Exams 21/22 + 22/23

1. Describe the data structures and algorithms involved in the implementation of a buffer pool in the relational DBMS.

   → A buffer pool is an in-memory cache of pages read from disk. It uses a page table to track pages in memory and a replacement policy to decide which pages to evict when space is needed.

2. Present briefly the main ideas of the query optimization method based on dynamic programming.

   → Dynamic programming in query optimization breaks down a complex problem into simpler subproblems, solves each subproblem once, and stores their results. It's used to find the most efficient way to execute a database query.

3. Present the referential integrity in relational databases. Explain the mechanisms involved.

   → Referential integrity ensures that all references in data are valid and consistent. It's maintained through the use of primary and foreign keys, and rules for insertions, updates, and deletions.

4. Describe the organization of data on a disk in a database management system.

   → In a DBMS, data is organized on a disk in blocks, the smallest units of data that a DBMS can read or write. The operating system manages disk space, tracking the location of every file part, and handles tasks like disk formatting, booting, and bad block recovery.

5. Explain briefly the referential integrity in relational databases. List the control mechanisms of an RDBMS based on the referential integrity.

   → Referential integrity in relational databases ensures relationships between tables remain consistent. It's maintained through the use of primary and foreign keys, and rules for insertions, updates, and deletions

6. Describe the implementation of a hash join. Include an estimation of the number of blocks read from the disk.

   → A hash join builds a hash table from one relation and scans the other relation, looking up each tuple in the hash table.

7. Describe the internal representation of a relational table in main memory and on disk.

   → A table in a relational database is represented in main memory as a set of data structures that hold the table data. On disk, the table data is stored in a set of blocks.

8. Present possible implementations of the operation GROUP-BY.

   → The GROUP-BY operation in a database is implemented by sorting the data based on the GROUP-BY attributes, and then aggregating the data based on these sorted groups.

9. Describe the bulk loading algorithm for B+ trees. What are the differences to using a simple B+ tree add operation?

   → Bulk loading in B+ trees involves inserting a large number of entries into the tree at once, which is more efficient when inserting a large number of entries.

10. Present the meaning of integrity constraint. Describe each type of integrity constraint in one sentence.

    → Integrity constraints in databases are rules that maintain the accuracy and consistency of data. They include domain constraints, entity integrity constraints, referential integrity constraints, and key constraints.

1. Present the basic syntax of SQL query language.

```
SELECT [DISCTINCT] select-list
FROM relation
WHERE qualification;
```

   - compute the cross-product of the tables in the **relation**
   - delete rows in the cross-product that fail the **qualification** condition
   - delete all columns that do not appear in the **select-list**
   - if DISTINCT is specified, eliminate duplicate rows

2. Present the basic syntax of QBE.
   a. To write QBE query, we need to create example tables. QBE used domain variables as in DRC. They are determined by the column in which it appears, and variable symbols are prefixed with underscore. The fields that should appear int he result are specified with P., which stands for print. The fields containing this command are the same as in the SELECT clause in SQL.
3. Describe the concept of join in SQL and QBE.
   a. In SQL: → from the given tables, we compare the id's from both tables for equality. If they are equal, we get the full tuple from both tables. That means that the foreign key from second table is the same as the primary key in the first table.
   b. In QBE: → we place the same variable in the 'id' columns of the two or more example relations. Same logic as in SQL, they need to match, from both relations.
4. Present the nested queries in SQL.
   a. Nested query is a query that has a query embedded inside it. The embedded query is called subquery. This happens when we write a query and sometimes need to express a condition that refers to a table that must itself be computed. The subquery can appear in the WHERE clause, FROM clause, or HAVING clause.
5. Describe the role of statements [not] exists and [not] unique in SQL.
   a. EXISTS → allows us to test whether a set is nonempty.
   b. NOT EXISTS → allows us to test whether a set is empty, opposite of EXISTS
   c. UNIQUE → returns true if no row appears twice in the answer to the subquery (no duplicates)
   d. NOT UNIQUE → exactly the opposite of UNIQUE, removes unique rows
6. How to implement division (RA operation) in SQL and QBE?
   a. In relational databases, the division operation finds values in one table that have a relationship with all values in another table based on a common key. For the SQL part we need to use SELECTION, JOIN & WHERE clauses. The QBE part we can write it with selecting attributes from the first table where the condition matches the values in the second table.
7. Describe the use of aggregation functions in SQL and QBE.
   a. They can be applied to any column, and are used for some computation or summarization. It allows arithmetic operations. They can be used to perform some calculations oner some values and return single value. Often are performed using the GROUP BY clause of the SELECT statement.
      i. COUNT is counting the values we would specify in the query
      ii. SUM is summing up the values we would specify in the query

1. Name and describe a few external storage devices.
   a. Hard Disk Drive - most common, rotational parts and not as much fast, random access
   b. Solid State Drive - new gen., FLASH, no moving parts, more expensive than HDD and faster
   c. CD/DVD - Slower than HDD, better than Tapes
   d. Tapes - Used for archiving because cheap. sequential access
2. What is the file organization? What alternatives we have?
   a. A method for organizing and arranging a file of records on external storage.
   b. Alternatives: Heap files (unordered), Sorted files, Index files.
3. Describe the concepts a search key, a data entry, an index entry, and a data record.
   a. search key → used to search for a single entry or a set of entries in an index
   b. data entry → records stored in an index file
   c. index entry → copy of selected columns of data from a table, enabling very efficient search
   d. data record → collection of data about specific entity of an object
4. What are the alternatives for a data entry k*?
   a. a data entry k* is an actual data record, with search key value k
      i. data record with key value k
   b. a data entry is a <k, rid>, pair, where rid is the record id of a data record with search key k
      i. <k, rid of data record with search key value k>
   c. a data entry is a <k, rid-list>, a pair, where rid-list is a list of record ids of data records with key value k
      i. <k, list of rids of data records with search key k>
5. What is a primary/secondary index? What is a clustered/un-clustered index?
   a. clustered index → when a file is organized so that the ordering of the data records is the same, or close, to the ordering of data entries in some index.
      i. If order of data records is the same as, or `close to', order of data entries, then called clustered index.
   b. un-clustered index → the opposite of clustered index, the ordering of data records in a file are not the same, or not close, to the ordering of data entries in some index.
   c. primary index → if its search key also contains the primary key of the table.
   d. secondary index → all the other indexes, whose set of fields that include primary key
6. Present ISAM indexes.
   a. Static data structure, can be unbalanced
   b. If a page is full, we continue with overflow pages
   c. The data is stored in the leaves, whereas non-leaf entries serve us for navigating from the root to the actual data entries.
7. Present B+ tree index.
   a. Dynamic data structure, always is balanced
   b. All nodes need to be at least 50% full
   c. each node contains $d \leq m \leq 2d$ entries (d is the order of the tree)
   d. efficient support for equality and range search operations
8. Describe the B+ tree operations insert and delete.
   a. Insert:
      i. find the correct leaf
      ii. put the data entry in the leaf if there is space, if not, split it in two leaves, left and right, copy the middle key and add index entry pointing to the right subtree to the parent.
   b. Delete:
      i. start from the root and find the leaf L
      ii. remove it
         1. if the node is still half full, we are done
         2. if not, borrow from sibling trees, that are such that have the same parent (redistribute)
         3. if redistribution fails, we merge with the sibling
9. Describe hash-based indexes. What are the alternatives?
   a. Hash-based indexes use a hash function to quickly quickly locate records based on a key. The hash function generates a unique address that directing searches specific bucket. The alternative solution for hash-based indexes are B+ trees.