Homework 4

Normalization, data storage and transaction management

Total points: 40

Complete the following exercises (some are from the textbook). Submit a document with your solutions.

Chapter 7

8.21 (4 pts.) Use the schema given in this exercise. Ignore the double arrow and assume that it is a single arrow that represents a functional dependency. Take each of the two relations and specify which normal form they are in. Then normalize them to 3NF. Hint: AccessionNo and userid are the primary keys.

books(accessionno, isbn, title, author, publisher)
users(userid, name, deptid, deptname)
accessionno → isbn
isbn → title
isbn → publisher
isbn →→ author
userid → name
userid → deptid
deptid → deptname

Response:
Both the relations books and users are in second normal form, as there are no partial keys.
Normalize to 3nf by removing the transitive relations:

books:	books1(accessionno, isbn)	books2(isbn, title, publisher, author)
users:	users1(userid, name, deptid)	users2(deptid, deptname)

8.35 (5 pts.) – This question is not included in the textbook.

Ø Consider the relation R(A,B,C,D,E,F,G,H) in which ABC is the primary key. If the following dependencies hold in this relation, is this relation in 3NF? If not, which normal form does it satisfy? Reduce it to 3NF.

AB -> D
C -> H
EF-> G

Response:
The relation is in 1nf.

R –> 3NF: R1(A,B,D) R2(C,H) R3(E,F,G)

Chapter 11

11.15 (2 pts.) When is it preferable to use a dense index rather than a sparse index? Explain your answer.

Response:

It is preferable to use a dense index in situations when the data isn't ordered;

using a sparse index only gives you a block pointer, in unordered situations that wouldn't be enough to find a record.

11.16 (2 pts.) What is the difference between a clustering index and a secondary index?

Response: Clustering and secondary indexes are both indexes that are based on keys that are not primary nor candidate keys, however clustering indexes are on search keys that are ordered, while secondary indexes are on not ordered keys, so clustering indexes can point to blocks, while secondary indexes need to point to individual records.

11.19 (3 pts.) Explain the distinction between closed and open hashing. Discuss the relative merits of each technique in database applications.

Response: Closed and open hashing are both responses to collisions and bucket overflow; closed hashing is the concept of creating overflow buckets and chaining or connecting the initial and new buckets together in some way, while open hashing is where collisions can allow keys to move into adjacent/other buckets. The value of open hashing is that it is a much faster way to solve collisions, however, it is completely useless in database applications where the data would then have to be retrieved later.

11.21 (2 pts.) Why is a hash structure not the best choice for a search key on which range queries are likely?

Response: Data in a hash structure is not necessarily stored sequentially, so range queries (where data is looked at within a sequence) would have be converted into something else.

11.29 (5 pts.) – This question is not included in the textbook. This material is covered in the lectures.

Ø Consider a disk with the following parameters:

B = 512 bytes
P = 6 bytes
P_r = 7 bytes

and an Employee data file with the following parameters:

r = 30000
R = 115 bytes
bfr = 4 records/block

Suppose the file is not ordered by the key field SSN (9 bytes) and we want to construct a *secondary index* with record pointers on SSN. Calculate the following:

a. The index blocking factor, bfr_i.
b. The number of first-level index entries and the number of number of first-level index blocks.
c. The number of levels needed if we make it into a multi-level index
d. The total number of blocks required by the multi-level index

e. The number of block accesses needed to search for and retrieve a record from the file – given its SSN value – using the primary index.

Response:
a) bfr_i = floor(B / P_r + bytes(SSN)) = floor(512 / (7 + 9)) = floor(512 / 16) = 32 index records per block
b) r_i = r = 30_000 index records needed; b_i = ceil(r_i / bfr_i) = ceil(30_000 / 32) = 938 first level index blocks needed
c) # of levels needed = ceil($\log_{fo} r_{i1}$) = ceil($\log_{32}$ 30_000) = 3 levels need for multi-level index
d) bfr_{i2} = floor(B / P + bytes(SSN)) = floor(512 / (6 + 9)) = floor(512 / 15) = 34; r_{i2} = b_i = 936; b_{i2} = ceil(r_{i2} / bfr_{i2}) = ceil(936 / 34) = 28; b_{i3} = ceil(b_{i2} / bfr_{i2}) = 1; 938 + 28 + 1 = 967 blocks required by the multi-level index
e) One block access per level, three levels, and one block access for data: 4 block accesses to retrieve a record

Chapter 14

14.12 (5 pts.) List the ACID properties. Explain the usefulness of each.

Response:

Atomicity – transactions are atomic, all steps must be completed or none of them are completed

Consistency – all integrity constraints must be held before and after execution of a transaction (not necessarily held during transaction)

Isolation – all transactions cannot gain access to an inconsistent database, transactions are treated as running independently of other transactions

Durability – once a transaction is completed, the data change is permanent

14.13 (2 pts.) No explanations needed. During its execution, a transaction passes through several states, until it finally commits or aborts. List all possible sequences of states through which a transaction may pass. ~~Explain why each state transition may occur.~~

Response:

Active -> partially committed -> committed

Active -> partially committed -> aborted

Active -> failed -> aborted

14.14 (2 pts.) Explain the distinction between the terms serial schedule and serializable schedule.

Response: A serial schedule is a schedule wherein each instruction in a transaction follows each other sequentially. A serializable schedule is a schedule that is equivalent to a serial schedule, equivalent either via conflict or view equivalence.

Chapters 15 & 16

15.40 (8 pts.) – This question is not included in the textbook.

Define or explain briefly the following terms.

Ø 2PL

Ø Deadlock

Ø Deferred database modification

Ø Checkpoint

Response:

2PL: Two phase locking: helps ensure secure concurrency access for database transactions by locking and unlocking data in shifts

Deadlock: a situation wherein two actions compete for use of resources and neither one of them completes

Deferred Database Modification: all transactions are written into temporary storage before being written into the database

Checkpoint: a log that keeps track of each committed transaction since the last checkpoint