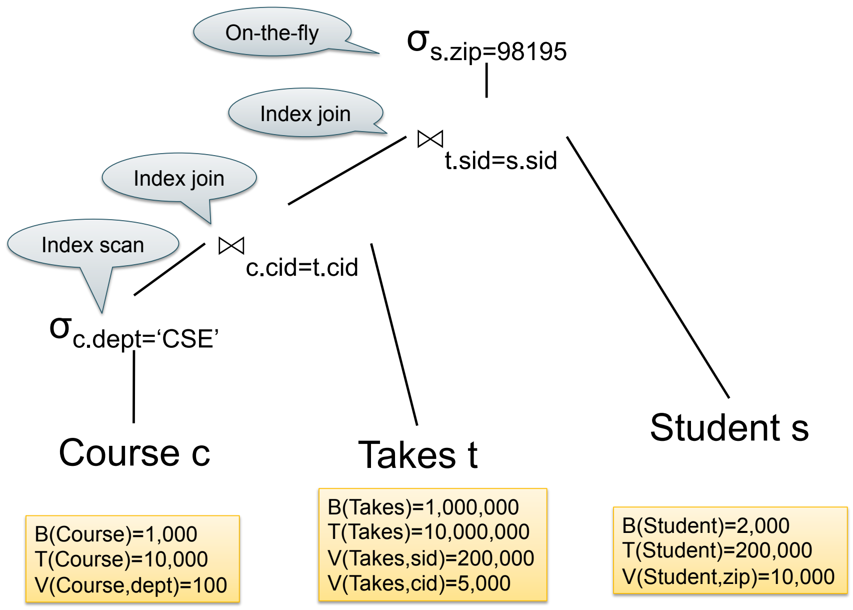
i. (5 points)

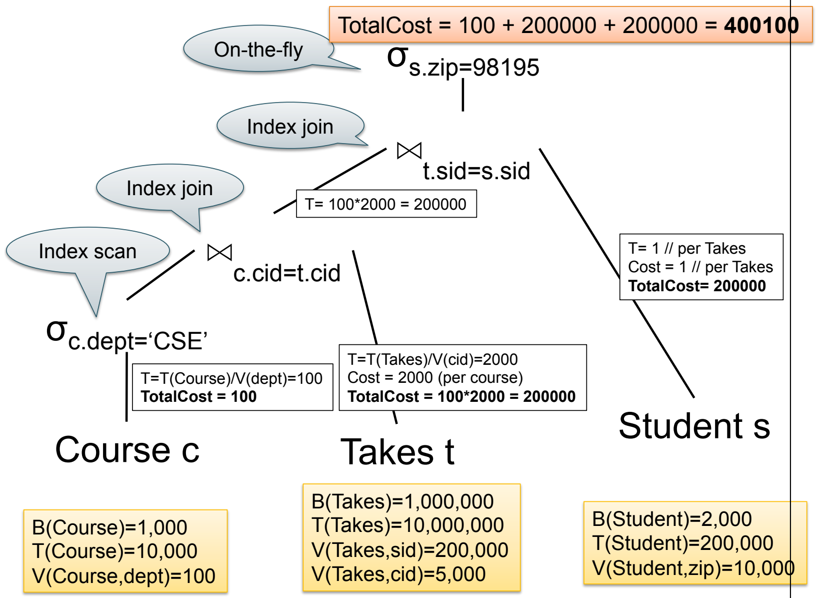


Solution:



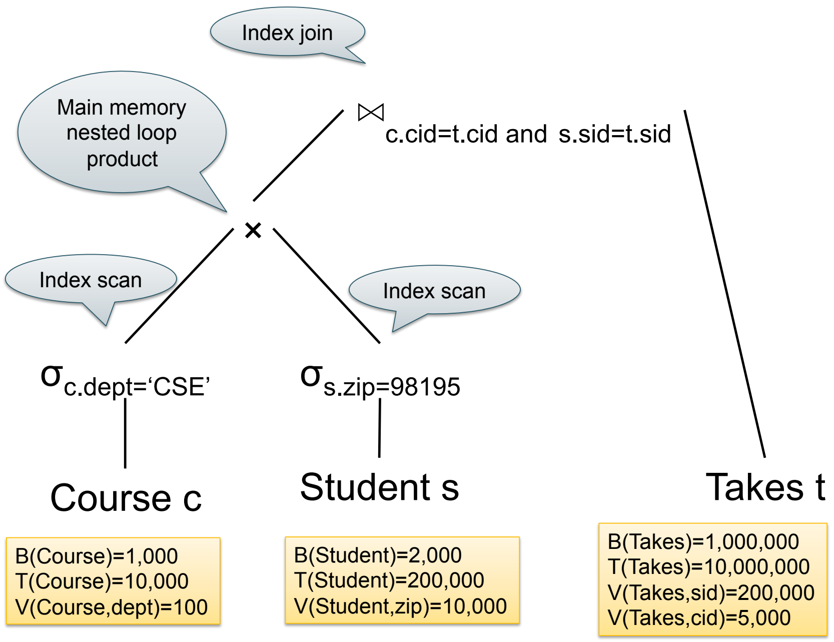
ii. (5 points)



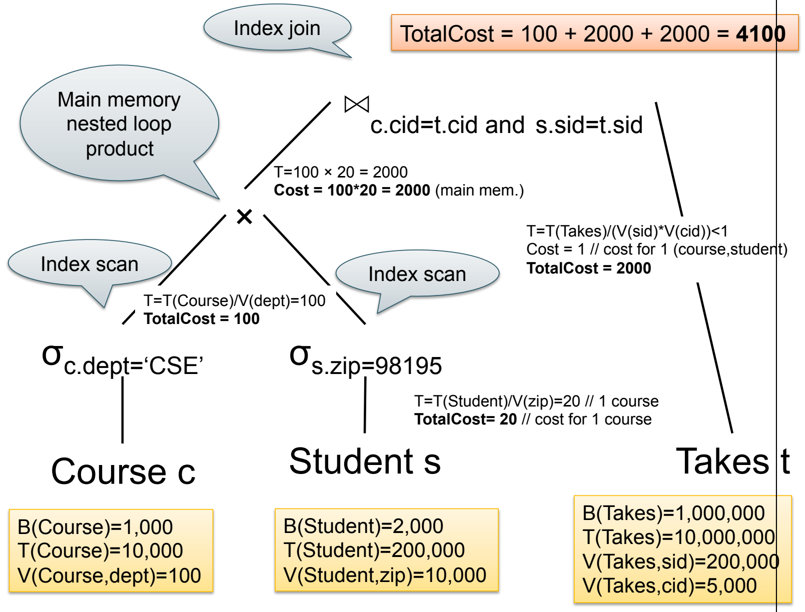


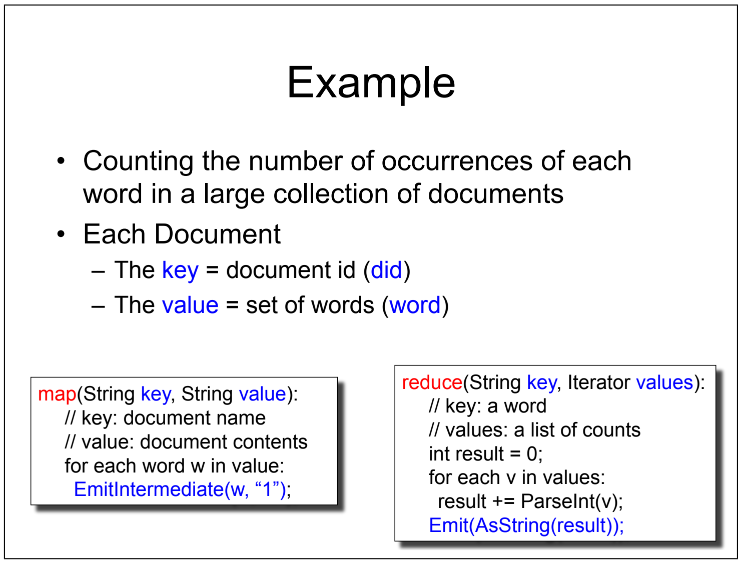
Solution:

iii. (5 points)



