Using Dynamic Invariant Detection to Support the Testing and Analysis of Database Applications

Gregory M. Kapfhammer†

Department of Computer Science
Allegheny College
http://www.cs.allegheny.edu/~gkapfham/

University of Ulm – January 17, 2012

† Joint with Mary Jean Harrold and Jake Cobb (GA Tech), James A. Jones (UC Irvine), Jonathan Miller Kauffman (Allegheny)

ALLEGHENY COLLEGE

Accessing the Presentation

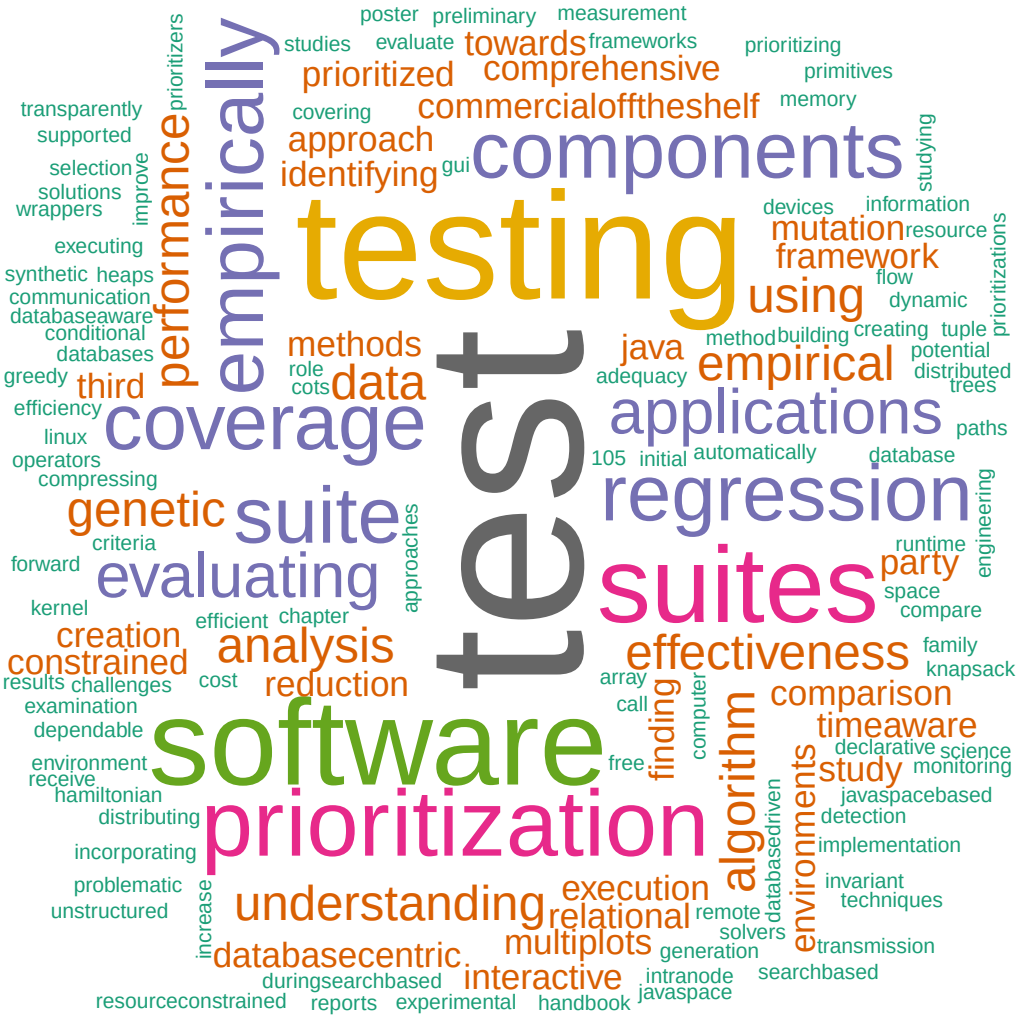
Scan this QR Code with your smartphone!

... or, visit this Web site:

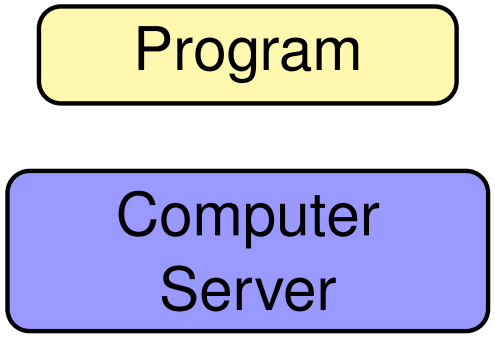
`http://www.cs.allegheny.edu/~gkapfham/ulm2012.pdf`

... or, ask me for a USB drive!

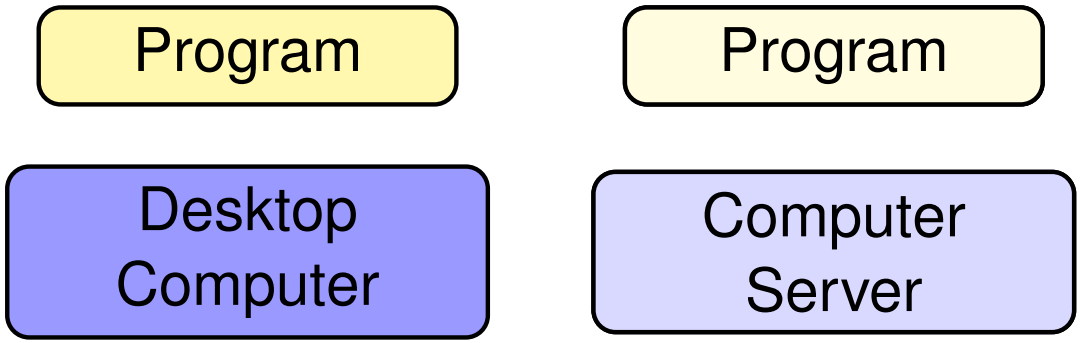
Presenter Introduction: Gregory M. Kapfhammer



