**“*ANU”***(Automated Normalisation Utility)

Ideas of this project evolve out from the central concept of RDBMS Normalisation, & how well that theory can be applied into practise.

Project will composed of several Modules, each having its own motivation & function, they are listed below as:

1. **FDM (Functional Dependencies Miner)**,A Tool to Extract possible Canonical cover of possible set of functional dependencies from the available instance of DB, this topic is specific to Data Mining. Such a tool will be helpful if the set of FDs from which available schema is designed in unavailable which is common if original designer if not there, also user can go initially for Universal relation (a single big table containing all attributes), which can later be decomposed depending upon the need of scaling of performance or derived FDs
2. **DBDE (Data-Base Decomposition Engine),** based on available/extracted set of FDs, this is a classical problem in DBMS, with advent of computers this can be done very easily in for large set of FDs.

This tool will provide the schema of Data-Base in all possible forms from 1NF to BCNF (highest used standard in industry). Also, schema in all Normal forms will be available in form of Tree like structure, where each level from top represents schema in a standard normal form.

1. **QREE (Query Resource Evaluation Engine)**, this will help estimating the time & memoryrequirements needed in query processing to be operated on system under workload prepared based on statistical analysis of past queries’ records. This help usto select optimal decomposed schema.

It might also be possible that any schema in one standard normal form is not optimal, so we can go for the combination between two levels, this led to non-standard normal form but promises best performance

1. **DBRM (Data-Base Reorganisation Module),** to convert initial instance of database to final schema. This module will be capable of performing both decomposition or merging of schema depending upon the optimal schema sent by above described module

LOD (Level-of-Difficulty) for implementing these Modules will be 4<2<1<3

**A discussion about availability of such tools**

<https://dba.stackexchange.com/questions/24748/is-there-a-tool-to-check-if-my-database-is-normalized-to-the-third-normal-form>

**Kind of our Module 3 implementation,** butthere arefew drawbacks in it

<http://www.ict.griffith.edu.au/~jw/normalization/ind.php>

**Where this Idea Begins from?**

Normalisation Theory!! One of the most important part when it comes to discussion of RDBMS, but isn’t highly used by moderate level designer due to lack to experience of applicability & tools needed for its proper implementation.

A DBMS is designed with various aspects considered in mind of an expertdesigner (definition of which still varies person to person), Normalisation theory is not the only one, but one of the most important one at conceptual level design of RDBMS.

So initial idea is to find schema in various normal forms based on set of FDs, things are added later to which based on work done in this area before & seemingly needs to us

**Module 1:“FDM” (Functional Dependencies Miner)**

Our initial assumption is that set of FDs applicable on a schema will be available to be converted into any final schema & there is nothing like present module.

But things don’t work the same way we think, sooner we realised that it may be a common issue that our FD set is missing & all our plan for further processing got stuck at this step. There are work done previously in direction, so we decided to study them at first & then move on.

We have improved one of the latest technique used in this problem domain& tested in on the worst case data sets prepared by our code itself to test the complexity measures & theoretical estimations, efficiency in terms of time & space; and at last scalability on the parallel architecture, which can be utilised nowadays.

**Module 2:“DBDE” (Data-Base Decomposition Engine)**

Based on the simple theoretical concept, this module is initially designed to help student learn & understand applications of FDs in a general problems of finding Candidate keys, Minimal cover, & decomposition into standard normal forms.

Later on we’ve realised how computers can be used for such operation at design time for large set of FDs where humans can’t find solution in efficient & error free manner.

This module finds Canonical cover, Candidate Keys, & all normal forms (2NF, 3NF & BCNF) in a very elegant manner & can also show each of it’s step & logic behind it.

Also it is capable of finding whether the given set of FDs is valid or invalid for conversion to any normal form & if found invalid, it shows the FDs that led to invalidation of whole set & suggests user to remove the least important functional dependency among them.

Along with this, there are FDs that makes few decompositions from 3NF to BCNF non-dependency-preserving. In that case we don’t force user to go for BCNF, instead we show that which of FDs are lost during decomposition, now its choice of user to consider that FD & go with 3NF, or ignore the loss of that FD & go with BCNF.

This module returns the final schema in form of a hierarchical data structure, where universal relation (1NF) is at root node of the tree, each of its child will be a subset, & second level forms tha 2NF, this division based data structure has 4 levels (0 for 1NF & last 3 for BCNF), which is returned by this module.

**Module 3: “QREE” (Query Resource Evaluation Engine)**

Can’t say a shit about it now

**Module 4:“DBRM” (Data-Base Reorganisation Module)**

This module does the simplest yet valuable task, which is to convert initial schema into the final one, based on the schema supplied by module 3.

The role can be either to decompose any table into two or more, also it can be to merge few relations into one.

#CHECKPOINT

This begins with the availability of candidate keys of the initial & final schema, depending upon whether a decomposition or merge is to be done. On the side (initial or final) with more number of relations, natural join of keys has to be performed.