Solution:





Example

- Counting the number of occurrences of each word in a large collection of documents
- Each Document
  - The key = document id (did)
  - The value = set of words (word)

```
map(String key, String value):
   // key: document name
   // value: document contents
   for each word w in value:
    EmitIntermediate(w, "1");
```

```
reduce(String key, Iterator values):
   // key: a word
   // values: a list of counts
   int result = 0;
   for each v in values:
    result += ParseInt(v);
   Emit(AsString(result));
```

4 Parallel Databases

4. (40 points)

   (a) (10 points) Write a Map/Reduce program to compute the following SQL query:

```
select p.category, count(*)
from Product p
where p.price > 100
group by p.category
having sum(p.quantity) > 200
```

You need to write to functions: `Map` and `Reduce`. Use pseudo-code to write both functions, as we did in class (to refresh your memory, the relevant slide from the lectures is shown at the top of the page). You may assume that the `Map` function has a single input, namely the `Product` record.

Solution:

```
function map(String p):
  if p.price > 100
    then Emitintermediate(p.category, p)

function reduce(String c,iterator ps):
   s = 0;
   for each p in ps:
      s = s + p.quantity
   if s > 200 then
      r = 0
      for each p in ps:
        r = r+1
    Emit(c, AsString(r))
```

(b) Consider the following SQL query, computing for each person p the number of its followers under 20 years old

```
select p.pid, p.name, count(*)
from Person p, Follower f
where p.pid = f.follows and f.age <= 20
group by p.pid, p.name
```

We have $T(\texttt{Person}) = 10^7$, $T(\texttt{Follower}) = 10^9$, $\min(\texttt{age}) = 1$, $\max(\texttt{age}) = 99$, and $V(\texttt{Follower}, \texttt{follows}) = 10^7$. Assume the logical query plan is:

$$\gamma_{\texttt{pid,name,count(*)}}(\texttt{Person} \bowtie \sigma_{\texttt{age}<20}(\texttt{Follower}))$$

The system computes the query on 200 servers as follows.

Step 1 Initially both relations are distributed randomly and uniformly on all servers. The selection is computed on the fly

Communication Next, there is a communication step where `Person` is hash-partitioned on `pid`, and `Follower` is hash partitioned on `follows`.

Step 2 Each server computes the join and group by locally.

Assume the values of `age` are uniformly distributed among `Follower` and that the hash function is uniform. For anything else, assume the worst case distribution.

i. (2 points) How many `Person` records does each server hold during each step of the computation? Report the maximum number of records during steps one and two, i.e. do not add up the number of records held during steps one and two.

i. $10^7/200 = 5000$

Number of records:

ii. (2 points) How many `Follower` records does each server hold during each step of the computation? Like in the previous question, report the maximum number, not their sum.

ii. $T(\texttt{Follower})/5 = 2 \cdot 10^8$

Number of records:

Solution: There are $T(\texttt{Follower})/5$ followers with $\texttt{age} \leq 20$. Since `follows` may be skewed, all of them may be following the same person.

(c) Answer the following questions:

i. (2 points) If you can compute a query in parallel on 1000 servers in 15 minutes, then you can always compute it in parallel on 500 servers in at most 30 minutes.

i. **True**

True or false?

ii. (2 points) If you can compute a query in parallel on 500 servers in 30 minutes, then you can always compute it in parallel on 1000 servers in at most 15 minutes.

ii. **False**

True or false?

iii. (2 points) If one server of a parallel database system fails, then the entire job that the system was computing is restarted; if one server of a Map/Reduce job fails then the M/R system can recover and continue the job. Why the difference? Choose one answer below:

1. Parallel databases can answer complex SQL queries, while an M/R job is restricted to two functions (map and reduce).
2. Parallel databases were developed in the 80's when fault tolerance techniques were not yet developed, while M/R was developed in the early 2000's, when the technology exists
3. Parallel database systems run on a small number of servers (10-20) while M/R jobs can run on a very large number of servers (10000) making failure much more likely.
4. Parallel database systems must maintain statistics on the database to be used by the query optimizer, while M/R systems do not maintain statistics on the data.

iii. **3**

Choose one answer

(d) Answer the following questions about Map/Reduce:

i. (2 points) The number of map tasks is usually equal to the number of blocks of the input file.

i. **True**

True or false?

ii. (2 points) The number of reduce tasks is usually equal to the number of blocks of the output file.

ii. **False**

True or false?

iii. (2 points) The first reduce task can start as early as the first map task finishes.

iii. **False**

True or false?

iv. (2 points) The number of reduce tasks must be less than or equal to the number of servers.

iv. **False**

True or false?

v. (2 points) The combine function is executed by the map task.

v. **True**

True or false?

vi. (2 points) If a server running a reduce task fails during the computation, then the master will start that reduce task on another server (or the same server, if it recovers).

vi. **True**

True or false?

vii. (2 points) In a Map/Reduce job, *shuffle* refers to the process of distributing the input data to the map tasks.

vii. **False**

True or false?

(e) Answer the following questions about Spark:

i. (2 points) A *Resilient Distributed Dataset* (RDD) is another term for a distributed file on disks.

i. **False**

True or false?

ii. (2 points) Consider the function `map`(f) : `RDD` $< T > \rightarrow$ `RDD` $< U >$, where $f : T \rightarrow U$. The output RDD can never be strictly smaller than the input RDD. (In other words, the number of U elements in the output can never be strictly smaller than the number of T elements in the input.)

ii. **True**

True or false?

iii. (2 points) Consider the function `flatMap`(f) : `RDD` $< T > \rightarrow$ `RDD` $< U >$, where $f : T \rightarrow$ `Seq`(U). The output RDD can never be strictly smaller than the input RDD. (Recall: `Seq`(T) means *sequence of* T, or *array of* T.)

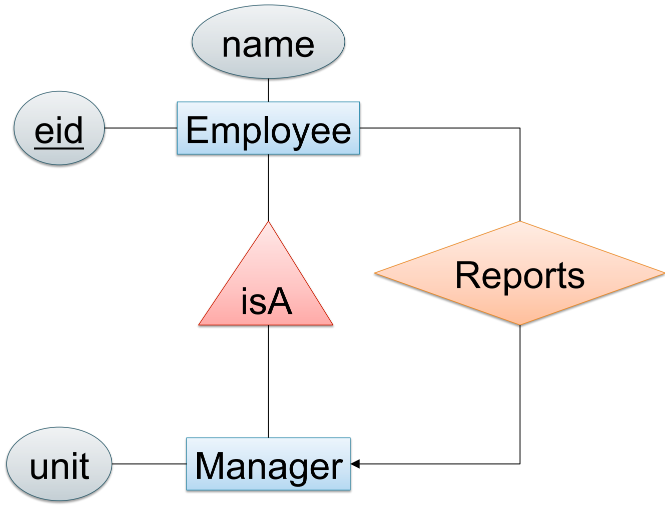
iii. **False**

True or false?

5 Conceptual Design

5. (40 points)

   (a) (5 points) Consider the following E/R diagram:



Write the CREATE TABLE statements to implement the E/R diagram in SQL. Your statements should contain all key and foreign key declarations. Assume that `eid` and `unit` are integers, and `name` is a text.

Solution:

```
create table Employee(eid int primary key, name text);
create table Manager(eid int primary key references Employee, unit int);
```

(b) (5 points) The relation $R(A, B, C, D, E, F)$ satisfies the following functional dependencies:

$$AB \rightarrow C$$
$$CD \rightarrow E$$

Find a BCNF decomposition for R. In each of the new relation indicate the key (by underlining the attribute, or set of attributes that form a key).