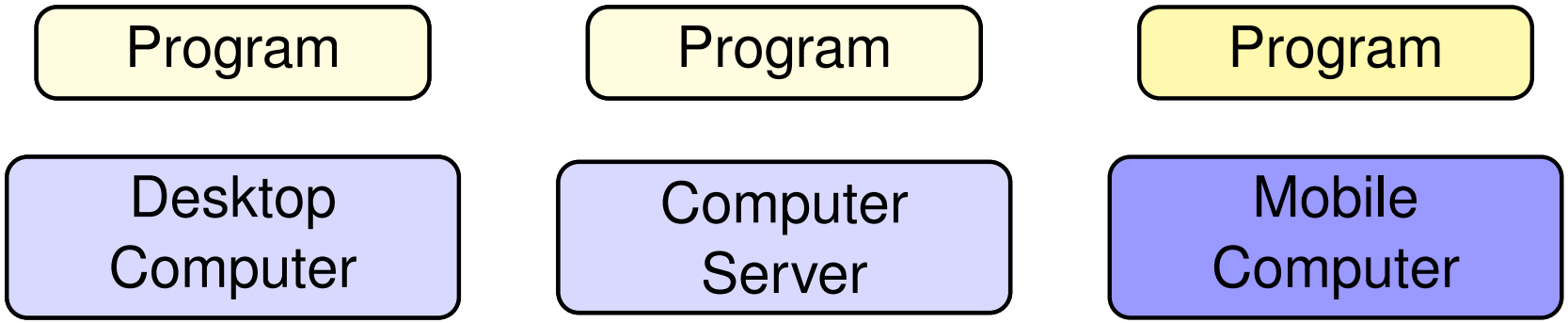
Software and Data are Everywhere



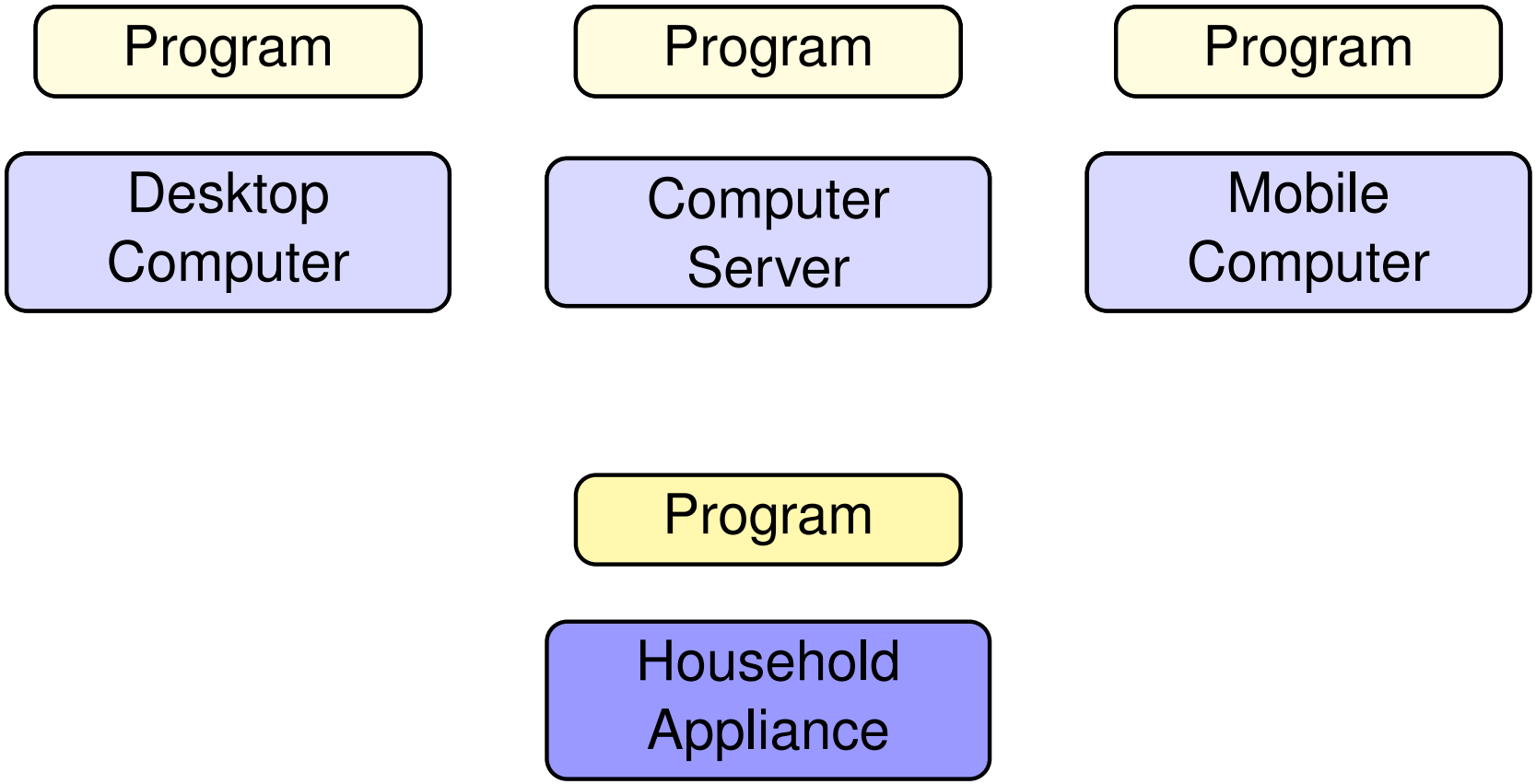
Software and Data are Everywhere



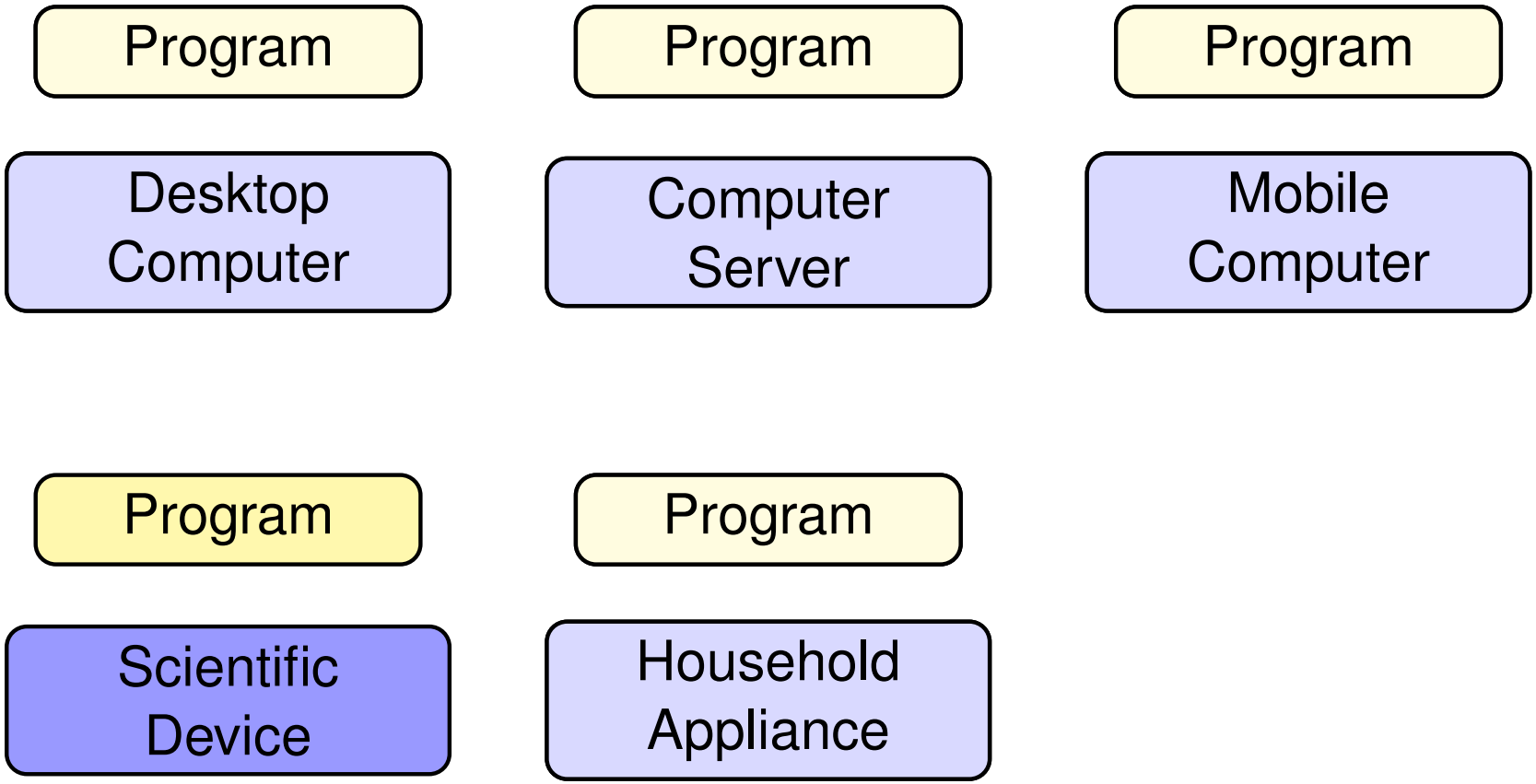
Software and Data are Everywhere



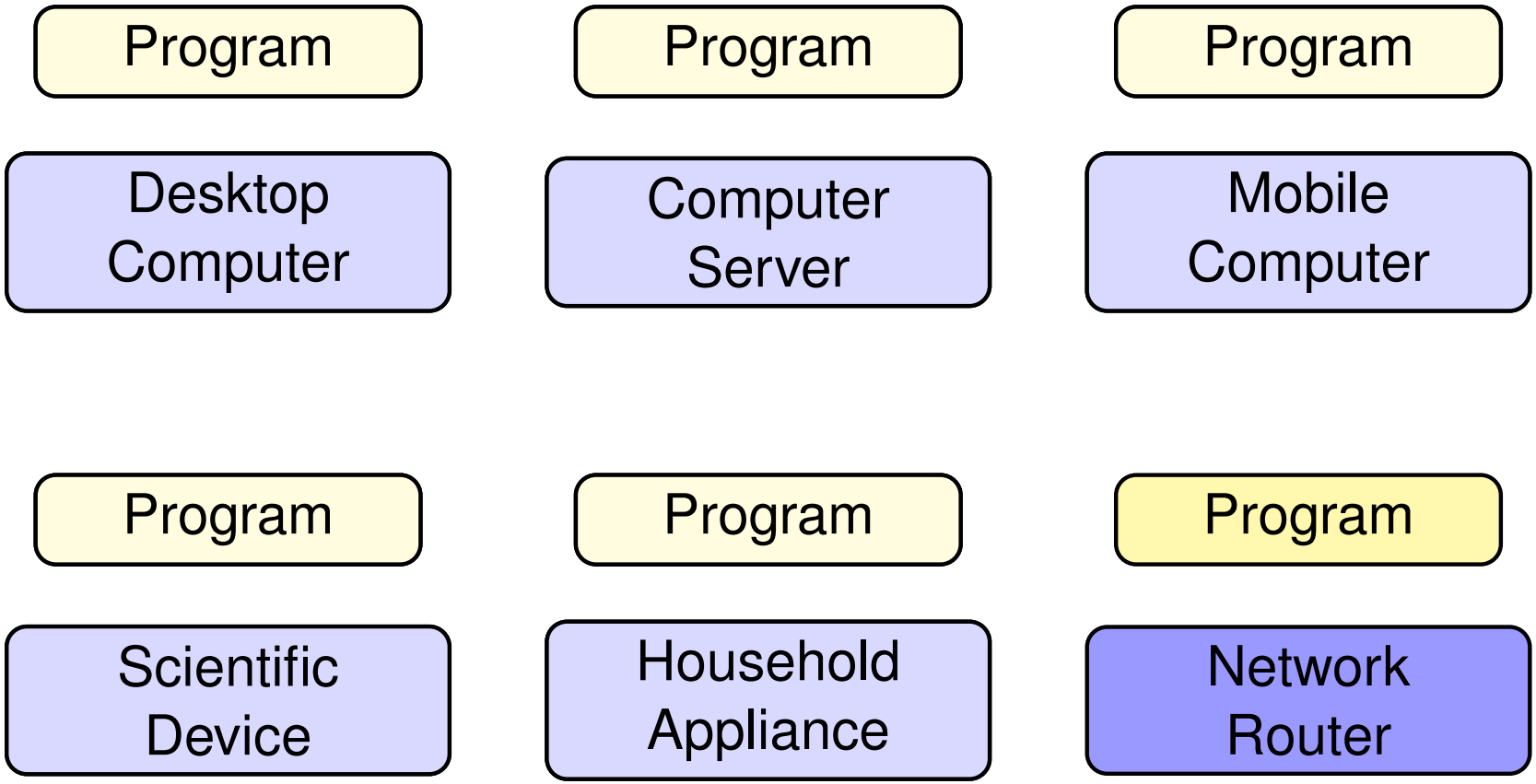
Software and Data are Everywhere



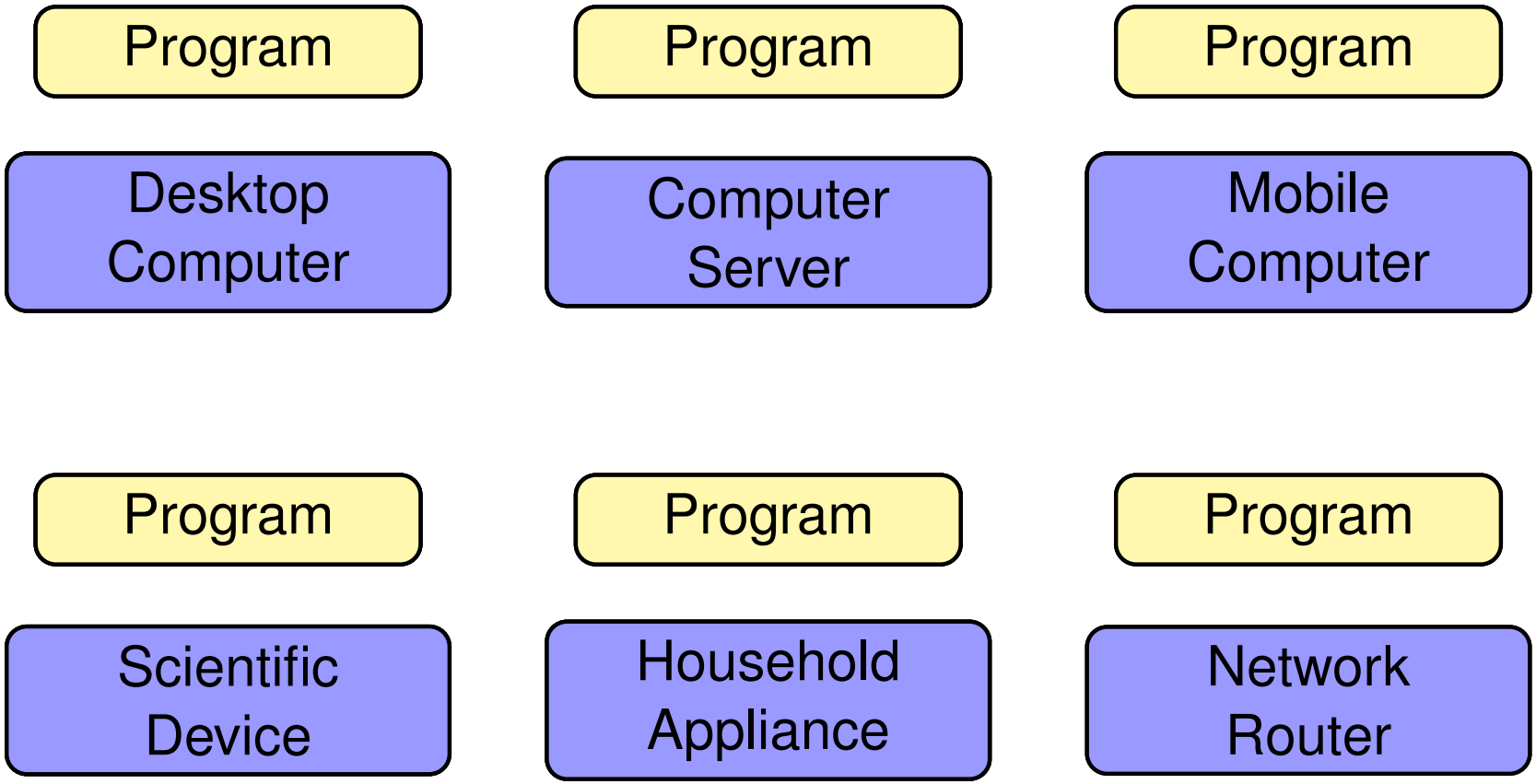
Software and Data are Everywhere



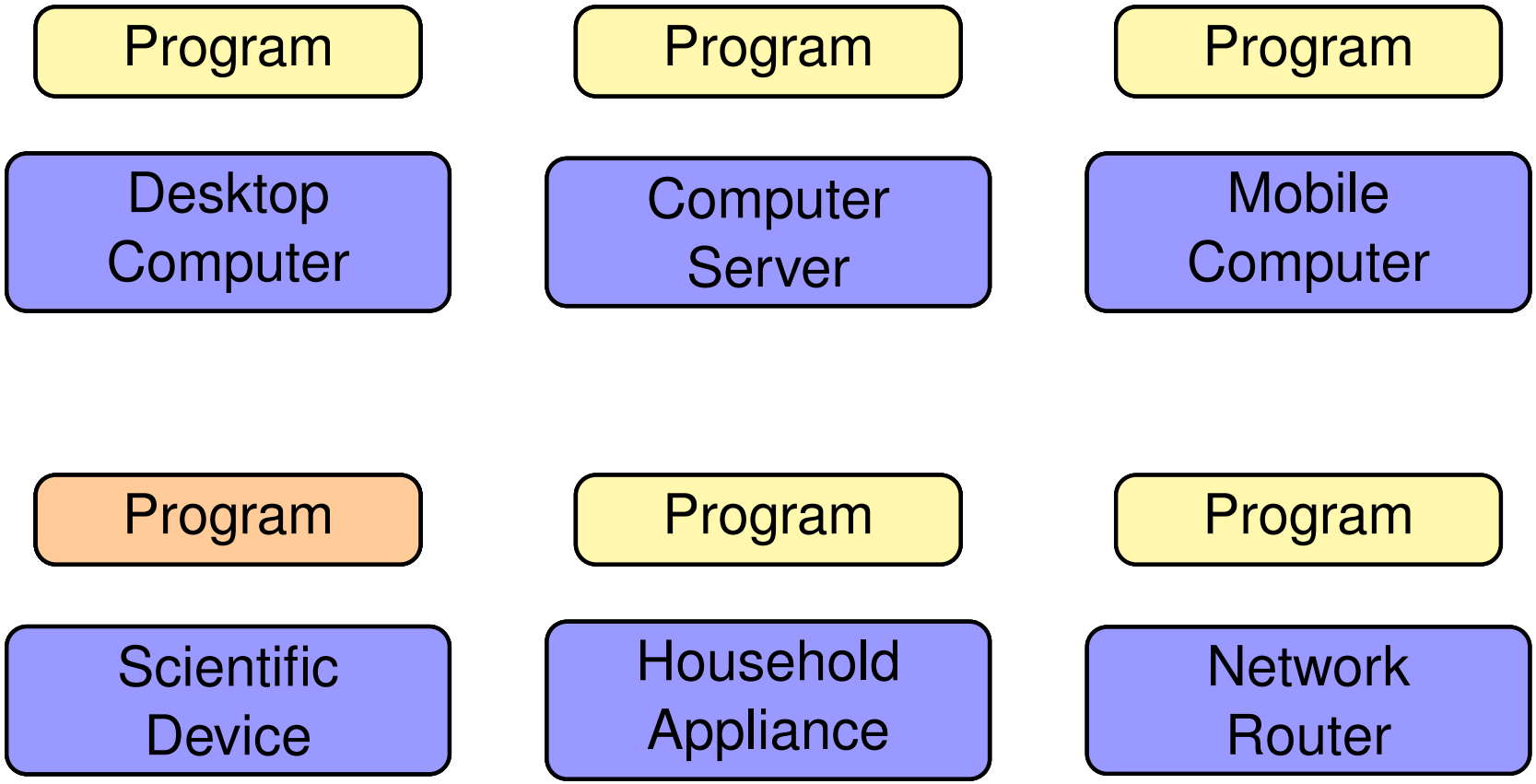
Software and Data are Everywhere



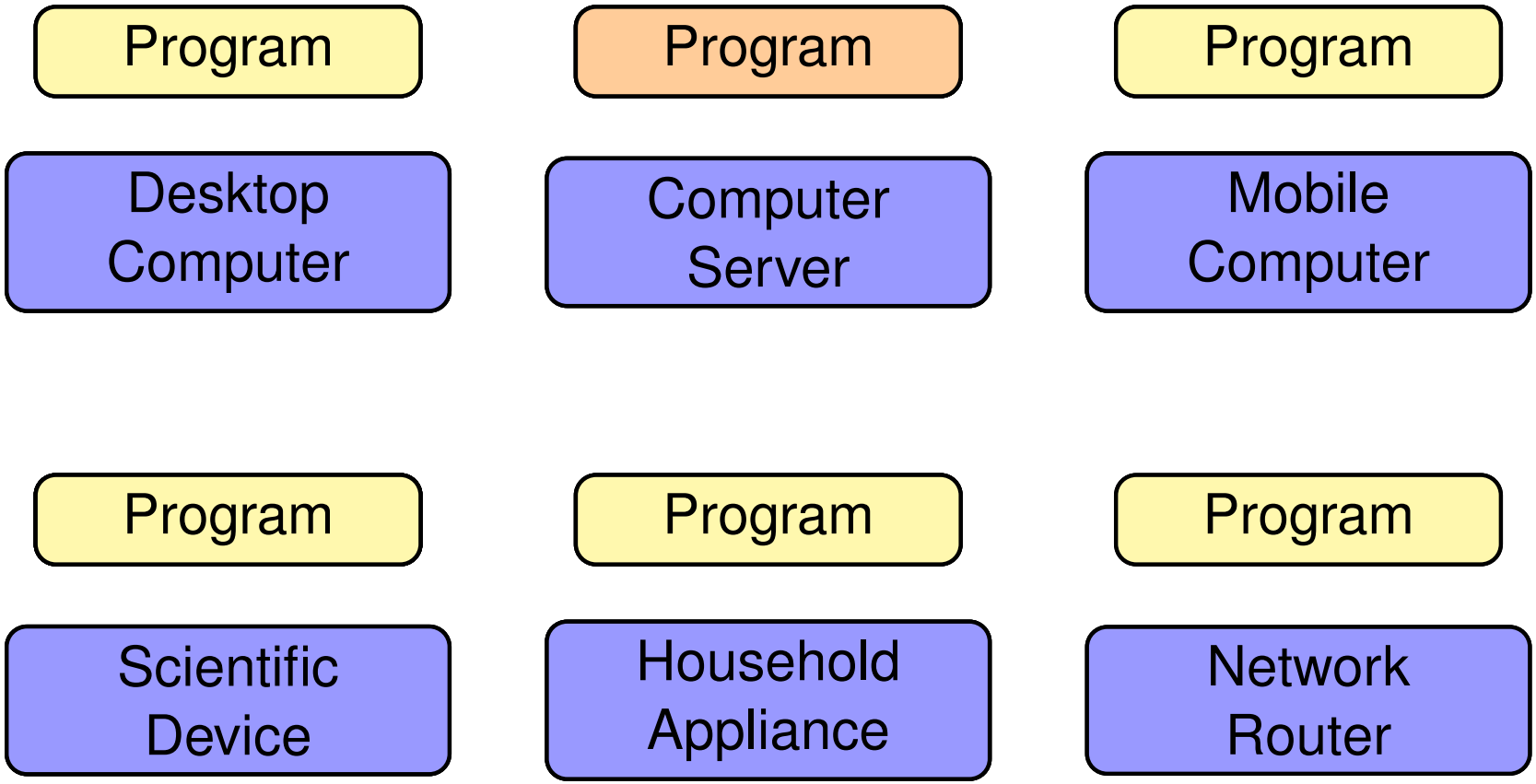
Software and Data are Everywhere



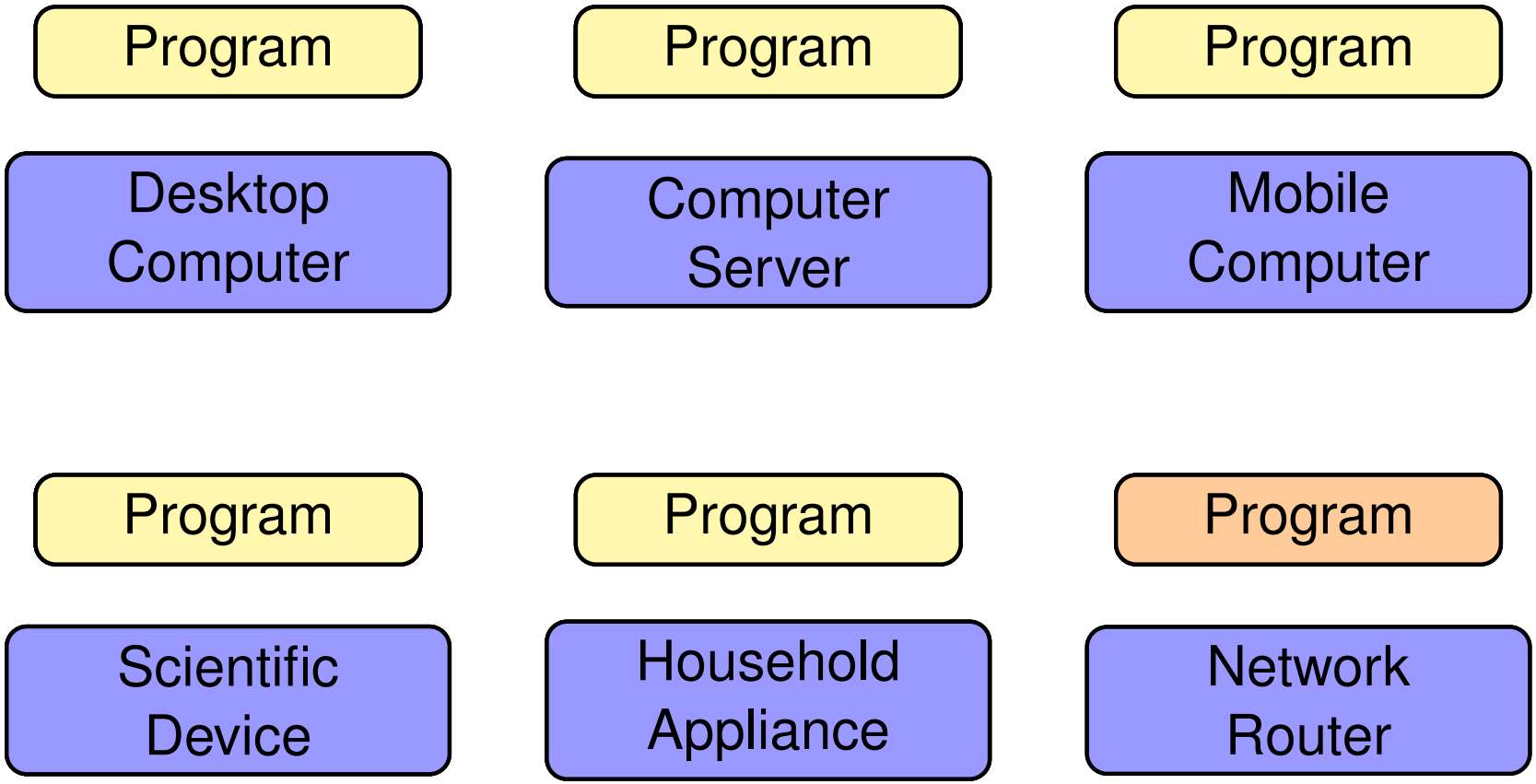
Software and Data are Everywhere



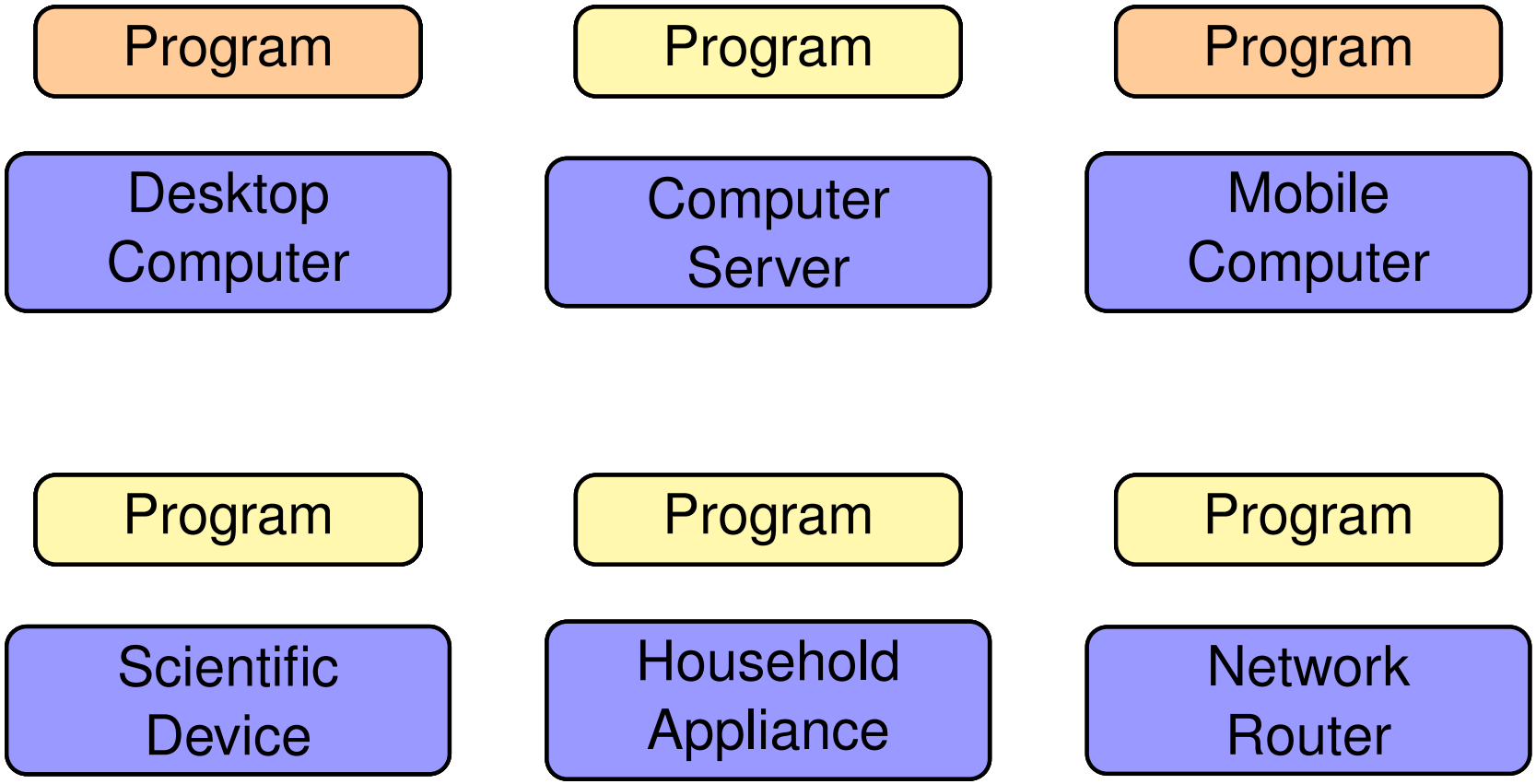
Software and Data are Everywhere



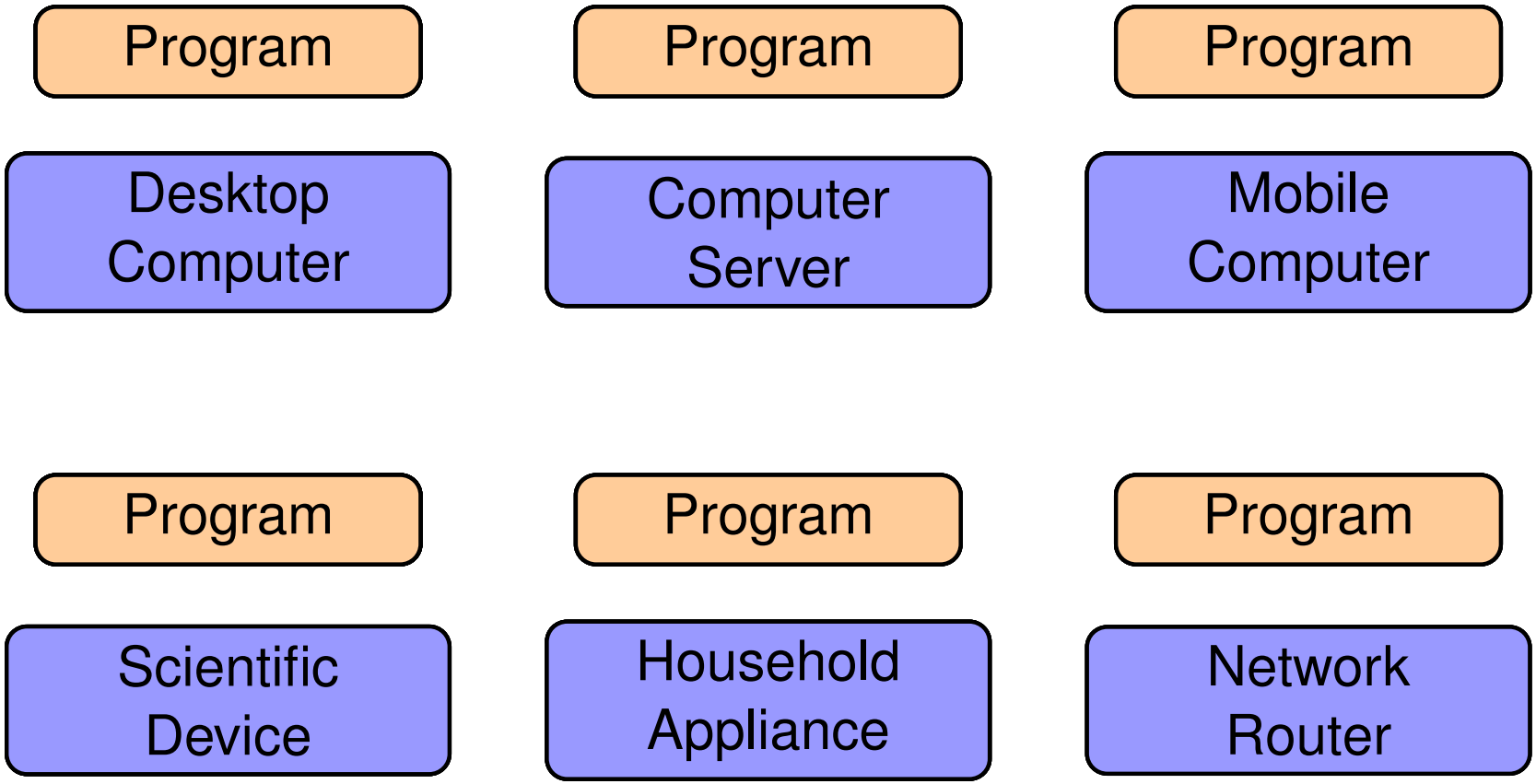
Software and Data are Everywhere



Software and Data are Everywhere



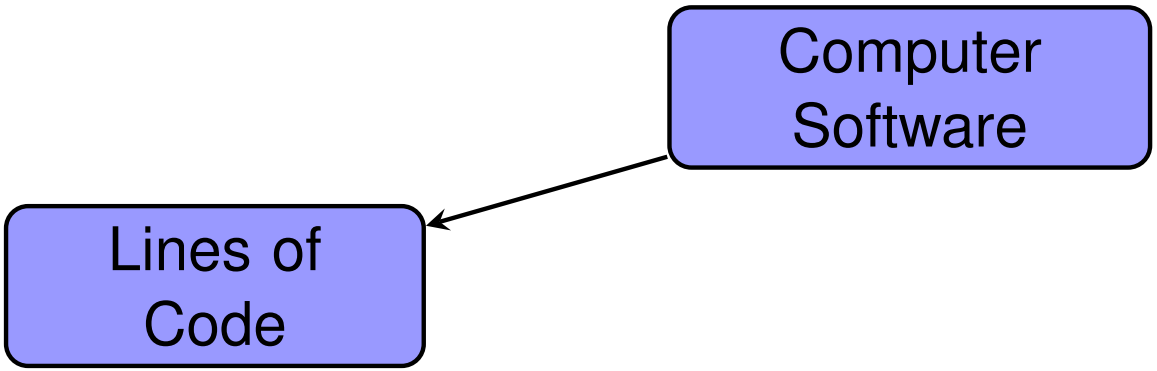
Software and Data are Everywhere



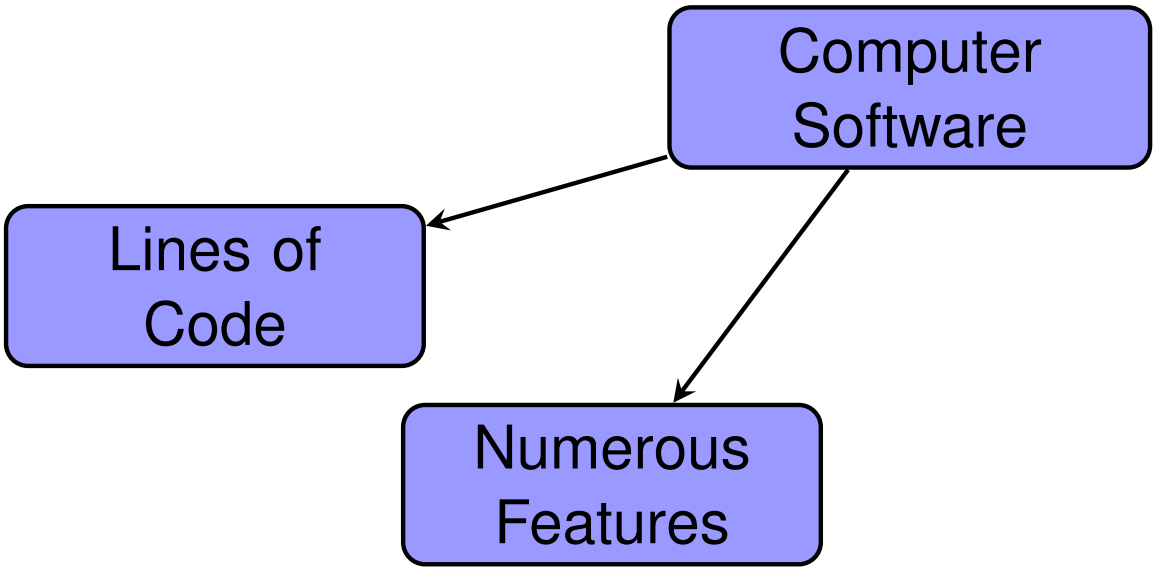
Software Complexity and Data Enormity

Computer Software

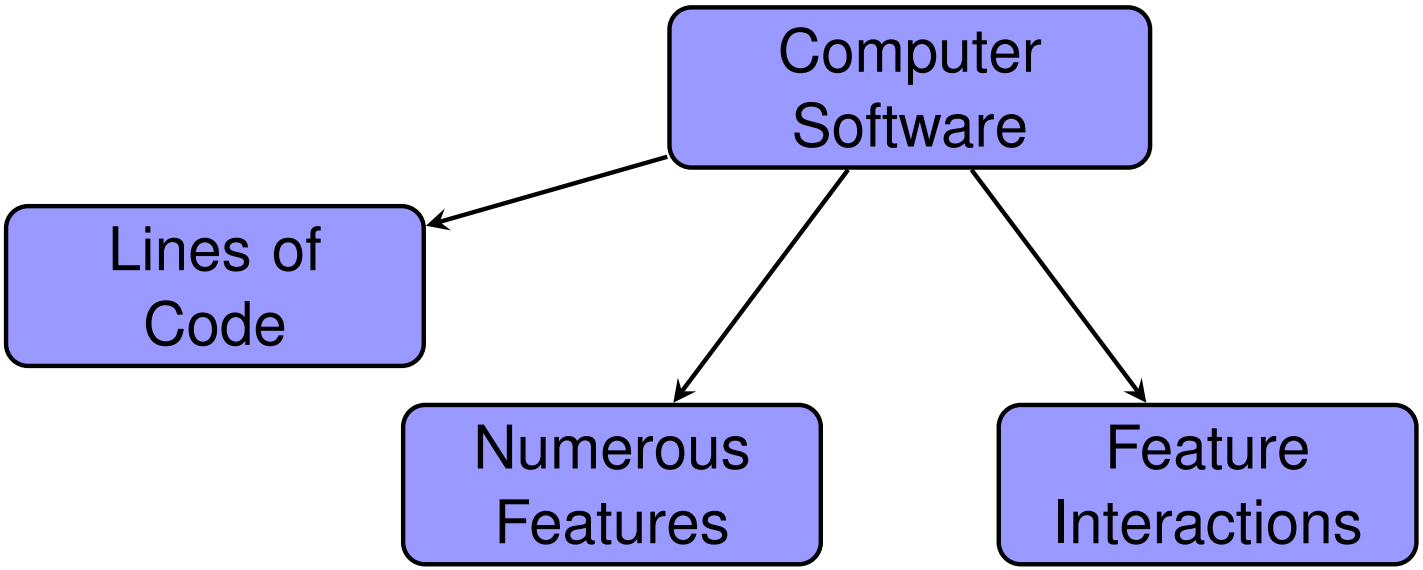
Software Complexity and Data Enormity



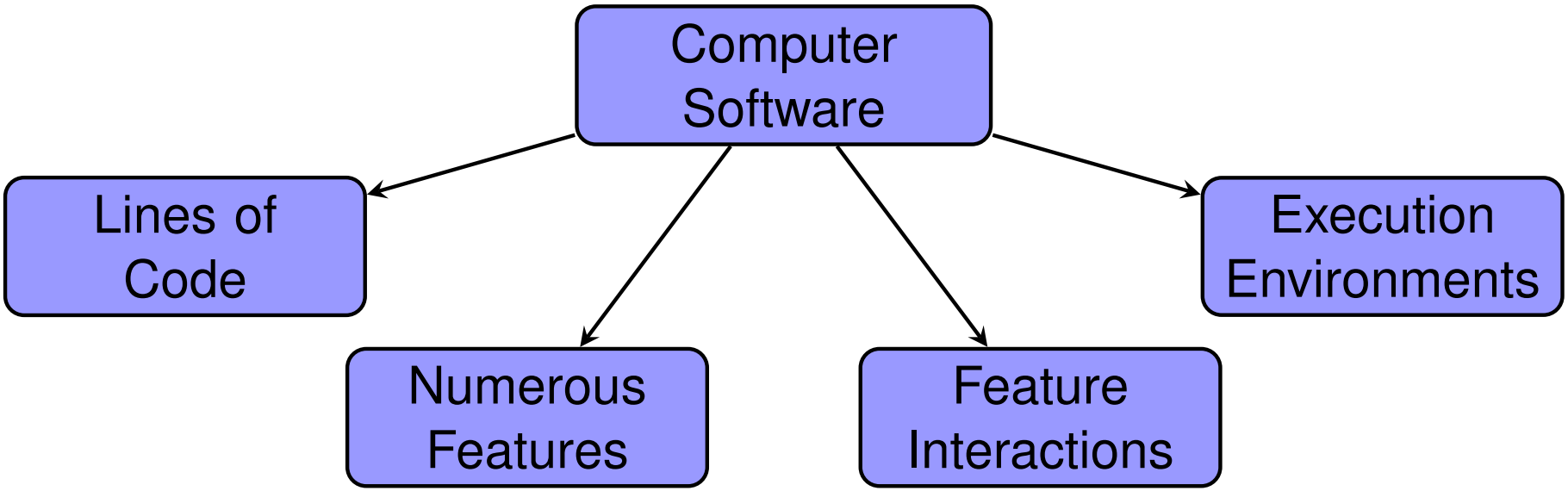
Software Complexity and Data Enormity



Software Complexity and Data Enormity

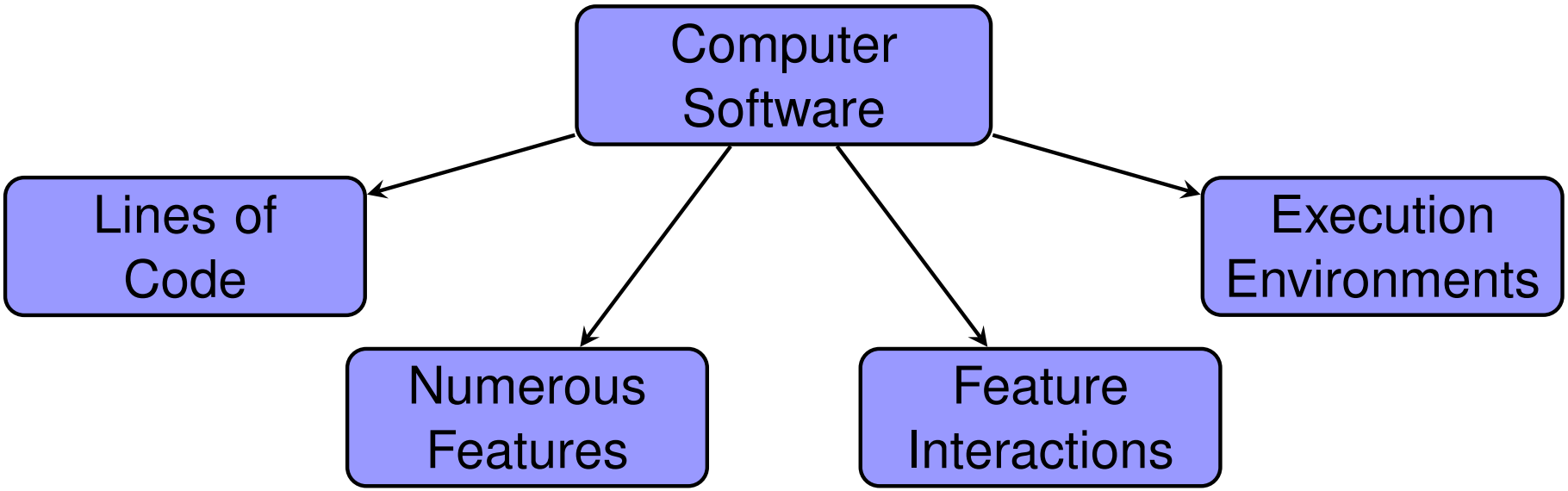


Software Complexity and Data Enormity



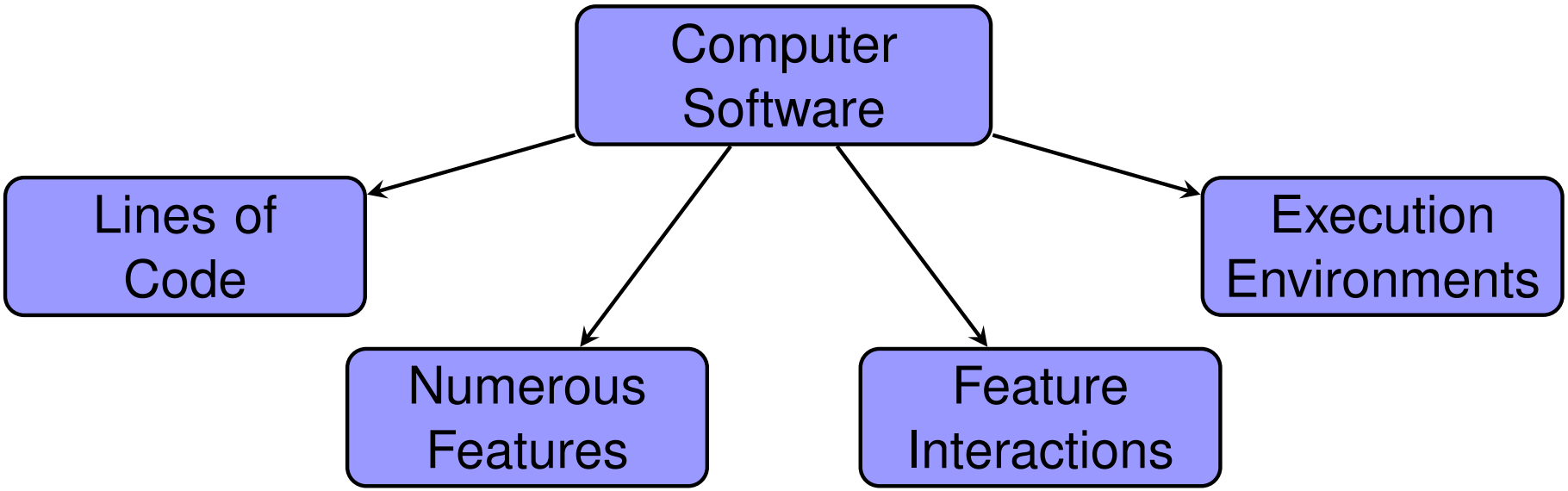
Software Complexity and Data Enormity

> Software entities are more complex for their size than perhaps any other human construct - Frederick P. Brooks, Jr.

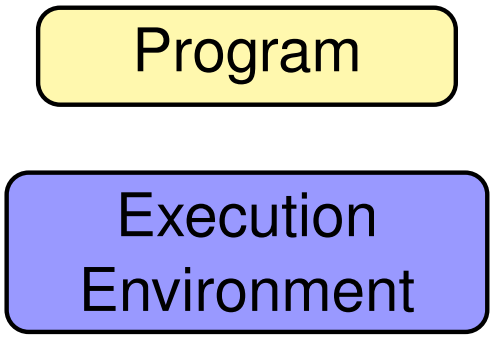


Software Complexity and Data Enormity

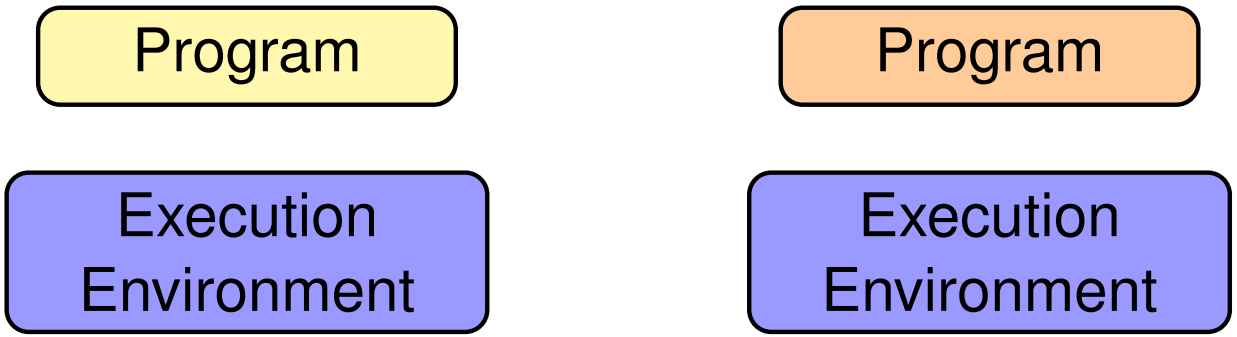
Prediction: in 2011, 1.8 zettabytes (i.e., 1.8 trillion gigabytes) of data will be created - IDC Digital Universe Study



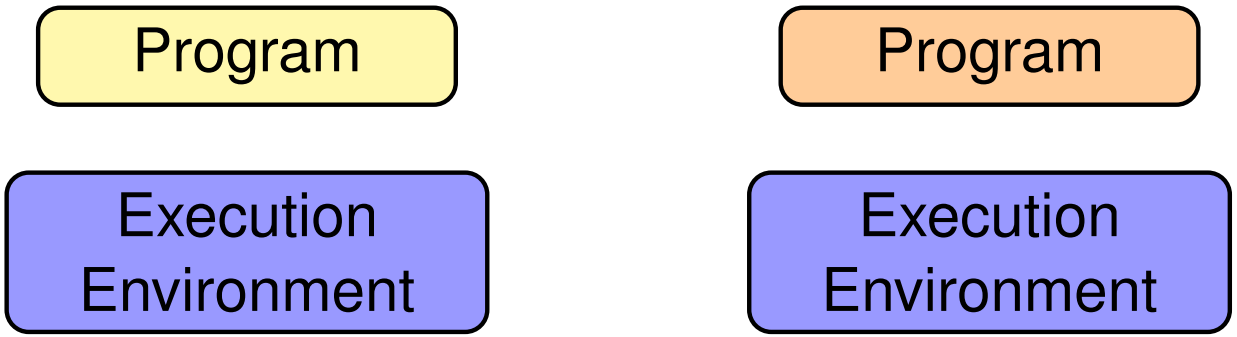
Software and Data are Evolving



Software and Data are Evolving

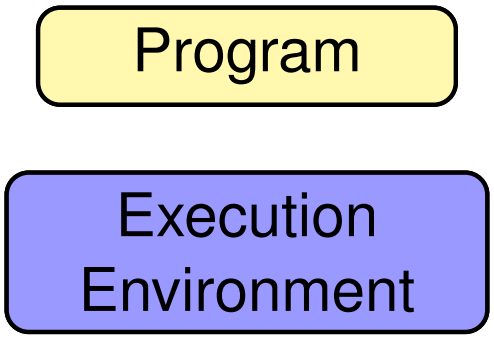


Software and Data are Evolving

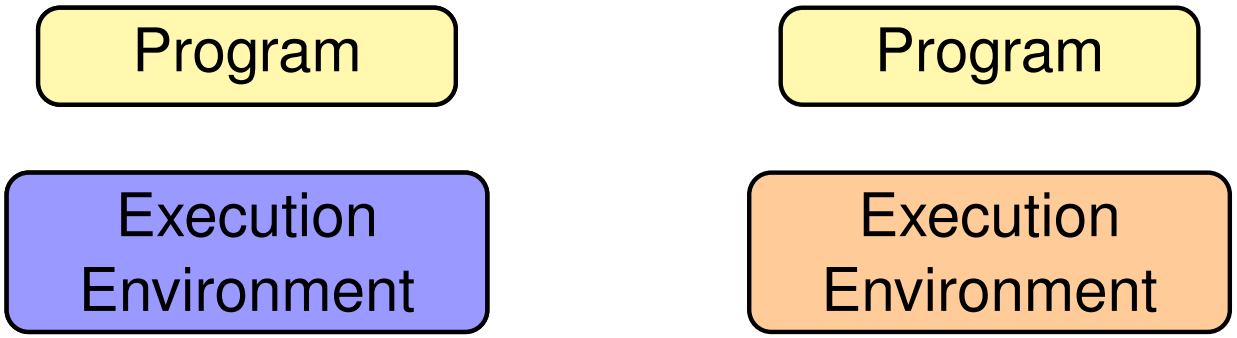


Program Changed because of the addition of a new feature or the correction of a defect

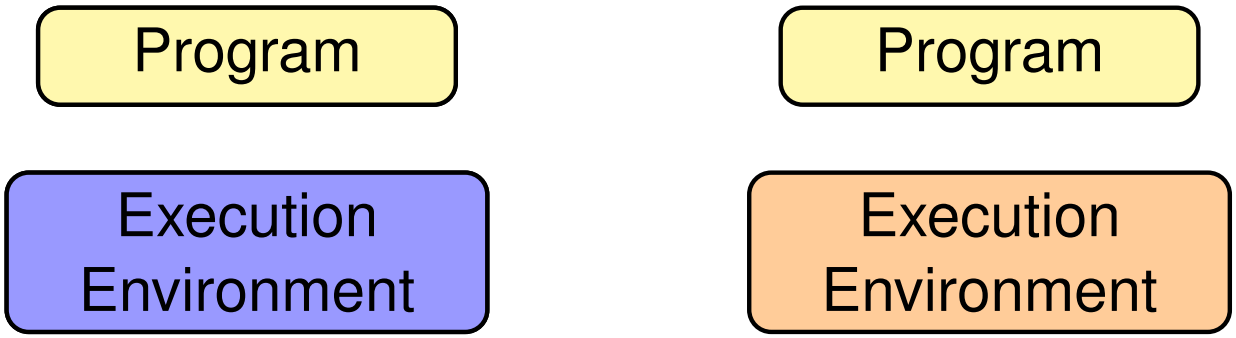
Software and Data are Evolving



Software and Data are Evolving



Software and Data are Evolving



Execution Environment Changed due to modification of a kernel, device driver, or relational database

An Interesting Defect Report

Database
Server Crashes

An Interesting Defect Report

Database
Server Crashes

When you run a complex query against Microsoft SQL Server 2000, the SQL Server scheduler may stop responding. Additionally, you receive an error message that resembles the following: **Date Time server Error: 17883 Severity: 1, State: 0 Date Time server Process 52:0 (94c) ...**

An Interesting Defect Report

Input-Dependent Defect

An Interesting Defect Report

Input-Dependent Defect

This problem occurs when one or more of the following conditions are true: The query contains a `UNION` clause or a `UNIONALL` clause that affects many columns. The query contains several `JOIN` statements. The query has a large estimated cost. **BUG 473858 (SQL Server 8.0)**

Real-World Defective Database Application

The Risks Digest, Volume 22, Issue 64, 2003

Jeppesen reports airspace boundary problems

About 350 airspace boundaries contained in Jeppesen NavData are incorrect, the FAA has warned. The error occurred at Jeppesen after a software upgrade when information was pulled from a database containing 20,000 airspace boundaries worldwide for the March NavData update, which takes effect March 20.

Real-World Defective Database Application

The Risks Digest, Volume 22, Issue 64, 2003
Jeppesen reports airspace boundary problems

About 350 airspace boundaries contained in Jeppesen NavData are incorrect, the FAA has warned. The error occurred at Jeppesen after a software upgrade when information was pulled from a database containing 20,000 airspace boundaries worldwide for the March NavData update, which takes effect March 20.

Practically all use of databases occurs from within application programs [Silberschatz et al., 2006, pg. 311]

Structured Query Language

The *structured query language* (SQL) is an established standard and a query and manipulation language for *relational database management systems* (RDBMS)

Structured Query Language

The *structured query language* (SQL) is an established standard and a query and manipulation language for *relational database management systems* (RDBMS)

A *schema* is a collection of table definitions:

```
CREATE TABLE person (
id INT,
name VARCHAR(100) NOT NULL,
age INT(3),
PRIMARY KEY (id)
)
```

Structured Query Language

The *structured query language* (SQL) is an established standard and a query and manipulation language for *relational database management systems* (RDBMS)

The *data manipulation language* supports several operations:

```
SELECT name FROM person WHERE age >= 30 AND age <= 40
```

Structured Query Language

The *structured query language* (SQL) is an established standard and a query and manipulation language for *relational database management systems* (RDBMS)

The *data manipulation language* supports several operations:

```
UPDATE person SET name = Jan WHERE id = 2
```

Structured Query Language

The *structured query language* (SQL) is an established standard and a query and manipulation language for *relational database management systems* (RDBMS)

The *data manipulation language* supports several operations:

```
INSERT INTO person (id, name, age) VALUES
                   (1, John, 38)
```

Structured Query Language

The *structured query language* (SQL) is an established standard and a query and manipulation language for *relational database management systems* (RDBMS)

The *data manipulation language* supports several operations:

```
DELETE FROM person WHERE id = 2
```

Relational Database Tables

id	name	age
1	Chalker Conrad	12
2	Abby Clulow	14
3	David Rogan	18
4	Stacie Reckling	32
5	Megan Hartnup	29

Relational Database Tables

id	name	age
1	Chalker Conrad	12
2	Abby Clulow	14
3	David Rogan	18
4	Stacie Reckling	32
5	Megan Hartnup	29

Relational Database Tables

id	name	age
1	Chalker Conrad	12
2	Abby Clulow	14
3	David Rogan	18
4	Stacie Reckling	32
5	Megan Hartnup	29

Relational Database Tables

id	name	age
1	Chalker Conrad	12
2	Abby Clulow	14
3	David Rogan	18
4	Stacie Reckling	32
5	Megan Hartnup	29

Relational Database Tables

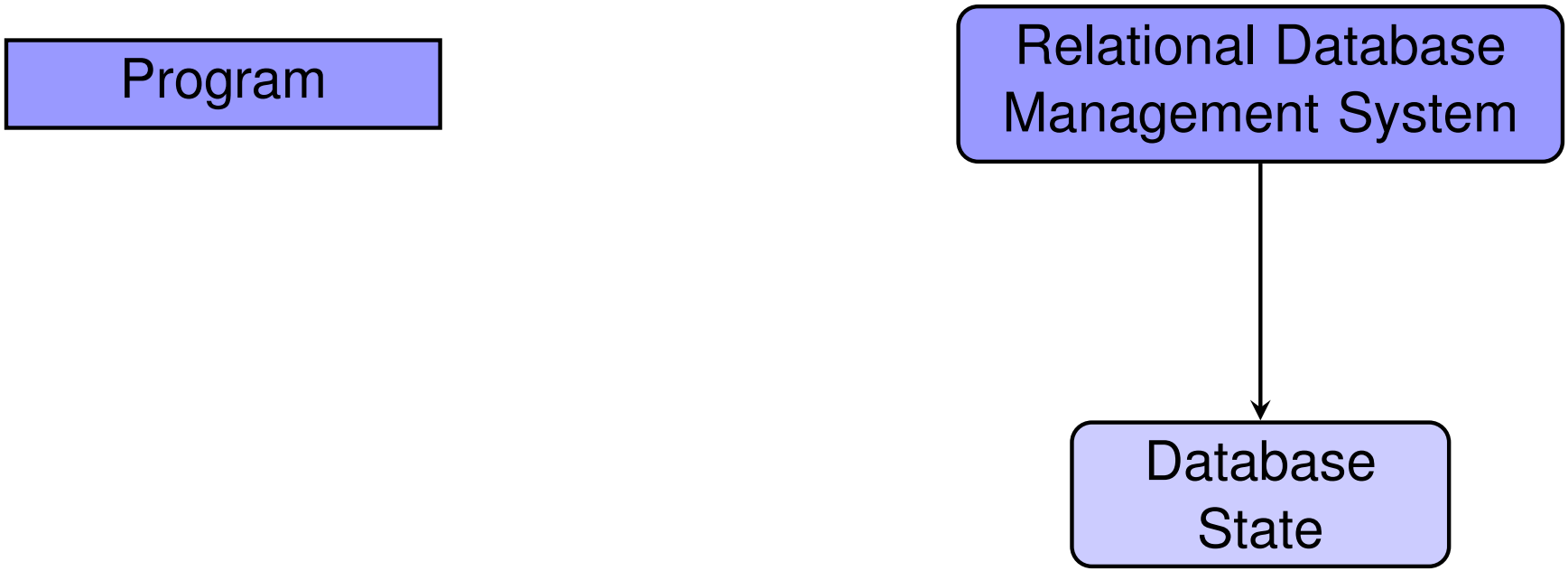
id	name	age
1	Chalker Conrad	12
2	Abby Clulow	14
3	David Rogan	18
4	Stacie Reckling	32
5	Megan Hartnup	29

Relational Database Tables

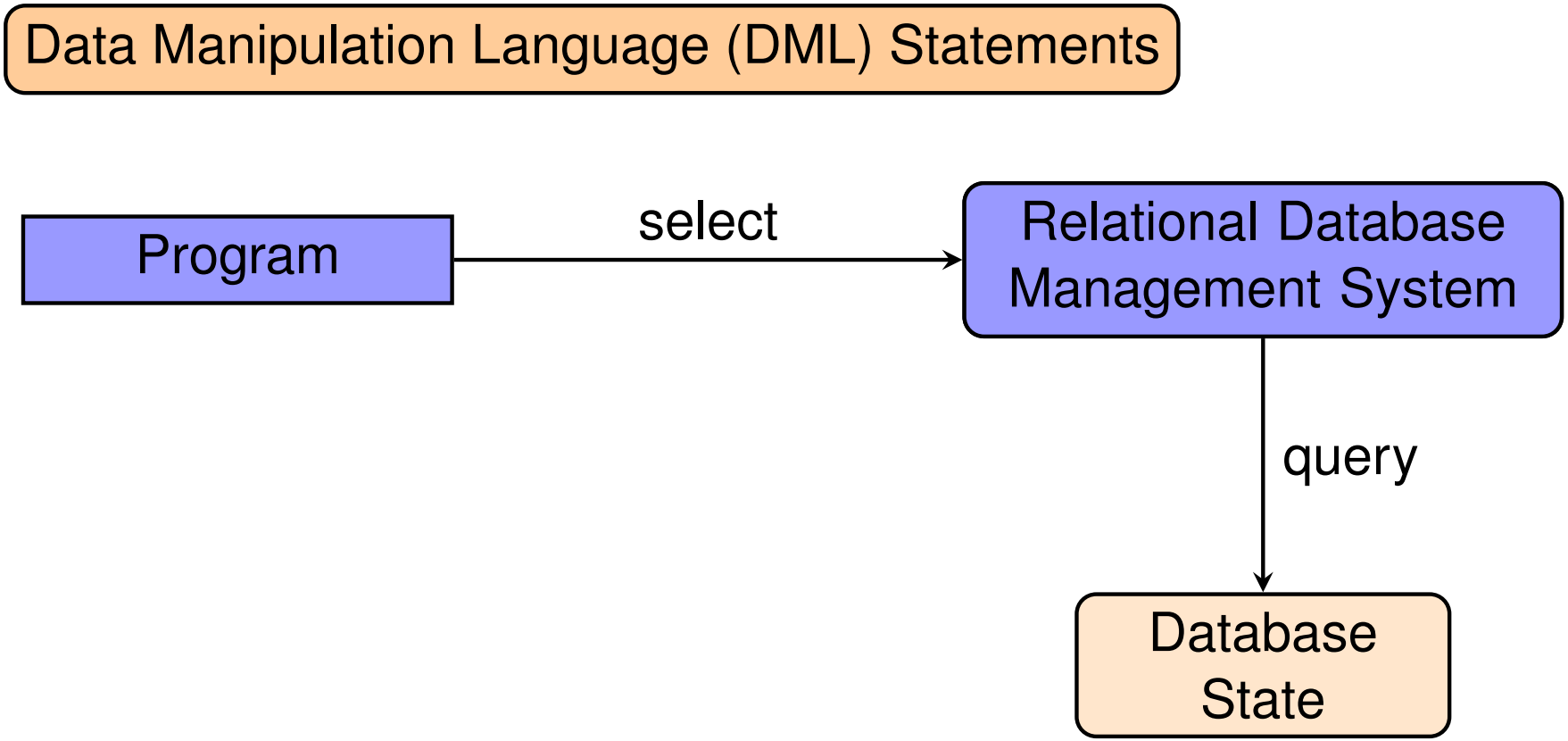
id	name	age
1	Chalker Conrad	12
2	Abby Clulow	14
3	David Rogan	18
4	Stacie Reckling	32
5	Megan Hartnup	29

