Regression Testing

of database applications under an incremental software development settings

By: RAÚL H. ROSERO1,2, OMAR S. GÓMEZ2, AND GLEN RODRÍGUEZ

Presenter: Aryan Ebrahimpour

Professor: Saeed Zahedi

Microsoft

What is regression testing?

- verifies previous features on a software product when it is modified or new features are added to it
- a costly process

Why not retest all?

- Executing the total set of test cases is an expensive and laborious endeavor
- Therefore determining a subset of test cases to be executed in a software regression test is an active topic of research

Different approaches

- **Minimization**: using greedy algorithms
- **Selection**: using graph based algorithms
- **Prioritization**: using capacity based fault detection
- Recently **Soft computing**

Why soft computing?

- the execution of the complete suite of test cases is an ***unacceptable practice*** for iterative and incremental development environments
- This raised the study of soft computing approaches, in conjunction to **agile approaches**
- Identify test cases that **Guarantee an acceptable level** of verification of software product

A more effective alternative

- Cluster test cases according to a pattern, criterion and characteristic
  - Cluster: Test cases that detect the same fault
  - How to cluster? Based on:
    - Execution Profile
    - Histories of test runs
    - Function calls
    - Partitioning programs
    - Filtering partitions
    - Access to database
- Finally: Sample the most representative test cases for each cluster

Challenges

- Determine the quality and size of clusters to balance the cost and effectiveness of approach
- Additional information to data such as numbers and sizes of clusters to be formed makes process expensive
  - Alternative: use unsupervised clustering
    - Probabilistic approaches determines number of clusters

Challenges

- Conventionally RT has been applied from **Code** perspective
  - Software is an FSM with constant Data
- Software with access to Database has two **Code** and **Data** perspective
  - **Code State** is a set of labeled memory
  - **Data State** organized by different by different data models, *such as relational*
  - Large amount of **Data** can be affected by a single line of **code**

Paper specs

- What are we presenting in this paper?
  - RT for software with DB access under incremental software dev context
- Our approach:
  - Clustering of DB access code along with DB Schema (by analyzing of code that accesses the DB)
  - Unsupervised
  - Random values that selects a set of test cases

Related Work

- Some of other RT approaches

Minimalization

```
let p = software product

let t = test cases associated with p

task -→ obtain a subset of test cases t’ where
        t’ does not include redundant or obsolete
        test cases to verify p
```

Selection

```
let p' = modifications to software product

task  -> determine a subset of test cases t' where
             p' are tested with t'
```

