




Interactive Decomposition of Relational Database Schemes Using Recommendations

Raji Ghawi^(✉) 

Bavarian School of Public Policy, Technical University of Munich,
Richard-Wagner-Straße. 1, 80333 Munich, Germany
raji.ghawi@tum.de

Abstract. Schema decomposition is a well known method for logical database design. Decomposition mainly aims at redundancy reduction and elimination of anomalies. A good decomposition should preserve dependencies and maintain recoverability of information. We propose a semi-automatic method for decomposing a relational schema in an interactive way. A database designer can build the subschemes step-by-step, guided by quantitative measures of decomposition “goodness”. At each step, a ranked set of recommendations are provided to the designer to guide him to the next possible actions that lead to a better design.

Keywords: Relational databases · Functional dependencies · Schema decomposition · Normal forms · Recommendation

1 Introduction

Logical database design is a wide field that has been very well studied in the past decades. Schema decomposition is a well known method for logical database design which mainly aims at eliminating anomalies and reducing redundancy. Decomposing a relation involves splitting its attributes to make the schemes of new relations.

In fact, careless selection of a relational database schema can lead to redundancy and related anomalies. It also introduces the potential for several kinds of errors. Therefore, one of the motivations for performing a decomposition is that it may eliminate anomalies and reduce redundancy. Normal forms have long been studied as a means of reducing redundancies caused by data dependencies in the process of schema design.

Besides anomaly elimination, literature suggests two other properties that are desired in a decomposition [1, 16, 27]: information recoverability and dependency preservation. Recoverability of information means that the original relation can be recovered by joining the relations in the decomposition. Dependency preservation means that the set of dependencies of the original schema are preserved within the subschemes of the decomposition. Among all the data dependencies

which have been proposed in the literature, functional dependencies (FDs) are the most common kind of constraints in a relational database.

In this paper, we propose a semi-automatic method for decomposing a relational schema in an interactive way. A database designer can build the sub-schemes step-by-step guided with quantitative measures of the decomposition “goodness”. Our goal is to provide richer insights that help a database designer have a better understanding of the consequences of different decomposition choices and make design decisions accordingly. Therefore, at each step, a ranked set of recommendations are provided to the designer that guide him to the possible next actions that lead to a better design. In order to do so, quantitative “goodness” measures of the decomposition need to be defined.

The paper is organized as follows. Section 2 provides some preliminaries, and Sect. 3 discusses the properties of good decomposition. Quantitative measures of decomposition goodness are defined in Sect. 4. Then, in Sect. 5 we present our proposed method for interactive decomposition using recommendations. Section 6 concludes the paper.

2 Preliminaries

Let $R(A_1, \dots, A_n)$ be a relation scheme, and X and Y be subsets of $\{A_1, \dots, A_n\}$. We say X functionally determines Y , denoted $X \rightarrow Y$ if, in whatever instance r of R , any two tuples that agree on X values they also agree on Y values. We say a relation instance r satisfies functional dependency $X \rightarrow Y$ if for every two tuples t_1 and t_2 in r such that $t_1[X] = t_2[X]$, it is also true that $t_1[Y] = t_2[Y]$.

Let F be a set of FD’s for relation scheme R , and let $X \rightarrow Y$ be a FD. We say F logically implies $X \rightarrow Y$, if every instance r for R that satisfies the dependencies in F also satisfies $X \rightarrow Y$. The closure of F , denoted F^+ , is the set of functional dependencies that are logically implied by F .

Armstrong [3] defined a complete and sound set of inference rules for functional dependencies:

- *Reflexivity*. If $Y \subseteq X$, then $X \rightarrow Y$ holds (called trivial FD).
- *Augmentation*. If $X \rightarrow Y$ holds, then $XZ \rightarrow YZ$ holds.
- *Transitivity*. If $X \rightarrow Y$ and $Y \rightarrow Z$ hold, then $X \rightarrow Z$ holds.

Let F be a set of FD’s on set of attributes U , and let X be a subset of U , then, the closure of X (with respect to F), denoted X^+ , is the set of attributes A such that $X \rightarrow A$ can be deduced from F by Armstrong’s axioms. The FD $X \rightarrow Y$ follows from a given set of dependencies F using Armstrong’s axioms if and only if $Y \subseteq X^+$; where the closure of X is taken with respect to F .

