**What is this document?**

* It contains the questions from midterm exam having some confusions from students, which are regarded as comments along with clarifications from instructors.
* We first highlight the question from midterm exam and provide the solution along with the comments from students.

**Why we create it?**

* To remove the confusions
* We will try our best to avoid these issues later in final exam.

**Important Note**

* We are thankful to students for identifying few legitimate cases and also request them to prepare for final exam based on covered topics.
* Midterm Exam Questions: [Link](https://drive.google.com/file/d/0B7tof8A3zkr3bHYwNWtiMXlsbU0/view)
* The following questions from mid-term exam ([Link](https://drive.google.com/file/d/0B7tof8A3zkr3bHYwNWtiMXlsbU0/view)) are excluded for grading, because, they were either too difficult or a bit ambiguous.
  + **Qs: 4, 7, 8, 17, 20, 23, 25, 28, 29, 30, 32, 37, 40, 42**

**Q 14.** Consider a fact table F(A, B, C, D, q, Price), and the following three queries: Q1: Select B, C, D, Sum(q\*Price) From F Group By B, C, D; Q2: Select B, D, Sum(q\*Price) From F Group By B, D; Q3: Select B, D, Sum(q\*Price) From F Where D < 10 Group By B, D. How many of the following statements are correct: 1) going from Q1 to Q2 is an example of roll-up. 2) going from Q2 to Q1 is an example of drill-down. 3) Q3 is an example of slicing of Q2. \*

1. • 0
2. • 1
3. • 2
4. • 3

**Answer:** D

**Comments:** 1 is roll-up since we are generalizing the results and since 1 is roll-up then 2, i.e. the reverse, would be drill-down. 3 is slicing in which we further restrict Q2 by slicing one dimension.

**Q 15.** Which of the join operations preserve non matched tuples \*

1. • Left outer join
2. • Natural join
3. • Right outer join
4. • All

**Answer:** D

**Comments:** Only INNER JOIN do not preserve since it returns all rows when there is at least one match in BOTH tables.

**Q 18.** Let relation S(A,B,C,D,E) satisfy the following functional dependencies: AB → C, BC → D, CD → E, DE → A, AE → B. How many of the following FDs are guaranteed to be satisfied by S?

1) AC → B; 2) CE → B; 3) AB → D; 4) A → AB. \*

1. • 0
2. • 1
3. • 2
4. • 3
5. • 4

**Answer:** B

**Comments:** 3 is the only guaranteed.

**Q 19.** Create table USERS(ID numeric not null, Name varchar(20) , deptname varchar(20), Salary numeric, unique(ID, Name)); S1: insert into USERS values(12, Sadegh, Dainfos, 100); S2: insert into USERS values(16,Qiang, Dainfos, ); S3: insert into USERS values(12,Joo, Dainfos,100); What will be the result of the query?: \*

1. • Error in create statement
2. • All statements executed
3. • Error in S2
4. • Error in S3

**Answer:** D

**Comments:** The not null specification prohibits the insertion of a null value for the attribute.

The unique specification says that no two tuples in the relation can be equal on all the listed attributes.

**Q 20.** Select all the incorrect statements: \*

1. • If a view involves only one base relation, the view is updatable.
2. • If all attributes of a design are key attributes, then the design at least meets 3NF
3. • Preserving dependencies up to 3NF
4. • If all keys are singleton, then the design at least meets 2NF

Answer B

Comments typo: 3NF->2NF. Not graded

**Q 21.** Given three relations, Students(SID, Sname, Sgrade, Semail, EntranceData), Rooms(RID, Rname, Rlocation, Rcapacity), and Study\_in\_Rooms (SID, RID), Students is a table with 100 thousands records stored in 1000 pages, and Study\_in\_Rooms is a table with 110 thousands

records stored in 550 pages. Consider to join operation Students x Study\_in\_Rooms, what is

the I/O cost if we use Blocked Nested Loops algorithm: \*

1. • 3300
2. • 551000
3. • 11000
4. • 8000
5. • 12850
6. • 79650

Answer F

Comments: 79650 = 1000 + Ceiling(1000/7)\*550

**Q 23.** Consider a relational DBMS that has two relations: Emp (employees) and Dept (departments). Emp(id, name, age, salary, dname) Dept(dname, location)

The Emp table has 500 tuples, and each tuple has a fixed length of 500 bytes. The primary key attribute "id" has a length of 40 bytes.