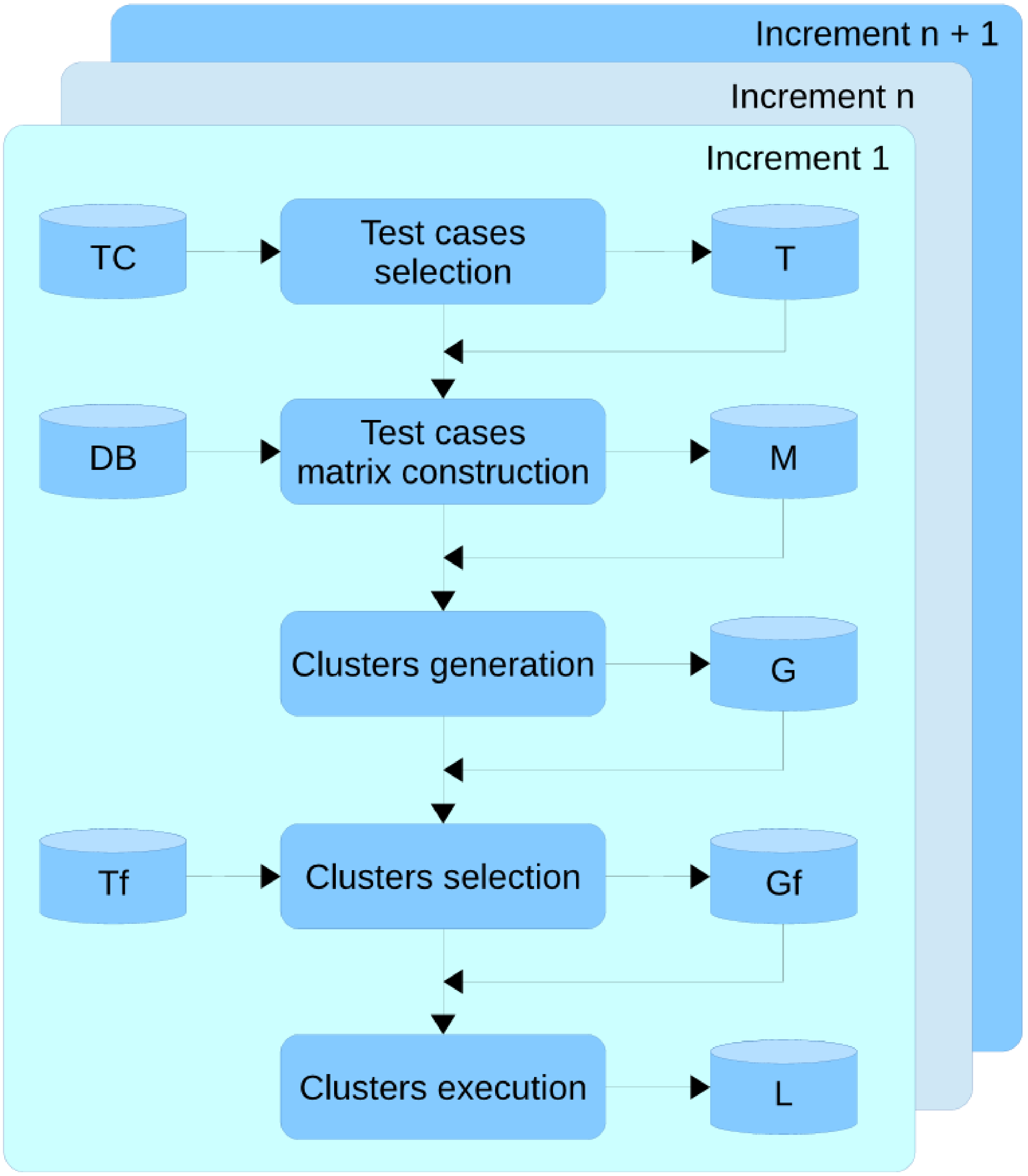
Prioritization

task -→ determine an ideal permutation of sequences of test cases to improve the RT technique