If R is a relation scheme with attributes A_1, \dots, A_n and functional dependencies F , and X is a subset of A_1, \dots, A_n , we say X is a *superkey* of R if $X \rightarrow A_1, \dots, A_n$ is in F^+ . That is, X functionally determines all attributes of R . We say X is a *key* of R if X is a superkey and no proper subset $Y \subset X$ is a superkey of R . Clearly, a relation can have more than one key. An attribute A

is called *prime* if it belongs to any key. Thus, an attribute that does not belong to any key is called *non-prime*.

Let F and E be two sets of FD's, we say that F and E are equivalent, denoted $F \equiv E$, if and only if $F^+ = E^+$. That is, we can deduce all FD's of E from F , and vice versa. Moreover, we say that E is a minimal cover of F if and only if $E \equiv F$ and there is no proper subset $E' \subset E$ such that $E' \equiv F$.

If R is a relation scheme with attributes A_1, \dots, A_n and functional dependencies F , and S is a subscheme of R with attributes $A_{i_1}, \dots, A_{i_k} \subseteq A_1, \dots, A_n$. The projection of F over S , denoted F_S is the set of functional dependencies $X \rightarrow Y$ in F^+ such that $(X \cup Y) \subseteq A_{i_1}, \dots, A_{i_k}$.

Let R be a relation schema with FDs F .

- We say that R is in *First Normal Form* (1NF) if all the attributes are atomic (single-valued).
- We say that R is in *Second Normal Form* (2NF) [8] if it is in 1NF and for a non-prime attribute A , whenever $X \rightarrow A$ is in F^+ , then X is not a proper subset of any key.
- We say that R is in *Third Normal Form* (3NF) if whenever $X \rightarrow A$ is in F^+ , then either X is a superkey of R or A is a prime attribute.
- We say that R is in *Boyce-Codd Normal Form* (BCNF) if whenever $X \rightarrow A$ is in F^+ , then X is a superkey of R .

Clearly, if R is in 3NF then it is also in 2NF; and if it is in BCNF then it is also in 3NF.

Other normal forms are formulated in the literature, such as Fourth Normal Form (4NF) [17], Fifth Normal Form (5NF) [11,18] (also known as Project-Join Normal Form, PJNF), Sixth Normal Form (6NF) [10,12], and Essential Tuple Normal Form (ETNF) [9]. However, these normal forms are based on other types of dependencies: multivalued-dependencies (MVD's) and join-dependencies (JD's). Therefore, they are beyond the scope of this paper.

A decomposition of a relation scheme $R(A_1, \dots, A_n)$ is its replacement by a collection $\delta = \{R_1, \dots, R_k\}$ of subsets of R , called *subschemes*, such that $R = R_1 \cup \dots \cup R_k$. The subsets R_i 's are not required to be disjoint. If r is an instance of R then the sub-instance of a subscheme R_i is the projection of r on R_i , that is, $r_i = \pi_{R_i}(r)$.

One of the motivations for performing a decomposition is that it may eliminate (insert, delete and update) anomalies and reduce redundancy.

3 What Is a Good Decomposition?

A decomposition of a relation scheme R is its replacement by a collection $\delta = \{R_1, \dots, R_k\}$ of subsets of R , called *subschemes*, such that $R = R_1 \cup \dots \cup R_k$. The subsets R_i 's are not required to be disjoint. If r is an instance of R then the sub-instance of a subscheme R_i is the projection of r on R_i , that is, $r_i = \pi_{R_i}(r)$.

Literature shows three main properties that a decomposition is desired to have: [4,21,22]: Elimination of Anomalies, Recoverability of Information, and Preservation of Dependencies.