Solution:

$$AB^+ = ABC \qquad \Rightarrow \text{Decompose } R \text{ into } S(\underline{A, B}, C), T(A, B, D, E, F)$$
$$CD^+ = CDE \qquad \Rightarrow \text{Decompose } T \text{ into } K(\underline{C, D}, E), L(C, D, F)$$

Final answer: $S(\underline{A, B}, C), K(\underline{C, D}, E), L(\underline{C, D, F})$.

(c) For each statement below indicate if whether it is true or false.

i. (3 points) The reason why we decompose a relation in Boyce Codd Normal Form is to save disk space when the relation becomes too large.

i. **False**

True or false?

ii. (3 points) Suppose that the BCNF decomposition of R consists of three relations S, T, K: then S, T, K are a lossless decomposition of R.

ii. **True**

True or false?

iii. (3 points) If $X \subseteq Y$ then $X^+ \subseteq Y^+$.

iii. **True**

True or false?

iv. (3 points) $(X^+)^+ = X$.

iv. **False**

True or false?

v. (3 points) If $(X \cup Y)^+ = X^+ \cup Y^+$

v. **False**

True or false?

(d) Answer the questions below:

i. (3 points) Consider the following three relations:

$$R(\underline{A}, B), S(B, C), T(C, A)$$

where $R.A$ is a key. Suppose they have the same cardinality N:

$$|R| = |S| = |T| = N$$

What is the largest possible size of the natural join $R \bowtie S \bowtie T$? Your answer should be a mathematical formula that depends on N, e.g. $N^2 \log(N)$ or $N + 5$ (not a real answer).

i. N

Maximum size of $R \bowtie S \bowtie T$ is:

ii. (3 points) Consider two relations $R(A, B, C)$ and $S(D, E, F)$ satisfying the following functional dependencies:

$$R : A \to B$$
$$S : DE \to F$$

Let $T(A, B, C, D, E, F)$ be the following table:

```
create table T as
  (select R.A, R.B, R.C, S.D, S.E, S.F
   from R, S
   where R.C = S.D and S.E = 5)
```

Find the key or keys of T. If T has more than one key, list all keys.

Solution: Keys: AC and AD.

iii. (3 points) Consider a relation with the schema $R(A, B, C)$. Suppose the relational instance satisfies the functional dependency

$$A \to B$$

and does not satisfy any other functional dependencies. We know the following statistics on R:

$|R| = 512$ $V(R, B) = 510$

Compute $V(R, A)$; your answer should consist of a number.

iii. 511

$V(R, A) =$

iv. (3 points) True or false? If R is a relation that satisfies a set of functional dependencies, then we can declare all of them in the CREATE TABLE R statement by using PRIMARY KEY and UNIQUE declarations.

iv. **False**

True or false?

v. (3 points) True or false? If R is a relation that satisfies a set of functional dependencies and is in BCNF, then we can declare all of them in the CREATE TABLE R statement by using PRIMARY KEY and UNIQUE declarations.

v. **True**

True or false?

6 Transactions

6. (40 points)

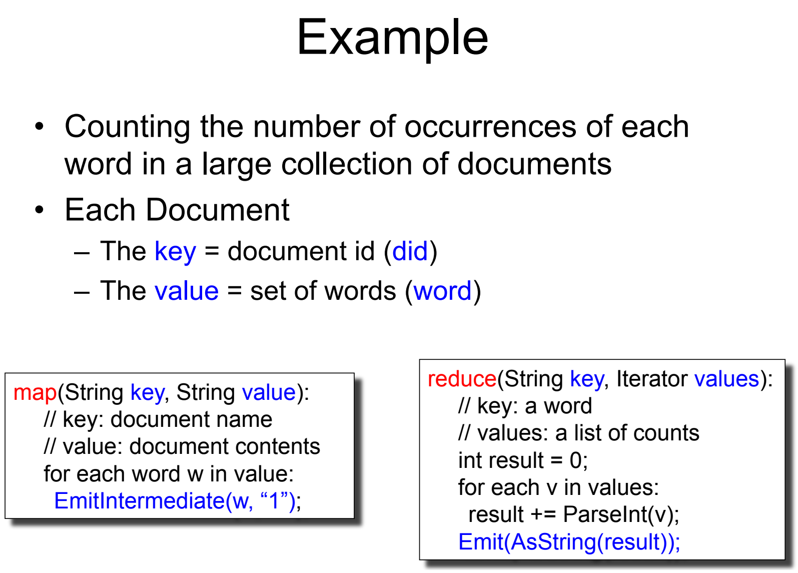
   (a) For each schedule below indicate whether it is conflict serializable and, if it is, indicate the equivalent serial schedule.