Proposed RT Approach

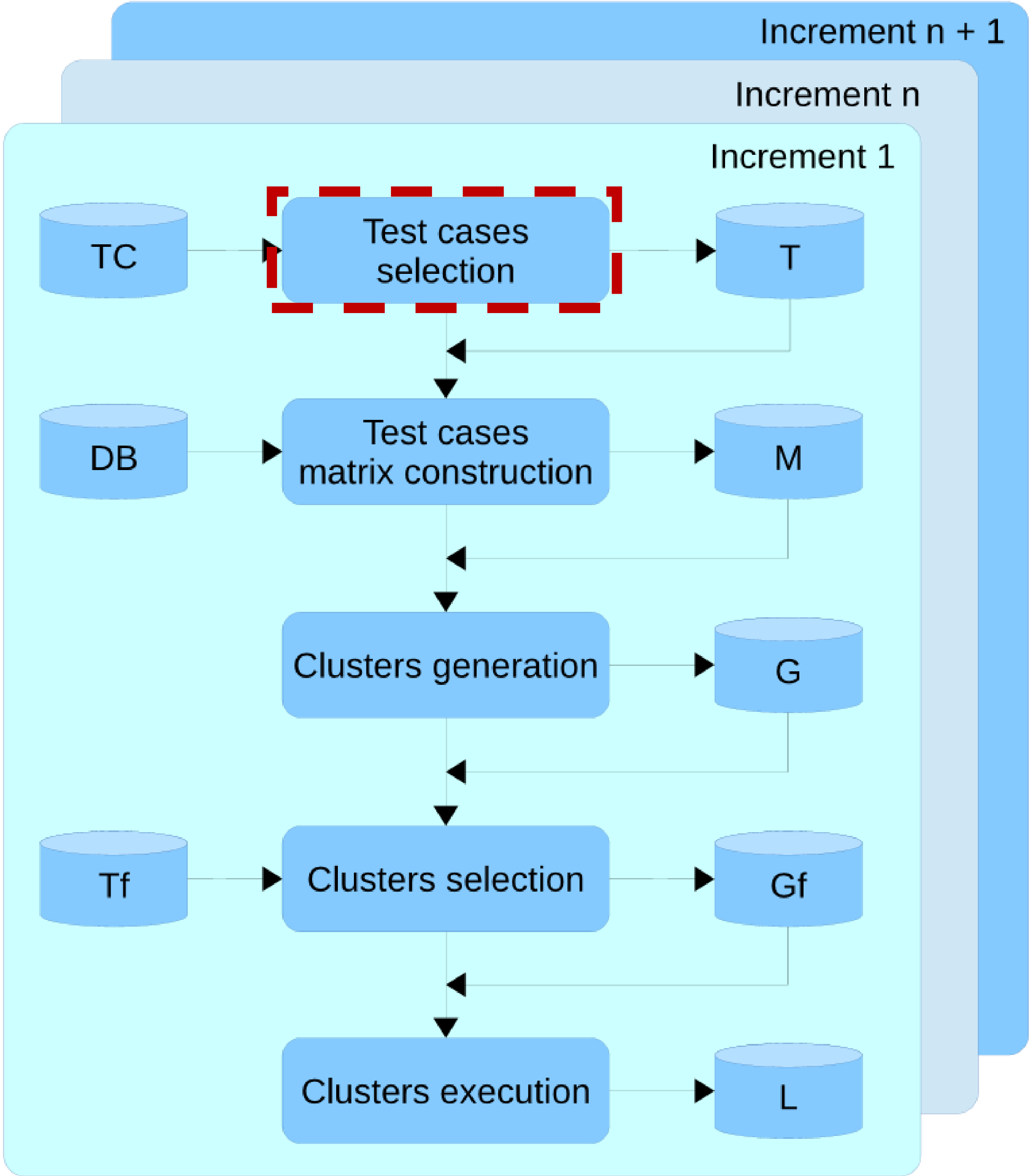
Terminology

TC:	Test cases of a given increment
T:	Set of test cases with access to the database
DB:	Database schema
M:	Matrix of similarities
G:	Clusters of test cases
Tf:	Set of test cases with faults
Gf:	Clusters of test cases with faults
L:	List of faults of the regression testing



Selection

```
let new = new features and modifications
let T = from new where item access fields
        of given table of DB
```



TC:	Test cases of a given increment
T:	Set of test cases with access to the database