1. *Elimination of Anomalies* can be described in terms of normal forms. The literature shows that certain undesirable anomalies can be avoided when the database scheme is in a normal form w.r.t the given dependencies [22].
2. *Recoverability of Information* (or lossless-join) means that the original relation can be recovered by taking the natural join of the relations in the decomposition. In fact, any decomposition gives back at least the tuples with which we start, but a carelessly chosen decomposition can give tuples in the join that were not in the original relation [19]. Formally, a decomposition $\delta = \{R_1, \dots, R_n\}$ of a schema R is recoverable iff for whatever instance r of R : $\pi_{R_1}(r) \bowtie \dots \bowtie \pi_{R_n}(r) = r$. Lossless-join property can be checked using *chase* test [2].
3. *Preservation of Dependencies* means that all the functional dependencies that hold in the original relation can be deduced from the FD's in the decomposed relations. Formally, a decomposition $\delta = \{R_1, \dots, R_n\}$ of a schema R with a set of FD's F is dependency preserving iff: $\bigcup_{i=1}^n F_{R_i} \equiv F$ where F_{R_i} is the projection of F over R_i . An algorithm to test the dependency preservation property is given in [5].

The literature shows several well known works on decomposition methods that achieve some of the above desired properties. Bernstein [6] proposed a “synthetic” approach for dependency-preserving decomposition into 3NF. Tsou and Fischer [26] proposed a lossless-join decomposition into BCNF. Biskup et al. [7] give a 3NF decomposition with a lossless join and dependency preservation. Demba [13] propose an algorithmic approach for database normalization up to third normal form.

However, it has been shown that there is no decomposition that guarantees all the three properties at once. That is, sometimes, decomposition into BCNF can lose the dependency-preservation property, while decomposition into 3NF does not guarantee to eliminate all redundancy due to FD's [19]. Moreover, it is important to remember that not every lossless-join decomposition step is beneficial, and some can be harmful [27]. Codd has argued that we should not insist that a relation schema be in a given normal form. Rather, the database designer should be aware of the issues and have a warning flag that if the relation schema is not in a given normal form, then certain problems may arise.

A database designer may wish to thoroughly investigate many decompositions in order to choose a good one for his design. However, the number of possible decompositions of a schema is exponential; thus, examining the whole the search space of decompositions is almost unfeasible. In order to overcome this drawback, we suggest to navigate step-wisely through this space. That is, from some ‘current’ decomposition, the designer can investigate the neighbors of that decomposition that are one step away; where a step means an action of adding or removing of a subscheme or an attribute.

In this paper, we propose an interactive decomposition method, that supports this idea of navigational search for good decomposition. Our goal is to guide the database designer throughout the decomposition process with quantitative measures that assesses the “goodness” of the decomposition.

In the literature, there are many works that have proposed tools and solutions to support database decomposition, such as Mirco [15], RDBNorma [14], and JMathNorm [28]. Some of such tools have educational purposes only, including: [20, 23–25]. However, to the best of our knowledge, there is yet no tool that support interactive decomposition in the manner we do in this paper, where recommendations (equipped with quantitative measures) for next steps are provided to guide the user during the decomposition process.

In the next section, we define decomposition “goodness” measures.

4 Decomposition Goodness Measures

Let R be a relation schema with FD’s F . Let D_R be the set of all possible decompositions over R , then a **goodness measure** is a function: $\vartheta : D_R \mapsto [0, 1]$. Let $\delta \in D_R$ be a decomposition of k subschemes: $\delta = \{R_1, \dots, R_k\}$, we define four goodness measures:

4.1 Join-Lossless Measure ϑ_J

This is a strict measure that takes value 1 when δ is join-lossless, and 0 otherwise:

$$\vartheta_J(\delta) = \begin{cases} 1 & \text{if } \delta \text{ is join lossless} \\ 0 & \text{otherwise} \end{cases}$$

4.2 Dependency-Preservation Measure ϑ_P

This measure can also be defined in a strict fashion: $\vartheta_P(\delta) = 1$ when δ is dependency-preserving, 0 otherwise.

$$\vartheta_P(\delta) = \begin{cases} 1 & \text{if } \delta \text{ is dependency-preserving} \\ 0 & \text{otherwise} \end{cases}$$

Alternatively, it can be defined in a relaxed fashion:

$$\vartheta_P(\delta) = \frac{|F_{pr}|}{|F|}$$

where $F_{pr} \subseteq F$ is the set of FD’s preserved by δ .

