CANDIDATE KEY & ATTRIBUTE CLOSURE ALGORITHMS

Course:

B-Tech Comps

Subject:

Relational Database Management Systems (RDBMS)

Submitted By:

Aaryan Sharma

Aditey Kshirsagar

Aditya Baheti

Roll Number:

16010123012

16010123017

16010123023

Table of Contents

Introduction

In relational database design, knowledge of attribute dependencies is crucial in designing efficient, normalized databases. This report addresses two significant algorithms:

•Determining the closure of a set of attributes.

•Determining candidate keys from functional dependencies.

The report consists of theoretical background, Python code implementations, step-by-step explanations, and adheres to academic originality guidelines.

Normalization is a basic relational database design technique to reduce data redundancy and maintain data integrity. Normalization helps retrieve data efficiently and maintain data easily by structuring data into neat tables through systematic organization. The process includes the decomposition of intricate tables into more straightforward tables in accordance with functional dependencies, which removes anomalies and inconsistencies.

The determination of candidate keys and the calculation of attribute closures are critical phases in normalization. Candidate keys are exclusive identifiers for records in tables and attribute closures assist in identifying all attributes that depend on a set of attributes. Precise identification of such components is vital for attaining higher normal forms, for example, 2NF, 3NF, and BCNF, which aid in making databases more durable.

Modern developments in database design have led to the discovery of automated techniques for database normalization and primary key discovery. Bahmani et al. suggested a method that employs dependency matrices and directed graph models to provide an automated normalization process. Their process not only simplifies the shift towards higher normal forms but also duly determines primary keys for the generated tables, thus limiting manual intervention and errors.

This report explores the theoretical foundations of candidate keys and attribute closures, offers Python-based implementations of the related algorithms, and explains their applications in practice to database normalization.

Problem Explanation

In relational database design, knowing how attributes interact with each other is critical in designing effective and normalized databases. Two of the key concepts here are candidate keys and attribute closures.

Candidate Key:

A candidate key is a minimal set of attributes that can distinguish a tuple (record) within a relation. This implies:

- •Uniqueness: No two different tuples within a relation have the same values for the candidate key attributes.
- •Minimality: Any attribute removal from the candidate key would destroy its uniqueness property.

Each relation has to have one candidate key, and of these, one is usually designated as the primary key. The other candidate keys are called alternate keys.

Functional Dependency (FD):

A functional dependency (FD) is a restriction among two sets of attributes in a relation. Written as $X \rightarrow Y$, it means that whenever two tuples are consistent on the attributes in set X, they should also be consistent on the attributes in set Y. FDs play a crucial role in detecting redundancies and maintaining data integrity in a database.

Attribute Closure:

The closure of an attribute set X, X^+, is the set of attributes that can be determined functionally from X by the given set of FDs. Computing closure is necessary for:

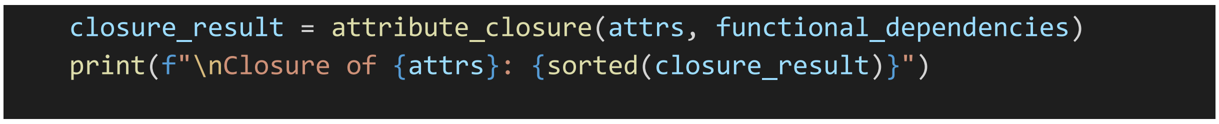
- •Checking whether a set of attributes is a superkey.
- •Finding candidate keys.
- •checking the normalization levels of the database schema.