      i. (5 points)

$$R_1(A), R_4(D), W_3(C), R_2(B), W_2(A), R_1(C), W_3(B)$$

      ii. (5 points)

$$R_1(A), W_2(B), R_3(C), W_3(A), R_3(D), R_4(B), W_4(D), W_2(C)$$



Example

- Counting the number of occurrences of each word in a large collection of documents
- Each Document
  - The key = document id (did)
  - The value = set of words (word)

```
map(String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
      EmitIntermediate(w, "1");
```

```
reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
      result += ParseInt(v);
    Emit(AsString(result));
```

Solution:

(b) An application uses a database of customers and their accounts:

```
Customer(cid,name,acc)
Account(acc, balance)
```

The application runs concurrently many instances of the following transaction (shown here in pseudocode):

```
function updateAccount(c, delta)
   BEGIN TRANSACTION
   myCustomer = "SELECT * FROM Customer WHERE cid = [c]";
   myAccount  = "SELECT * FROM Account WHERE acc = [myCustomer.acc]";
   if myAccount.balance + delta < 0 then ROLLBACK
   "UPDATE Account
    SET balance a = [a.balance + delta]
    WHERE acc = [myCustomer.acc]"
   COMMIT
```

Thus, the transaction first looks up the customer with `cid` $= c$, call him/her `myCustomer`, then looks up the account `myCustomer.acc`, and adds `delta` to the account balance, checking first that it doesn't become negative.

The concurrency control manager of the database uses a locking based concurrency control mechanism. It uses the strict 2PL mechanism for write locks, the mechanism decided by the isolation level for read locks, and table-level locking for phantoms. The only transactions on the system are instances of the `updateAccount` shown above.

Answer the questions on the next page.

In each case below you are asked to indicate (a) whether the scheduled is guaranteed to be serializable, and (b) whether a deadlock may occur.

i. (3 points) Suppose we run all transactions with:
`SET ISOLATION LEVEL READ UNCOMMITTED`

It is always serializable? i. **No**

Can deadlock occur? i. **No**

ii. (3 points) Suppose we run all transactions with:
`SET ISOLATION LEVEL READ COMMITTED`

It is always serializable? ii. **No**

Can deadlock occur? ii. **No**

iii. (3 points) Suppose we run all transactions with:
`SET ISOLATION LEVEL REPEATABLE READ`

It is always serializable? iii. **Yes**

Can deadlock occur? iii. **Yes**

iv. (3 points) Suppose we run all transactions with:
`SET ISOLATION LEVEL SERIALIZABLE`

It is always serializable? iv. **Yes**

Can deadlock occur? iv. **Yes**

v. (3 points) Assume that the concurrency control mechanism uses strict 2PL with exclusive locks only; in other words, it no longer uses separate read and write locks.

It is always serializable? v. **Yes**

Can deadlock occur? v. **No**

(c) For each of the following statements indicate whether it is true or false:

i. (3 points) In a static database, every serializable schedule is conflict serializable.

True or false?

i. **False**

ii. (3 points) In a dynamic database, every serializable schedule is conflict serializable.

True or false?

ii. **False**

iii. (3 points) In a static database, every conflict serializable schedule is serializable.

True or false?

iii. **True**

iv. (3 points) In a dynamic database, every conflict serializable schedule is serializable.

True or false?

iv. **False**

v. (3 points) In Sqlite all schedules are guaranteed serializable.

True or false?

v. **True**