Database Applications

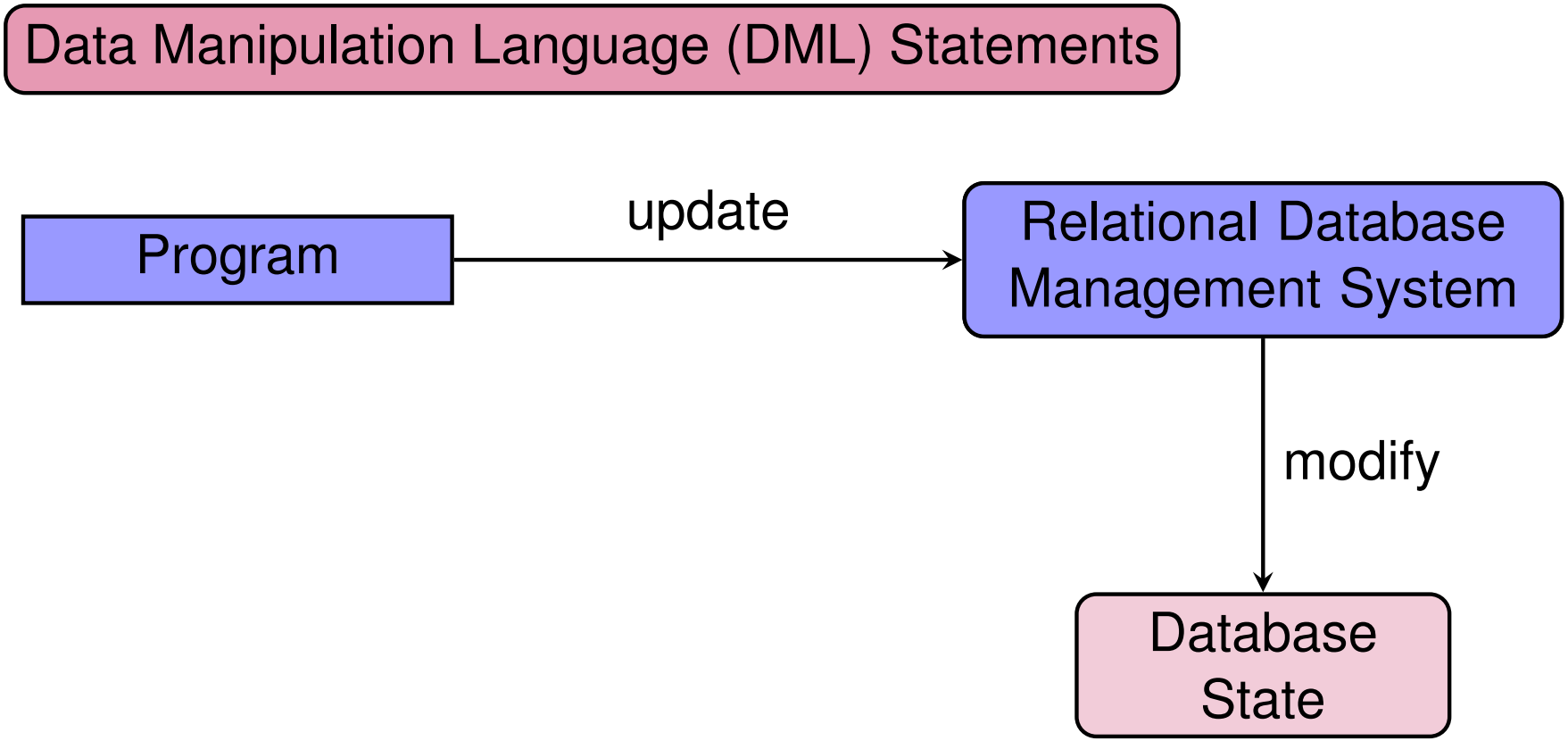
Database Applications



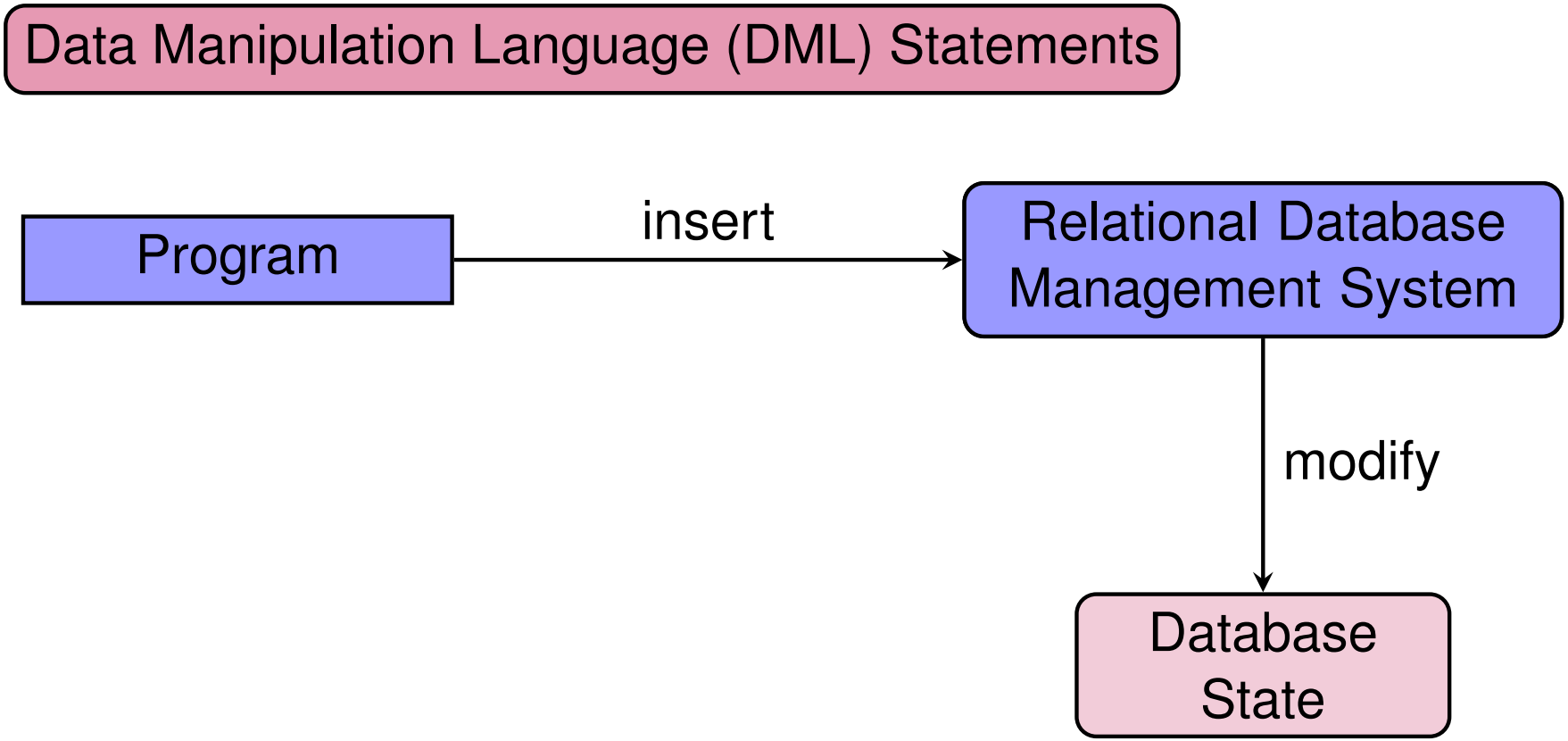
Database Applications



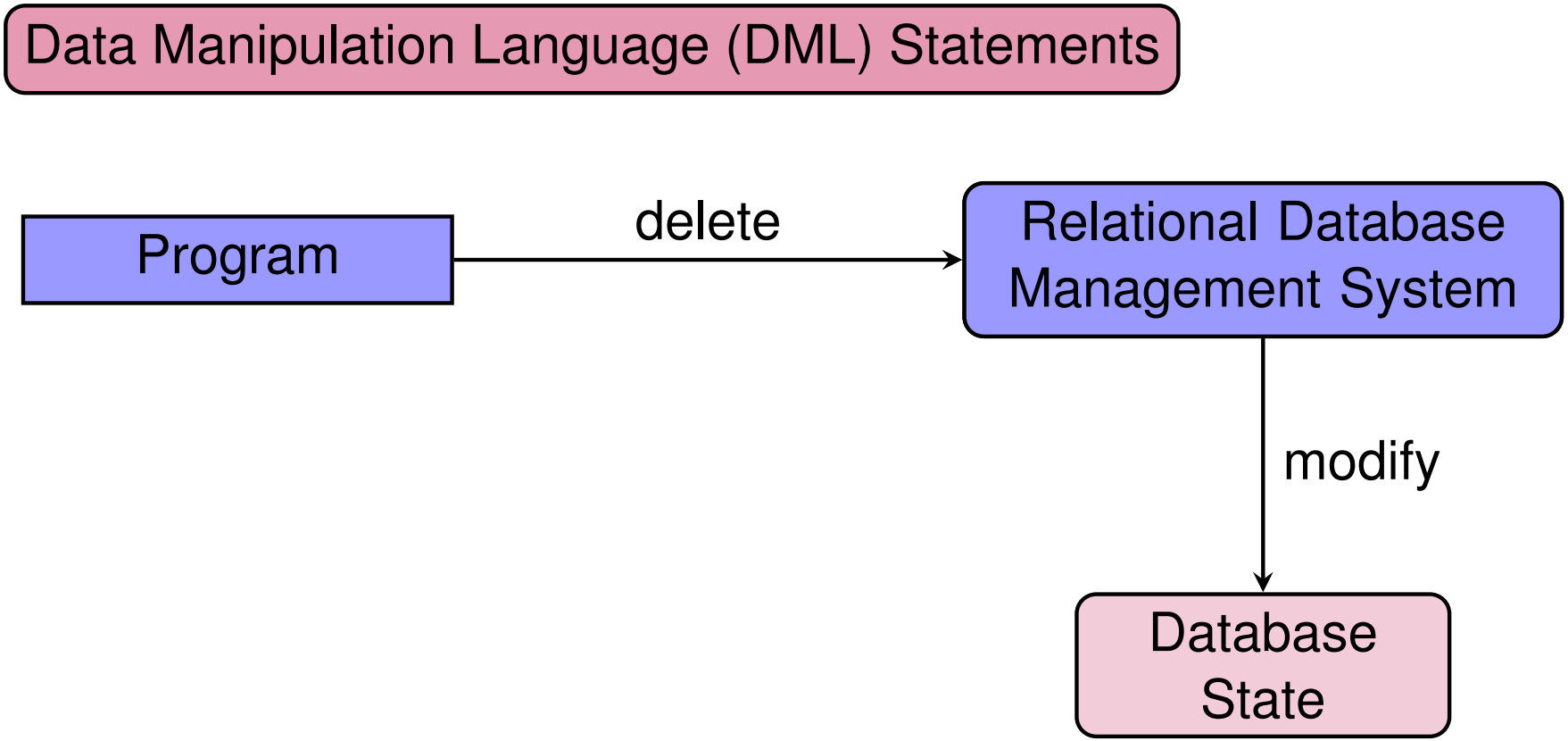
Database Applications



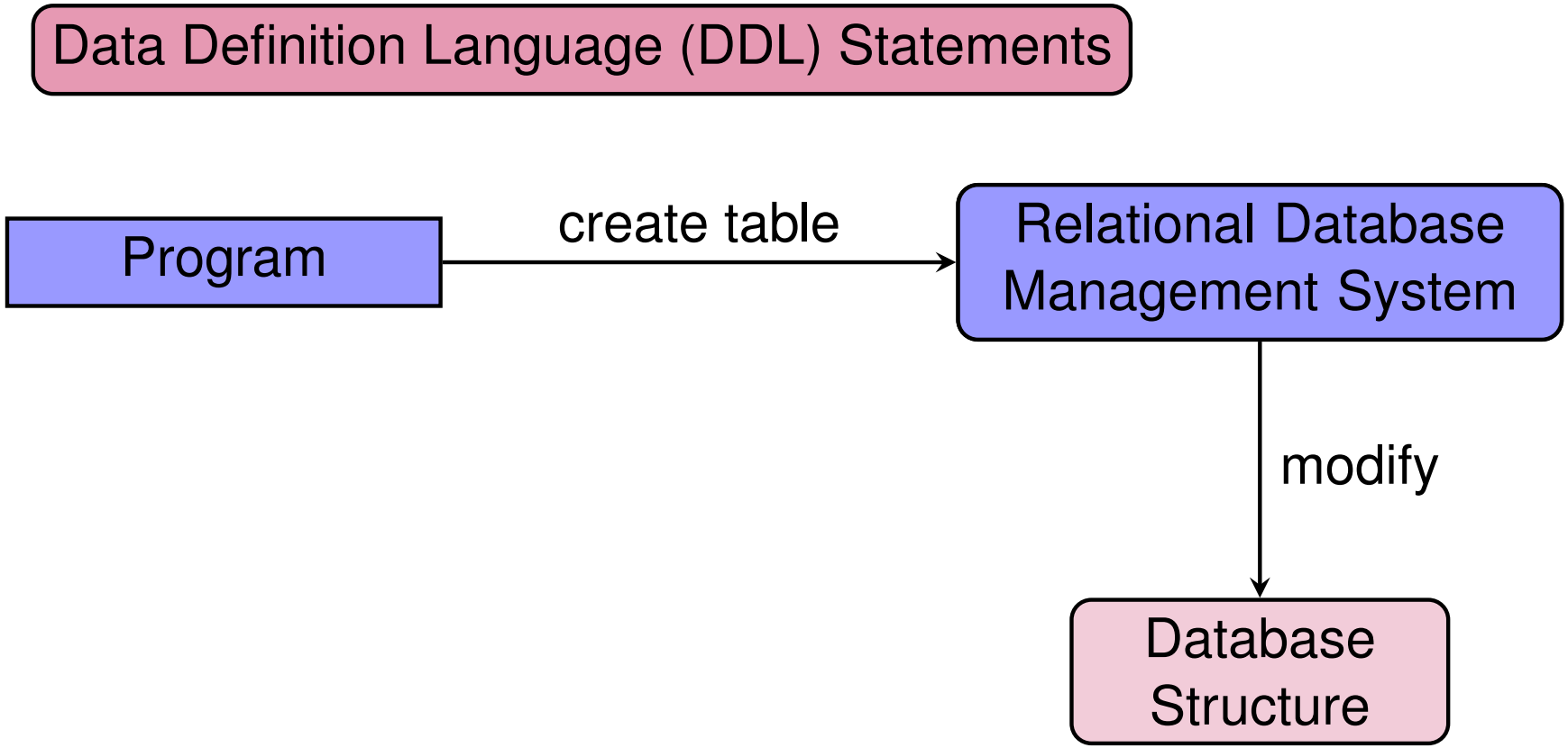
Database Applications



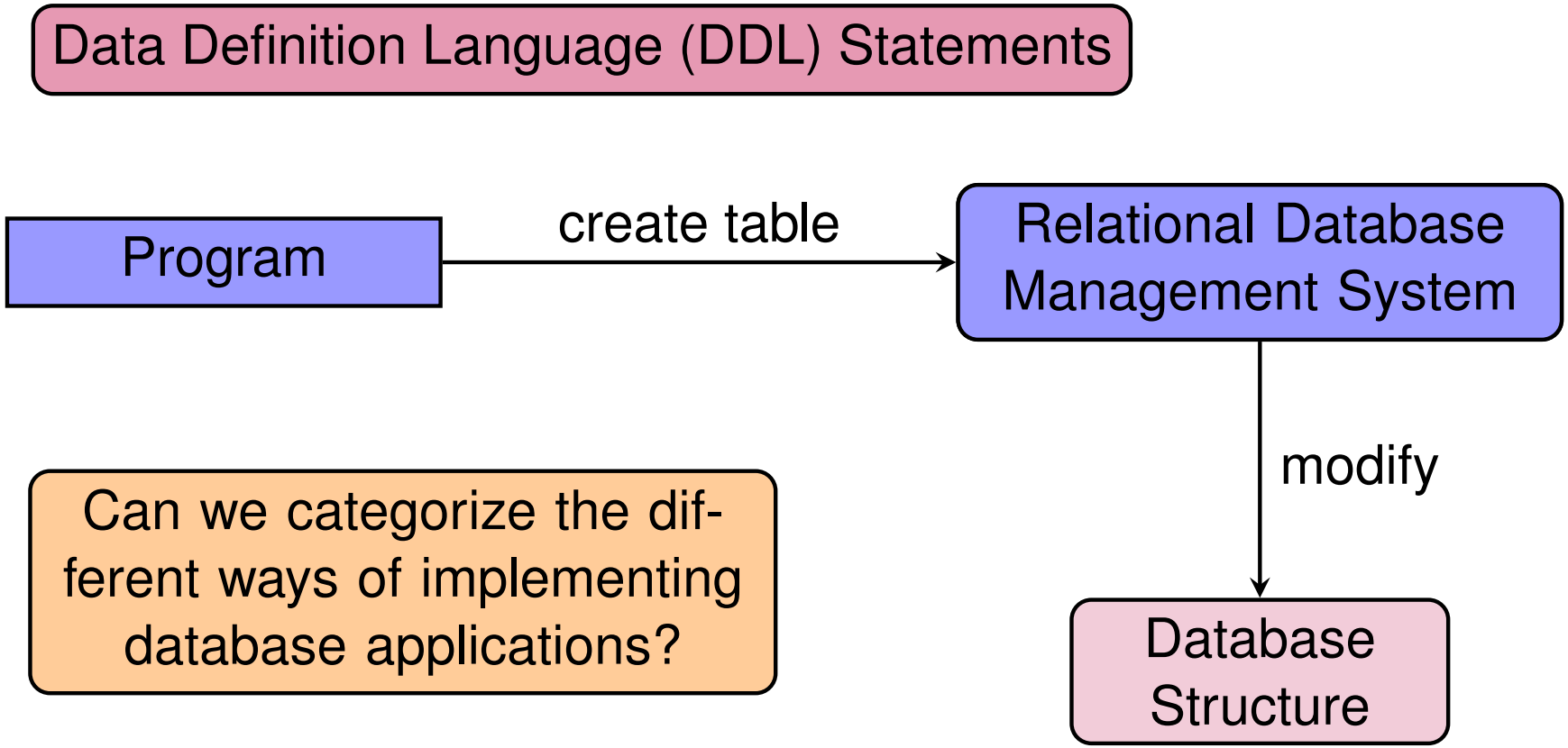
Database Applications



Database Applications

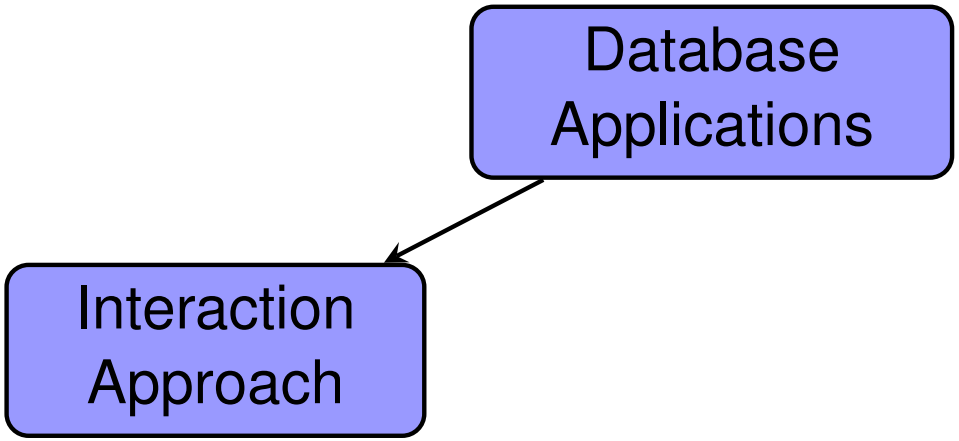


Database Applications

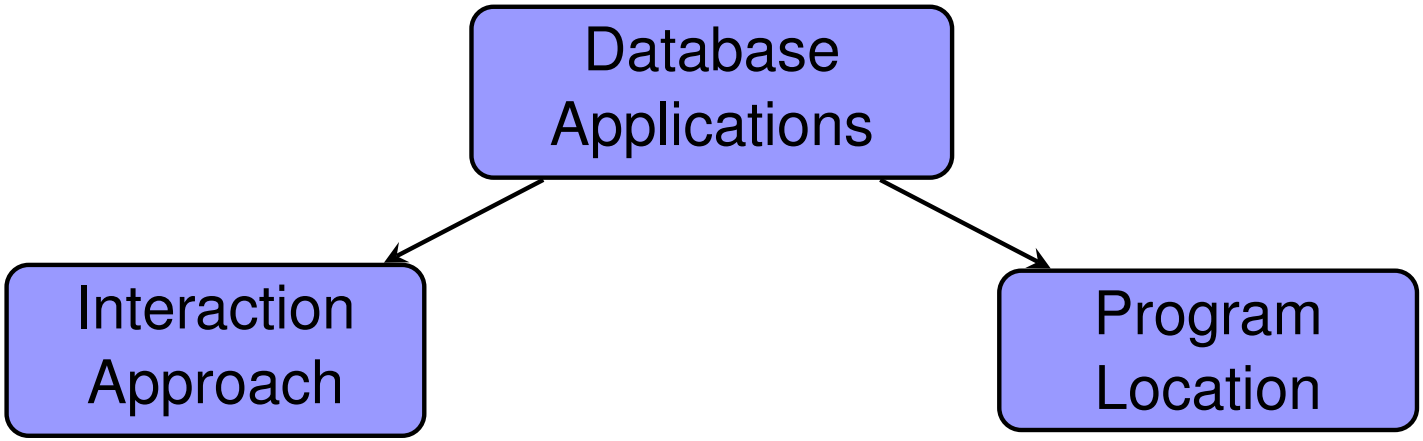


Categorizing Database Applications

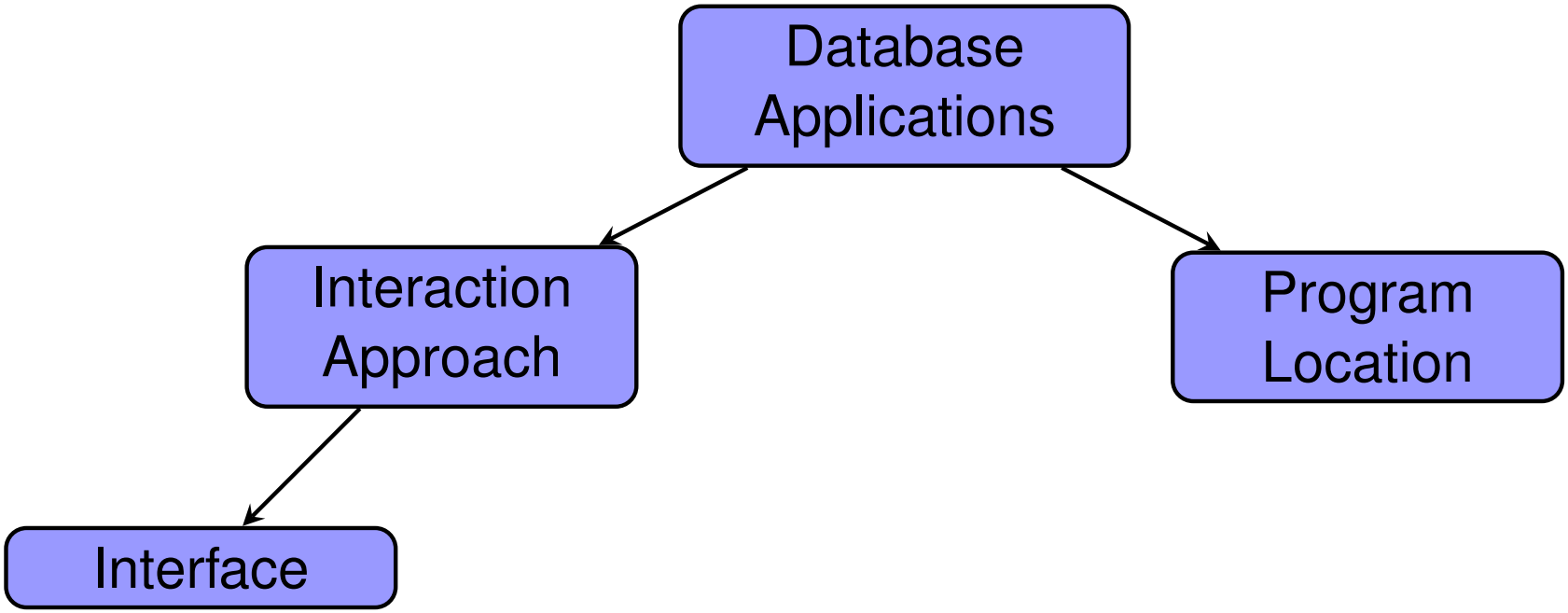
Categorizing Database Applications



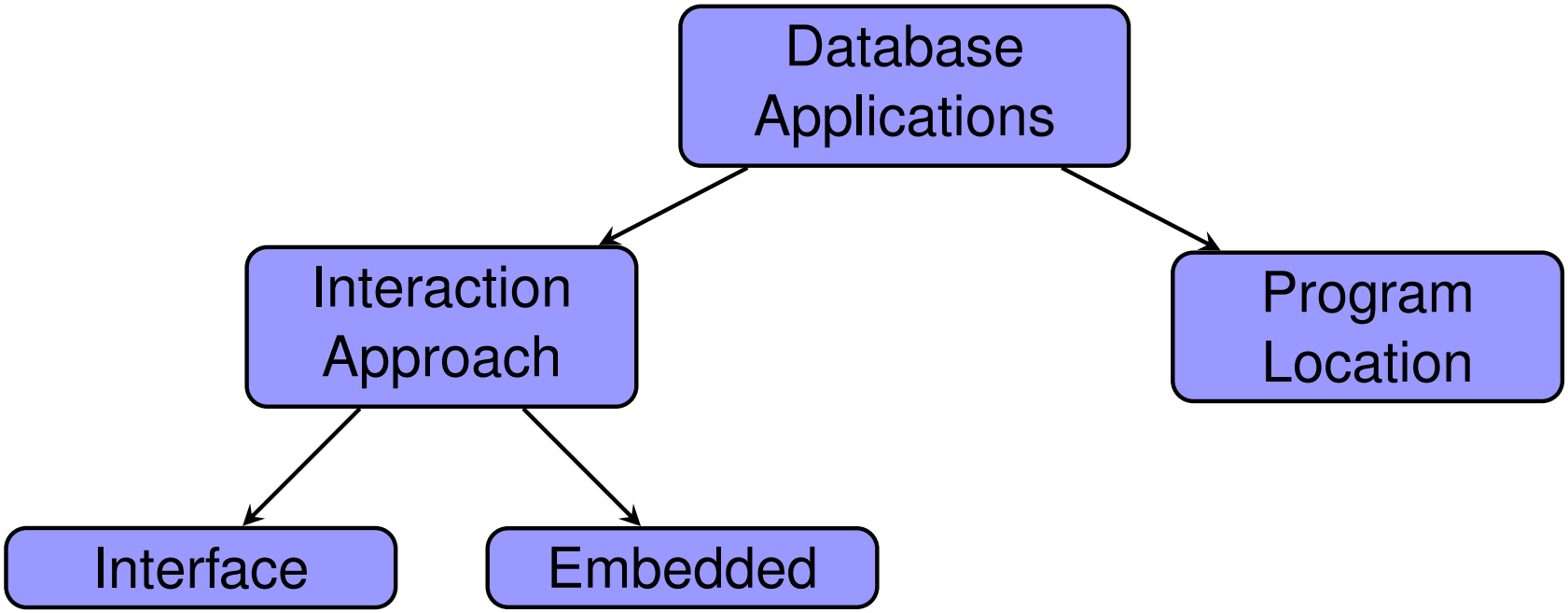
Categorizing Database Applications



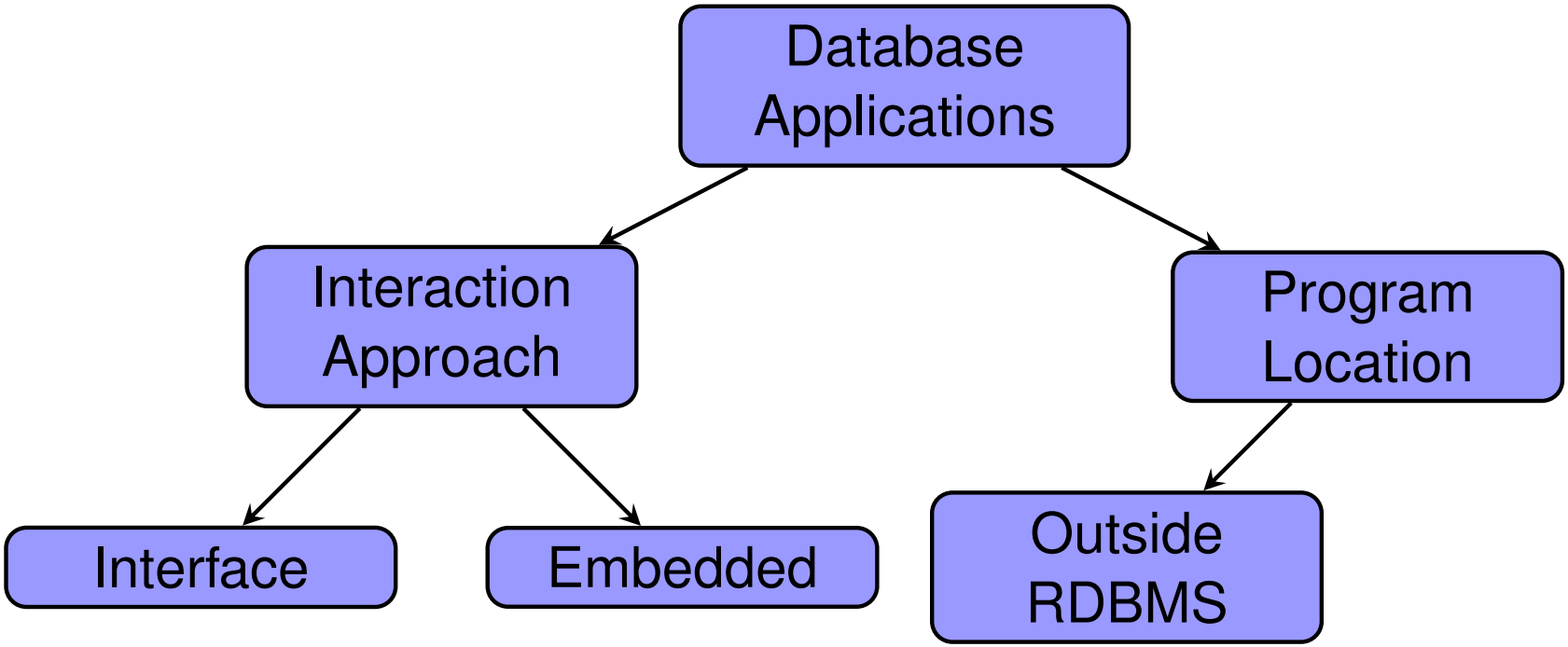
Categorizing Database Applications



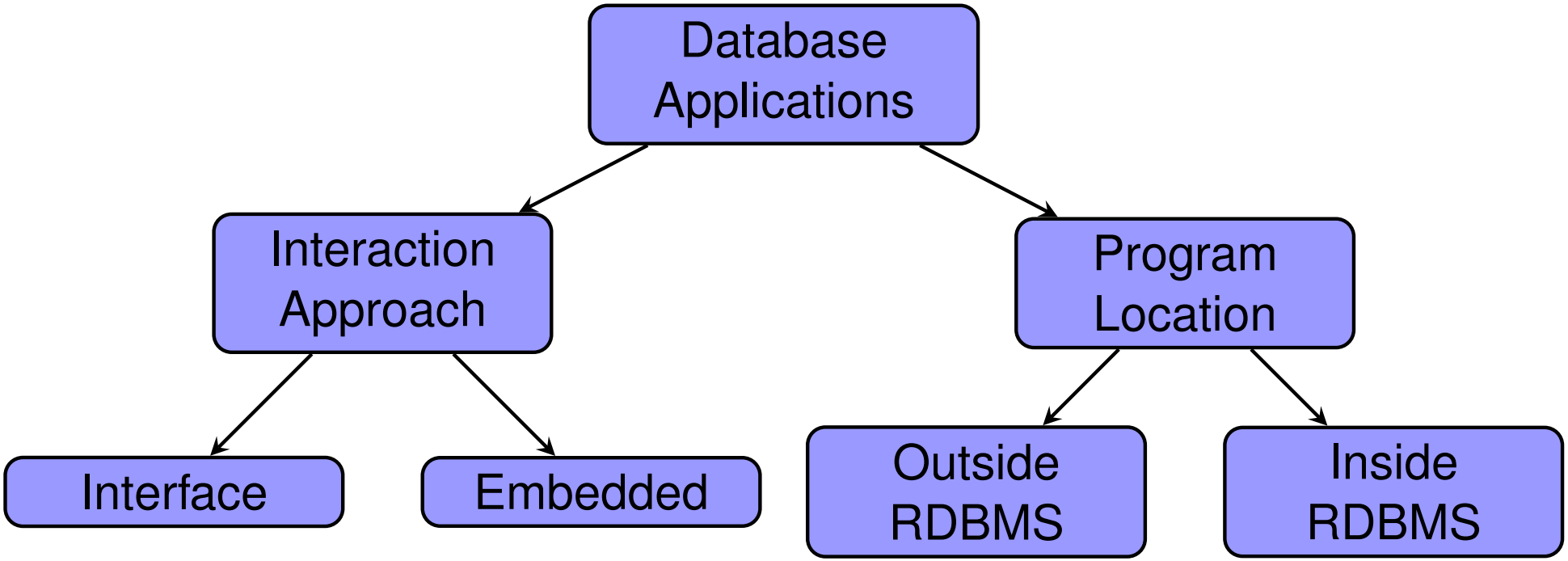
Categorizing Database Applications



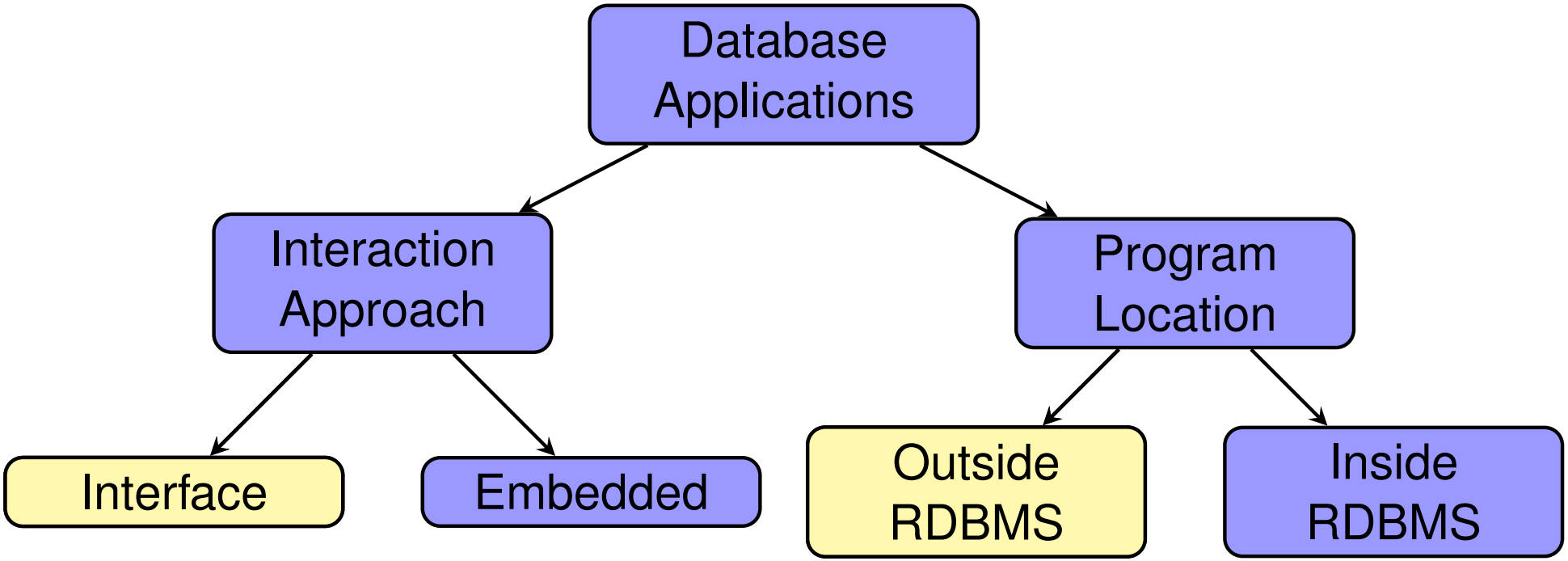
Categorizing Database Applications



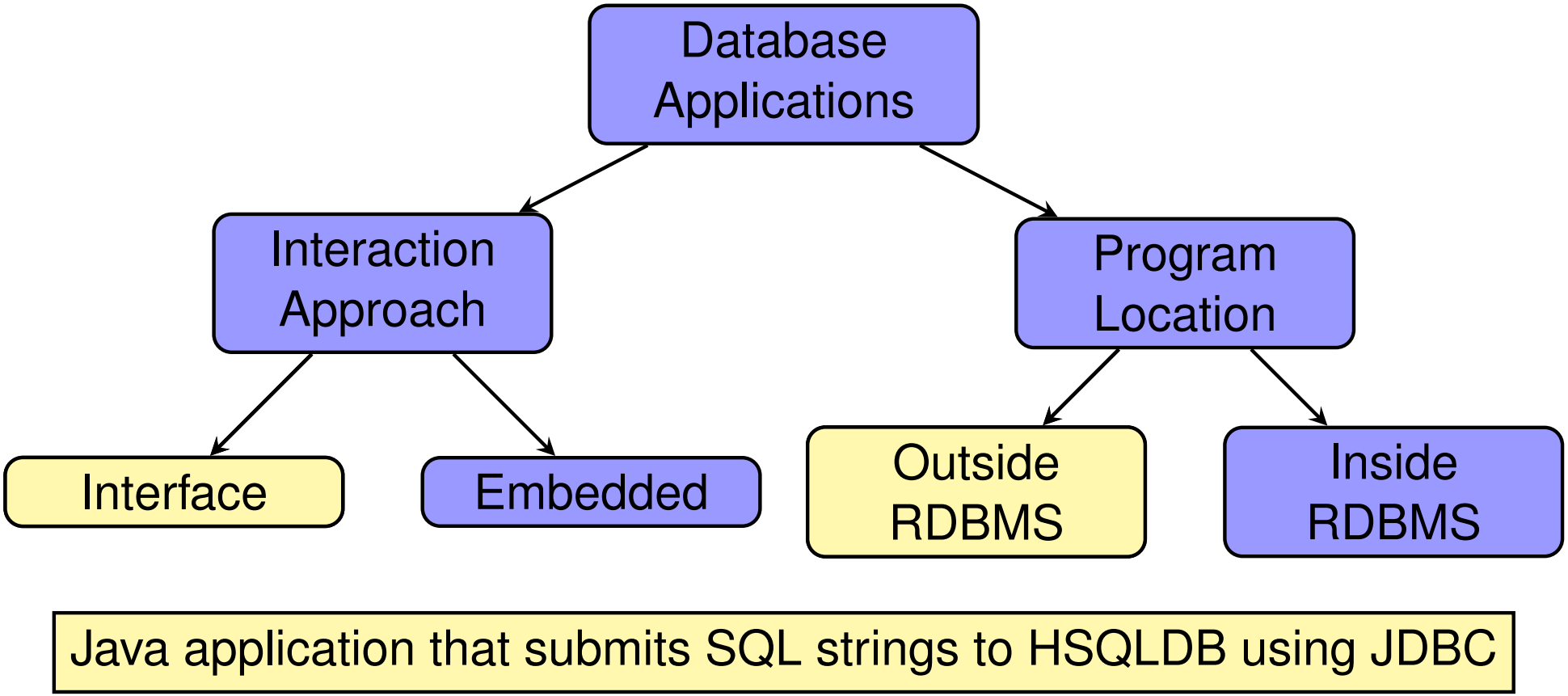
Categorizing Database Applications



Categorizing Database Applications

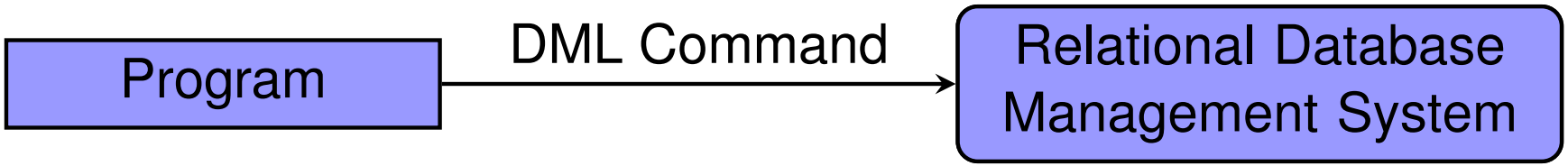


Categorizing Database Applications

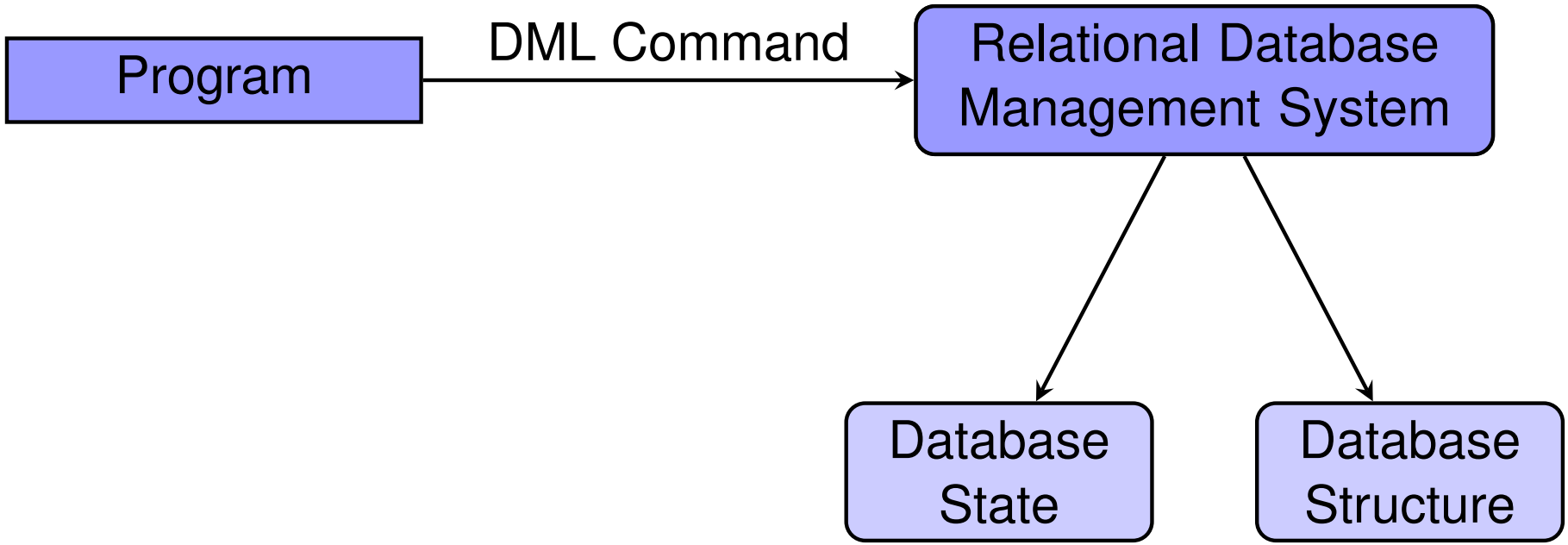


Java application that submits SQL strings to HSQLDB using JDBC

Evolution of Database Applications

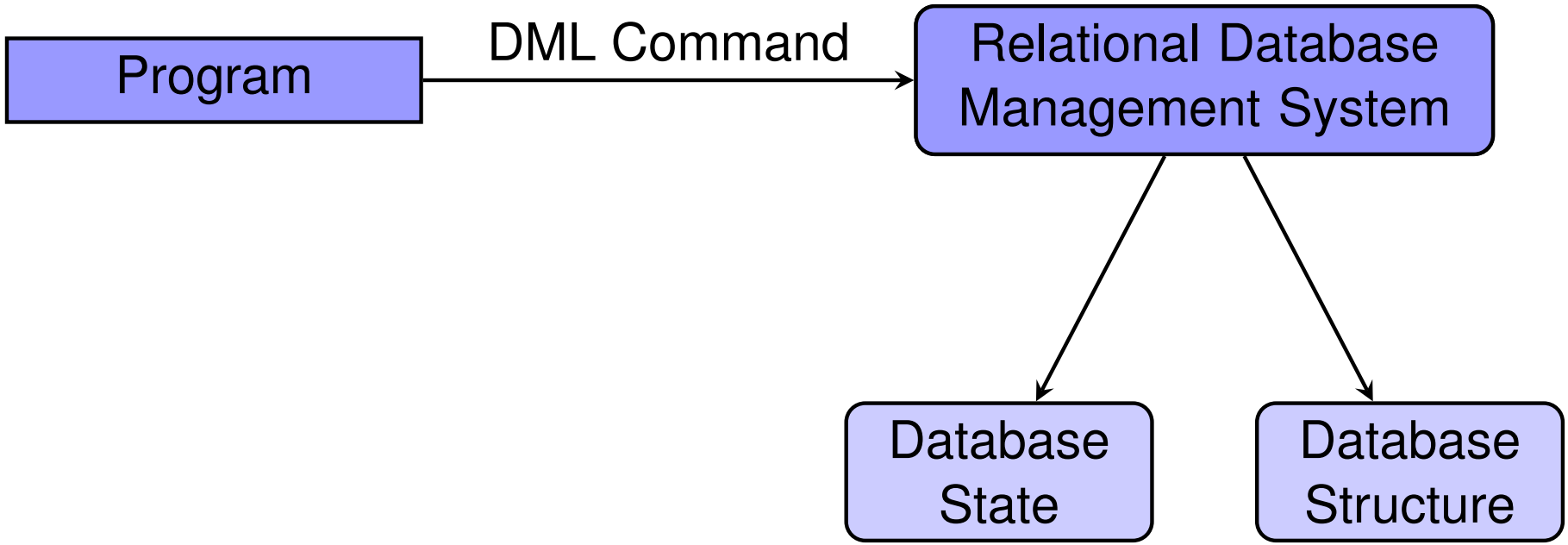


Evolution of Database Applications



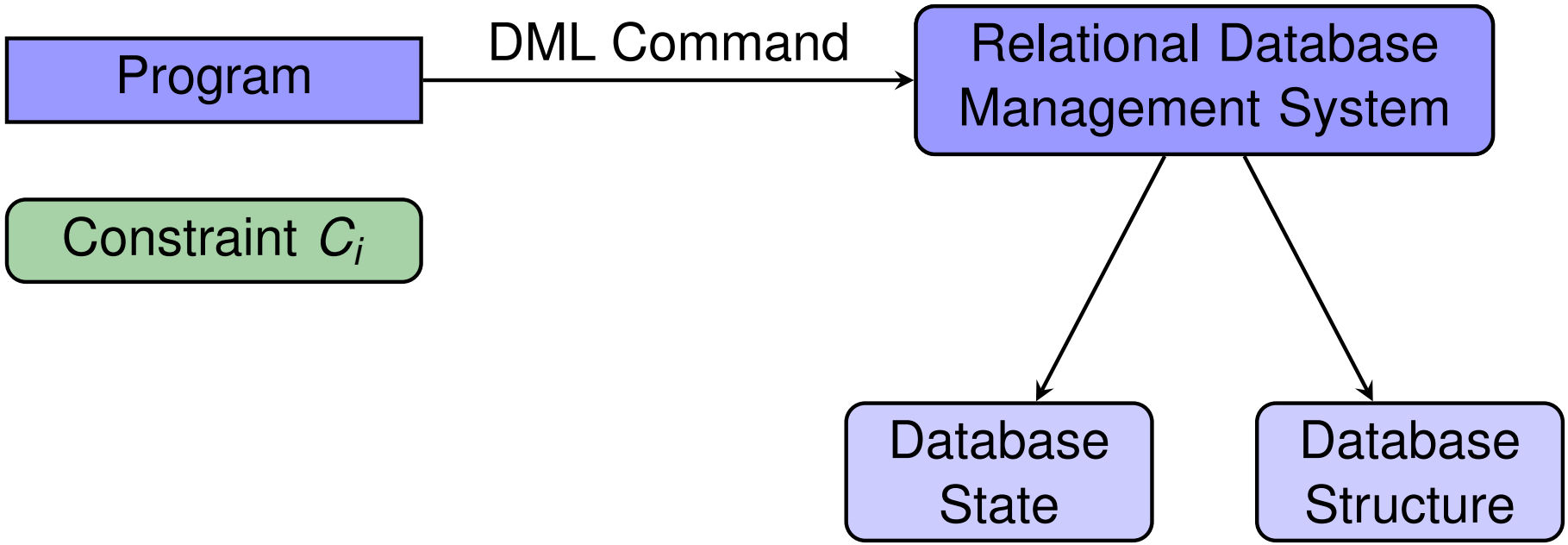
Evolution of Database Applications

Only the database administrator can add new constraints to the schema!



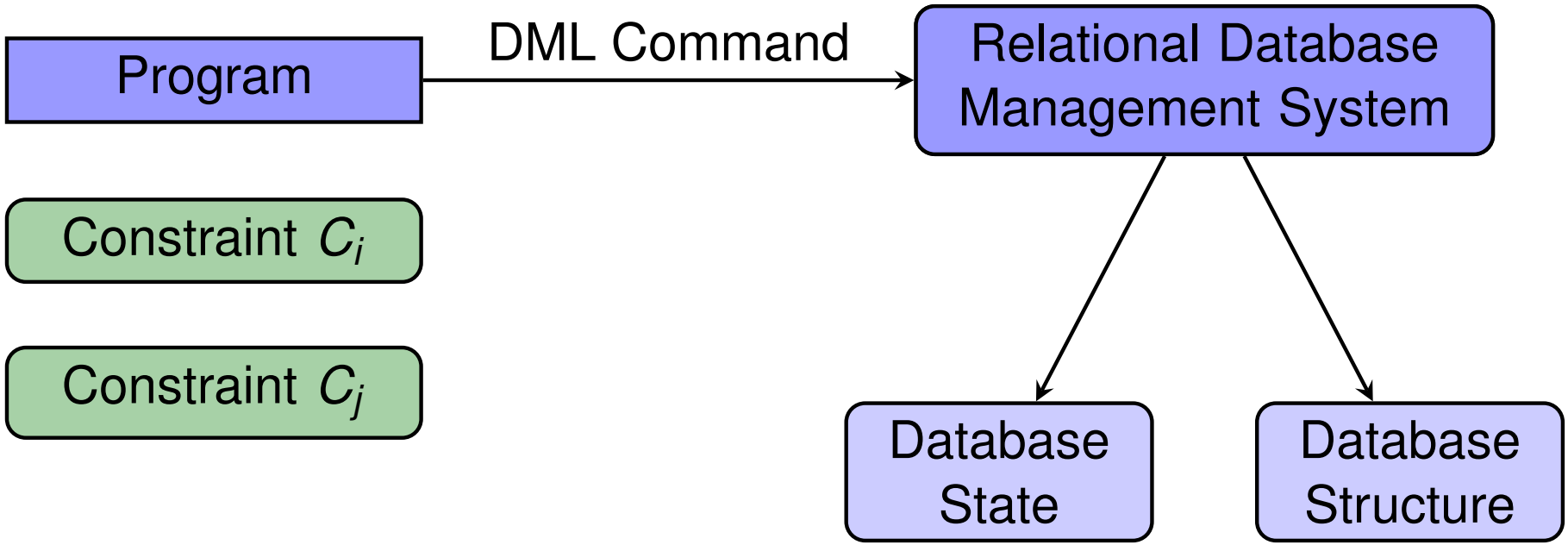
Evolution of Database Applications

The programmers encode the constraints in the program's source code!



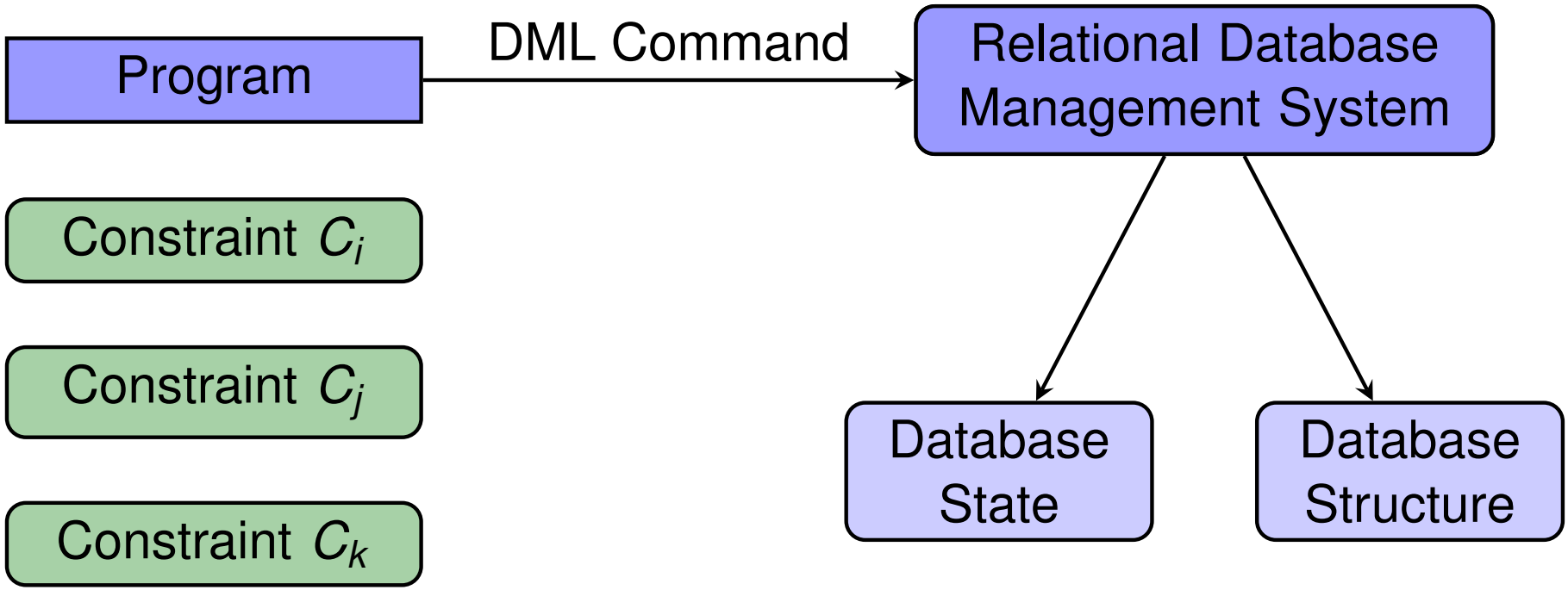
Evolution of Database Applications

The programmers encode the constraints in the program's source code!

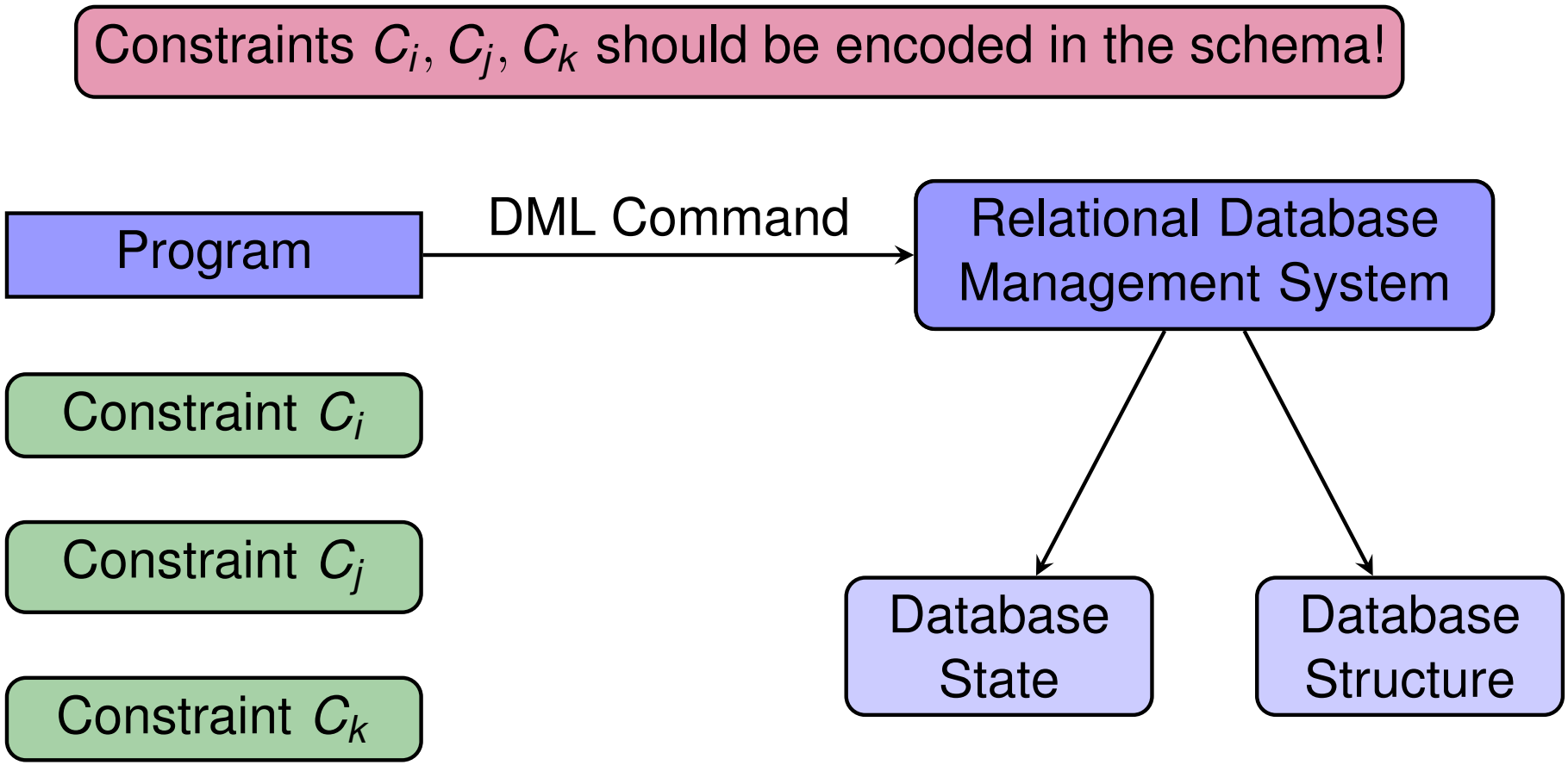


Evolution of Database Applications

The programmers encode the constraints in the program's source code!

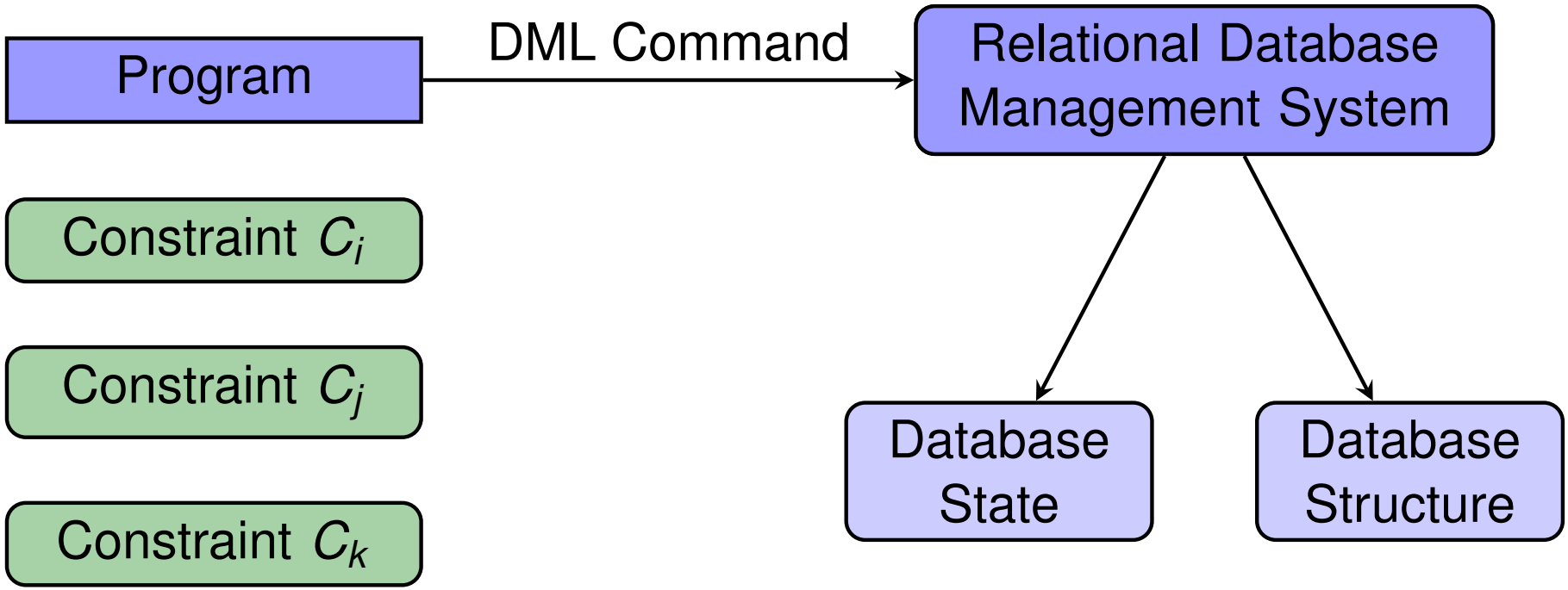


Evolution of Database Applications



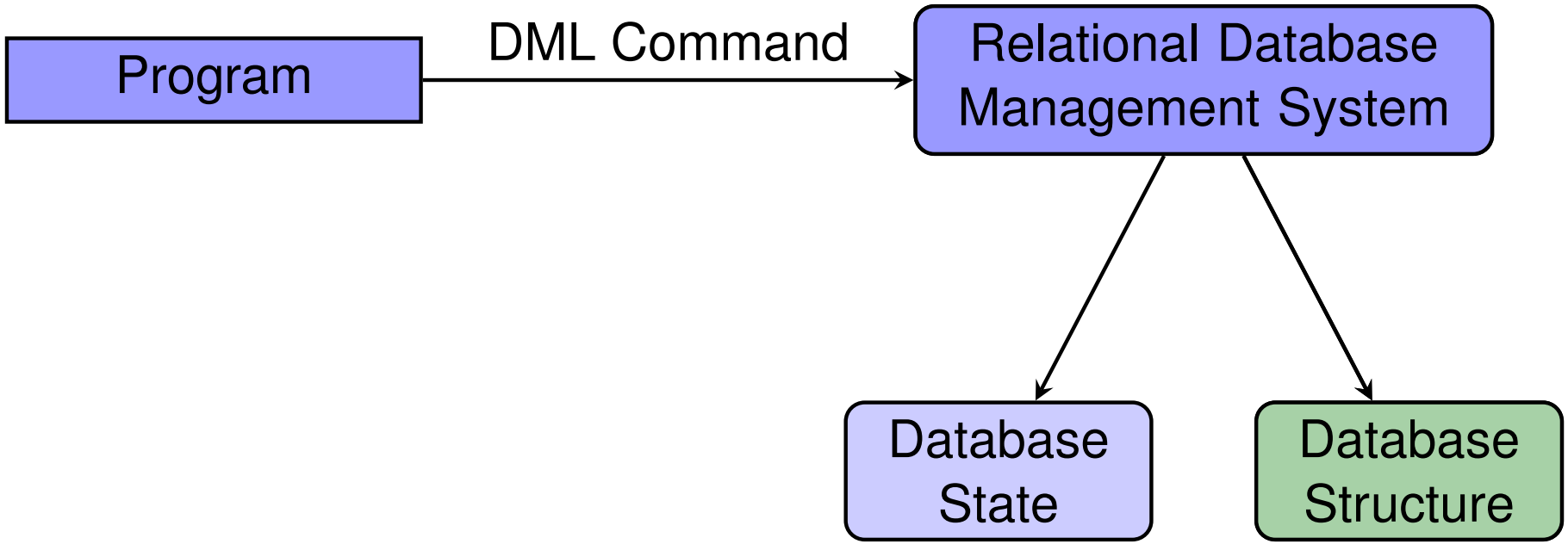
Evolution of Database Applications

Goal: extract C_i, C_j, C_k from the source code of the program



Evolution of Database Applications

Goal: extract C_i, C_j, C_k from the source code of the program



Invariants

Definition

An *invariant* is a mathematical property that holds through some set of transformations

Motivating Examples

- $0 \times y = 0$
- $|x| \geq 0$
- $\frac{c}{d} = \pi$

Invariants

Definition

An *invariant* is a mathematical property that holds through some set of transformations

Motivating Examples

- $0 \times y = 0$
- $|x| \geq 0$
- $\frac{c}{d} = \pi$

Invariants

Definition

An *invariant* is a mathematical property that holds through some set of transformations

Motivating Examples

- $0 \times y = 0$
