**Introduction to normalization :**

<https://www.geeksforgeeks.org/introduction-of-database-normalization/>

**Normal forms :**

<https://www.geeksforgeeks.org/normal-forms-in-dbms/>

**Functional dependency :**

<https://www.geeksforgeeks.org/what-is-functional-dependency-in-dbms/>

**Decomposition :**

<https://www.geeksforgeeks.org/decomposition-in-dbms/>

**Dependency preservation and lossless join :**

<https://www.geeksforgeeks.org/lossless-join-and-dependency-preserving-decomposition/>

**multivalued dependencies :**

<https://www.geeksforgeeks.org/multivalued-dependency-mvd-in-dbms/>

**Query Optimization**

<https://www.geeksforgeeks.org/query-optimization-in-relational-algebra/>

**operations of relational algebra :**

<https://www.geeksforgeeks.org/introduction-of-relational-algebra-in-dbms/>

**heuristic based optimization :**

<https://www.geeksforgeeks.org/what-is-heuristic-optimization-in-dbms/>

**cost estimation based optimization :**

<https://www.geeksforgeeks.org/cost-based-optimization/>

**problems with null valued and dangling tuples**

Handling null values and dangling tuples in databases poses several challenges. Here's a detailed look at the problems associated with each:

**Problems with Null Values**

Null values represent missing or unknown data in a database. While useful, they introduce several issues:

**1. Ambiguity in Meaning:**

**- Unknown vs. Not Applicable:** Null can mean that the value is unknown, does not apply, or is not available, which can be ambiguous

**2. Logical Operations and Comparisons:**

**- Three-Valued Logic (3VL):** SQL uses three-valued logic (true, false, unknown) when dealing with nulls, complicating logical operations.

**- Comparisons with Null:** Comparisons involving nulls result in unknown, leading to unexpected behavior in queries.

**3. Aggregations and Functions:**

**- Aggregate Functions:** Functions like `SUM`, `AVG`, `COUNT`, etc., may yield unexpected results when nulls are involved. For instance, `COUNT` does not include nulls, potentially causing misinterpretation of data.

**4. Joins and Conditions:**

**- Join Operations:** Null values can affect the results of join operations, especially in outer joins, leading to incomplete or misleading results.

**5.Indexing and Performance**:

**- Indexing:** Null values might not be indexed, affecting query performance and optimization.

**6. Data Integrity:**

**- Constraints:**Nulls can violate data integrity constraints, such as unique or foreign key constraints, if not handled properly.

**7. Application Logic:**

**- Complexity in Code:** Handling null values increases the complexity of application code, requiring additional checks and conditions.

**Problems with Dangling Tuples**

Dangling tuples refer to rows in a database that reference non-existent rows in another table, typically due to referential integrity issues. They pose several problems:

**1. Referential Integrity Violations:**

- **Broken Relationships:** Dangling tuples indicate broken relationships between tables, leading to inconsistent data.

**2. Data Integrity:**

- **Orphan Records**: Dangling tuples, or orphan records, can result from deletions in the referenced table without corresponding deletions in the referencing table.

**3. Query Errors and Inaccuracies:**

**- Erroneous Results:** Queries involving dangling tuples may return incorrect or incomplete results, leading to potential misinterpretation of data.

4. **Maintaining Data Consistency:**

- **Cascade Actions:** Ensuring consistency through cascade updates and deletes can be complex and error-prone.

**5. Application Logic and Performance**:

**- Additional Checks:** Applications need additional logic to handle dangling tuples, increasing complexity and affecting performance.

**6. User Trust:**

- \*\*Trust Issues:\*\* Inconsistent and incomplete data due to dangling tuples can erode user trust in the database's accuracy and reliability.

**Solutions**

To mitigate these problems, several strategies can be employed:

**1. For Null Values:**

- \*\*Default Values:\*\* Use default values instead of nulls where appropriate.

- \*\*Nullable Columns:\*\* Carefully design which columns should allow nulls.

- \*\*Database Constraints:\*\* Use constraints and triggers to enforce business rules.

- \*\*Application Logic:\*\* Implement consistent handling of nulls in application code.

**2. For Dangling Tuples:**

- \*\*Foreign Key Constraints:\*\* Enforce foreign key constraints to maintain referential integrity.

- \*\*Cascade Actions:\*\* Use cascade delete and update actions to automatically maintain consistency.

- \*\*Integrity Checks:\*\* Regularly perform integrity checks to identify and clean up dangling tuples.

- \*\*Database Triggers:\*\* Use triggers to enforce referential integrity rules.