The Dept table has 100 tuples, and each tuple has 200 bytes. The primary key attribute "dname" has a length of 20 bytes. For simplicity, we assume each employee belongs to one and only one department, and each department has 5 employees. Each block has 8K (8192) bytes, in which 100 bytes are reserved for the block header. No records span two or more blocks. Consider the following disk organization strategy:

Sequential: All the Emp records are placed sequentially based on their id's. Similarly, all Dept records are stored sequentially based on their names. Suppose we want to build a primary index INDEX1 on Emp.id, and a secondary index INDEX2 on Dept.location. Treat duplicates in the secondary index in a straightforward way; that is, each tuple should have an index entry.

Assume: Each block pointer is 6 bytes. Each record pointer is 8 bytes. Index entries do not span blocks. For each index type (dense and sparse), compute the minimum number of blocks needed for INDEX1 and INDEX2, respectively. (Notice that some type of index may not make sense.) That is, compute the number of blocks for the following combinations: INDEX1 + Dense; INDEX1 + Sparse; INDEX2 + Dense; INDEX2 + Sparse. Which of the following are right? \*

1. • INDEX2 + Sparse: 3 blocks.
2. • None
3. • INDEX1 + Sparse: 4 blocks.
4. • INDEX2 + Dense: 3 blocks.
5. • INDEX2 + Sparse: 2 blocks.

**Answer:** D

**Comments:** INDEX1 + Dense; Each index-entry size: 40 + 8 = 48 bytes. Number of indexes per block (assuming INDEX1 is not spanned): floor((8192 -100)/48) = 168. Number of index blocks: ceiling(500/168) = 3.

INDEX1 + Sparse; Each index-entry size: 40 + 6 = 46 bytes. So the number of index entries per block: floor((8192 -100)/46) = 175. The number of Emp tuples in each block is: floor((8192 - 100) / 500) = 16. So the number of data blocks: ceiling(500/16) = 32. Number of index blocks: ceiling(32/175) = 1.

INDEX2 + Dense; Each index-entry size: 180 + 8 = 188 bytes. The number of data blocks per index block: floor(8092 / 188) = 43. The number of index blocks: ceiling(100/43) = 3.

INDEX2 + Sparse: it does not make sense, since the records are not stored sequentially based on the Dept.location attribute.

**Q 24.** What are the reasons we need to normalize database schema? \*

1. • To eliminate redundancies
2. • To reduce anomalies during insert, delete and updates
3. • To promote schema integration
4. • To reduce number of operations to execute a query

**Answer** A, B

**Comments**

A: A normalized database has fewer repeated values.

B: The main purpose of the normalization.

C: Normalization has nothing to do with schema integration.

D: Normalization tends to create more tables which increases required number of joins.

**Q 26.** Construct a B-tree for the following set of key values: (2, 3, 5, 7, 11, 17, 19, 23, 29, 31) Assume that the tree is initially empty, values are added in ascending order, and the fanout is n = 3. We split a node whenever necessary. We do NOT borrow the space from a sibling for a new record. Which of the following are true after the tree has been constructed? \*

1. • The root has only one key value 19
2. • None
3. • Key values 3 and 5 are in the same leaf node
4. • The root has two key values 11 and 17.
5. • The root has two key values 17 and 19

**Answer:** A

**Comments:** ·

Key values 2 and 3 are in the same leaf node.

· Key values 5 and 7 are in the same leaf node.