- $|x| \geq 0$
- $\frac{c}{d} = \pi$

Invariants

Definition

An *invariant* is a mathematical property that holds through some set of transformations

Motivating Examples

- $0 \times y = 0$
- $|x| \geq 0$
- $\frac{c}{d} = \pi$

Program Invariants

Invariant with respect to:

- State
- Input/Output

Simple Examples

- $0 \leq x \leq 10$
- $1 \leq nextX() \leq 11$
- $nextX() = (x+1) \mod 11$

```
1  class Invariant {
2    static int x = 0;
3    public static int nextX() {
4      if( ++x > 10 )
5        x = 0;
6      return x + 1;
7    }
8  }
```

Program Invariants

Invariant with respect to:

- State
- Input/Output

Simple Examples

- $0 \leq x \leq 10$
- $1 \leq \texttt{nextX()} \leq 11$
- $\texttt{nextX()} = (x + 1) \mod 11$

```
1  class Invariant {
2    static int x = 0;
3    public static int nextX() {
4      if( ++x > 10 )
5        x = 0;
6      return x + 1;
7    }
8  }
```

Program Invariants

Invariant with respect to:

- State
- Input/Output

Simple Examples

- $0 \leq \texttt{x} \leq 10$
- $1 \leq \texttt{nextX()} \leq 11$
- $\texttt{nextX()} = (\texttt{x} + 1) \mod 11$

```
1  class Invariant {
2    static int x = 0;
3    public static int nextX() {
4      if( ++x > 10 )
5        x = 0;
6      return x + 1;
7    }
8  }
```

Program Invariants

Invariant with respect to:

- State
- Input/Output

Simple Examples

- $0 \leq \texttt{x} \leq 10$
- $1 \leq \texttt{nextX()} \leq 11$
- $\texttt{nextX()} = (\texttt{x} + 1) \mod 11$

```
1  class Invariant {
2    static int x = 0;
3    public static int nextX() {
4      if( ++x > 10 )
5        x = 0;
6      return x + 1;
7    }
8  }
```

Dynamic Invariants

Definition

A dynamic invariant is a property that is observed to hold during a *series of executions*

- Not guaranteed for all possible executions
- May reflect property of:
  - Program
  - Inputs

Dynamic Invariants

Definition

A dynamic invariant is a property that is observed to hold during a *series of executions*

- Not guaranteed for all possible executions
- May reflect property of:
  - Program
  - Inputs

Daikon Invariant Detector

Daikon [Ernst et al. 2001] is a dynamic invariant detection engine originally designed for traditional C and Java programs

Detection Process

- Collect data traces for variables at *program points*
- Compare to pool of potential invariants
- Output remaining invariants that meet confidence threshold

Daikon Invariant Detector

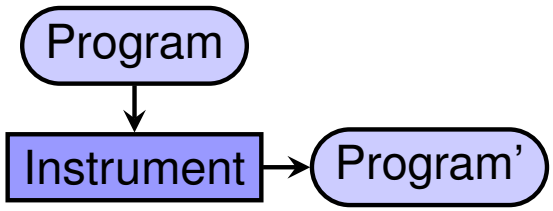
Daikon [Ernst et al. 2001] is a dynamic invariant detection engine originally designed for traditional C and Java programs

Detection Process

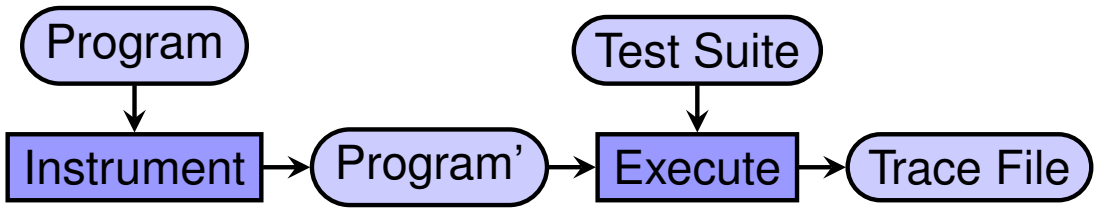
- Collect data traces for variables at *program point*s
- Compare to pool of potential invariants
- Output remaining invariants that meet confidence threshold