DB:	Database schema
M:	Matrix of similarities
G:	Clusters of test cases
Tf:	Set of test cases with faults
Gf:	Clusters of test cases with faults
L:	List of faults of the regression testing

Similarity matrix construction

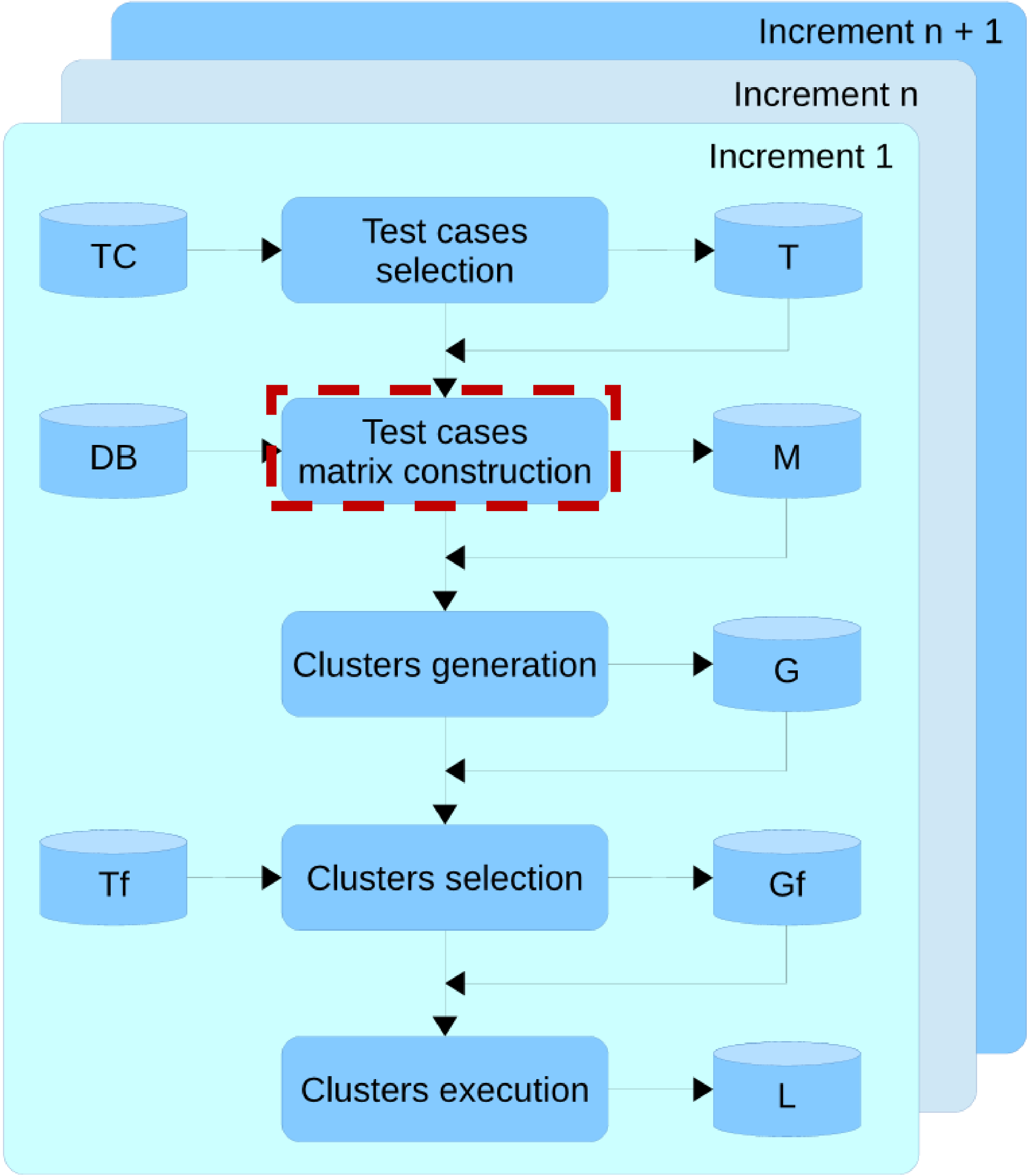
```
task -→ obtain info of tables and columns
         of each test case T

        yield binary matrix M[m, n] where
                0 -→ no DB access
                1 -→ DB access operation

        rows(m) -→ test cases
        cols(n) -→ field of DB tables
```