4.3 Normal-Forms Measure ϑ_N

Let R_i be a subscheme in δ , then the normal-forms measure of R_i is defined as:

$$\vartheta_N(R_i) = \begin{cases} 3 & \text{if } R_i \text{ is in BCNF} \\ 2 & \text{if } R_i \text{ is not in BCNF, but in 3NF} \\ 1 & \text{if } R_i \text{ is not in 3NF, but in 2NF} \\ 0 & \text{if } R_i \text{ is not in 2NF} \end{cases}$$

The normal-forms measure ϑ_N of a decomposition δ is then the normalized average of the normal-forms measure of its components:

$$\vartheta_N(\delta) = \frac{1}{3k} \sum_{i=1}^k \vartheta_N(R_i)$$

Clearly, ϑ_N would be 0 when none of the subschemes is in 2NF, and reaches 1 when all the subschemes are in BCNF.

4.4 Structural Issues Measure ϑ_I

When a decomposition is built in an interactive way by adding or removing subschemes and/or attributes, structural *issues* may arise. These issues can be classified in different types as shown in Table 1.

Let Ω_δ be the set of all the structural issues occurring in a decomposition δ , the Structural-Issues Measure ϑ_I is defined as:

$$\vartheta_I(\delta) = \frac{1}{1 + |\Omega_\delta|}$$

Clearly, ϑ_I can not be 0; it would be 1 when the decomposition δ does not suffer any structural issues ($\Omega_\delta = \emptyset$).

Table 1. Types of structural issues

Issue	Formulation
A subscheme has no attributes	$\exists R_i \in \delta : R_i = \emptyset$
A subscheme has one attribute only	$\exists R_i \in \delta : R_i = 1$
A subscheme is the same as the original schema	$\exists R_i \in \delta : R_i = R$
An attribute is not mentioned in any relation	$\exists a \in R, \forall R_i \in \delta : a \notin R_i$
Two subschemes have exactly the same attributes	$\exists R_i, R_j \in \delta : R_i = R_j$
A subscheme is a proper subset of another	$\exists R_i, R_j \in \delta : R_i \subset R_j$
A subscheme has no shared attributes with others	$\exists R_i \in \delta, \forall R_j \in \delta / R_i : R_i \cap R_j = \emptyset$

The list shown in Table 1 contains the most common and frequent types of structural issues. These types are what we consider in the current version of our tool. However, we do not claim that this list of issues is exhaustive.

Moreover, the formula of Structural-Issues Measure can be modified to obtain a smoother decay of score when the number of issues increases, for example, using $\vartheta_I(\delta) = \frac{3}{3 + |\Omega_\delta|}$.

Total Score θ

To summarize the above goodness measures, a total score θ is defined as their weighted average. Let w_J, w_P, w_N , and w_I be non-negative weights that sum up to 1. These weights signify the degree of importance of the goodness measures, $\vartheta_J, \vartheta_P, \vartheta_N$, and ϑ_I , respectively. The total score θ is computed as:

$$\theta(\delta) = w_J\vartheta_J(\delta) + w_P\vartheta_P(\delta) + w_N\vartheta_N(\delta) + w_I\vartheta_I(\delta)$$

Example 1. Let $R(A, B, C, D, E)$ be a schema with FD's $F = \{A \rightarrow B, E \rightarrow G, B \rightarrow DE\}$, let $\delta = \{R_1(B, D, E), R_2(C, E, G)\}$ be a decomposition of R . Then, the FD's of the subschemes are: $F_{R_1} = \{B \rightarrow DE\}$ and $F_{R_2} = \{E \rightarrow G\}$.

- δ is not join-lossless: $\vartheta_J(\delta) = 0$.
- FD's $E \rightarrow G$ and $B \rightarrow DE$ are preserved, but $A \rightarrow B$ is not: $\vartheta_P(\delta) = 2/3$.
- R_1 is in BCNF, and R_2 is not in 2NF: $\vartheta_N(R_1) = 3$, $\vartheta_N(R_2) = 0$, thus $\vartheta_N(\delta) = (3 + 0)/(3 \times 2) = 1/2$.
- δ suffers one issue: attribute A is not mentioned in any relation, thus $|\Omega| = 1$, and $\vartheta_I(\delta) = 1/2$.
- Assuming balanced weights, the total score is: $\theta = (0 + \frac{2}{3} + \frac{1}{2} + \frac{1}{2})/4 = \frac{5}{12}$.