Invariant Detection Process

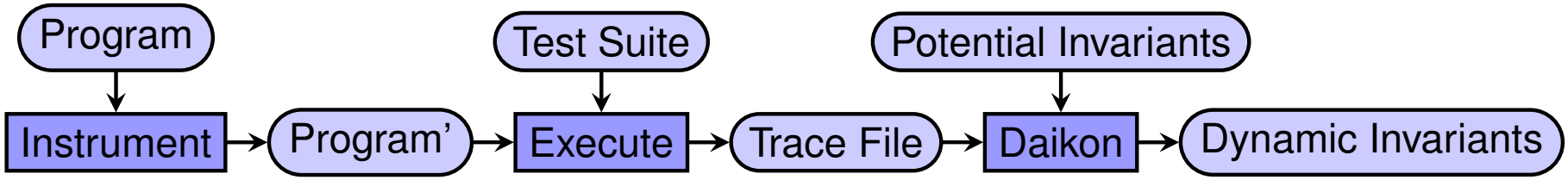
Invariant Detection Process



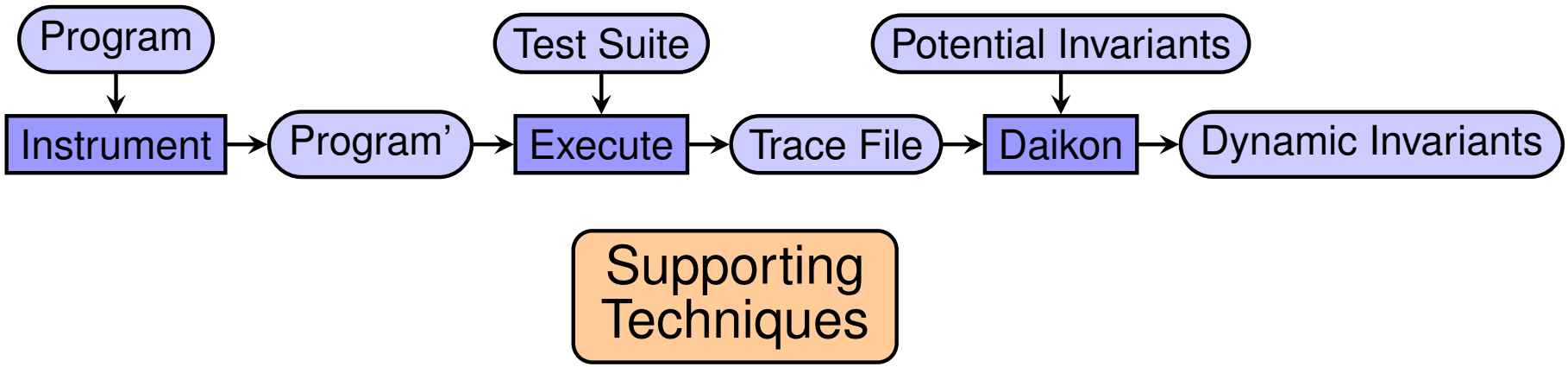
Invariant Detection Process



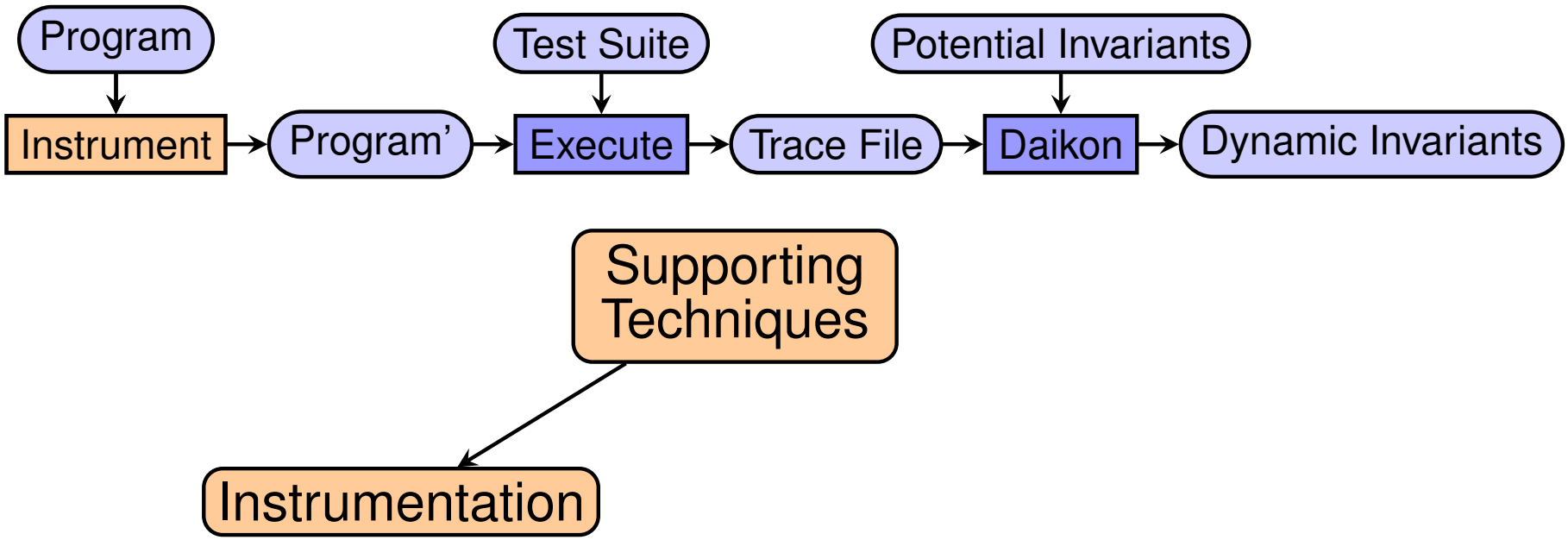
Invariant Detection Process



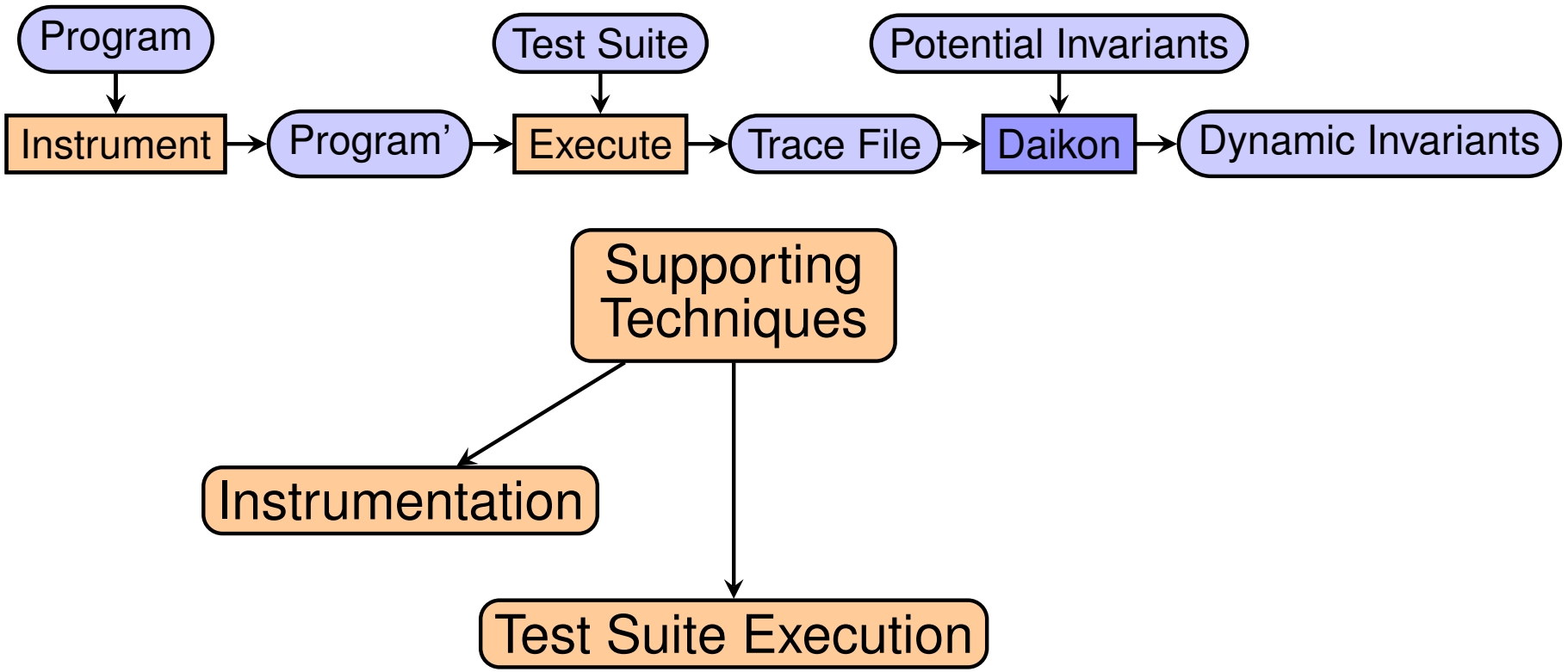
Invariant Detection Process



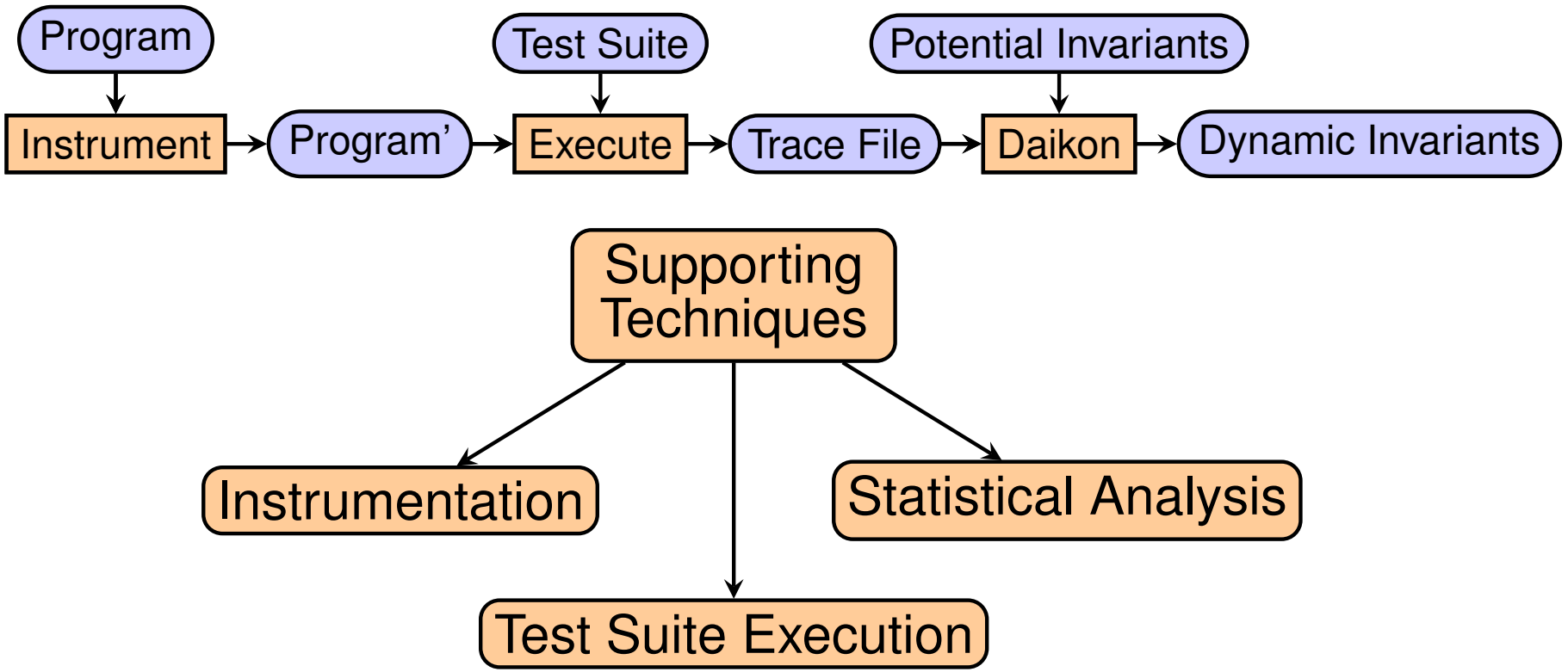
Invariant Detection Process



Invariant Detection Process

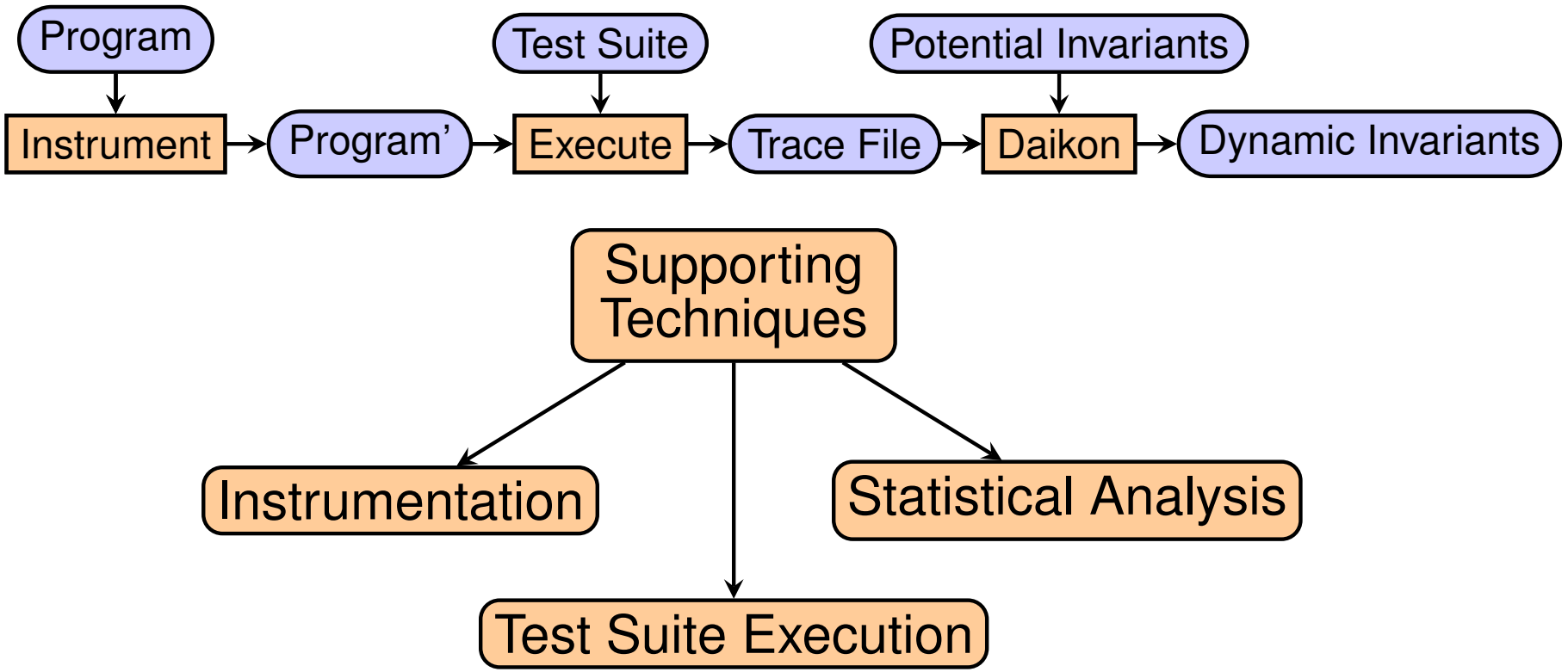


Invariant Detection Process



Invariant Detection Process

Refer to [Ernst et al. 2001] for additional details about these techniques



Applications of Daikon

Applications of dynamic invariants in software engineering:

- Programmer understanding
- Run-time checking
- Integration testing
- Interface discovery
- Test-input generation
- Test-suite reduction
- Many additional techniques

Structural Mapping

Program Element	Database Element
Program Point	Table
Variable	Column
Occurence	Row

Detect invariants for:

- Individual columns
- Between columns in a given row

id	name	age	employed	...
1	'John Smith'	38	5	...
2	'Jan Downing'	22	2	...

Structural Mapping

Program Element	Database Element
Program Point	Table
Variable	Column
Occurence	Row

Detect invariants for:

- Individual columns
- Between columns in a given row

id	name	age	employed	...
1	'John Smith'	38	5	...
2	'Jan Downing'	22	2	...

Structural Mapping

Program Element	Database Element
Program Point	Table
Variable	Column
Occurence	Row

Detect invariants for:

- Individual columns
- Between columns in a given row

id	name	age	employed	...
1	'John Smith'	38	5	...
2	'Jan Downing'	22	2	...

Data Mapping

Daikon Concepts

- Representation type
  - `int`
  - `double`
  - `String`
  - `int[]`
- Comparability

Data Mapping

Group	Name	SQL Types	Java Type
1	Text	CHAR VARCHAR TEXT	String
2	Integer	INTEGER NUMERIC BIT	int
3	Decimal	FLOAT DOUBLE REAL DECIMAL	double
4	Binary	BLOB BIT	byte[]
5	Text Set	SET	String[]
6	Datetime	DATETIME TIMESTAMP	String
7	Date	DATE	String
8	Time	TIME	String
9	Interval	INTERVAL	int
10	Primary Key	INTEGER	*reference*

Data Mapping

Handling `NULL` Values

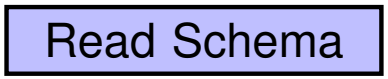
- `NULL` is a possible value for any SQL type
- Daikon does not accept `null` for primitive representation types such as `int`
- Introduce synthetic variable for each `NULL`-able column
  - Representation type is `hashcode` (*reference*)
  - Value is either `null` or a constant

Data Mapping

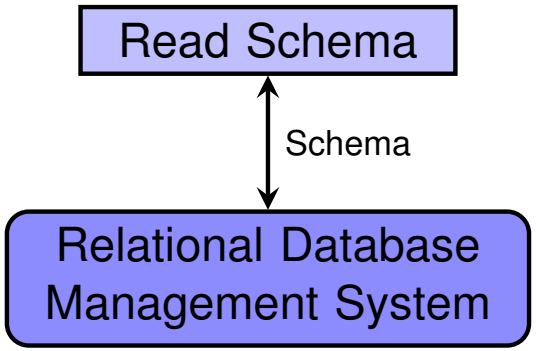
Handling `NULL` Values

- `NULL` is a possible value for any SQL type
- Daikon does not accept `null` for primitive representation types such as `int`
- Introduce synthetic variable for each `NULL`-able column
  - Representation type is `hashcode` (*reference*)
  - Value is either `null` or a constant

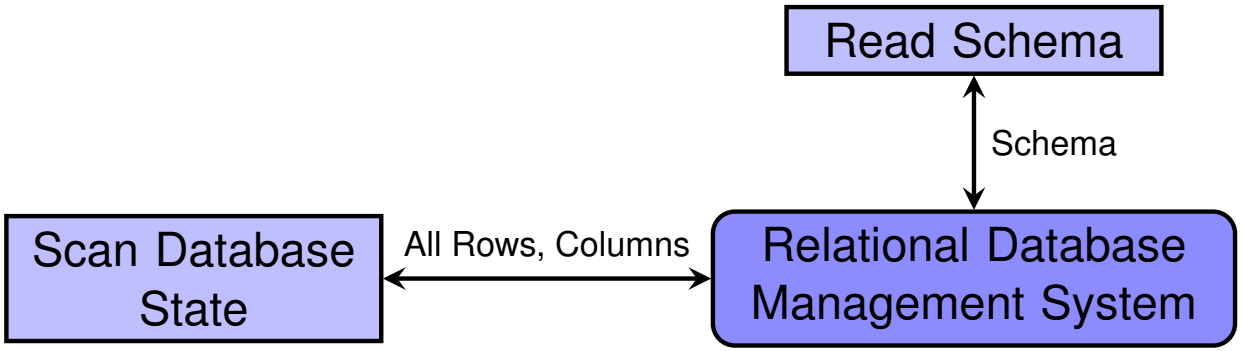
Database-Aware Procedure



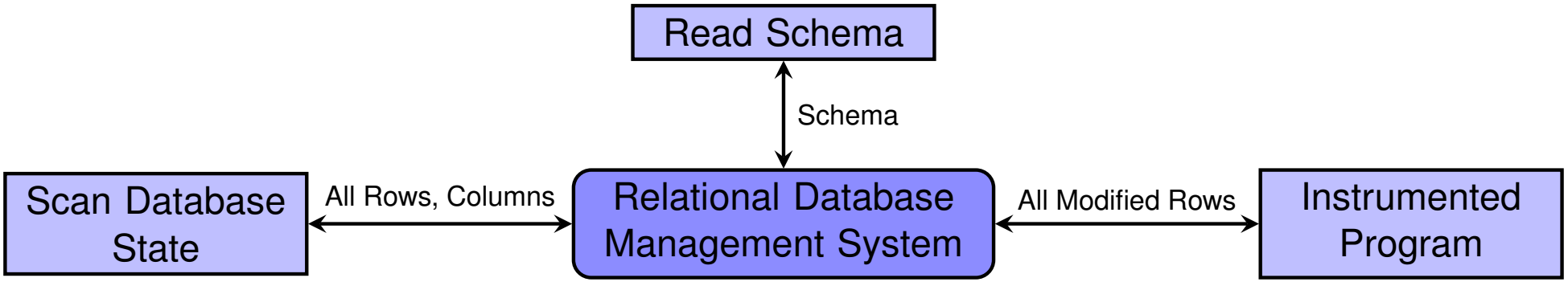
Database-Aware Procedure



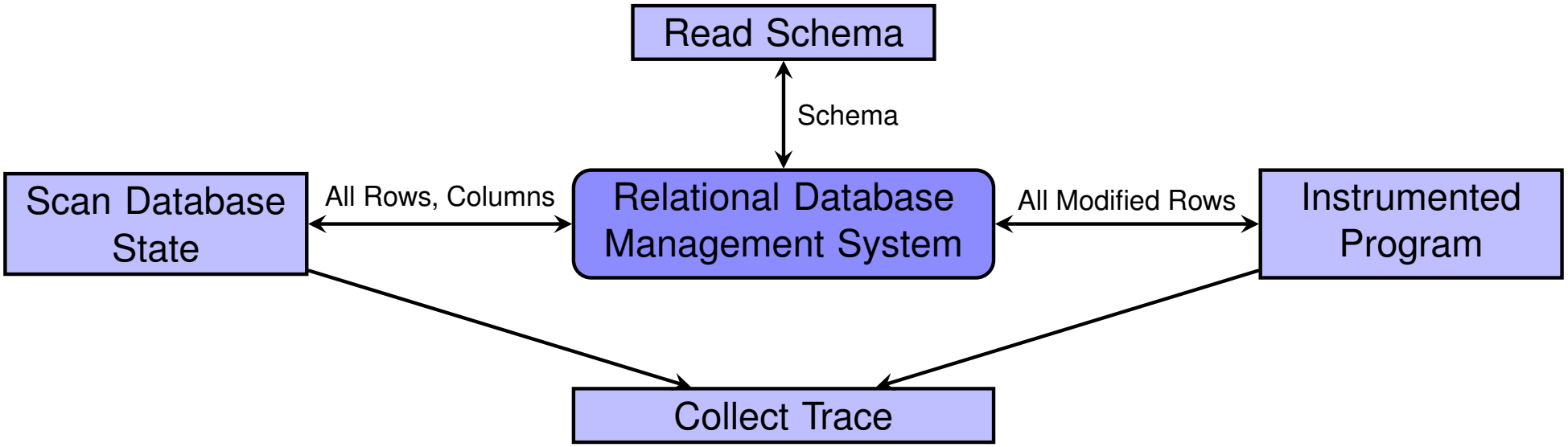
Database-Aware Procedure



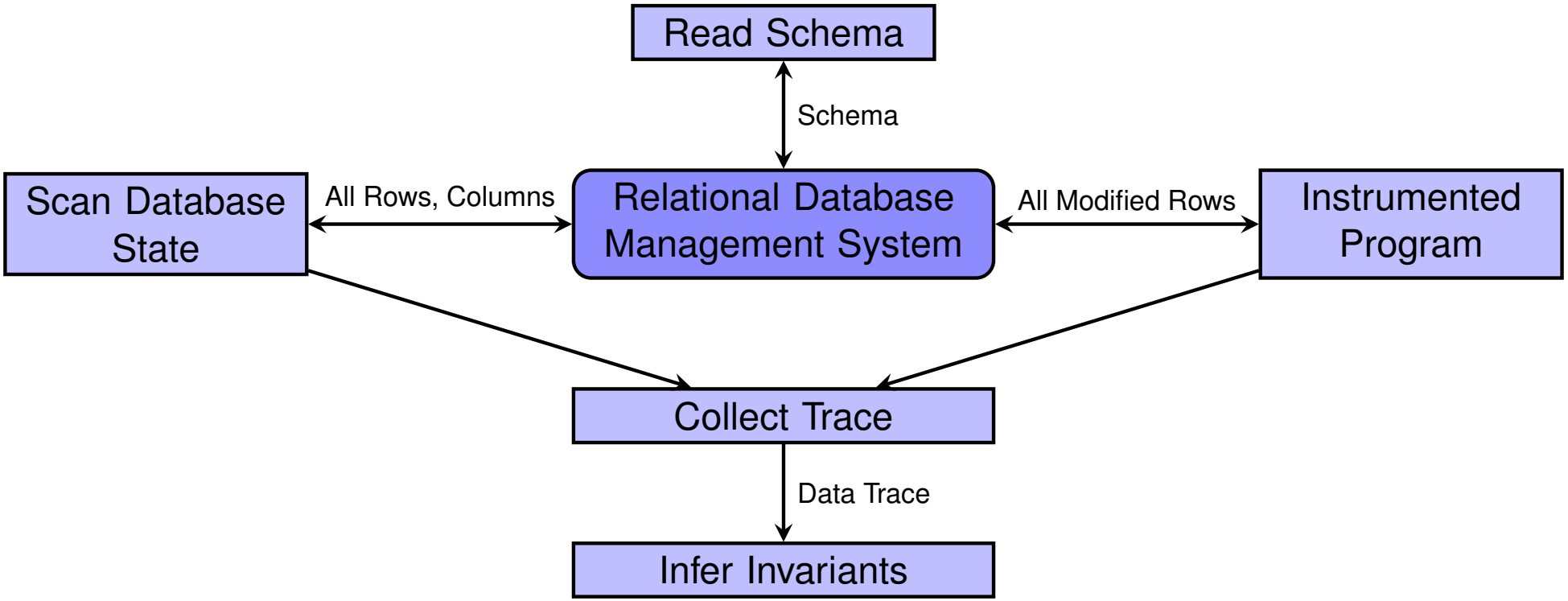
Database-Aware Procedure



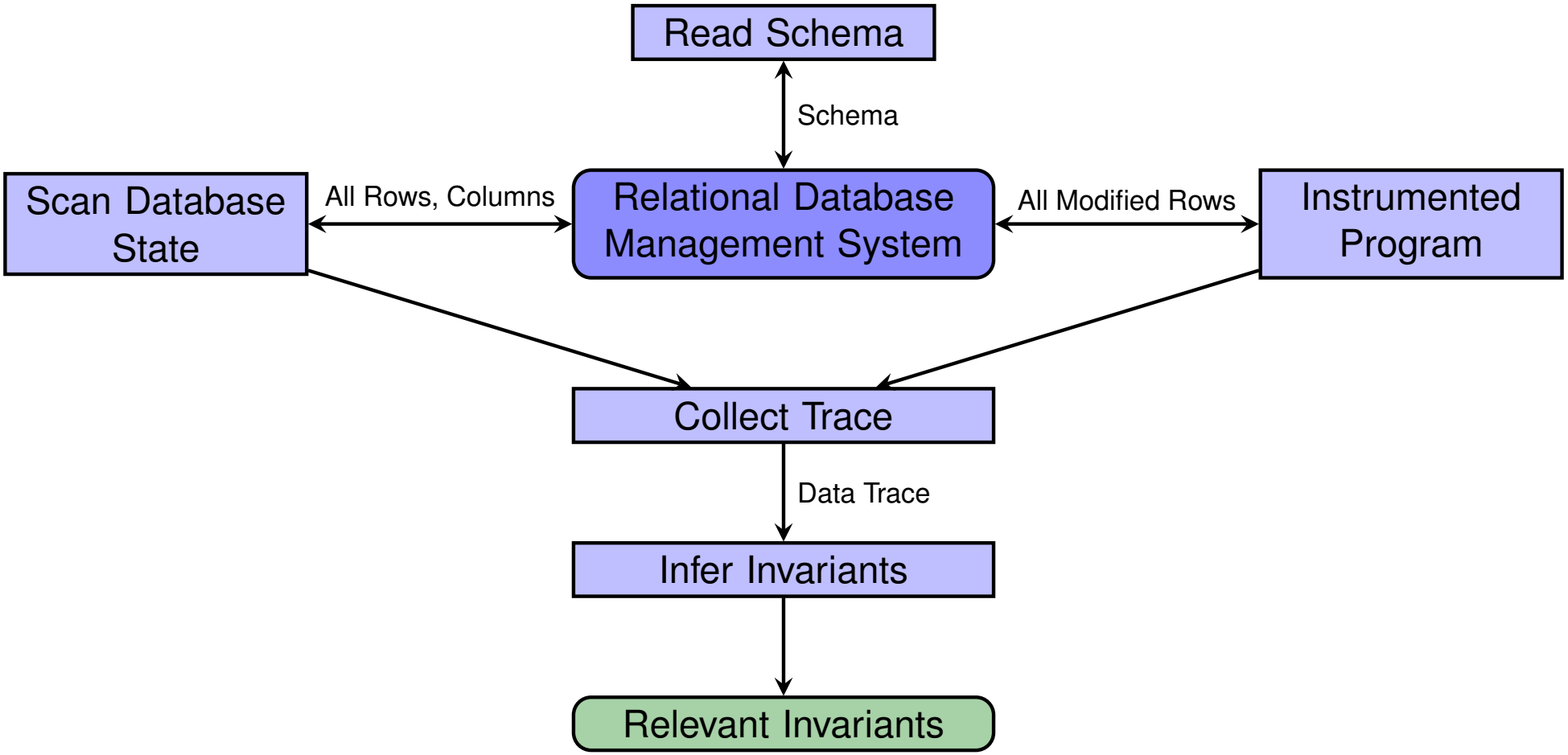
Database-Aware Procedure



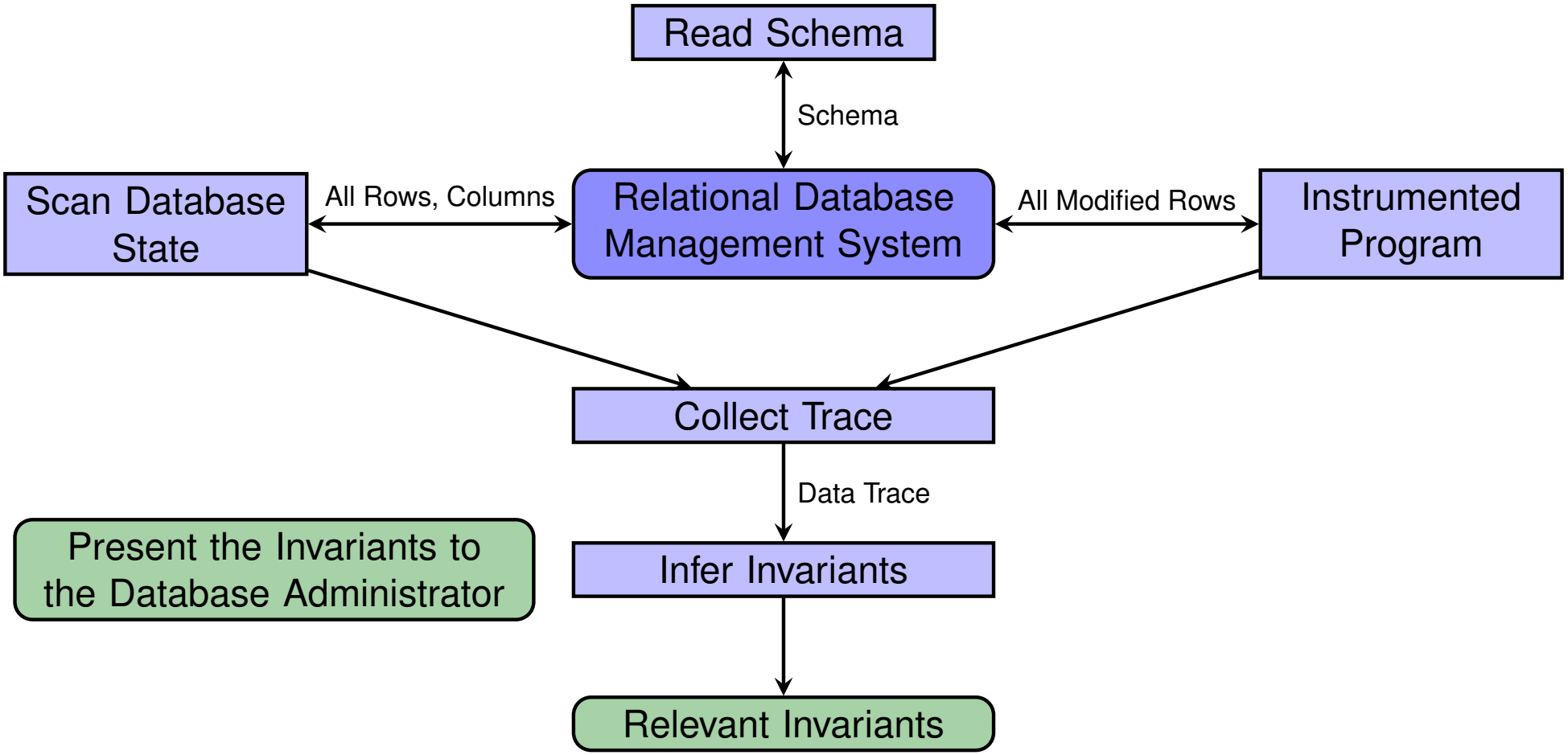
Database-Aware Procedure



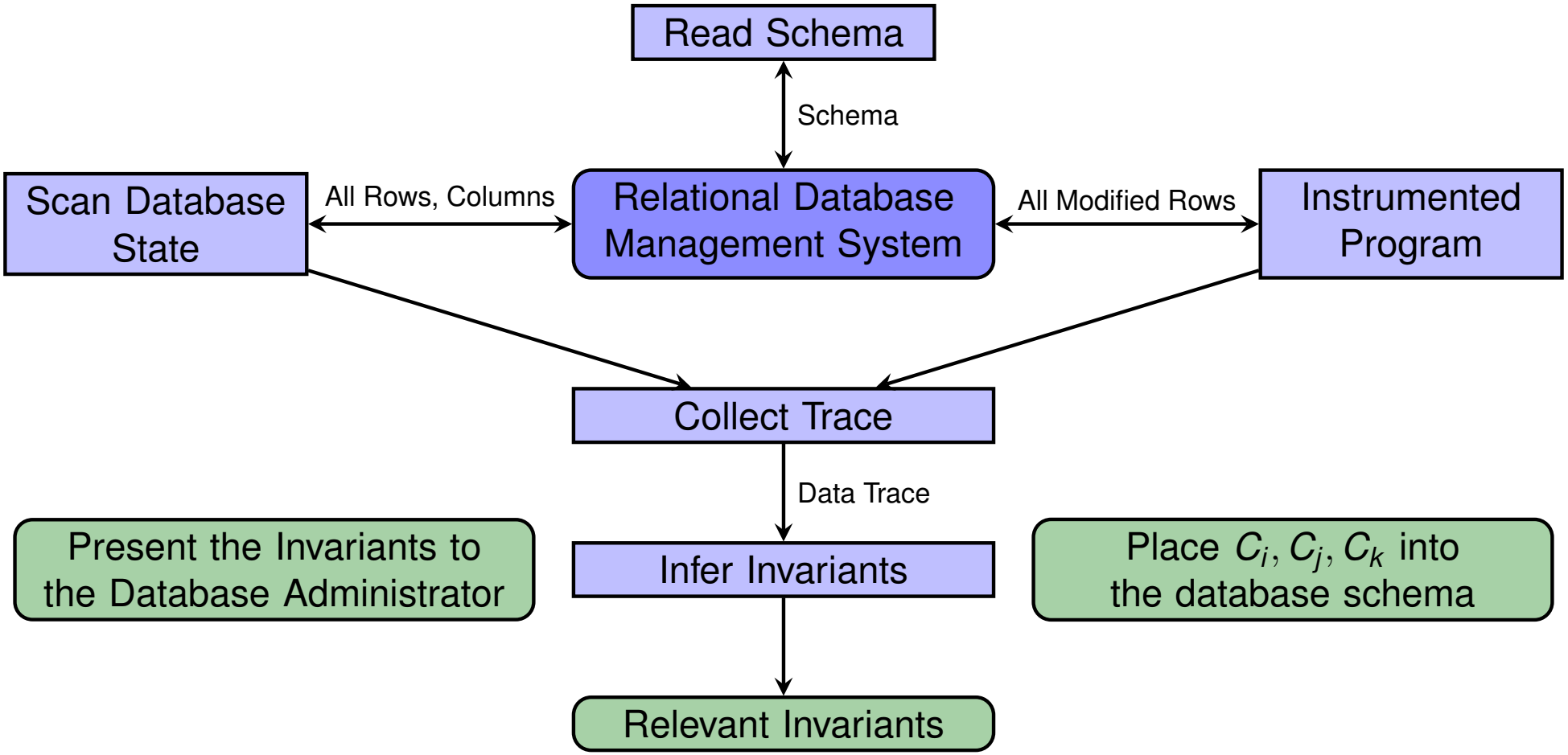
Database-Aware Procedure



Database-Aware Procedure



Database-Aware Procedure



Implementation

Trace Collector

- Python[a] program:
  - Input: Database connection information
  - Output: Daikon declarations and data trace files
- Process:
  1. Read schema metadata to determine tables, columns, and data mapping
  2. Write declarations file and serialize mapping info for reuse
  3. `SELECT` table contents, transform data by mapping, write to GZip'd trace file
- Supports various RDBMS with the SQLAlchemy toolkit

[a] . . . plus a small amount of Cython

Implementation

Trace Collector

- Python[a] program:
  - Input: Database connection information
  - Output: Daikon declarations and data trace files
- Process:
  1. Read schema metadata to determine tables, columns, and data mapping
  2. Write declarations file and serialize mapping info for reuse
  3. `SELECT` table contents, transform data by mapping, write to GZip'd trace file
- Supports various RDBMS with the SQLAlchemy toolkit

[a] . . . plus a small amount of Cython

Implementation

Trace Collector

- Python[a] program:
  - Input: Database connection information
  - Output: Daikon declarations and data trace files
- Process:
  1. Read schema metadata to determine tables, columns, and data mapping
  2. Write declarations file and serialize mapping info for reuse
  3. `SELECT` table contents, transform data by mapping, write to GZip'd trace file
- Supports various RDBMS with the SQLAlchemy toolkit

[a] . . . plus a small amount of Cython

Implementation

Instrumentation Wrapper

- Modified P6Spy JDBC driver wrapper
- On connection, capture information and initiate initial metadata read and trace
- On statement execution, append the trace file if the database could be modified
  - `INSERT` statement
  - `UPDATE` statement
  - Unknown (e.g., a stored procedure call)
  - Ignore others, including `DELETE` and `TRUNCATE`

Implementation

Instrumentation Wrapper

- Modified P6Spy JDBC driver wrapper
- On connection, capture information and initiate initial metadata read and trace
- On statement execution, append the trace file if the database could be modified
  - `INSERT` statement
  - `UPDATE` statement
  - Unknown (e.g., a stored procedure call)
  - Ignore others, including `DELETE` and `TRUNCATE`

Implementation

Instrumentation Wrapper

- Modified P6Spy JDBC driver wrapper
- On connection, capture information and initiate initial metadata read and trace
- On statement execution, append the trace file if the database could be modified
  - `INSERT` statement
  - `UPDATE` statement
  - Unknown (e.g., a stored procedure call)
  - Ignore others, including `DELETE` and `TRUNCATE`

Objects of Analysis

Fixed Data Sets

Database	Tables	Columns	Rows
world	3	24	5302
sakila	23	131	50,086
menagerie	2	10	19
employees	6	24	3,919,015

- MySQL sample databases commonly used for training, certification, and testing
- Trace the entire dataset during invariant detection

Objects of Analysis

Database Applications

Program	iTrust	JWhoisServer	JTrac
Tables	30	7	13
Columns	177	57	126
KLOC	25.5 (Java), 8.6 (JSP)	6.7	12
Test Cases	787	67	41

- Java applications that interact with a relational database
- Wrap real database driver in a modified P6Spy driver
- Execute the entire test suite during invariant detection

Invariant Quality

Meaningful Invariants

Invariants that capture a semantic relationship

- `dept_emp.from_date <= dept_emp.to_date`
- `employees.gender one of { "F", "M" }`
- `employees.birth_date < employees.hire_date`
- `country.Population >= 0`
- `icdcodes.Chronic one of { "no", "yes" }`

All of these invariants were automatically generated!

Invariant Quality

Meaningful Invariants

Invariants that capture a semantic relationship

- `dept_emp.from_date <= dept_emp.to_date`
- `employees.gender one of { "F", "M" }`
- `employees.birth_date < employees.hire_date`
- `country.Population >= 0`
- `icdcodes.Chronic one of { "no", "yes" }`

All of these invariants were automatically generated!

Invariant Quality

Meaningful Invariants

Invariants that capture a semantic relationship

- `dept_emp.from_date <= dept_emp.to_date`
- `employees.gender one of { "F", "M" }`
- `employees.birth_date < employees.hire_date`
- `country.Population >= 0`
- `icdcodes.Chronic one of { "no", "yes" }`

All of these invariants were automatically generated!

Invariant Quality

Meaningful Invariants

Invariants that capture a semantic relationship

- `dept_emp.from_date <= dept_emp.to_date`
- `employees.gender one of { "F", "M" }`
- `employees.birth_date < employees.hire_date`
- `country.Population >= 0`
- `icdcodes.Chronic one of { "no", "yes" }`

All of these invariants were automatically generated!

Invariant Quality

Meaningful Invariants

Invariants that capture a semantic relationship

- `dept_emp.from_date <= dept_emp.to_date`
- `employees.gender one of { "F", "M" }`
- `employees.birth_date < employees.hire_date`
- `country.Population >= 0`
- `icdcodes.Chronic one of { "no", "yes" }`

All of these invariants were automatically generated!

Invariant Quality

Meaningful Invariants

Invariants that capture a semantic relationship

- `dept_emp.from_date <= dept_emp.to_date`
- `employees.gender one of { "F", "M" }`
- `employees.birth_date < employees.hire_date`
- `country.Population >= 0`
- `icdcodes.Chronic one of { "no", "yes" }`

All of these invariants were automatically generated!

Invariant Quality

Meaningful Invariants

Invariants that capture a semantic relationship

- `dept_emp.from_date <= dept_emp.to_date`
- `employees.gender one of { "F", "M" }`
- `employees.birth_date < employees.hire_date`
- `country.Population >= 0`
- `icdcodes.Chronic one of { "no", "yes" }`

All of these invariants were automatically generated!