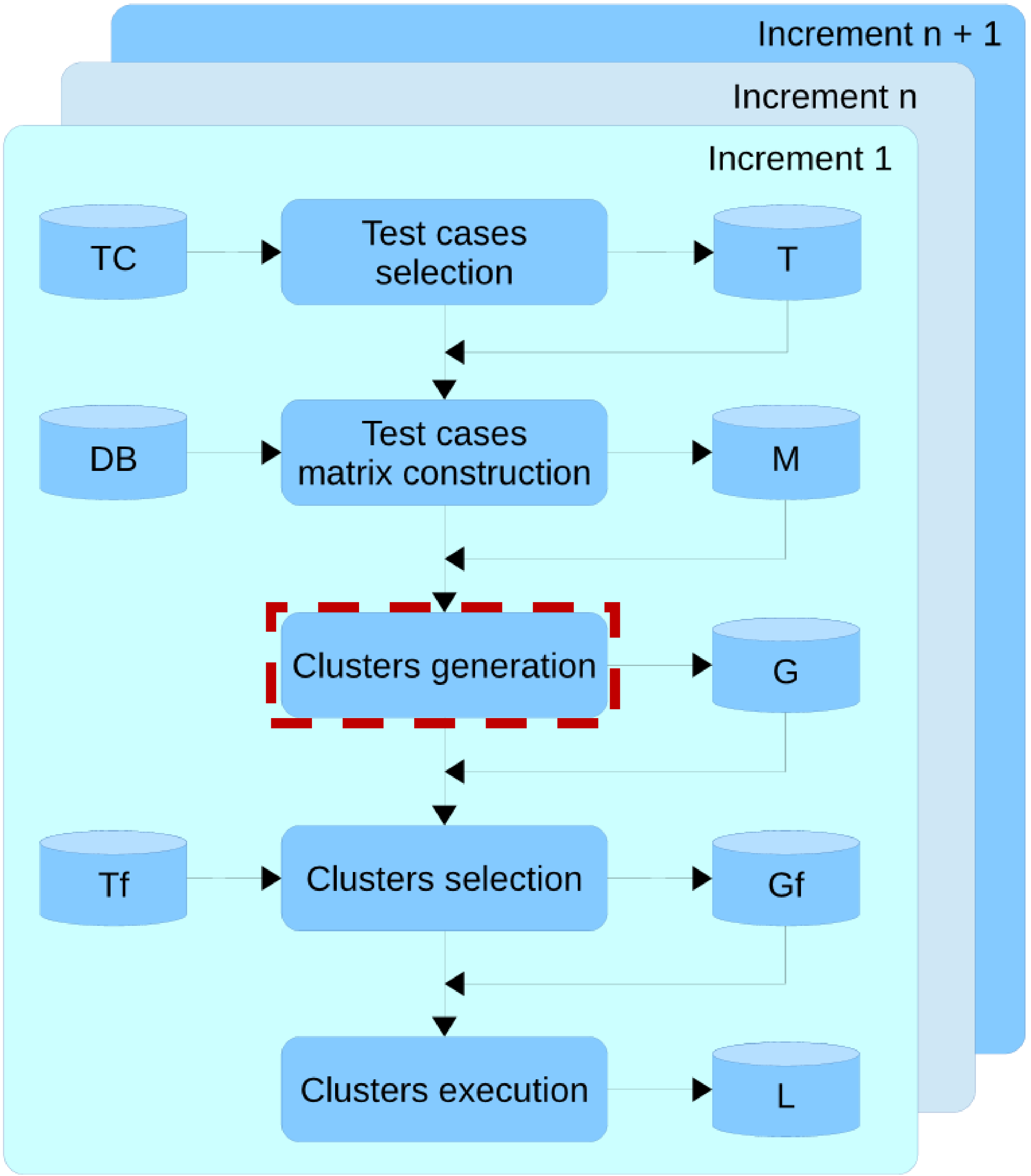
TABLE 2. Matrix of test cases accessing the DB.