5 Interactive Decomposition and Recommendations

In order to help a database designer make a good decomposition, s/he first needs to be able to measure the “goodness” of the decomposition, and second to better understand the consequences of different decomposition choices; thus s/he can make design decisions accordingly. We propose a semi-automatic method for decomposing a relational schema in an interactive way, where the designer can build the subschemes in a step-by-step way.

Given an original database schema R , and a set of FD's F , the designer starts with one empty subscheme (with no attributes). Then, s/he can add attributes to this subscheme or add another subscheme. Gradually, the decomposition becomes richer and other types of actions become possible, such as removing an attribute from certain subscheme or removing an entire subscheme. In general, four types of actions are possible at any step:

1. Add a new subscheme. $\delta \leftarrow \delta \cup R'$
2. Remove a subscheme. $\delta \leftarrow \delta - R'$
3. Add an attribute to a subscheme. $\delta \leftarrow (\delta - R') \cup (R' \cup \{a\})$
4. Remove an attribute from a subscheme. $\delta \leftarrow (\delta - R') \cup (R' - \{a\})$

Every time an action is taken, two things happen. First, the “goodness” measures (Sect. 4) are computed for the current decomposition δ , including the total score $\theta(\delta)$. This involves finding the FD projections and normal forms of each subscheme, and the set of structural issues, if any.

Second, a list of next-possible-actions $\Psi = \{\psi_1, \dots, \psi_m\}$ is generated (Sect. 5.1). For each next-possible-action $\psi_j \in \Psi$, the corresponding decomposition δ_{ψ_j} (resulting if the action ψ_j is taken) is computed behind the scene, along with its goodness metrics including the total score $\theta(\delta_{\psi_j})$. The total score is used to rank the list of actions, such that the actions with higher scores are displayed first. Moreover, each action is annotated as *positive*, *equivalent* or *negative*, respectively, based on whether the action score $\theta(\delta_{\psi_j})$ is, respectively, *higher than*, *equal to*, or *less than* the score of the current decomposition $\theta(\delta)$.

This way, a ranked list of action-score pairs is presented to the designer: $\{\langle \psi_j, \theta(\delta_{\psi_j}) \rangle\}$. In this list, *positively* annotated actions form the recommendations that are given to the designer to choose one from them. Those recommendations are guaranteed to give, if taken, a better decomposition than the current one in terms of the total score of the goodness measures.

When an action (unnecessarily positive) is taken, the list of recommendations will change (re-computed). That is because the next possible actions would be different, and their corresponding decompositions will vary accordingly.

Obviously, the designer is not obligated to take the first recommendation in the list, s/he is free to take any action. However, taking top recommended actions would rapidly lead to better decomposition. It is important to note that an optimal decomposition is not always possible. Therefore, the designer may stop the decomposition process whenever s/he feels satisfied with the goodness measures s/he gets.

A prototype of this interactive method is implemented in Java as a user-friendly GUI (Fig. 1). The source code, examples, and other resources are available at: <https://goo.gl/rSPOin>.

5.1 Algorithm for Recommendation Generation

The list of recommendations of each step is generated using Algorithm 1. For each subscheme, we consider actions of adding an attribute that is not currently in the subscheme (lines 3:7), and removing an attribute from the subscheme (lines 8:12). If there are more than one subscheme, we also consider removing each subscheme (lines 13:17). For each key, we consider adding a subscheme that consists of the attributes of that key, if such a subscheme is not already present in the decomposition (lines 18:22). For each FD in the minimal cover of F , we consider adding a subscheme that consists of the left and right side attributes of that FD, if such a subscheme is not already in the decomposition (lines 23:28).

For each one of these actions, ψ , the corresponding decomposition δ_ψ is found and its total score $\theta(\delta_\psi)$ is computed. The pairs $\langle \psi, \theta_\psi \rangle$ are then added to the list of recommendations Ψ , which is, at the end, ordered by the values of θ_ψ and returned.