Spurious Invariants

Spurious Invariants

- Vacuous invariants reflect a meaningless relationship.
  - `patients.phone1 <= patients.BloodType`
  - `patients.lastName >= patients.address1`
  - `cptcodes.Description != cptcodes.Attribute`
- Lack-of-data invariants result from limited data samples.
  - `mntnr.login == "mntnt"`
  - `inetnum.changed == "2006-10-14 16:21:09"`
  - `person.name one of { "no name company", "persona non grata"}`

Spurious Invariants

Spurious Invariants

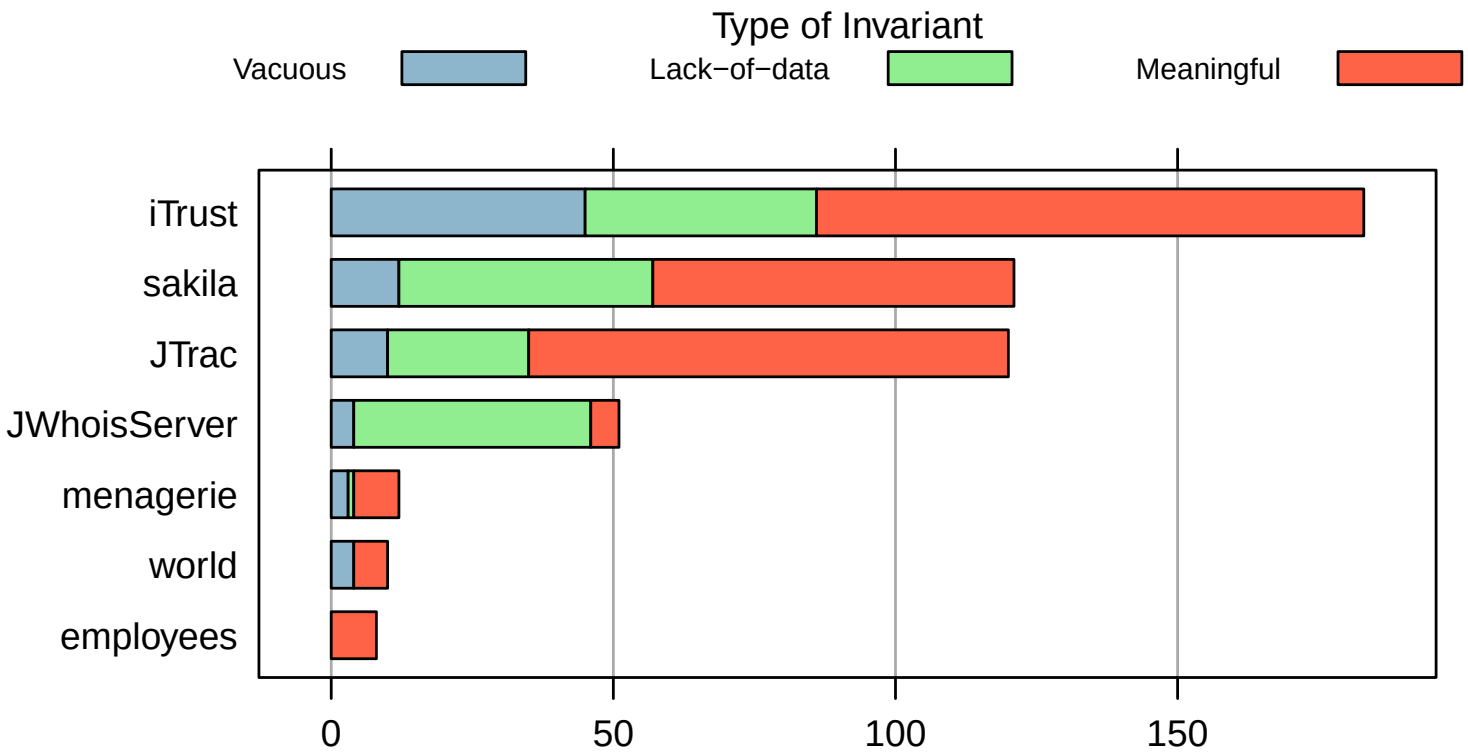
- Vacuous invariants reflect a meaningless relationship.
  - `patients.phone1 <= patients.BloodType`
  - `patients.lastName >= patients.address1`
  - `cptcodes.Description != cptcodes.Attribute`
- Lack-of-data invariants result from limited data samples.
  - `mntnr.login == "mntnt"`
  - `inetnum.changed == "2006-10-14 16:21:09"`
  - `person.name one of { "no name company", "persona non grata"}`

Spurious Invariants

Spurious Invariants

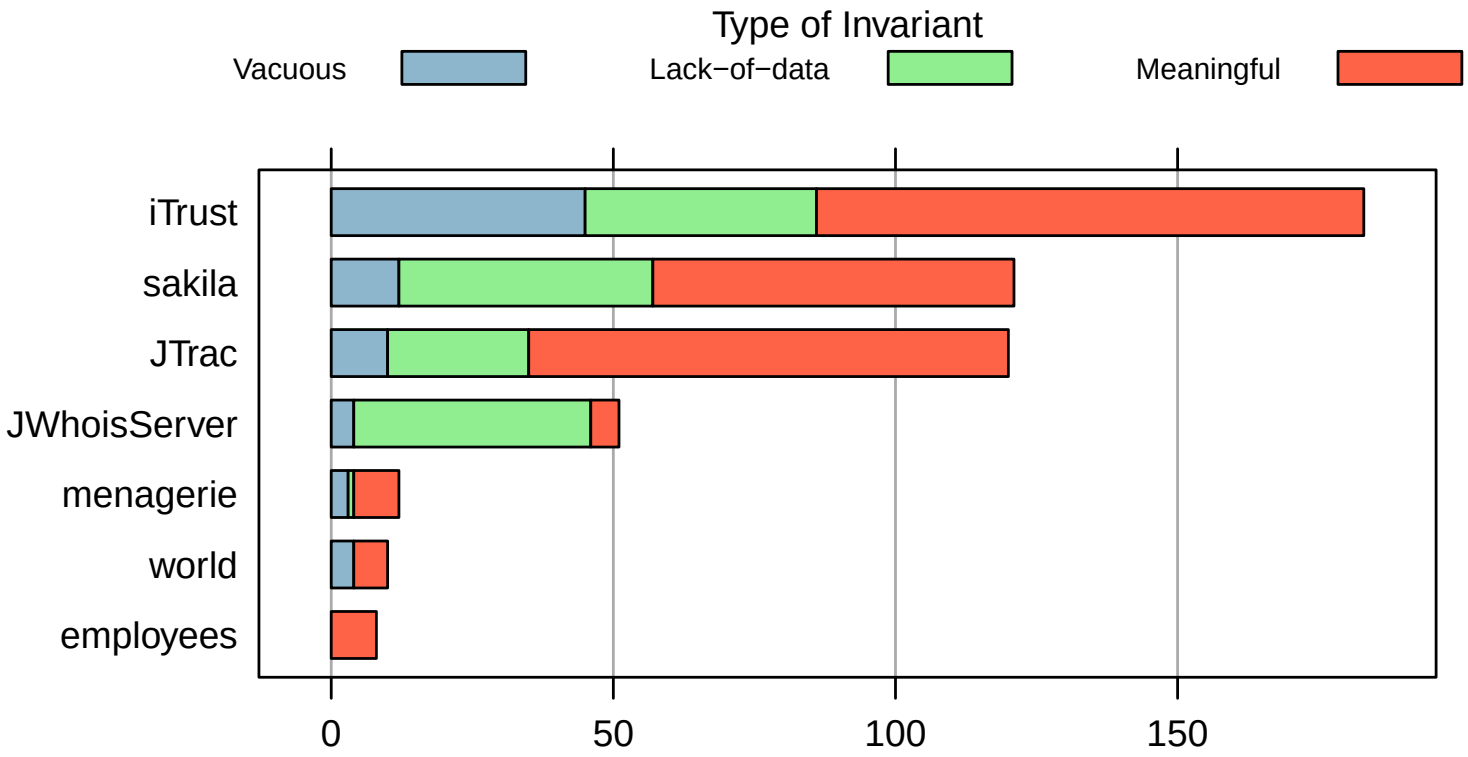
- Vacuous invariants reflect a meaningless relationship.
  - `patients.phone1 <= patients.BloodType`
  - `patients.lastName >= patients.address1`
  - `cptcodes.Description != cptcodes.Attribute`
- Lack-of-data invariants result from limited data samples.
  - `mntnr.login == "mntnt"`
  - `inetnum.changed == "2006-10-14 16:21:09"`
  - `person.name one of { "no name company", "persona non grata"}`

Invariant Quality



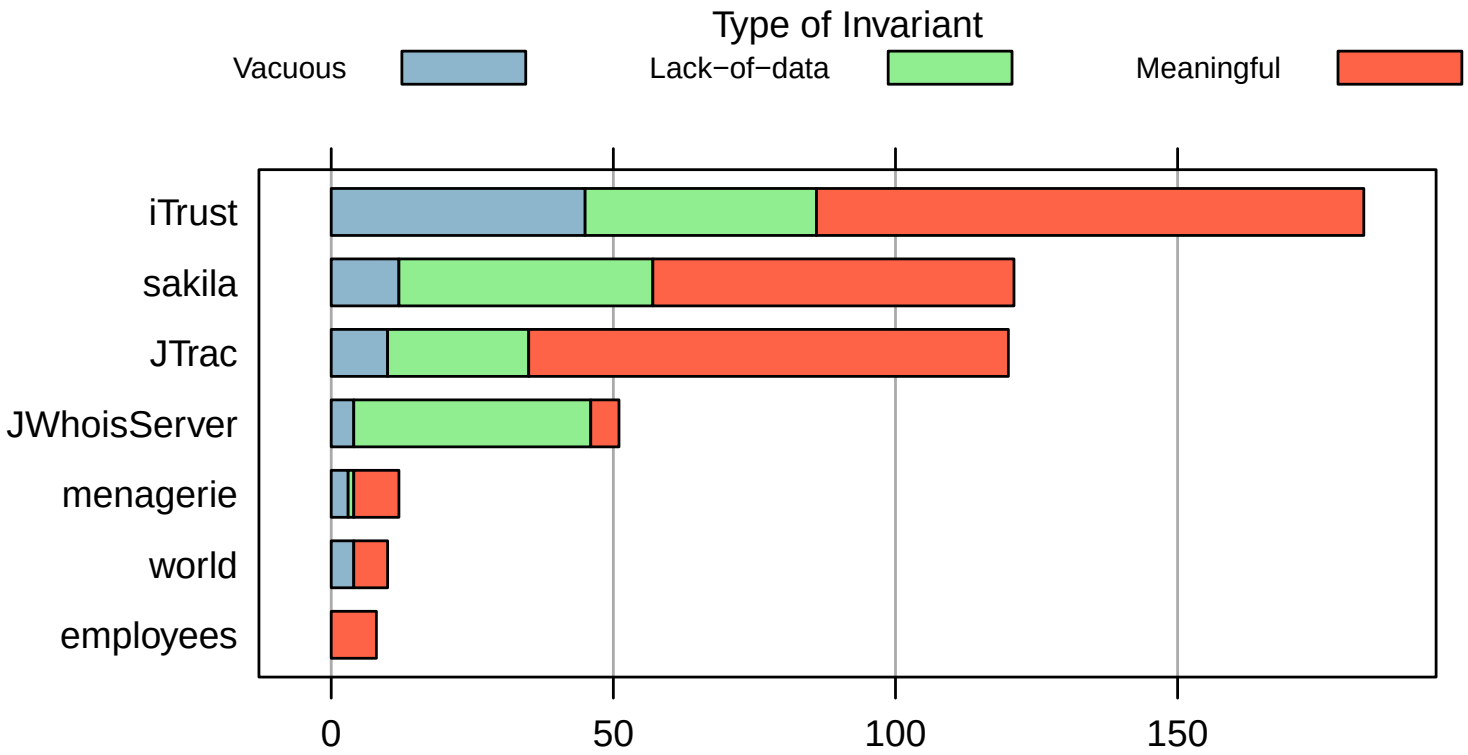
The majority of detected dynamic invariants are not spurious

Invariant Quality



employees' invariants are meaningful due to wealth of data

Invariant Quality



JWhoisServer's few meaningful invariants suggests poor tests

Schema Modification

Using Dynamic Invariants

- Some invariants can be enforced by the schema definition
- Schema enforcement:
  - Provides a stronger assurance of data integrity than application enforcement
  - Enables easy long-term maintenance of the program and the relational database
- Analyze enforceable invariants:
  - Already enforced by the schema
  - Suggest modification to enforce the invariant

Schema Modification

Using Dynamic Invariants

- Some invariants can be enforced by the schema definition
- Schema enforcement:
  - Provides a stronger assurance of data integrity than application enforcement
  - Enables easy long-term maintenance of the program and the relational database
- Analyze enforceable invariants:
  - Already enforced by the schema
  - Suggest modification to enforce the invariant

Schema Modification

Schema Enforced

Invariant	Schema Definition
`employees.gender one of { "F", "M" }`	`ENUM('F','M')`
`countrylanguage.IsOfficial one of { "F", "T" }`	`ENUM('F','T')`
`customer.active one of { 0, 1 }`	`TINYINT(1)`
`inventory.film_id >= 1`	`SMALLINT(5) UNSIGNED`
`spaces.guest_allowed one of { 0, 1 }`	`BIT(1)`

Reverse engineered many constraints already enforced by the schema

Schema Modification

Schema Enforced

Invariant	Schema Definition
`employees.gender one of { "F", "M" }`	`ENUM('F','M')`
`countrylanguage.IsOfficial one of { "F", "T" }`	`ENUM('F','T')`
`customer.active one of { 0, 1 }`	`TINYINT(1)`
`inventory.film_id >= 1`	`SMALLINT(5) UNSIGNED`
`spaces.guest_allowed one of { 0, 1 }`	`BIT(1)`

Reverse engineered many constraints already enforced by the schema

Schema Modification

Schema Enforceable

Invariant	Schema	Modification
`isnull(message.message) != null`	`TEXT`	`NOT NULL`
`isnull(film_text.description) != null`	`TEXT`	`NOT NULL`
`isnull(history.time_stamp) != null`	`DATETIME`	`NOT NULL`
`user_space_roles.user_id >= 1`	`BIGINT(20)`	`UNSIGNED`
`pet.sex one of { "f", "m" }`	`CHAR(1)`	`ENUM('m','f')`
`country.Population >= 0`	`INT(11)`	`UNSIGNED`
`isnull(titles.to_date) != null`	`DATE`	`NOT NULL`

Many detected constraints can easily be added to the schema

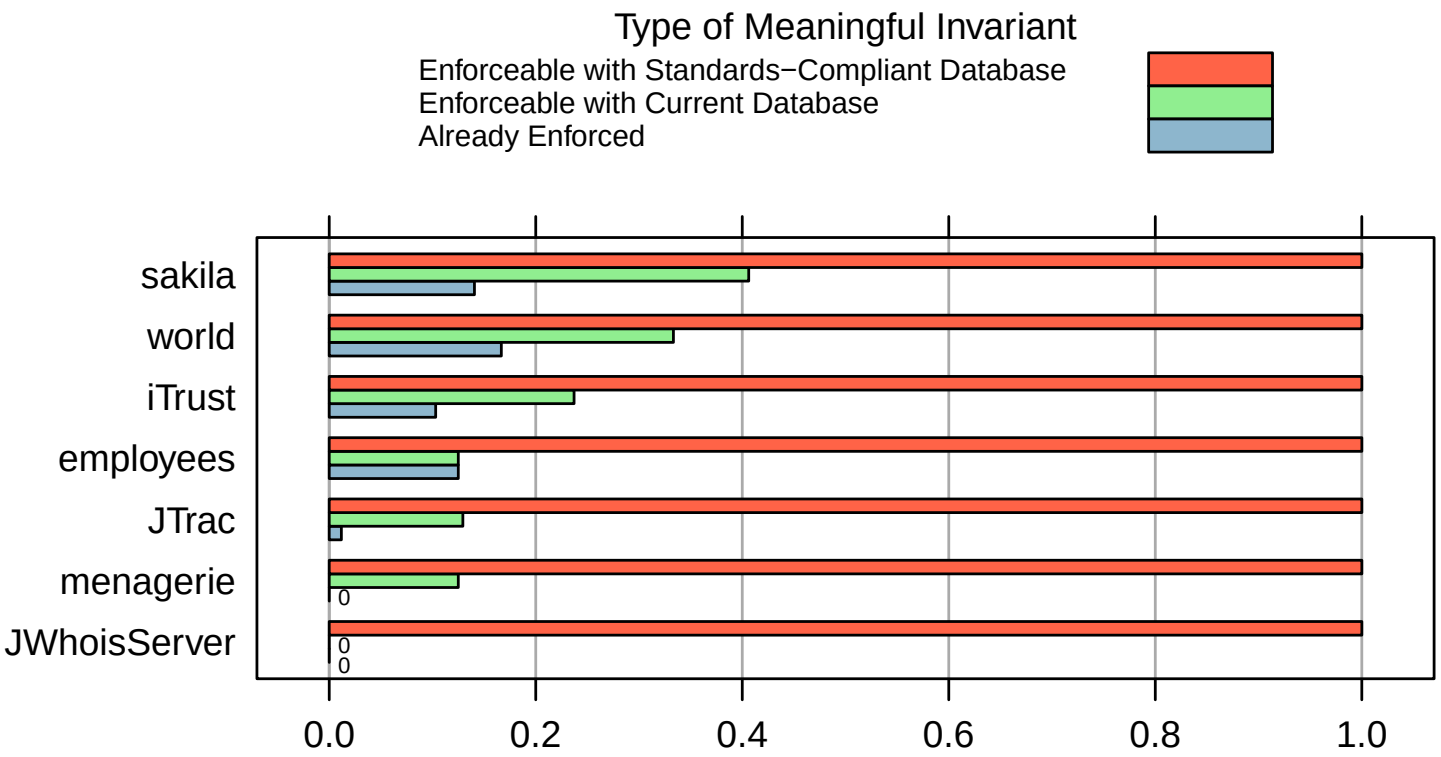
Schema Modification

Schema Enforceable

Invariant	Schema	Modification
`isnull(message.message) != null`	`TEXT`	`NOT NULL`
`isnull(film_text.description) != null`	`TEXT`	`NOT NULL`
`isnull(history.time_stamp) != null`	`DATETIME`	`NOT NULL`
`user_space_roles.user_id >= 1`	`BIGINT(20)`	`UNSIGNED`
`pet.sex one of { "f", "m" }`	`CHAR(1)`	`ENUM('m','f')`
`country.Population >= 0`	`INT(11)`	`UNSIGNED`
`isnull(titles.to_date) != null`	`DATE`	`NOT NULL`

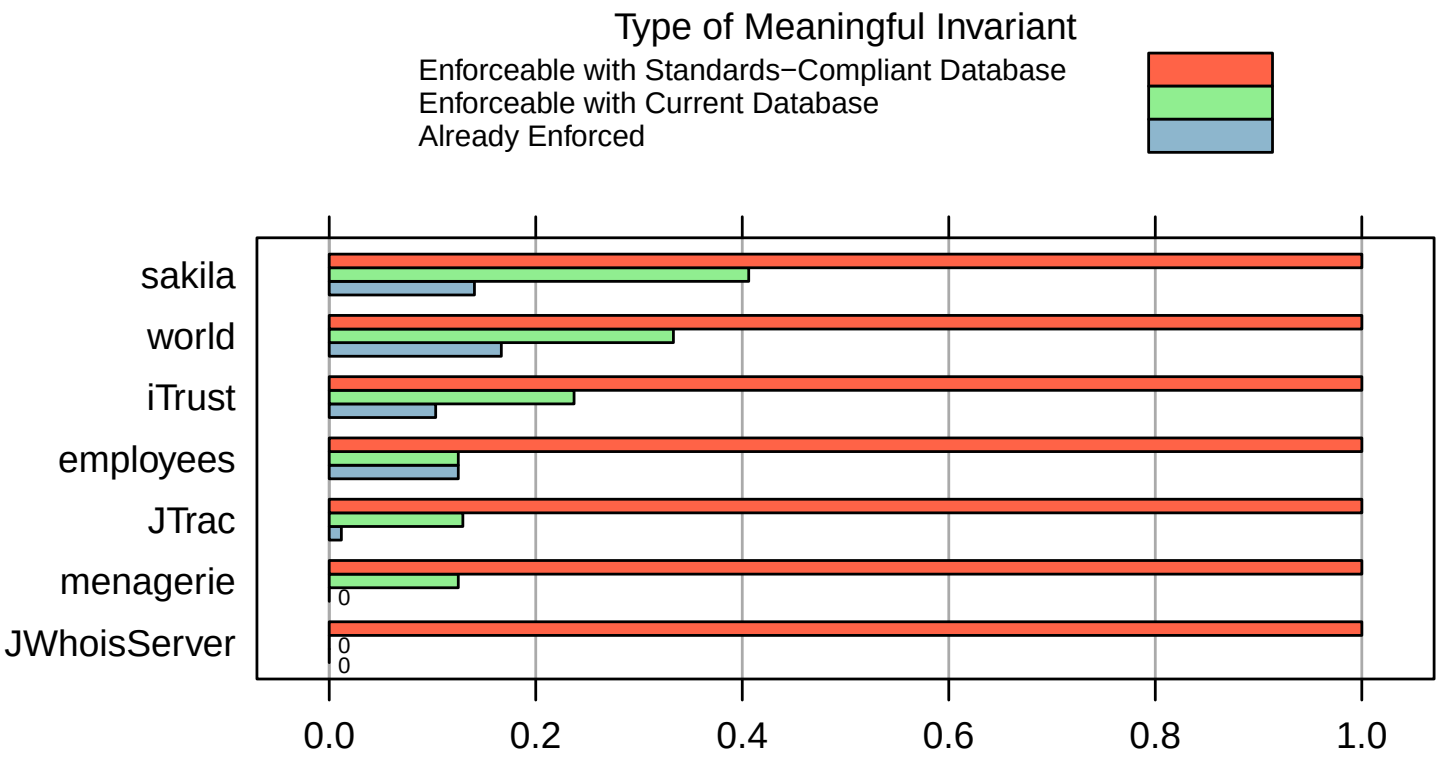
Many detected constraints can easily be added to the schema

Schema Modification



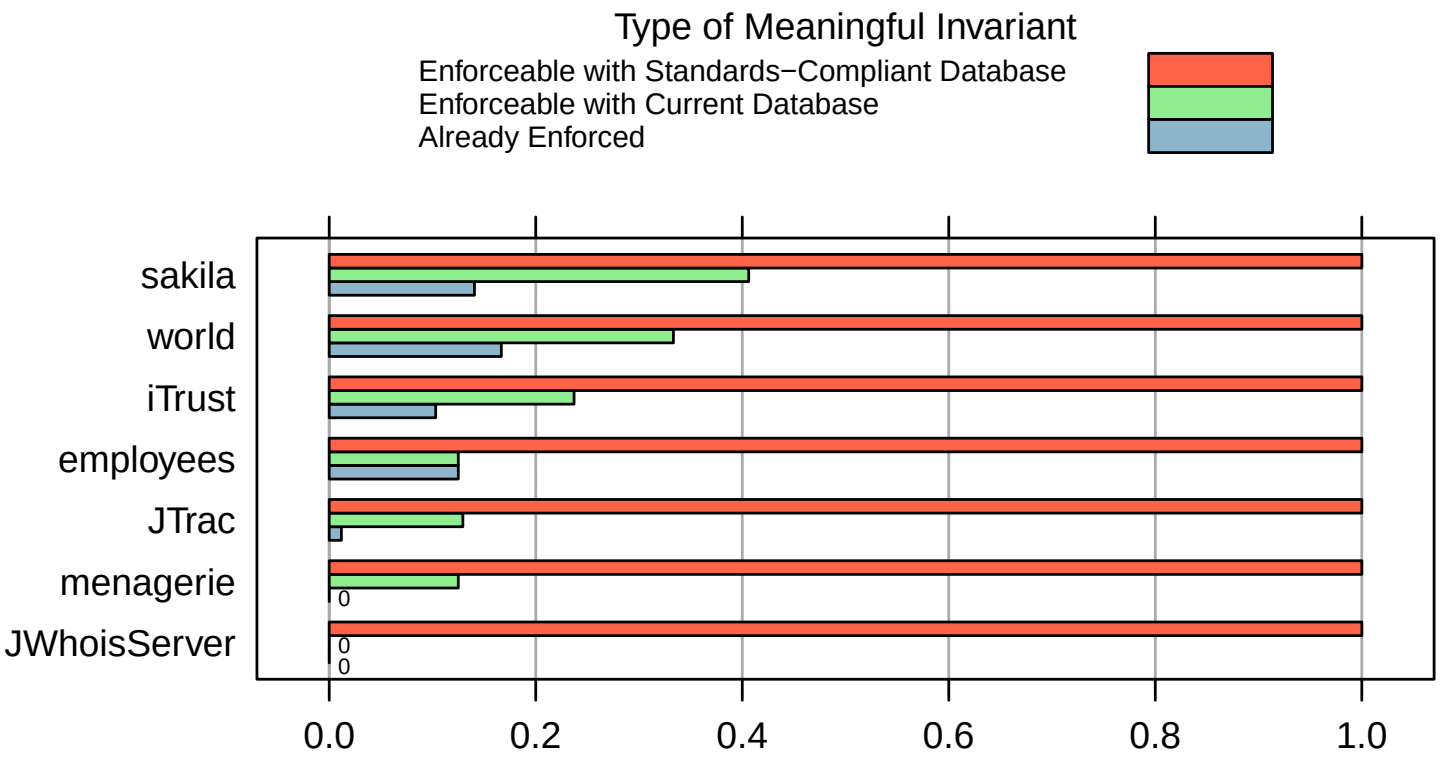
Percentages relative to the total number of non-spurious invariants

Schema Modification



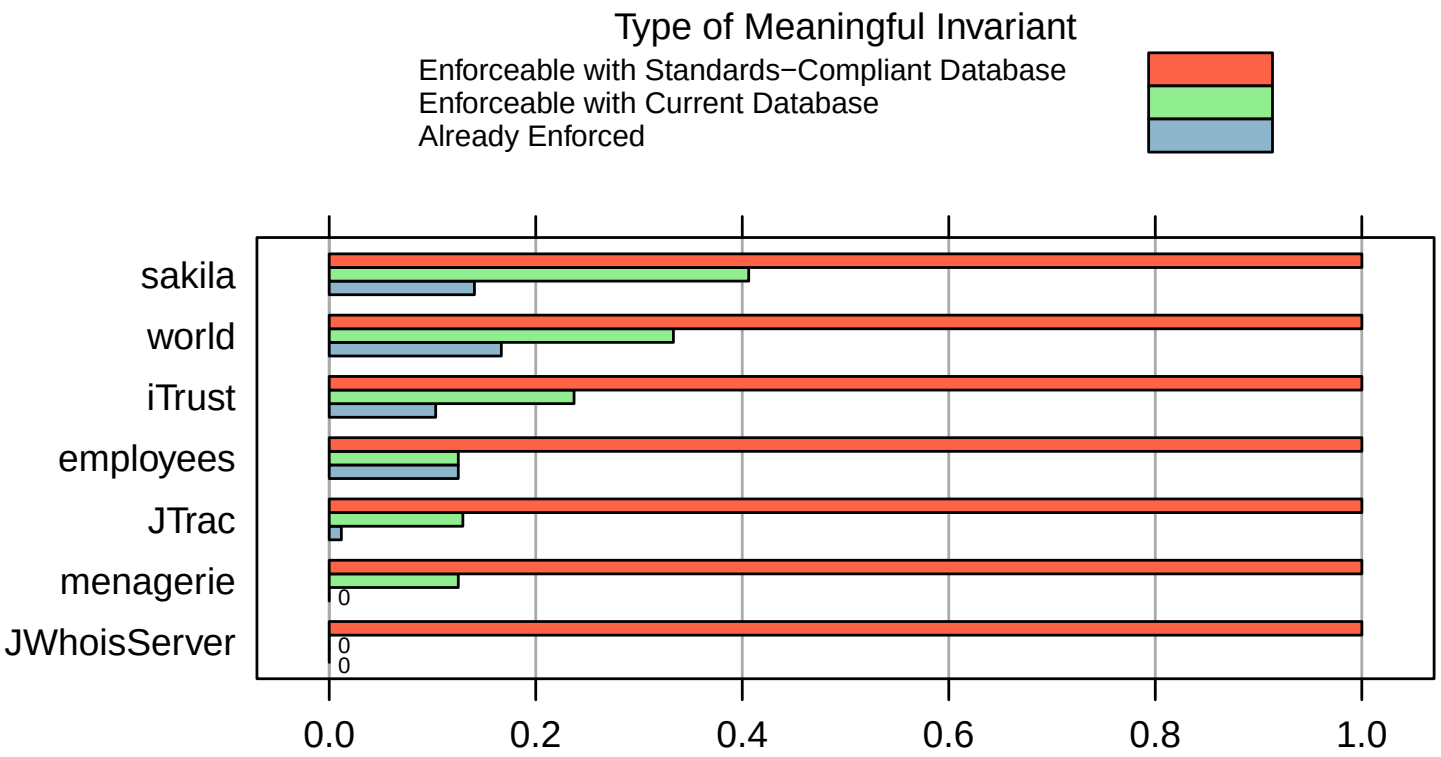
All constraints enforceable by a standards-compliant database

Schema Modification



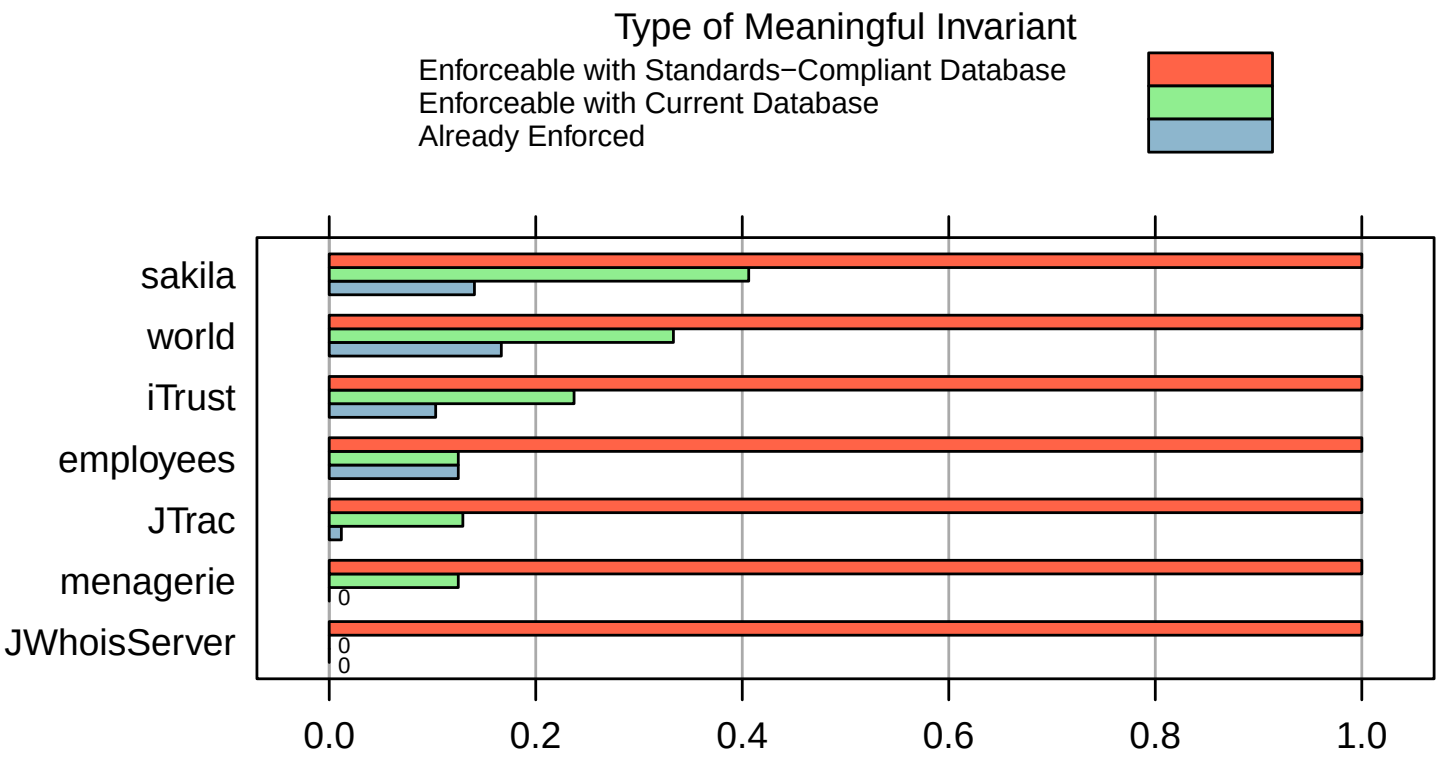
Three schemas can be enhanced with many new constraints

Schema Modification



menagerie did not already enforce any of the meaningful invariants

Schema Modification



JWhoisServer's MySQL doesn't support constraint enforcement

Empirical Conclusions and Future Research

Conclusions

- Meaningful invariants may be mined from both relational databases and database applications
- Invariant quality depends on existence of diverse data
- Data integrity may be enhanced by using invariants for modification of the database's schema

Future Research

- Invariants between multiple tables
- Invariants for individual queries
- Explore additional client applications

Empirical Conclusions and Future Research

Conclusions

- Meaningful invariants may be mined from both relational databases and database applications
- Invariant quality depends on existence of diverse data
- Data integrity may be enhanced by using invariants for modification of the database's schema

Future Research

- Invariants between multiple tables
- Invariants for individual queries
- Explore additional client applications

Empirical Conclusions and Future Research

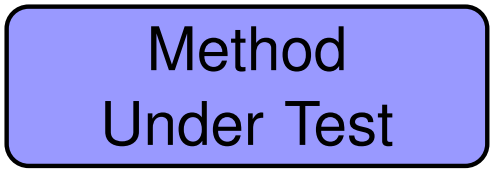
Conclusions

- Meaningful invariants may be mined from both relational databases and database applications
- Invariant quality depends on existence of diverse data
- Data integrity may be enhanced by using invariants for modification of the database's schema

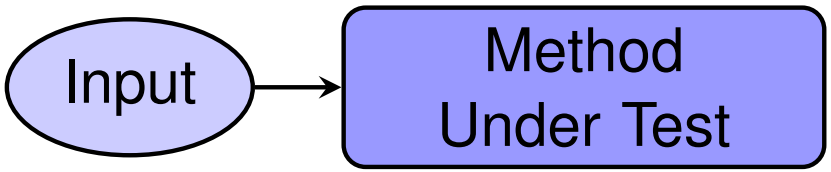
Future Research

- Invariants between multiple tables
- Invariants for individual queries
- Explore additional client applications

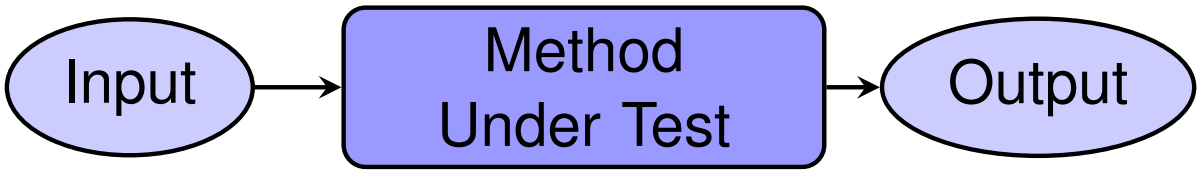
What is a Test Case?



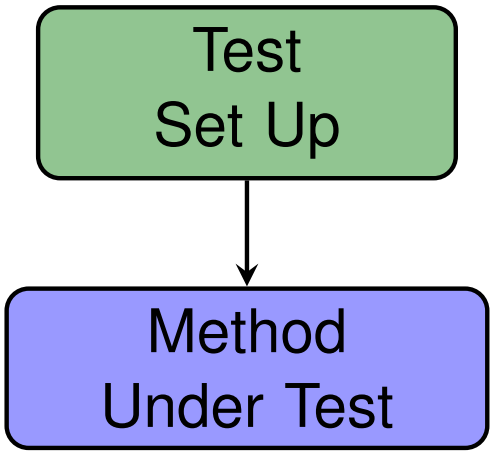
What is a Test Case?



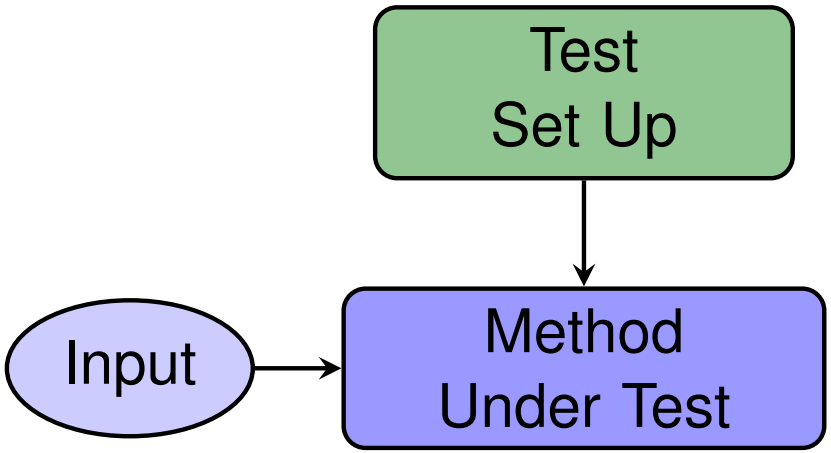
What is a Test Case?