	Tables						
	$Table_1$			…	$Table_n$		
	Col_1	Col_2	Col_i	…	Col_1	Col_2	Col_i
TC_1	1	0	1	…	0	1	0
	.	.	.	….	….	…	…
TC_n	.	.	.		….	…	…



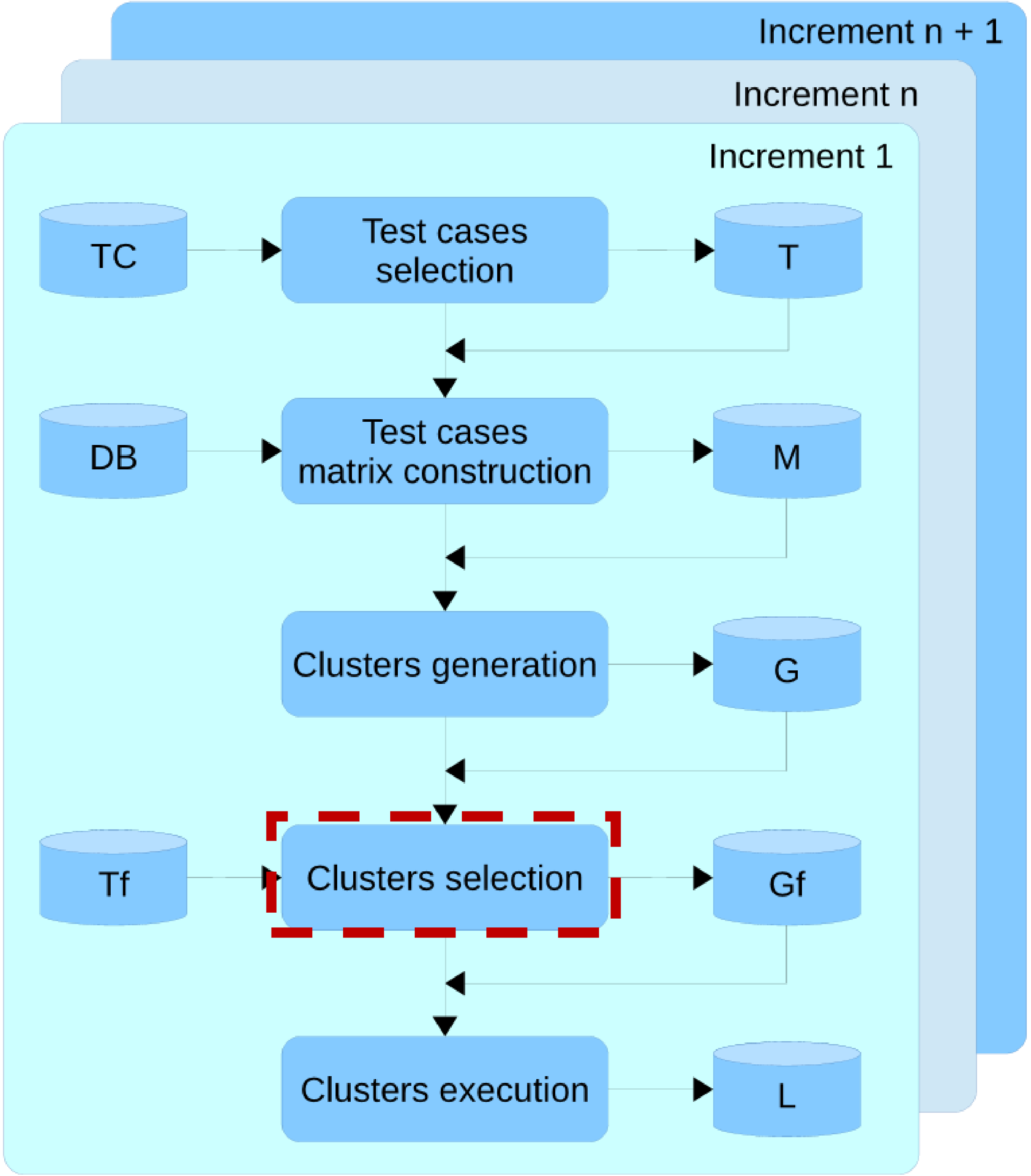
Clusters Generation

- ✓ Unsupervised clustering
- ✓ Expectation maximization
- ✓ We determine a **probabilistic distribution** function to assign a membership of a **test case** to a **cluster** based on **similarity matrix**
- ✓ Finite gaussian mixture model which all attributes are random
- ✓ **Result is one or more clusters**