Algorithm 1. Compute a List of Recommendations**Require:** R original schema, F a set of FD's, δ a decomposition of R .**Ensure:** Ψ a set of recommendations.

```

1:  $\Psi := \emptyset$ 
2: for each subscheme  $R_i \in \delta$  do
3:   if  $|R - R_i| > 1$  then
4:     for each attribute  $A \in R - R_i$  do
5:        $\psi := (\text{Add } A \text{ to } R_i)$ ;
6:        $\delta_\psi := (\delta - R_i) \cup (R_i \cup \{A\})$ ;
7:        $\theta_\psi := \theta(\delta_\psi)$ ;  $\Psi := \Psi \cup \{\langle \psi, \theta_\psi \rangle\}$ 
8:   if  $|R_i| > 1$  then
9:     for each attribute  $B \in R$  do
10:       $\psi := (\text{Remove } B \text{ from } R_i)$ ;
11:       $\delta_\psi := (\delta/R_i) \cup (R_i/\{A\})$ ;
12:       $\theta_\psi := \theta(\delta_\psi)$ ;  $\Psi := \Psi \cup \{\langle \psi, \theta_\psi \rangle\}$ 
13: if  $|\delta| > 2$  then
14:   for each subscheme  $R_i \in \delta$  do
15:      $\psi := (\text{Remove } R_i \text{ from } \delta)$ ;
16:      $\delta_\psi := \delta/R_i$ ;
17:      $\theta_\psi := \theta(\delta_\psi)$ ;  $\Psi := \Psi \cup \{\langle \psi, \theta_\psi \rangle\}$ 
18: for each key  $K$  of  $R$  do
19:   if  $K \notin \delta$  then
20:      $\psi := (\text{Add } K \text{ to } \delta)$ ;
21:      $\delta_\psi := \delta \cup K$ ;
22:      $\theta_\psi := \theta(\delta_\psi)$ ;  $\Psi := \Psi \cup \{\langle \psi, \theta_\psi \rangle\}$ 
23: for each FD  $X \rightarrow Y \in E = \text{minCover}(F)$  do
24:    $S := X \cup Y$ 
25:   if  $S \notin \delta$  then
26:      $\psi := (\text{Add } S \text{ to } \delta)$ ;
27:      $\delta_\psi := \delta \cup S$ ;
28:      $\theta_\psi := \theta(\delta_\psi)$ ;  $\Psi := \Psi \cup \{\langle \psi, \theta_\psi \rangle\}$ 
29: order  $\Psi$  by  $\theta_\psi$  desc.
30: return  $\Psi$ 

```

5.2 Example

Consider the schema $R(A, B, C, D, E, G)$ with FD's $F = \{AB \rightarrow CD, A \rightarrow E, B \rightarrow G, EG \rightarrow C\}$. Let us start a new decomposition δ with a new subscheme R_1 and add attribute A to R_1 . At this point, the goodness measures are: $\vartheta_J = 0$, $\vartheta_P = 0$, $\vartheta_N = 1$, $\vartheta_I = 0.125$ (due to 7 issues), and $\theta = 0.28125$. The top 3 recommendations are:

$\langle \psi_{11} :: \text{Add attribute } E \text{ to subscheme } R_1, 0.3542 \rangle$
 $\langle \psi_{12} :: \text{Add new subscheme: } \{C, E, G\}, 0.3542 \rangle$
 $\langle \psi_{13} :: \text{Add new subscheme: } \{A, E\}, 0.3482 \rangle$

If we take the first recommendation, we get $\delta = \{R_1(A, E)\}$ with $F_{R_1} = \{A \rightarrow E\}$. This will make $\vartheta_P = 0.25$ (1 FD is preserved), and $\vartheta_I = 0.1667$ (issues are reduced to 5); therefore, $\theta = 0.3542$ (as the recommendation promised). The top-3 next recommendations are:

- $\langle \psi_{21} :: \text{Add new subscheme: } \{C, E, G\}, 0.4583 \rangle$
- $\langle \psi_{22} :: \text{Add new subscheme: } \{B, G\}, 0.425 \rangle$
- $\langle \psi_{23} :: \text{Add new subscheme: } \{A, B, D\}, 0.3958 \rangle$

If we again take the first recommendation, we get $\delta = \{R_1(A, E), R_2(C, E, G)\}$ with $F_{R_2} = \{EG \rightarrow C\}$. This will make $\vartheta_P = 0.5$ (2 FD's are preserved), and $\vartheta_I = 0.333$ (2 issues remaining: attributes B, D are not mentioned in any subscheme); therefore, $\theta = 0.4583$ (Fig. 1).