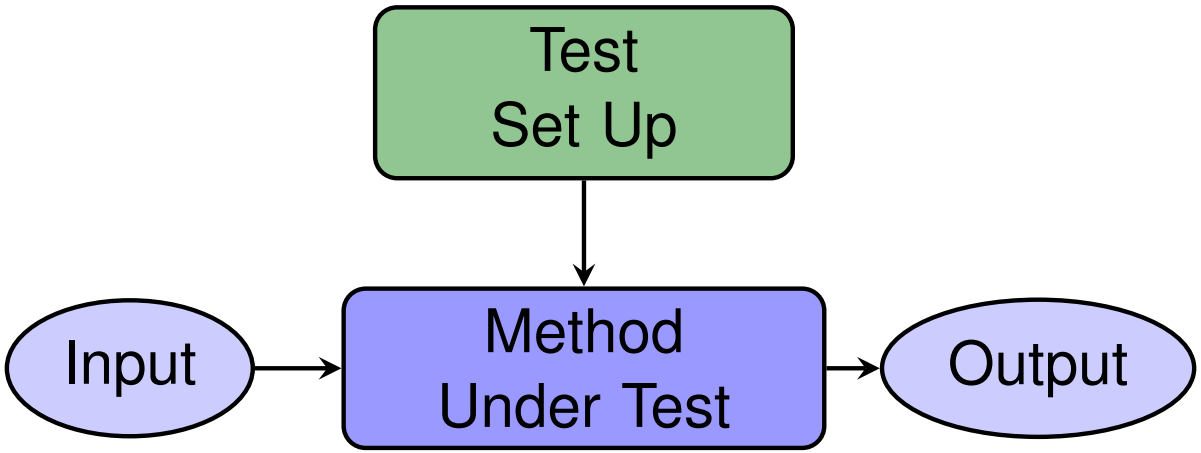
What is a Test Case?



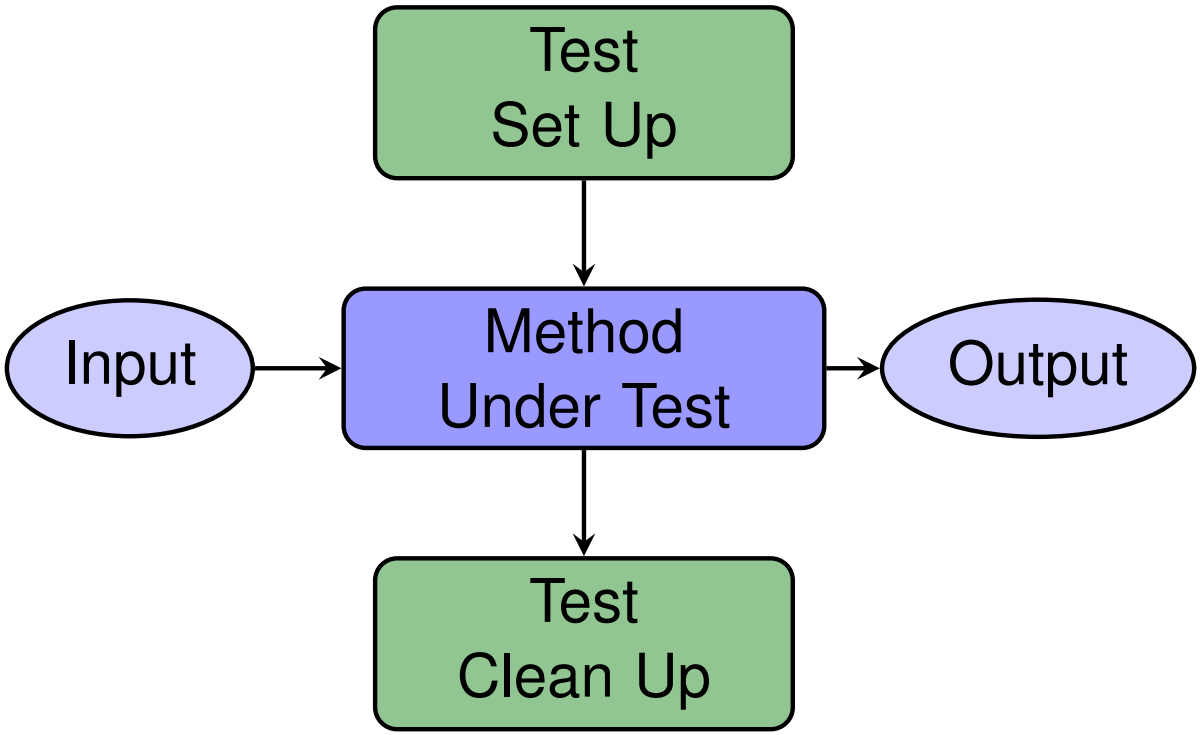
What is a Test Case?



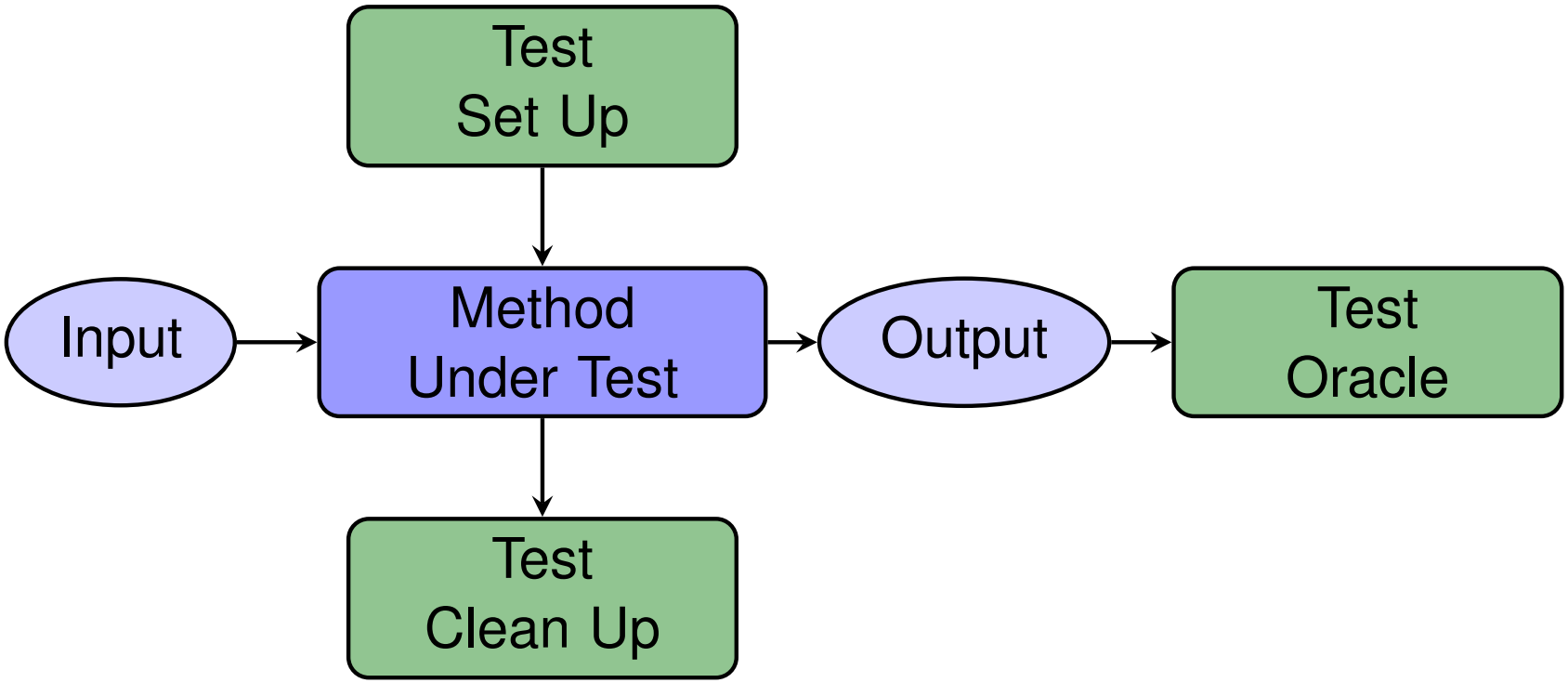
What is a Test Case?



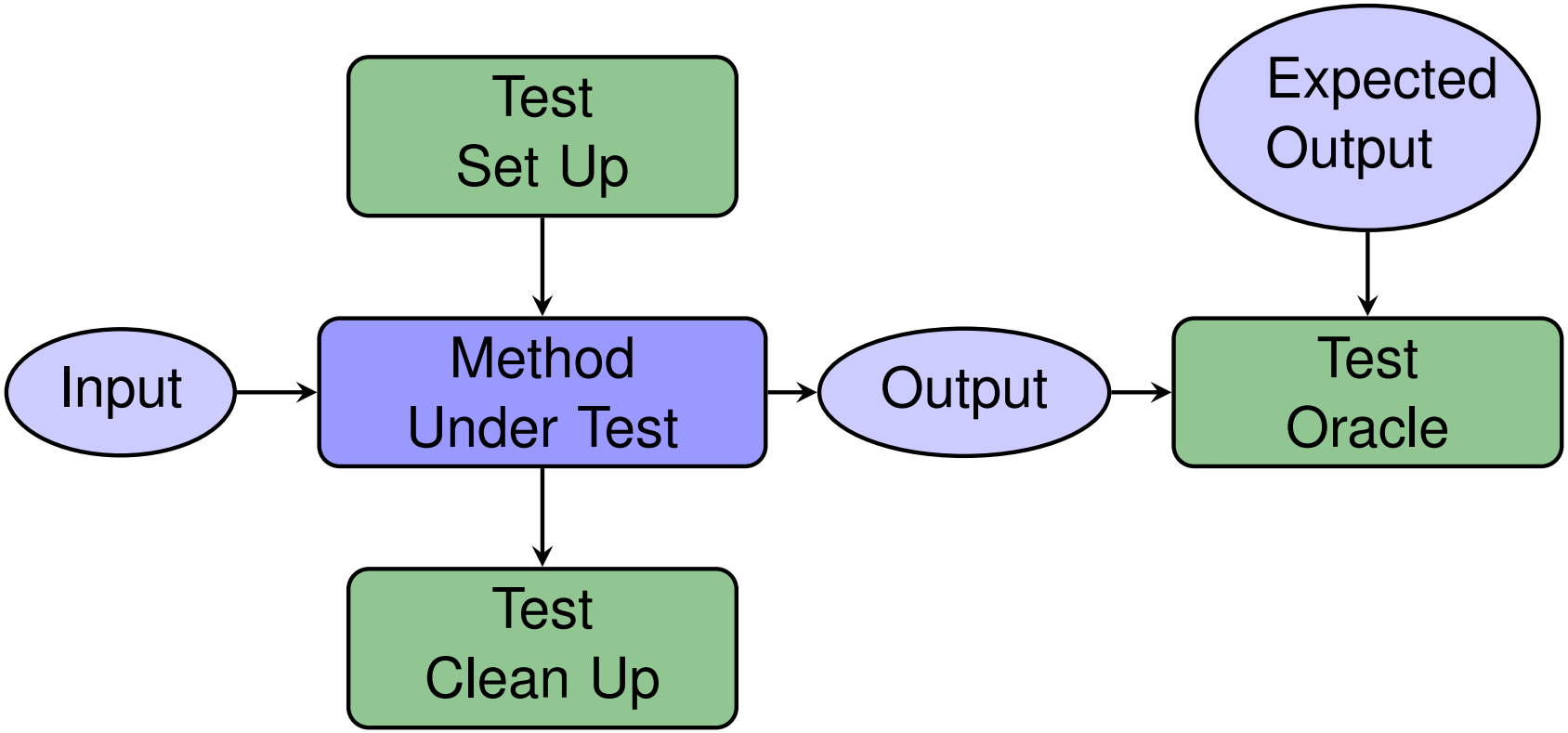
What is a Test Case?



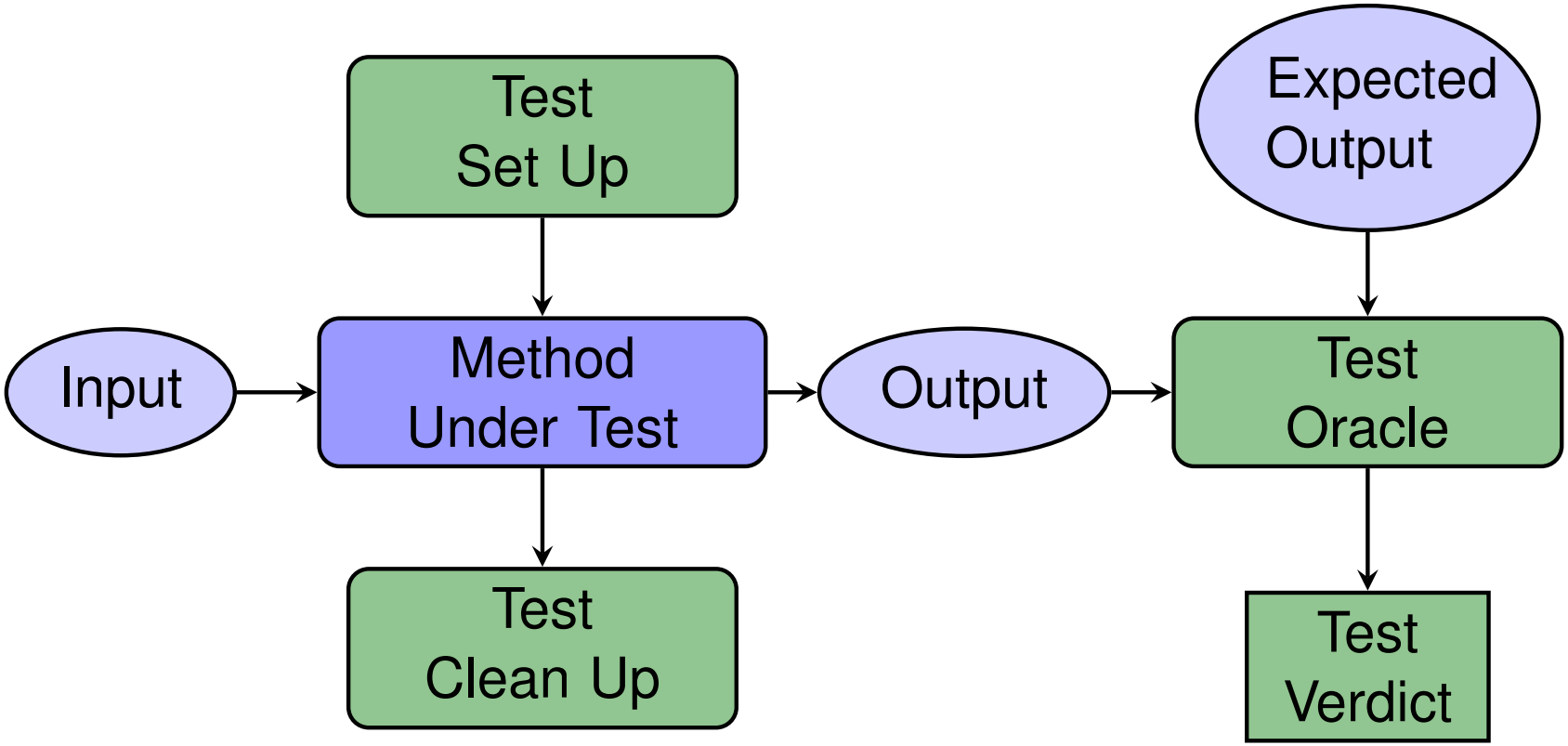
What is a Test Case?



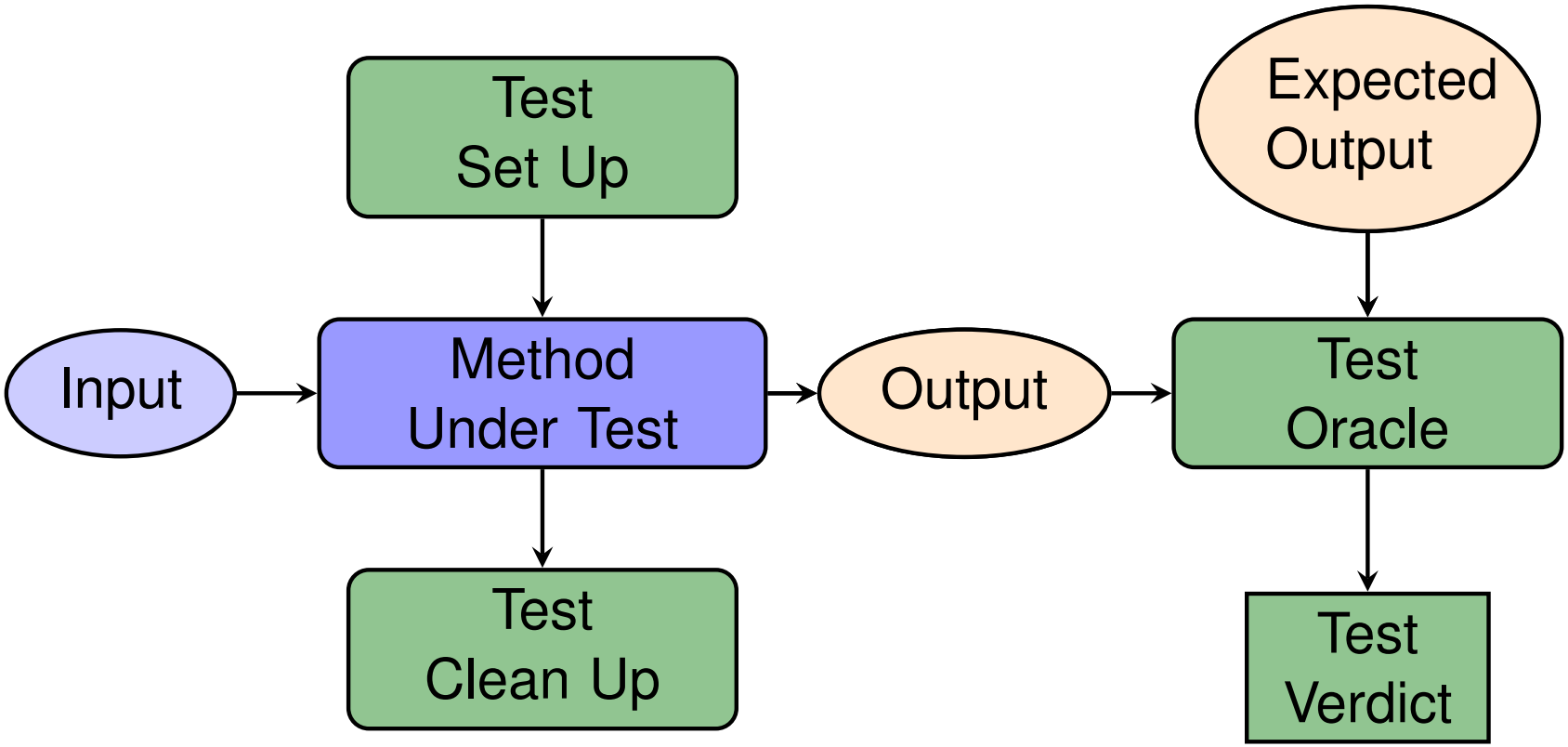
What is a Test Case?



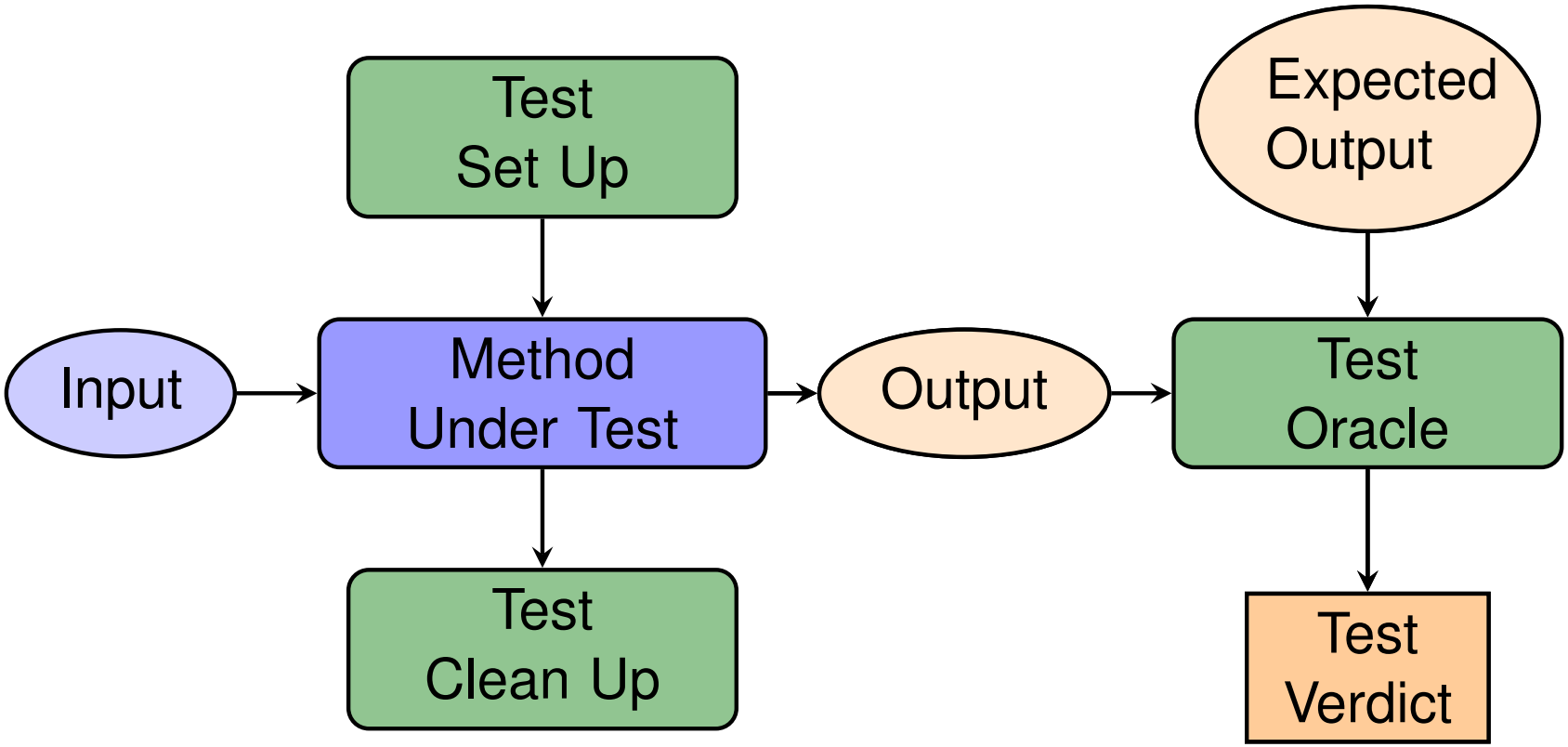
What is a Test Case?



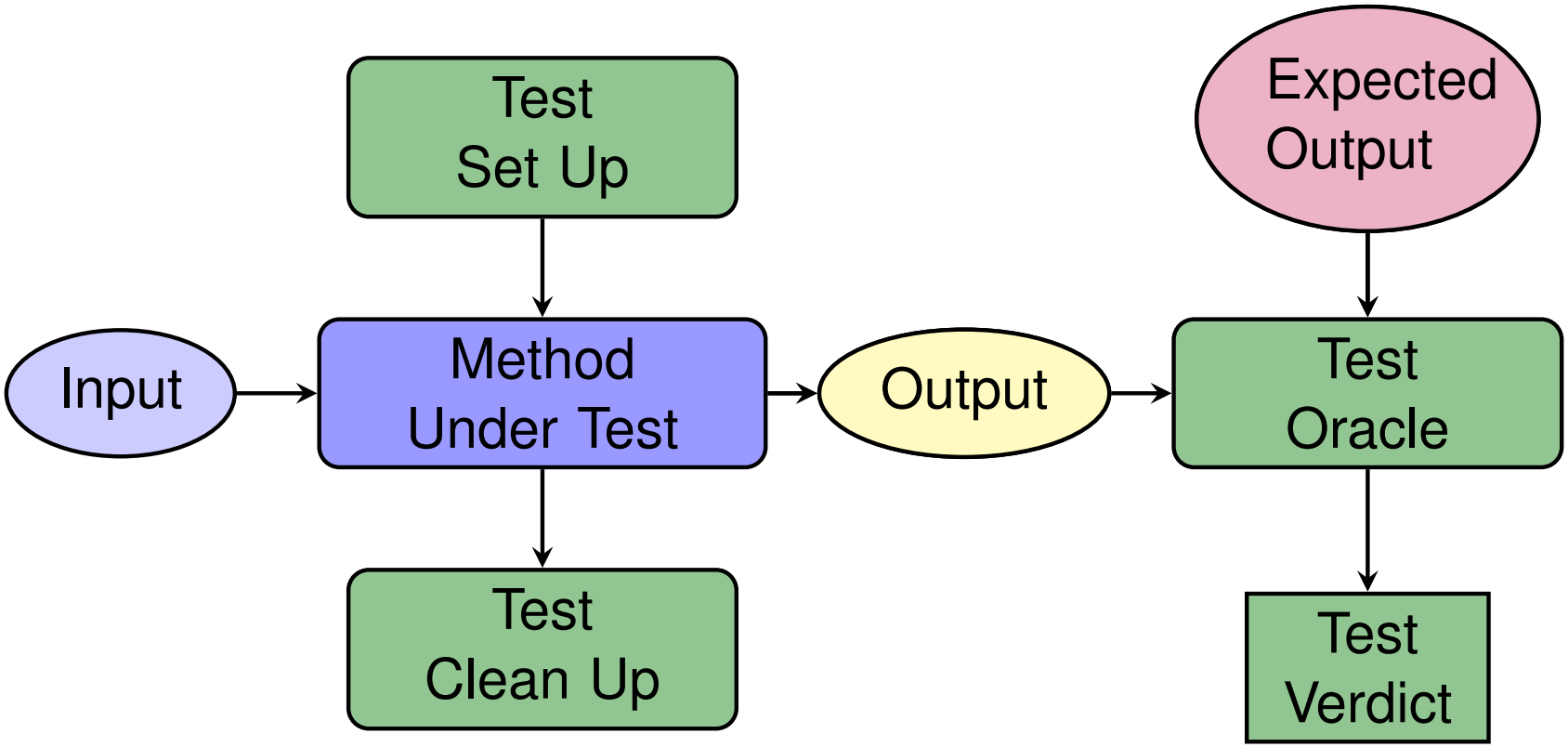
What is a Test Case?



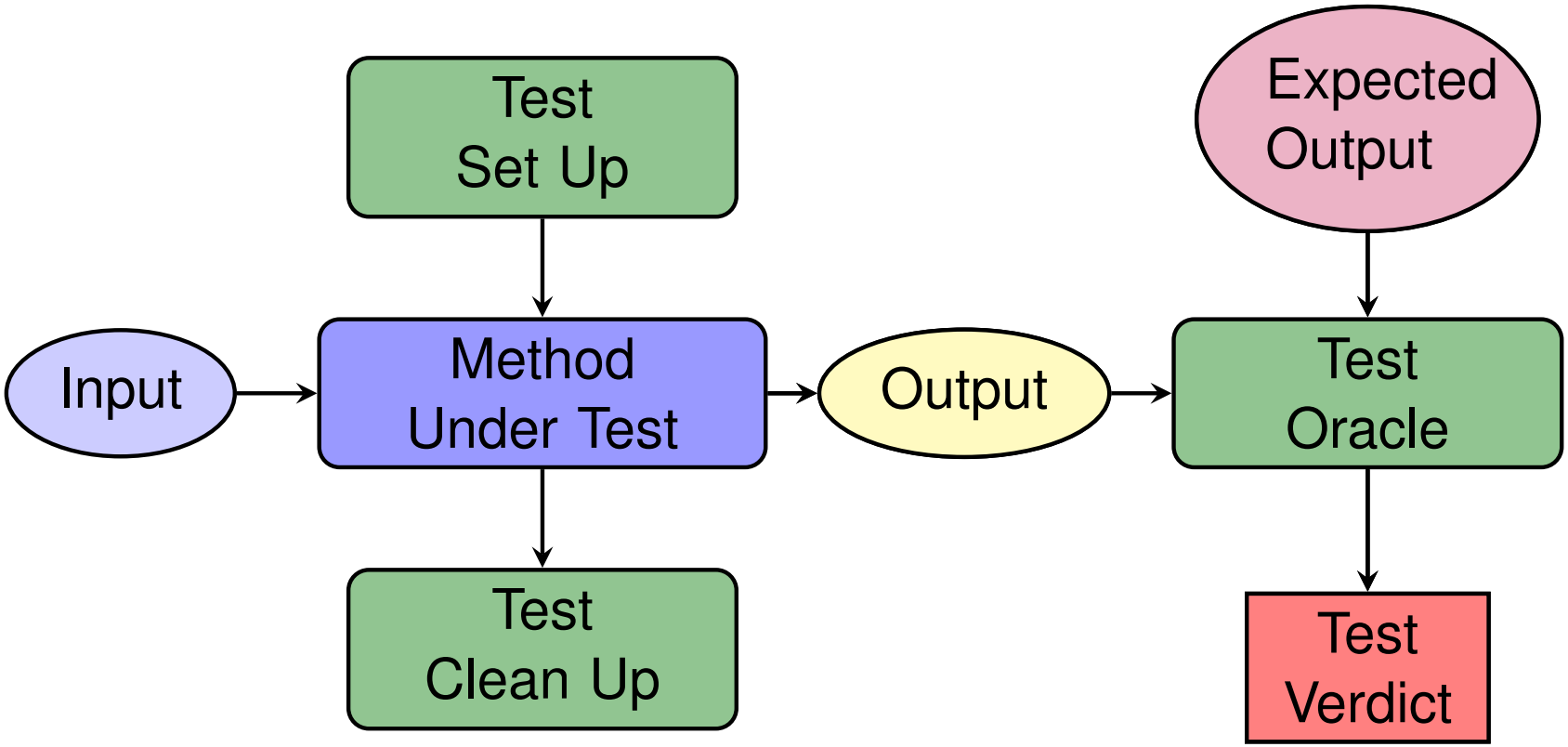
What is a Test Case?



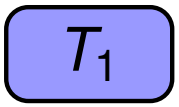
What is a Test Case?



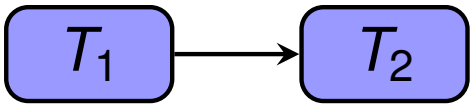
What is a Test Case?



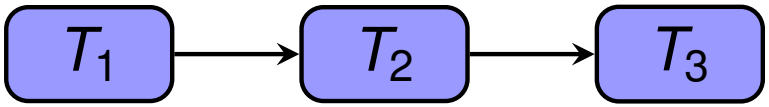
What is a Test Suite?



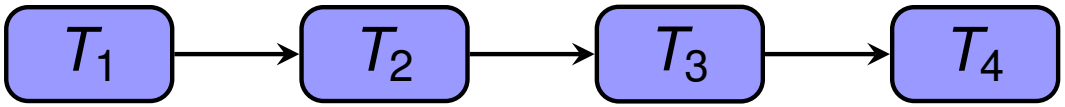
What is a Test Suite?



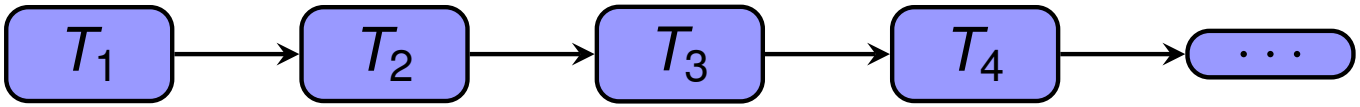
What is a Test Suite?



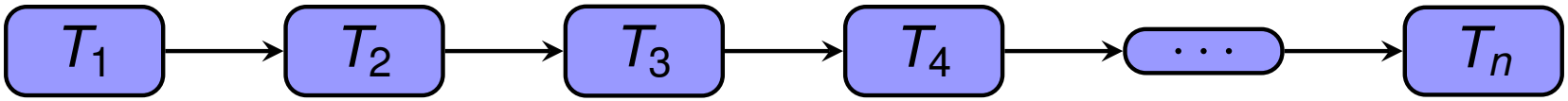
What is a Test Suite?



What is a Test Suite?

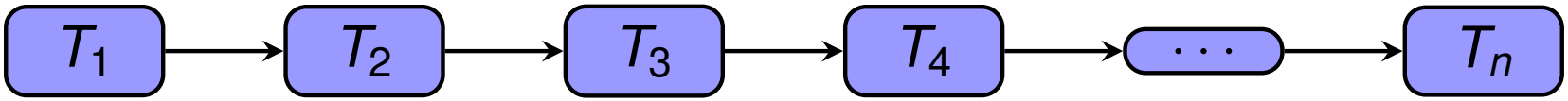


What is a Test Suite?



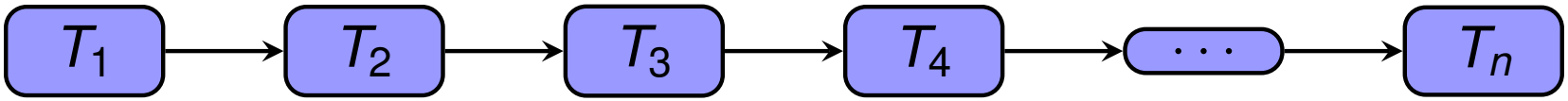
What is a Test Suite?

Organize the Test Cases into a Test Suite



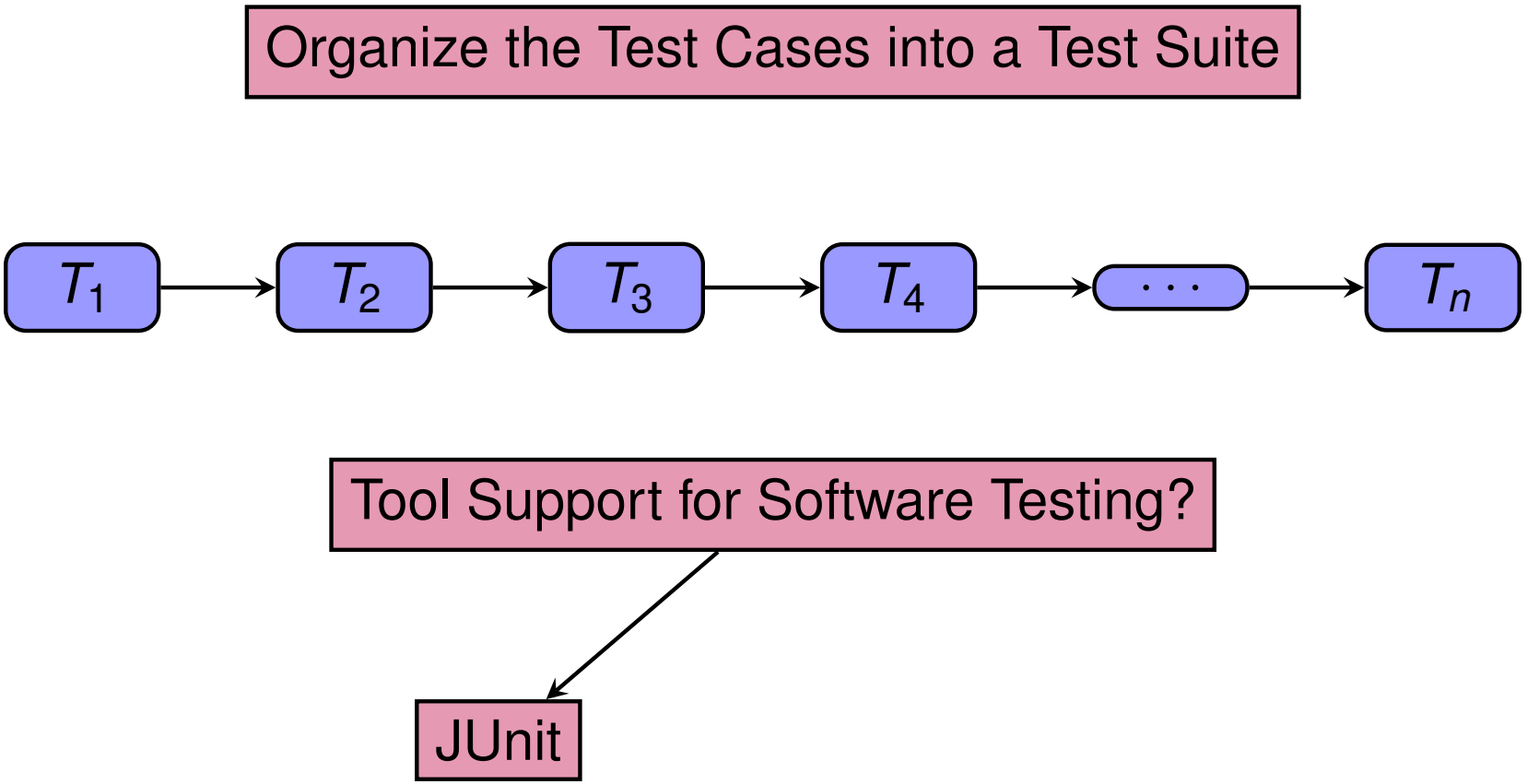
What is a Test Suite?