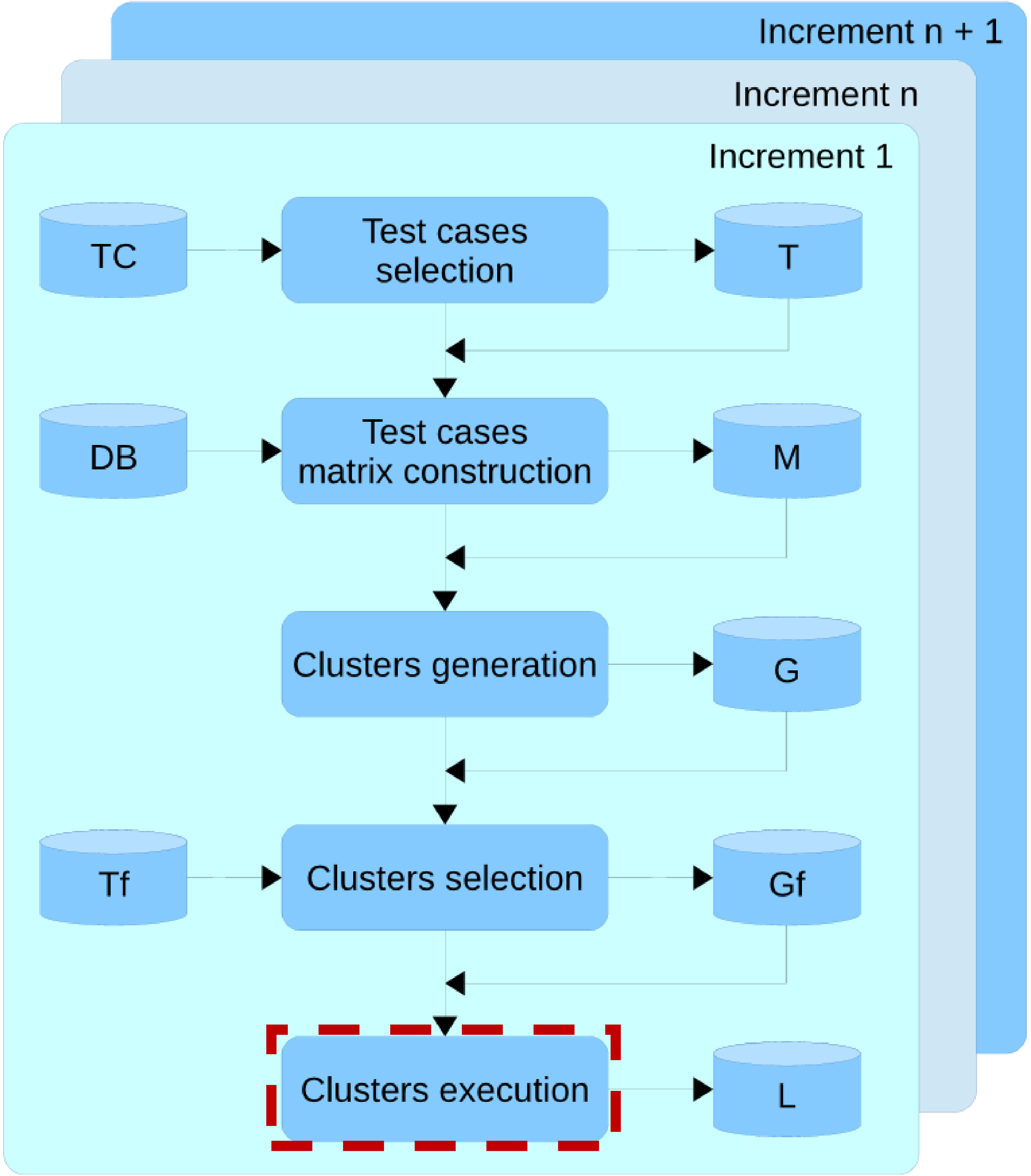
Clusters Selection

- ✓ Result: What clusters are going to be executed for RT
- ✓ Which clusters (New or modified test cases = Gf) are located
- ✓ Considering Tf: Test cases with faults



Clusters execution

- ✓ In this level, Gf are executed
- ✓ Generates a list L of faults related to test cases



Algorithm used in proposal

Algorithm Identifiers

Variable	Description
DB	Database
T	Set of test cases
F	Set of features
Fi	i-th feature, $f_i \in F$
TC_i	Set of test cases from i-th feature $f_i \in F$
tc_i	i-th Test case of a feature
$tc_{i,j}$	i-th Test case $tc_i \in TC_i$ from j-th feature
TestDB	Set of test cases with access to DB
TestDBfailed	Set of test cases failed with access to DB
MatTestDB	Test cases matrix with access DB
MatClusterTestDB	Test cases cluster matrix with access DB
MatClusterRegressionTestDB	Failed matrix of clustered test cases with access to DB
FaultsList	Set of faults for the regression test
NumTestDB	Number of test cases with access to the DB

Main algorithm

1. For each feature (fi), a selection of test cases that access DB is performed
2. Similarity matrix construction
3. Clustering analysis
4. Execute clusters containing failed clusters, and their faults examined

Algorithm 1 clusterSelectionRegression Test

Input: f_i, TC_i
Output: Faults List

clusterSelectionRegressionTest process
repeat
  for each f_i do //fi $\in$ F
    TestDB $\leftarrow$ selectTestDB(f_i, TC_i)
    MatTestBD $\leftarrow$ buildMatrixTestDB(testDB)
    MatClusterTestDB $\leftarrow$ clusterizationEM(MatTestBD)
      MatClusterRegressionTestDB $\leftarrow$ (MatCluster TestDB $\cap$ TestDBfailed)
      executeClusterRegressionTestDB (MatClusterRegressionTestDB)
  end for
until $\nexists$ f_i
end clusterSelectionRegressionTest process

SelectTestDb Algorithm

1. For implementing this, authors created a tool that analyzes DB Schema (yeah, it's magic)

Algorithm 2 selectedTestDB

Input: f_i, TC_i
Output: TestDB, NumTestDB

```
selectedTestDB process
numTestDB ← 0
repeat
    if TCi access DB then
        TestDB ← TestDB ∪ TCi
        NumTestDB++
    end if
until ∄ TCi
end selectedTestDB process
```

Similarity matrix construction Algorithm

1. For implementing this, authors created a tool that analyzes DB Schema (yeah, it's magic)

Algorithm 3 buildMatrixTestDB

Input: TestDB, numTestDB
Output: MatTestDB

buildMatrixTestDB process
  $i \leftarrow 1$
  while $i <$ numTestBD do
    MatTestDB $\leftarrow$ (MatTestDB $\cup$ ($TestDB_i$))
  end while
end buildMatrixTestDB process

Cluster generation and selection Algorithm

1. For implementing this, authors used Weka data mining tool for practical reasons
2. Also used probabilistic algorithm EM (Expectation maximization) which has the advantage of **determining a k number of clusters** based on the information of the test cases of the similarity matrix