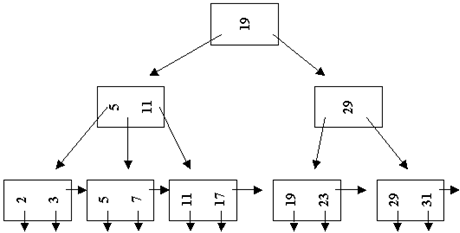
· Key values 11 and 17 are in the same leaf node.

· Key values 19 and 23 are in the same leaf node.

· Key values 29 and 31 are in the same leaf node.

· The root has only one key value 19.

The height of the tree (not including the records) is 3.



**Q 27.** Suppose we have three tables, Students(SID, Sname, Sgrade, Semail), Rooms(RID, Rname, Rlocation, Rcapacity), and Study\_in\_Rooms (SID, RID). Which queries return the names of rooms where more than 100 students study? \*

1. • None
2. • SELECT Rname FROM Rooms WHERE RID IN (SELECT RID FROM Study\_in\_Rooms GROUP BY RID HAVING Count(\*) > 100)
3. • SELECT Rname FROM Rooms r1 WHERE 100 < (SELECT Count(\*) FROM Study\_in\_Rooms r2 WHERE r2.RID = r1.RID)
4. • SELECT Rname FROM Rooms r1, Study\_in\_Rooms r2 WHERE r1.RID = r2.RID GROUP BY r1.RID HAVING Count(\*) > 100
5. • SELECT Rname FROM Study\_in\_Rooms GROUP\_BY RID WHERE Count(\*) > 100

Answer: bcd

**Q 29.** Consider an extensible hash table that uses 4-bit hash keys and stores two records per bucket. Simulate the insertion, into an initially empty hash table, of records with (hash values

of) keys 1111, 1110, 1101,..., 0001, 0000, in that order. Find in the list below a combination of i

(the number of bits used to index the bucket array) and a set of keys belong to one bucket, that appears at some time during the sequence of insertions. \*

1. • i = 3; 1000 and 1001
2. • i = 3; 1110 only
3. • i = 2; 1100 and 1101
4. • i = 3; 0010 only
5. • None

**Answer:** A

**Comments:**

To solve this problem, it is necessary to simulate all 16 insertions. Here is a table indicating what happens after each insertion. We refer to buckets by the value of their index, which is also the first *i* bits of the key for any record in that bucket.

|  |  |  |
| --- | --- | --- |
| **Inserted Record** | ***i* After Insertion** | **Changed Blocks** |
| 1111 | 1 | Block 1 = {1111} |
| 1110 | 1 | Block 1 = {1110, 1111} |
| 1101 | 3 | Block 110 = {1101}; Block 111 = {1110, 1111} |
| 1100 | 3 | Block 110 = {1100, 1101} |
| 1011 | 3 | Block 10 = {1011} |
| 1010 | 3 | Block 10 = {1010, 1011} |
| 1001 | 3 | Block 10 becomes block 101. Block 100 = {1001} |
| 1000 | 3 | Block 100 = {1000, 1001} |
| 0111 | 3 | Block 0 = {0111} |
| 0110 | 3 | Block 0 = {0110, 0111} |
| 0101 | 3 | Block 0 becomes block 011. Block 010 = {0101} |
| 0100 | 3 | Block 010 = {0100, 0101} |
| 0011 | 3 | Block 00 = {0011} |
| 0010 | 3 | Block 00 = {0010, 0011} |
| 0001 | 3 | Block 00 becomes block 001. Block 000 = {0001} |
| 0000 | 3 | Block 000 = {0000, 0001} |

**Q 32.** For the "more efficient" version of two-pass sort-join where given M memory blocks, we first divide each of the relations R and S into sorted sublists of length M blocks, and then merge all these lists, taking the join as we go. In the following, assume for simplicity that relation R does not have two tuples that agree on the join attributes, and likewise for S. Suppose r and s are the number of blocks occupied by relations R and S, respectively. What is the smallest number of main-memory blocks M needed to carry out the two-pass sort-join described? Identify, from the list below, the triple of values for which it is impossible to carry out this sort-join. \*

1. • r = 3000; s = 3000; M = 80
2. • r = 5000; s = 4000; M = 100
3. • r = 10,000; s = 2000; M = 100
4. • None is correct
5. • r = 30,000; s = 1000; M = 200

**Answer:** C

**Comments:** The rule for whether or not the efficient two-pass sort-join can be carried out is that B(R) + B(S) must be no greater than *M*2 in order for the join to be possible.