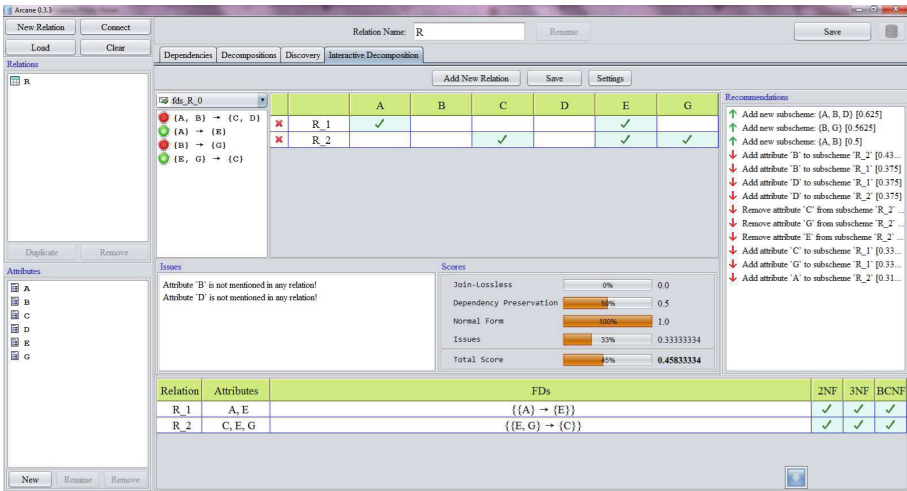


Fig. 1. Decomposition after adding subscheme $R_2\{C, E, G\}$

The top-3 next recommendations are:

- $\langle \psi_{21} :: \text{Add new subscheme: } \{C, E, G\}, 0.4583 \rangle$
- $\langle \psi_{22} :: \text{Add new subscheme: } \{B, G\}, 0.425 \rangle$
- $\langle \psi_{23} :: \text{Add new subscheme: } \{A, B, D\}, 0.3958 \rangle$

Continuing the decomposition process with recommended actions, we shall add $R_3(A, B, D)$ and $R_4(B, G)$, hence an optimal decomposition can be achieved: $\delta = \{R_1(A, E), R_2(C, E, G), R_3(A, B, D), R_4(B, G)\}$ which is lossless-join, all FD's are preserved, all subschemes are in BCNF, and free from structural issues; thus, $\theta(\delta) = 1$.

6 Conclusions and Future Work

We have proposed a semi-automatic method for interactive decomposition of relational databases. A database designer is guided throughout the decomposition process by a list of recommendations that tell him what are the next possible actions that can lead to a better decomposition.

Interactive decomposition seems to be a good approach to help the designer better understand the pros and cons of every possible action by quantitatively assessing the goodness of the decomposition. This method can be considered as an interesting addition to the arsenal of already established methods and tools within database design literature. As a future work, an extensive experimentation is needed to evaluate the method, and to study the impact of each goodness property on the overall quality of database design. Looking ahead, interactive decomposition can be extended to involve extra goodness measures such as: minimality, freedom from globally redundant attributes, and freedom from attribute replication. Moreover, it can be extended to consider other types of dependencies (e.g. MVDs) and higher normal forms (e.g. 4NF).

Other directions of future work concern improving the tool and making the interface more user-friendly; by introducing, for instance, graphical representation on functional dependencies in the form of FD diagrams, and graphical representation of join- diagram of the decomposition.

Finally, future work should address the usefulness of the decomposition method in real-life situations where the search space is huge. Therefore, it is necessary to conduct some experiments with real-life databases consisting of many, possibly non-normalized, relations.