Algorithm 4 clusterizationEM

Input: MatTestDB

Output: MatClusterTestDB

clusterizationEM process

Θ Vector desconocido de parámetros

$\Theta^0, \Theta^1, ..., \Theta^T$, T criterio de convergencia

$T \leftarrow 0$

$\Theta^0 \leftarrow 0$

Repetir

$Q(\Theta, \Theta^t) \leftarrow E[\log p(x^g, x^m I\ \Theta)\ I\ x^g, \Theta^t]$

$\Theta^{t+1} \leftarrow \arg\max_\Theta Q(\Theta, \Theta^t)$

$t \leftarrow t+1$

until convergence criterion

end clusterizationEM process

$$\boldsymbol{\tau}^{(t+1)} = \arg\max_{\boldsymbol{\tau}} Q(\theta|\theta^{(t)})$$
$$= \arg\max_{\boldsymbol{\tau}} \left\{ \left[\sum_{i=1}^{n} T_{1,i}^{(t)}\right] \log \tau_1 + \left[\sum_{i=1}^{n} T_{2,i}^{(t)}\right] \log \tau_2 \right\}$$

$$\tau_j^{(t+1)} = \frac{\sum_{i=1}^{n} T_{j,i}^{(t)}}{\sum_{i=1}^{n} (T_{1,i}^{(t)} + T_{2,i}^{(t)})} = \frac{1}{n} \sum_{i=1}^{n} T_{j,i}^{(t)}$$

$$(\boldsymbol{\mu}_1^{(t+1)}, \Sigma_1^{(t+1)}) = \arg\max_{\boldsymbol{\mu}_1, \Sigma_1} Q(\theta|\theta^{(t)})$$
$$= \arg\max_{\boldsymbol{\mu}_1, \Sigma_1} \sum_{i=1}^{n} T_{1,i}^{(t)} \left\{ -\tfrac{1}{2} \log |\Sigma_1| - \tfrac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_1)^\top \Sigma_1^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_1) \right\}$$

$$, \boldsymbol{\mu}_2^{(t+1)} = \frac{\sum_{i=1}^{n} T_{2,i}^{(t)} \mathbf{x}_i}{\sum_{i=1}^{n} T_{2,i}^{(t)}} , \Sigma_1^{(t+1)} = \frac{\sum_{i=1}^{n} T_{1,i}^{(t)} (\mathbf{x}_i - \boldsymbol{\mu}_1^{(t+1)})(\mathbf{x}_i - \boldsymbol{\mu}_1^{(t+1)})^\top}{\sum_{i=1}^{n} T_{1,i}^{(t)}} , \boldsymbol{\mu}_1^{(t+1)} = \frac{\sum_{i=1}^{n} T_{1,i}^{(t)} \mathbf{x}_i}{\sum_{i=1}^{n} T_{1,i}^{(t)}}$$

$$\Sigma_2^{(t+1)} = \frac{\sum_{i=1}^{n} T_{2,i}^{(t)} (\mathbf{x}_i - \boldsymbol{\mu}_2^{(t+1)})(\mathbf{x}_i - \boldsymbol{\mu}_2^{(t+1)})^\top}{\sum_{i=1}^{n} T_{2,i}^{(t)}}$$

$$E_{Z|\theta^{(t)},\mathbf{x}}[\log L(\theta^{(t)}; \mathbf{x}, \mathbf{Z})] \leq E_{Z|\theta^{(t-1)},\mathbf{x}}[\log L(\theta^{(t-1)}; \mathbf{x}, \mathbf{Z})] + \epsilon$$

Cluster generation and selection Algorithm

1. For implementing this, authors used Weka data mining tool for practical reasons
2. Also used probabilistic algorithm EM (Expectation maximization) which has the advantage of **determining a k number of clusters** based on the information of the test cases of the similarity matrix

Algorithm 4 clusterizationEM

Input: MatTestDB
Output: MatClusterTestDB

```
clusterizationEM process
    Θ Vector desconocido de parámetros
    Θ^0, Θ^1, ...., Θ^T, T criterio de convergencia
    T←0
    Θ^0 ←0
    Repetir
        Q(Θ, Θ^t) ← E[log p(x^g, x^m I Θ) I x^g, Θ^t]
        Θ^{t+1} ← arg max_Θ Q(Θ, Θ^t)
        t ← t+1
    until convergence criterion
end clusterizationEM process
```

Thank you!