ADVANCED DATABASE SYSTEMS Coursework

Part B1

Indexing is a critical aspect of database management systems (DBMS) that improves data retrieval and query performance. There are various indexing techniques that can be utilized depending on the needs of a database.

One well-known indexing technique is B-Tree. It is commonly used in relational databases such as Oracle, MySQL, and SQL Server. B-Tree indexes are efficient for both large and small data sets, and they are especially useful for columns that are frequently searched or used in JOIN operations. They are balanced trees, which means they are always kept in a balanced state, ensuring data retrieval is fast.

Another technique is Hash indexing, it is used in databases such as MySQL, and it is particularly useful for exact-match lookups. Hash indexes work by creating a hash value for each data value, which is then used to quickly locate the corresponding data record. They are best used for columns that are used in WHERE clauses with equality conditions, but not recommended when range conditions are used in the WHERE clause.

Bitmap indexing is another technique commonly used in data warehousing and business intelligence systems, it is best suited for low-cardinality columns with a large number of distinct values. Bitmap indexes work by creating a bitmap for each distinct value, which is then used to quickly locate the corresponding data records. They are particularly useful for columns that are used in WHERE clauses with multiple conditions.

Clustered indexing is a technique that determines the physical order of data in a table, it is typically used in columns that are frequently searched or used in JOIN operations. This type of index is only available in certain types of DBMS such as SQL Server.

Lastly, full-text indexing is used in databases such as MySQL, SQL Server, and Oracle, it is used to speed up text search operations by creating an index of the words in a column. This type of indexing is particularly useful for columns that contain large amounts of text data, such as product descriptions or customer reviews.

In conclusion, indexing is an essential process in database management systems that allows for faster data retrieval and improved query performance. The most appropriate indexing technique to use depends on the specific requirements of a database, such as the types of data and the types of queries that are commonly used. Popular indexing techniques include B-Tree, Hash, Bitmap, clustered and full-text indexing. Each one has its own advantages and disadvantages, and the best one depends on the specific scenario and the data characteristics. It's important to have a good understanding of the data and the queries that will be executed on the data to choose the best indexing technique.

B-Tree indexing is a commonly used indexing technique in relational databases such as Oracle, MySQL, and SQL Server. It is efficient for both small and large data sets, and it is especially useful for columns that are frequently searched or used in JOIN operations. B-Tree indexes are balanced trees, which means they are always kept in a balanced state, ensuring that data retrieval is always fast.

In the above database, B-Tree indexing can be applied on columns such as the unique ID columns of the owners, apartments, tenants, rentals and reviews tables. This will improve the query performance when searching for specific records by their unique ID. Additionally, B-Tree indexing can be applied on composite key columns such as the composite key of the rentals table (apartment_id, tenant_id, start_date) and the composite key of the reviews table (apartment_id, tenant_id, review_id) to ensure the efficient querying of the specific rentals and reviews.

B-Tree indexing is also suitable for the TIN column in the owners table as it is frequently used to join with other tables and it's used to search for specific owners.

In summary, B-Tree indexing is a great choice for the above database as it can be applied to the unique and composite key columns, ensuring efficient data retrieval and improved query performance.

Part B2

A star schema or a snowflake schema could be used for the database created in Part A.

A star schema is a type of database schema where a central fact table is connected to one or more dimension tables through foreign keys. The fact table contains the measures or quantitative data, while the dimension tables contain the attributes or qualitative data.

A snowflake schema is a variation of the star schema, where the dimension tables are normalized and connected to other dimension tables through foreign keys. This allows for more detailed data and reduces data redundancy.

Here's an example of how we can implement a snowflake schema for the apartment rental database:

Create a fact table called rentals that contains the rental ID, apartment ID, tenant ID, review ID, start date, rental days, number of tenants and total rental price.

Create a dimension table called apartments that contains the apartment ID, owner TIN, street, street number, postal code, city, floor, surface, and price.

Create a dimension table called owners that contains the owner TIN, name, surname, email, city, bank account, contact number and date of birth.

Create a dimension table called tenants that contains the tenant ID, name, surname, tenant TIN, gender, email, date of birth, country, contact number and bank account.

Create a dimension table called reviews that contains the review ID, date of the review, star review point, and review text.

- DIM_OWNERS:
    - O_TIN (Primary Key)
    - Name
    - Surname
    - Email
    - City
    - Bank account
    - Contact number
    - Date_of_birth
- DIM_APARTMENTS:
    - Apartment_ID (Primary Key)
    - O_TIN (Foreign Key to DIM_OWNERS)
    - Street
    - Street_num
    - Postal_code
    - City
    - Floor
    - Surface
    - Price
- DIM_TENANTS:
    - Tenant_ID (Primary Key)
    - Name
    - Surname
    - Ten_TIN
    - Gender
    - Email
    - Date_of_birth
    - Country
    - Contact number
    - Bank account

- FACT_RENTALS:
    - Rental_ID (Primary Key)
    - Apartment_ID (Foreign Key to DIM_APARTMENTS)
    - Tenant_ID (Foreign Key to DIM_TENANTS)
    - Review_ID (Foreign Key to DIM_REVIEW)
    - Start_date
    - Rental days
    - Number of tenants
    - Total rental price
- DIM_REVIEWS:
    - Review_ID (Primary Key)
    - Review_date
    - Stars
    - Review_text

In this case, the data related to the reviews, tenants and apartments are included in the fact table FACT_RENTALS and linked to the corresponding dimension tables through the use of foreign keys (Review_ID, Tenant_ID, Apartment_ID). The snowflake schema is implemented with the link of owners table to the apartments table using the o_tin as foreign key.

SNOWFLAKE SCHEMA