**Q 34.** What combination of SQL equivalents to relational algebra operators is valid? \*

1. • SELECT selection FROM table UNION SELECT union from TABLE 2
2. • SELECT projection AS rename FROM table WHERE selection
3. • All
4. • SELECT selection FROM table INNER JOIN table2 ON theta-join
5. • None

**Answer**: B

**Comments**:

* projection is essentially limiting set of returned attributes.
* rename allows to assign different names to relation attribute in result set
* selection operator limits result set by some condition

SQL SELECT syntax (simplified):

SELECT (list of returned columns, attribute\_name AS alias\_name)

FROM (table name)

WHERE (filtering condition)

by replacing parenthesis with corresponding relational algebra ops, we get answer B.

**Q 36.** What of the following list is not true? \*

1. • Weak entity can not be have less than two relationships
2. • Weak entity can not have own attributes
3. • None
4. • Weak entity can not have relationships with other weak entities
5. • Weak entity should have at least one attribute
6. • All

**Answer**: F

Comments:

definition of the weak entity does not prevent any of the options (except C)

**Q 37.** What of the following list is true? \*

1. • Multivalued attribute can only take limited number of possible values
2. • None
3. • All
4. • Multivalued attribute can not be implemented by SQL engine without ENUM support
5. • Multivalued attribute can take values of a mixed type (e.g. both int and string)

**Answer**: A

**Comments**: this is exactly the definition of multivalued attribute. D can not be true simply because concept of multivalued attribute is in ER models, not SQL. E can not be true because in ER model does not allow for mixed types.

**Q 42.** Aggregate functions, such as AVG, MAX and SUM can NOT be used in: \*

1. • None
2. • All
3. • Without GROUP BY statement
4. • SELECT statement without condition
5. • INSERT .. SELECT statement
6. • DELETE statement
7. • Aggregate functions can be used in all of the above cases

**Answer**: G, except it does not make sense with options A and B added

**Comments**: these are syntactically correct statements:

C, D: SELECT MAX(id) FROM Students; -- returns one row

E: INSERT INTO temp\_table(foo) SELECT MAX(id) FROM Students;

F: DELETE FROM Students WHERE id=MAX(id);

**Q 44.** Main reason not to use stored procedures \*

1. • All
2. • None
3. • Mix of storage and business logic is bad for maintainability
4. • Additional disk space requirements, because we need to store procedures
5. • Performance degradation, because SQL is slower than native code
6. • Replication issues, because we need to stop all replicas to run procedure

**Answer**: C

**Comments**: reasons why others are not true:

D: we need to store code performing the same function anyway

E: true in some cases, but pays off with advantage of reducing network communication overhead

F: simply not true, stored procedures are not different from the rest of SQL

**Q 45.** Main reason to use table constraints is: \*

1. • Better data integrity
2. • Move business logic to database layer
3. • All
4. • None
5. • Save space
6. • Increase of performance

**Answer**: A

**Comments**:

B: constraints do not implement logic

E: constraints require virtually no storage space

F: Constraints actually decrease performance

**Q 50.** UPDATE anomaly is: \*

1. • When you can not update value and have to delete and recreate an entry
2. • UPDATE command results in deletion of a data
3. • Multiple queries are required to update single attribute
4. • Multiple records are inevitably updated by single UPDATE command
5. • None
6. • All

**Answer**: C

**Comments**: definition of update anomaly straight from the textbook

**Q 51.** Main advantage of database normalization is: \*

1. • Disk space
2. • None
3. • Performance
4. • All
5. • Maintainability

Answer is C

Comment: denormalization. not graded. the exact answers for the purpose of normalization is to avoid anomaly and redundancy.