References

1. Abiteboul, S., Hull, R., Vianu, V.: *Foundations of Databases*. Addison-Wesley, Boston (1995)
2. Aho, A.V., Beeri, C., Ullman, J.D.: The theory of joins in relational databases. *ACM Trans. Database Syst.* **4**(3), 297–314 (1979)
3. Armstrong, W.W.: Dependency structures of data base relationships. In: *IFIP Congress*, pp. 580–583 (1974)
4. Arora, A.K., Carlson, C.R.: The information preserving properties of relational database transformations. In: *Proceedings of the Fourth International Conference on Very Large Data Bases, VLDB 1978*, vol. 4, pp. 352–359. VLDB Endowment (1978)
5. Beeri, C., Honeyman, P.: Preserving functional dependencies. *SIAM J. Comput.* **10**(3), 647–656 (1981)
6. Bernstein, P.A.: Synthesizing third normal form relations from functional dependencies. *ACM Trans. Database Syst.* **1**(4), 277–298 (1976)
7. Biskup, J., Dayal, U., Bernstein, P.A.: Synthesizing independent database schemas. In: *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, SIGMOD 1979*, pp. 143–151. ACM, New York (1979)
8. Codd, E.F.: A relational model of data for large shared data banks. *Commun. ACM* **13**(6), 377–387 (1970)

9. Darwen, H., Date, C.J., Fagin, R.: A normal form for preventing redundant tuples in relational databases. In: Proceedings of the 15th International Conference on Database Theory, ICDT 2012, pp. 114–126. ACM, New York (2012)
10. Date, C.J., Darwen, H., Lorentzos, N.A.: Temporal Data and the Relational Model. Elsevier, Amsterdam (2002)
11. Date, C.: An Introduction to Database Systems, 8th edn. Addison-Wesley Longman Publishing Co. Inc., Boston (2003)
12. Date, C., Darwen, H., Lorentzos, N.: Time and Relational Theory, Second Edition: Temporal Databases in the Relational Model and SQL, 2nd edn. Morgan Kaufmann Publishers Inc., San Francisco (2014)
13. Demba, M.: Algorithm for relational database normalization up to 3NF. *Int. J. Database Manag. Syst.* **5**, 39–51 (2013)
14. Dongare, Y., Dhabe, P., Deshmukh, S.: RDBNorma: a semi-automated tool for relational database schema normalization up to third normal form. arXiv preprint [arXiv:1103.0633](https://arxiv.org/abs/1103.0633) (2011)
15. Du, H., Wery, L.: Micro: a normalization tool for relational database designers. *J. Netw. Comput. Appl.* **22**(4), 215–232 (1999)
16. Elmasri, R., Navathe, S.: Fundamentals of Database Systems, 6th edn. Addison-Wesley Publishing Company, Boston (2010)
17. Fagin, R.: Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst.* **2**(3), 262–278 (1977)
18. Fagin, R.: Normal forms and relational database operators. In: Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, SIGMOD 1979, pp. 153–160. ACM, New York (1979)
19. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database Systems: The Complete Book, 2nd edn. Prentice Hall Press, Upper Saddle River (2008)
20. Kung, H.J., Tung, H.L.: A web-based tool to enhance teaching/learning database normalization. In: Proceedings of the 2006 Southern Association for Information Systems Conference. Jacksonville (2006)
21. Maier, D.: The Theory of Relational Databases. Computer Science Press, Rockville (1983)
22. Maier, D., Mendelzon, A.O., Sadri, F., Dullman, J.: Adequacy of decompositions of relational databases. *J. Comput. Syst. Sci.* **21**(3), 368–379 (1980)
23. Piza-Dávila, H.I., Gutiérrez-Preciado, L.F., Ortega-Guzmán, V.H.: An educational software for teaching database normalization. *Comput. Appl. Eng. Educ.* **25**(5), 812–822 (2017)
24. Stefanidis, C., Koloniari, G.: An interactive tool for teaching and learning database normalization. In: Proceedings of the 20th Pan-Hellenic Conference on Informatics, PCI 2016, pp. 18:1–18:4. ACM, New York (2016)
25. Taofiki, A.A., Tale, A.O.: A visualization tool for teaching and learning database decomposition system. *J. Inf. Comput. Sci.* **7**(1), 003–010 (2012)
26. Tsou, D.M., Fischer, P.C.: Decomposition of a relation scheme into Boyce-Codd normal form. *SIGACT News* **14**(3), 23–29 (1982)
27. Ullman, J.D.: Principles of Database and Knowledge-base Systems, vol. I. Computer Science Press Inc., New York (1988)
28. Yazici, A., Karakaya, Z.: JMathNorm: a database normalization tool using mathematics. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4488, pp. 186–193. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72586-2_27