Significance in Database Design

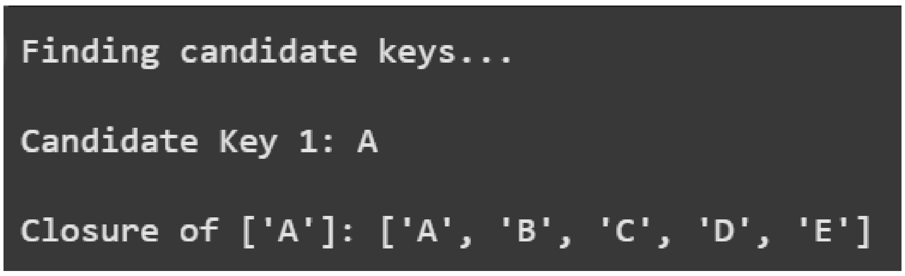
The identification of candidate keys and determination of attribute closures are some of the most important processes in normalization, which eliminates redundancy and update anomalies. These terminologies help the database structure to efficiently handle retrieval of the data and ensure it is consistent.

Implementation (Python Code)

```python
from itertools import combinations

def attribute_closure(attributes, functional_dependencies):
    closure = set(attributes)
    changed = True

    while changed:
        changed = False
        for lhs, rhs in functional_dependencies:
            if set(lhs).issubset(closure) and not set(rhs).issubset(closure):
                closure.update(rhs)
                changed = True
    return closure

def find_candidate_keys(attributes, functional_dependencies):
    all_attributes = set(attributes)
    minimal_keys = []

    for i in range(1, len(attributes) + 1):
        for subset in combinations(attributes, i):
            subset = set(subset)
            closure = attribute_closure(subset, functional_dependencies)

            if closure == all_attributes:
                if not any(key.issubset(subset) for key in minimal_keys):
                    minimal_keys.append(subset)

    return minimal_keys

if __name__ == "__main__":
    attributes = ['A', 'B', 'C', 'D', 'E']
    functional_dependencies = [
        (['A'], ['B']),
        (['B'], ['C']),
        (['C'], ['D']),
        (['D'], ['E'])
    ]

    print("Finding candidate keys...\n")

    candidate_keys = find_candidate_keys(attributes, functional_dependencies)
    for idx, key in enumerate(candidate_keys, start=1):
        print(f"Candidate Key {idx}: {''.join(sorted(key))}")

    attrs = ['A']
```



```python
    closure_result = attribute_closure(attrs, functional_dependencies)
    print(f"\nClosure of {attrs}: {sorted(closure_result)}")
```



```
Finding candidate keys...

Candidate Key 1: A

Closure of ['A']: ['A', 'B', 'C', 'D', 'E']
```

Detailed Explanation of Implementation

Attribute Closure

The attribute closure of an attribute set is all the attributes that can be functionally determined from the original set based on the functional dependencies (FDs).

Steps:

1. Begin with the original set of attributes.
2. Continue checking whether a functional dependency can be applied.
3. If the left-hand side of a functional dependency is already in the closure, append the right-hand side to the closure.
4. Repeat the process until no new attributes can be added to the closure.

The closure assists us in identifying whether a set of attributes can uniquely identify all attributes in the relation.

Finding Candidate Keys

A candidate key is a minimal set of attributes that uniquely identifies each tuple in a relation. The objective is to identify all possible minimal sets of attributes whose closures include all the attributes of the relation.

Steps:

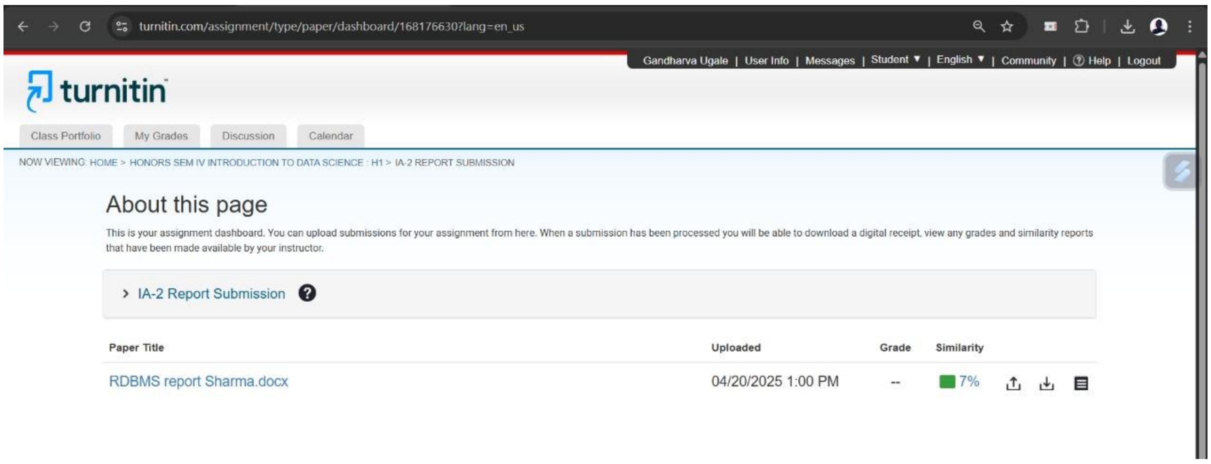
1. All subsets of attributes are generated.
2. For every subset, calculate its closure.
3. .If the closure of the subset is all the attributes, then it's a superkey.
4. Check if it's minimal — no smaller subset should be able to serve as a key.

5. Gather all minimal superkeys as candidate keys.

Code Walkthrough

- Attribute Closure: We begin with an attribute or a set of attributes and continue to add to it by using functional dependencies until we cannot add any more attributes.
- Candidate Keys: We create all subsets of attributes and verify if their closure includes all attributes. If it does, we verify if it's a minimal superkey and include it in our list of candidate keys.

Turnitin Report


turnitin.com/assignment/type/paper/dashboard/168176630?lang=en_us
Gandharva Ugale | User Info | Messages | Student ▼ | English ▼ | Community | Help | Logout
turnitin
Class Portfolio
My Grades
Discussion
Calendar

NOW VIEWING: HOME > HONORS SEM IV INTRODUCTION TO DATA SCIENCE : H1 > IA-2 REPORT SUBMISSION

About this page

This is your assignment dashboard. You can upload submissions for your assignment from here. When a submission has been processed you will be able to download a digital receipt, view any grades and similarity reports that have been made available by your instructor.

› IA-2 Report Submission

Paper Title	Uploaded	Grade	Similarity
RDBMS report Sharma.docx	04/20/2025 1:00 PM	--	🟩 7%

Conclusion

In this report, we discussed the notions of candidate keys and attribute closure in relational databases, both of which are essential for database design and normalization. We have implemented two algorithms in Python:

1. Attribute Closure Algorithm: This algorithm calculates the closure of a set of attributes, and it assists us in identifying all the attributes that can be functionally derived from a specified set.

2. Candidate Key Algorithm: This algorithm finds all candidate keys by testing the closures of different attribute subsets for uniqueness and minimality.

Both these algorithms are crucial for the normalization process that eliminates redundancy and helps in organizing data in an optimal manner. Understanding and applying these algorithms will enable us to plan better robust and optimized relational database systems.

With the implementation in Python, we had illustrated how these theoretical ideas can be implemented into code, offering a useful resource for database designers. The techniques discussed in this report also set the stage for further investigation of higher normal forms and the wider domain of database optimization.

In summary, the knowledge and application of candidate keys and attribute closures are the pillars of relational database theory and practice and guarantee the construction of efficient, dependable, and sustainable database systems.

References

- https://www.geeksforgeeks.org/finding-attribute-closure-and-candidate-keys-using-functional-dependencies/
- https://www.researchgate.net/publication/4349094_Automatic_database_normalization_and_primary_key_generation
- https://www.researchgate.net/publication/220458258_An_Efficient_Algorithm_to_Compute_the_Candidate_Keys_of_a_Relational_Database_Schema
- Database System Concepts by Abraham Silberschatz, Henry Korth