Organize the Test Cases into a Test Suite

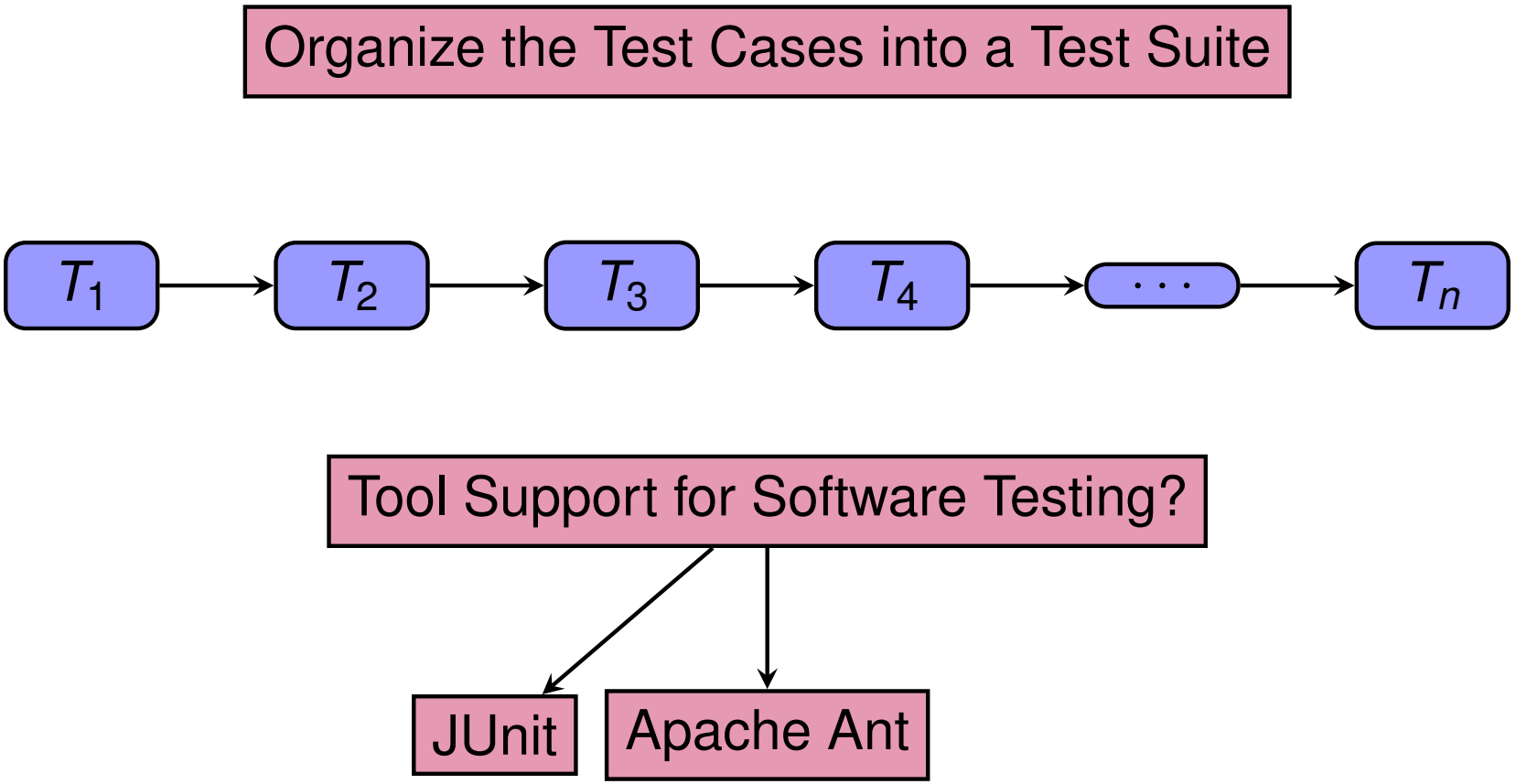


Tool Support for Software Testing?

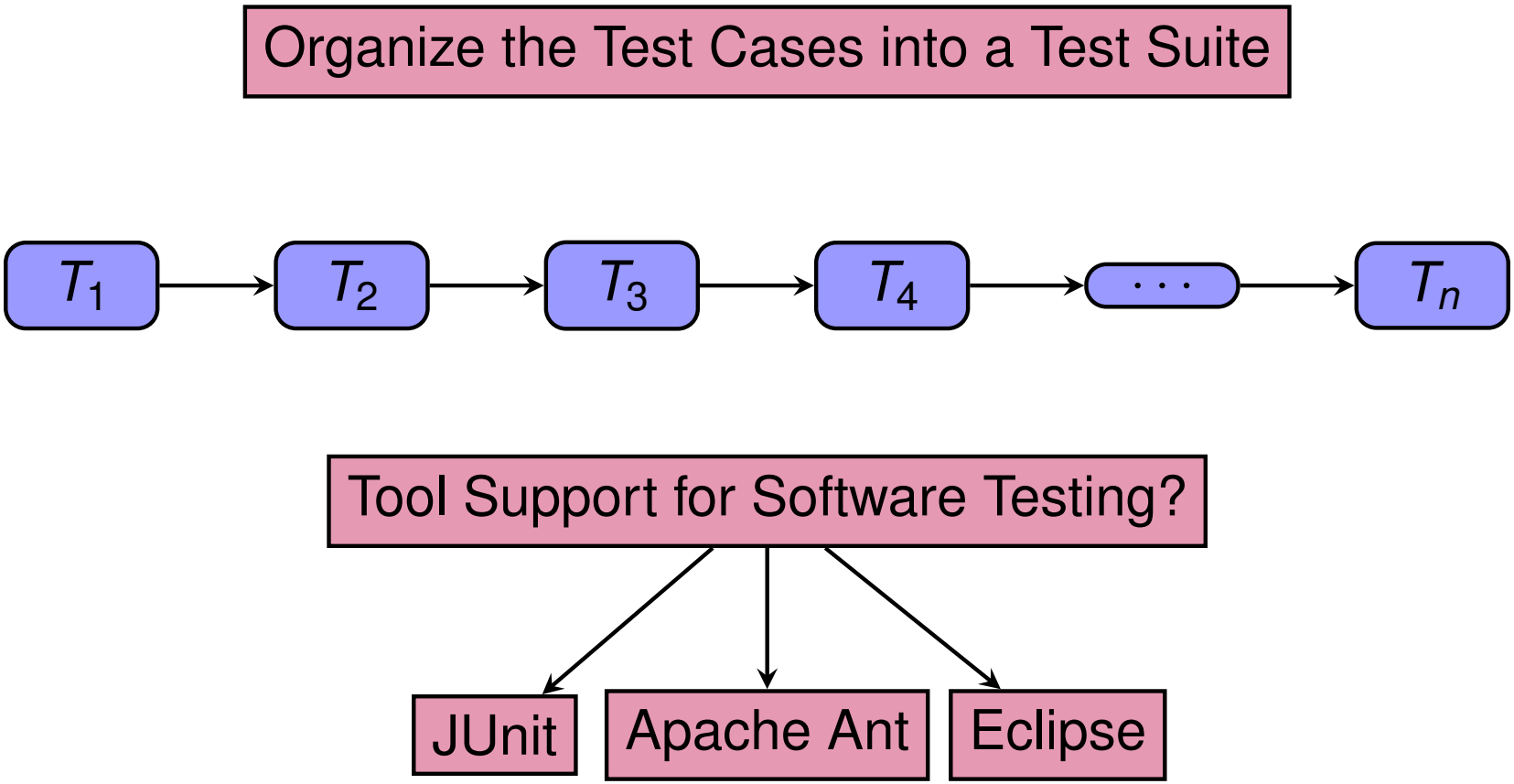
What is a Test Suite?



What is a Test Suite?

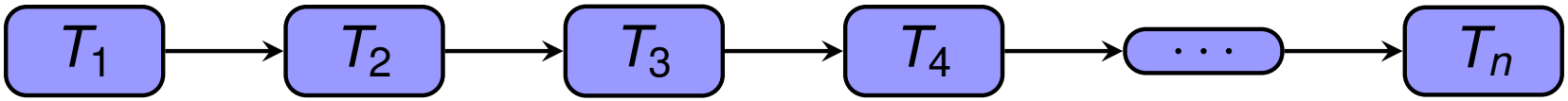


What is a Test Suite?

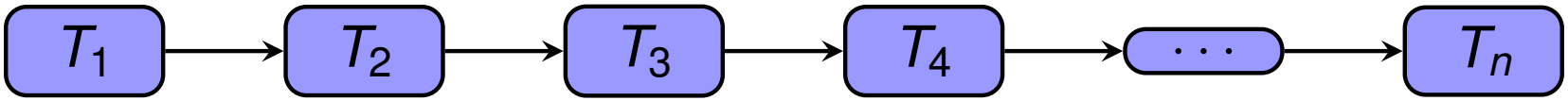


Test Suite Management

Organize the Test Cases into a Test Suite



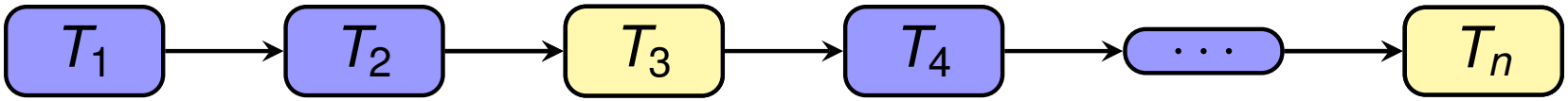
Test Suite Management



Regression Testing Technique

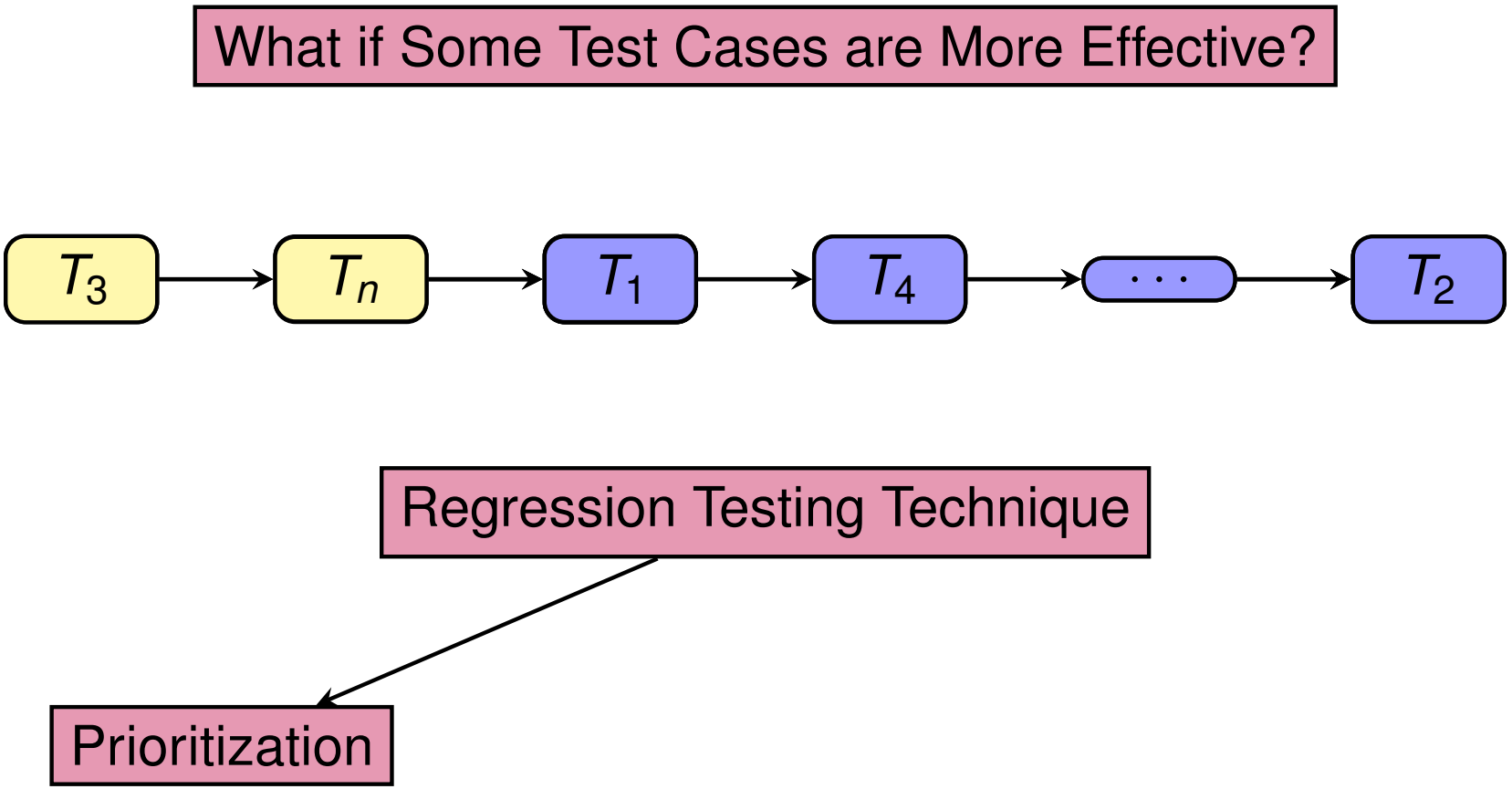
Test Suite Management

What if Some Test Cases are More Effective?

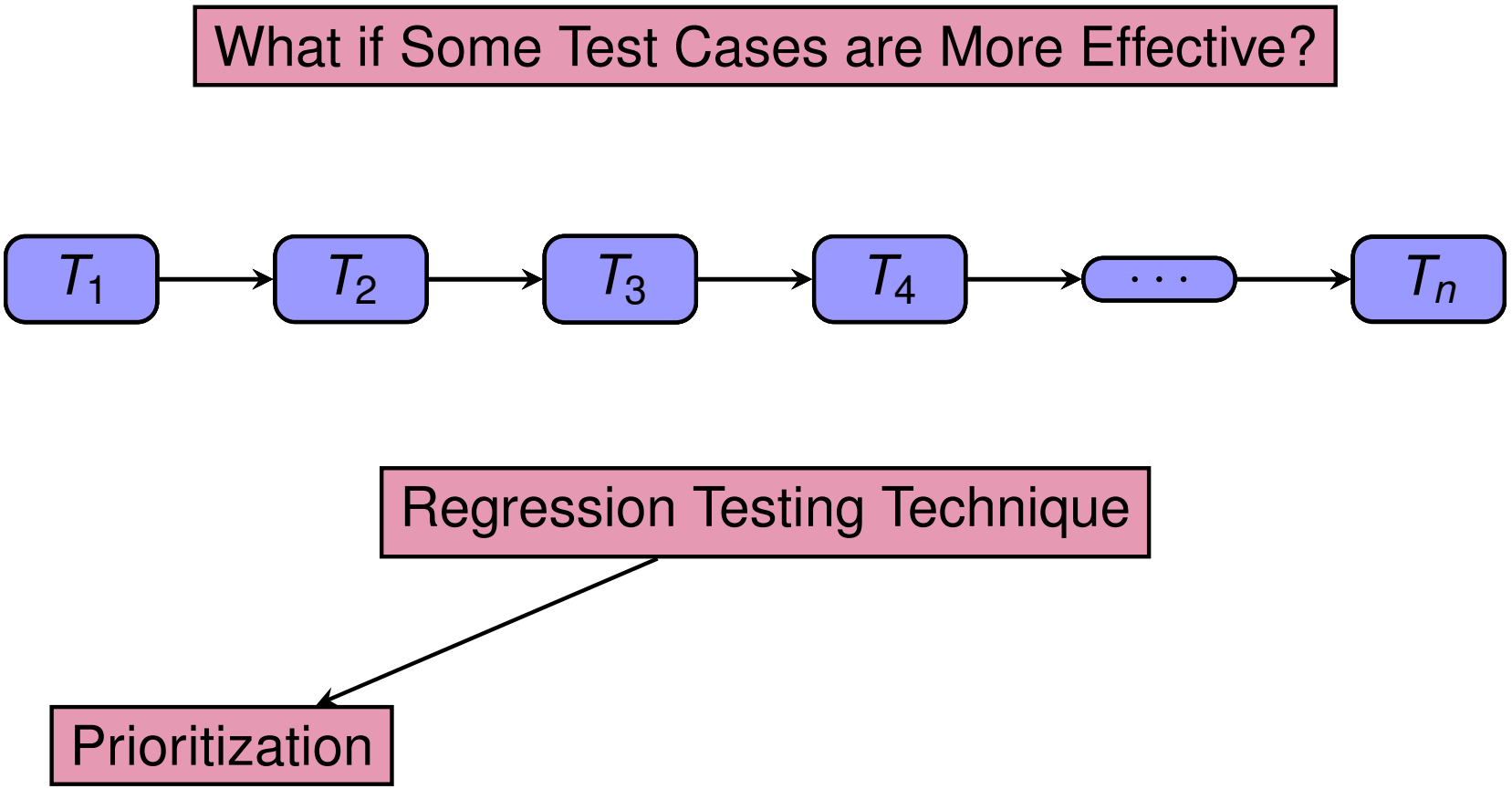


Regression Testing Technique

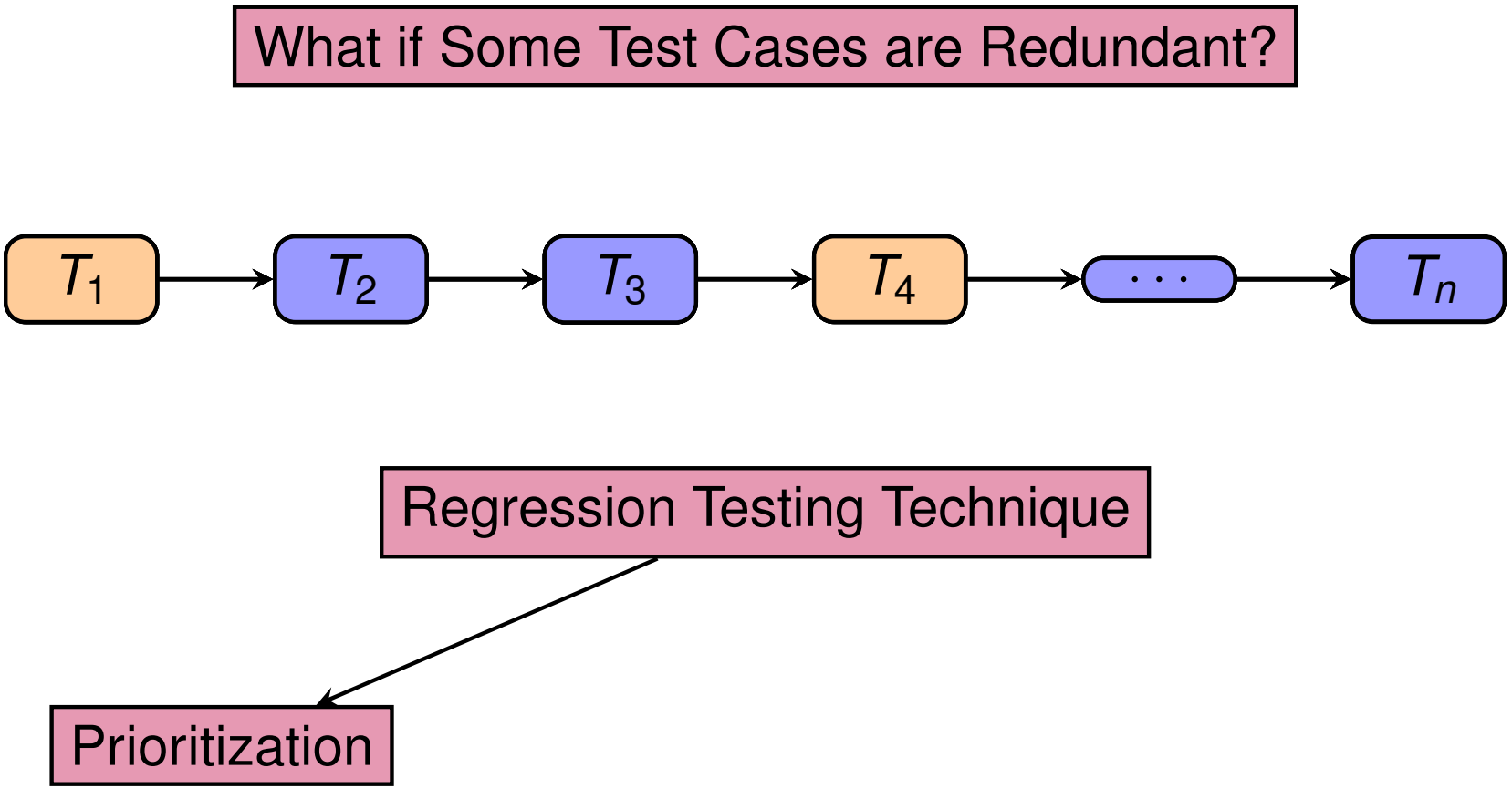
Test Suite Management



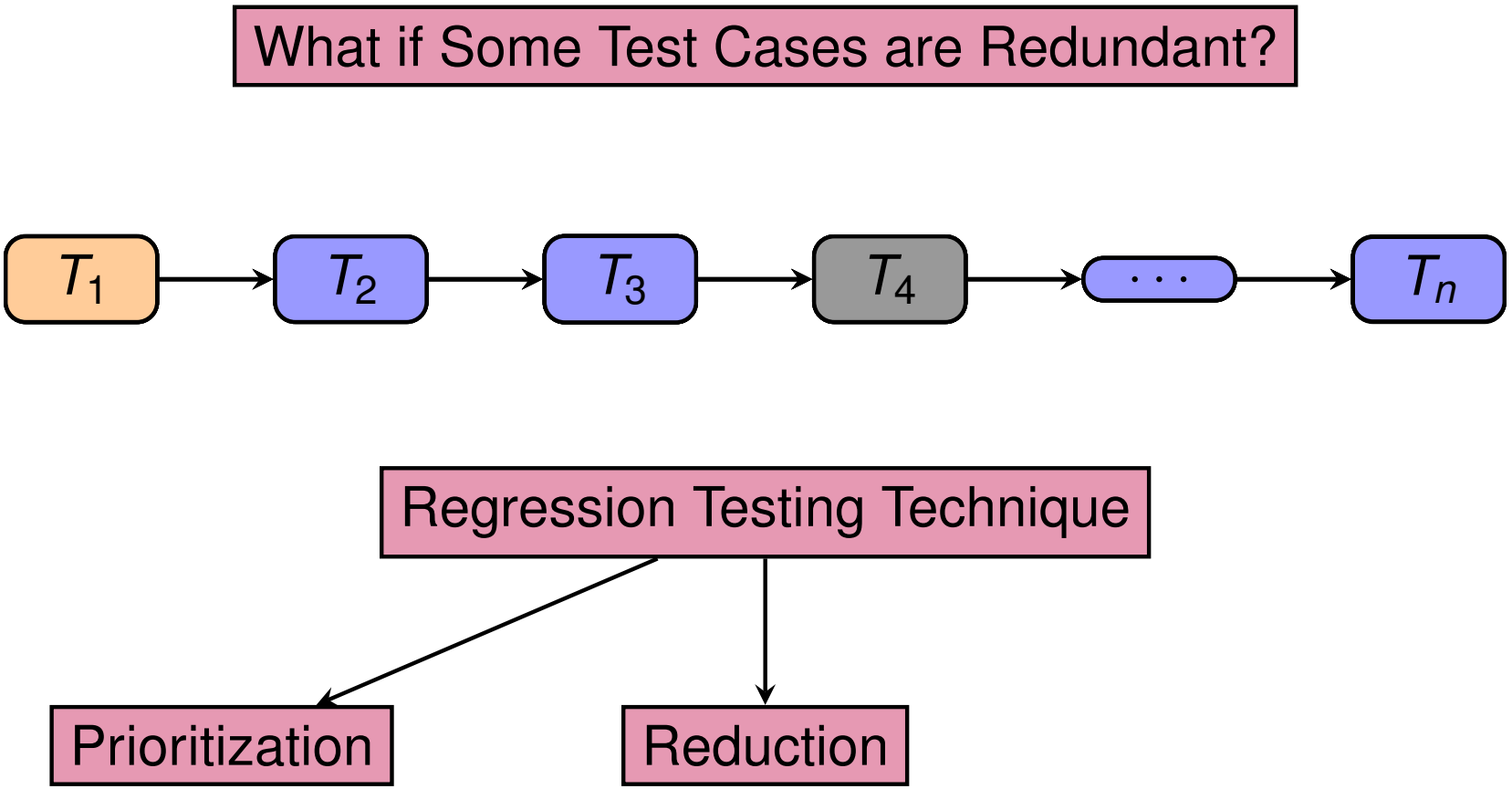
Test Suite Management



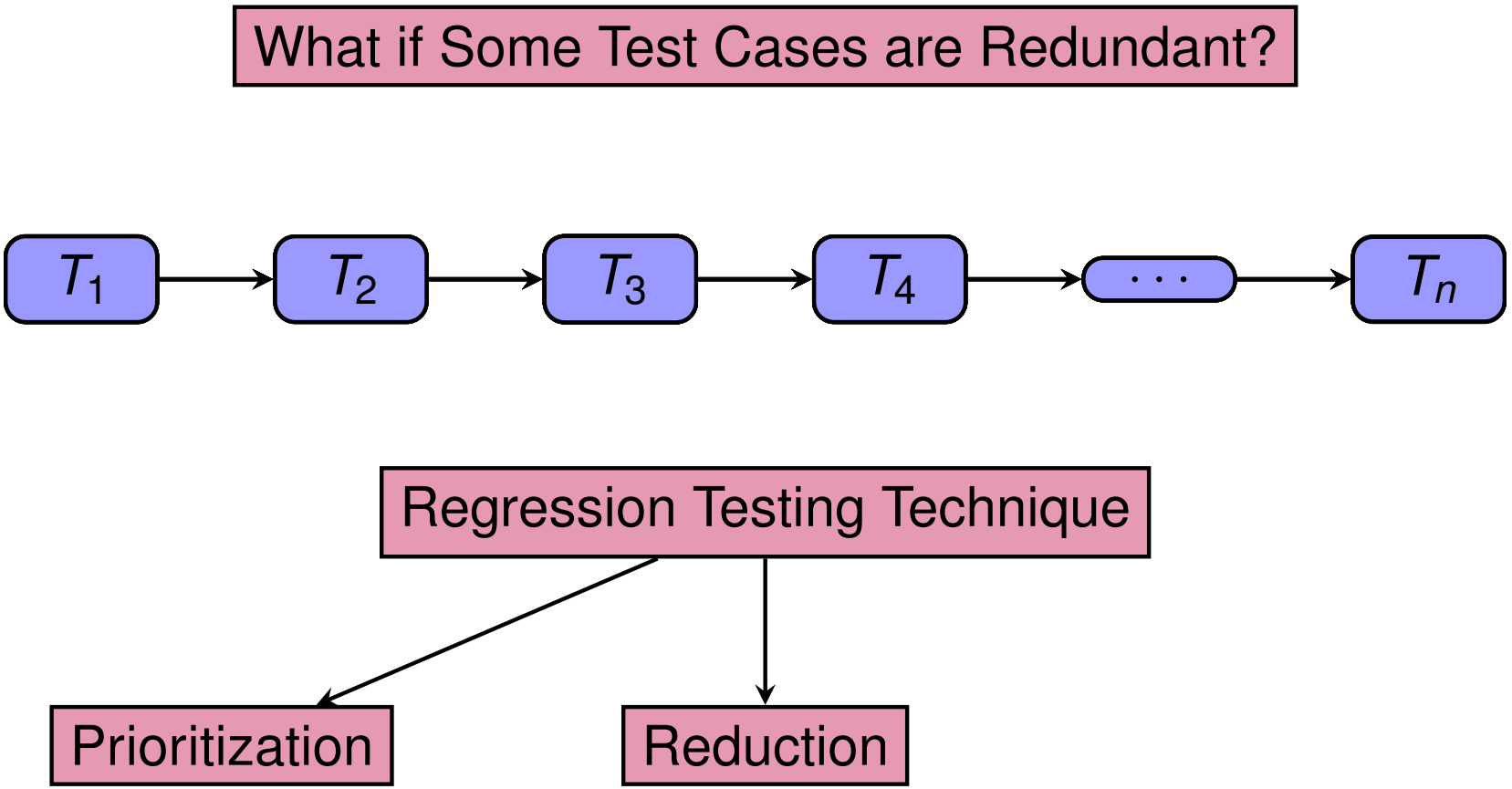
Test Suite Management



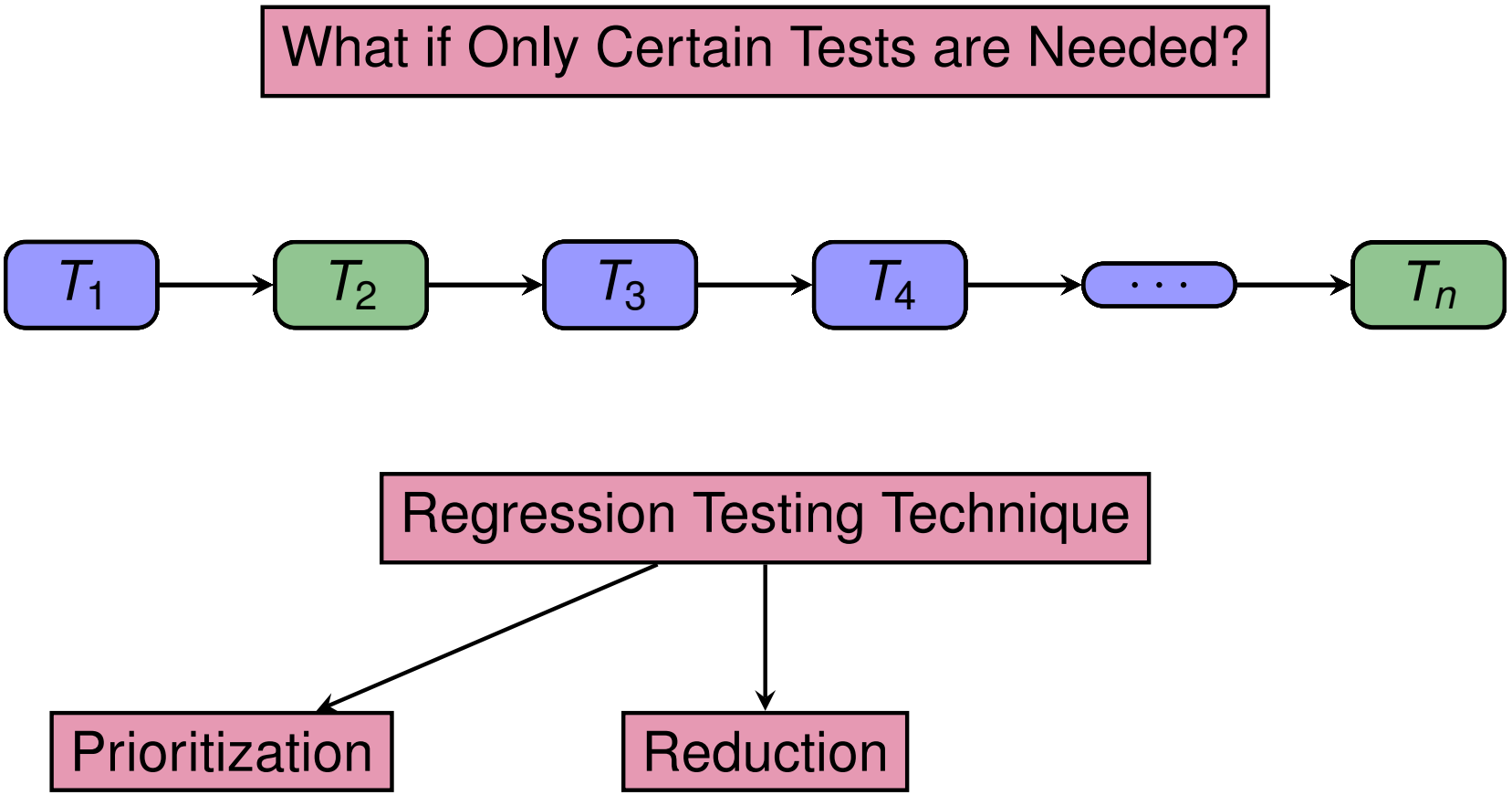
Test Suite Management



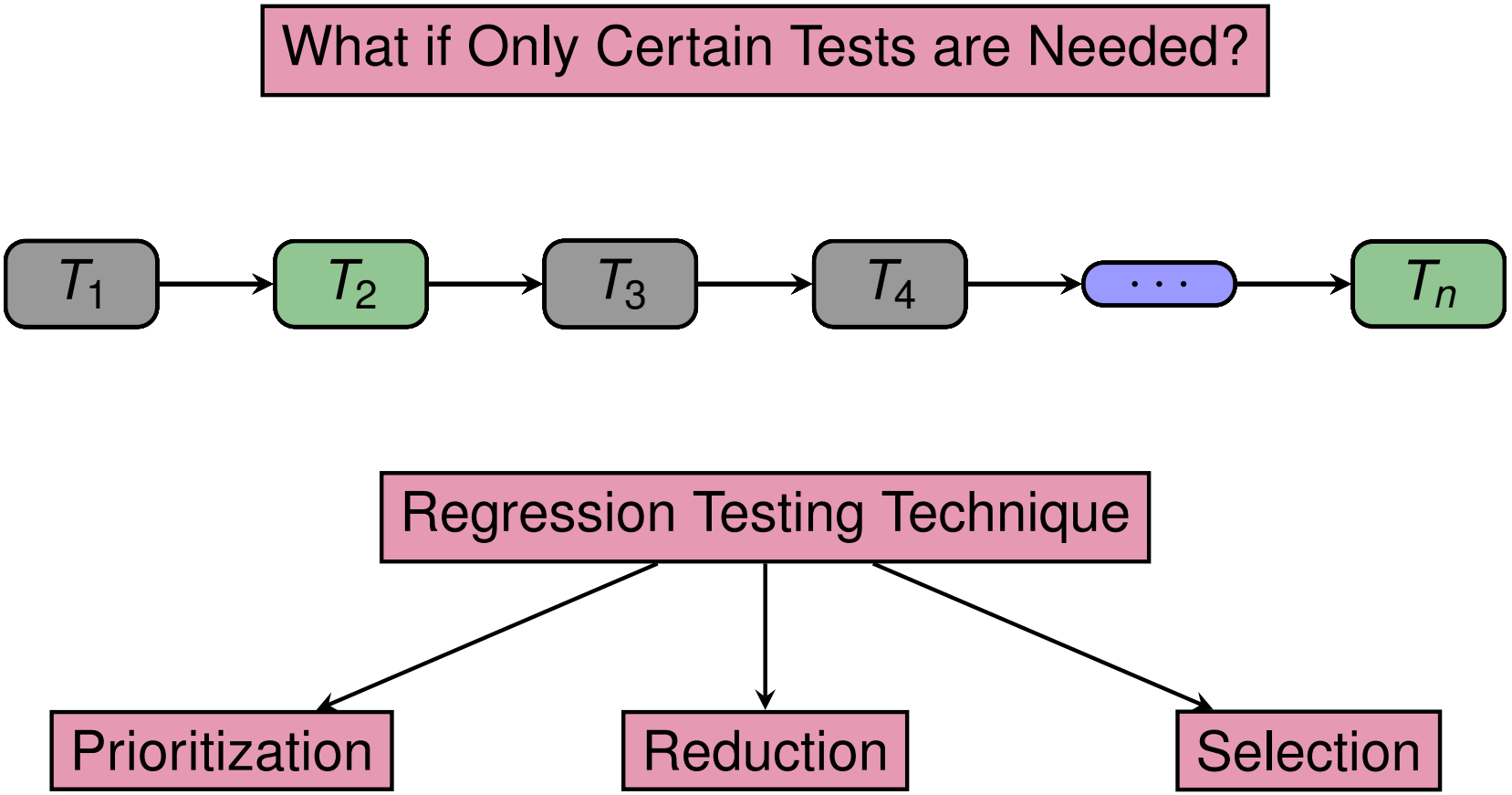
Test Suite Management



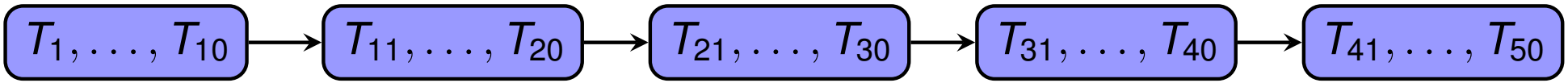
Test Suite Management



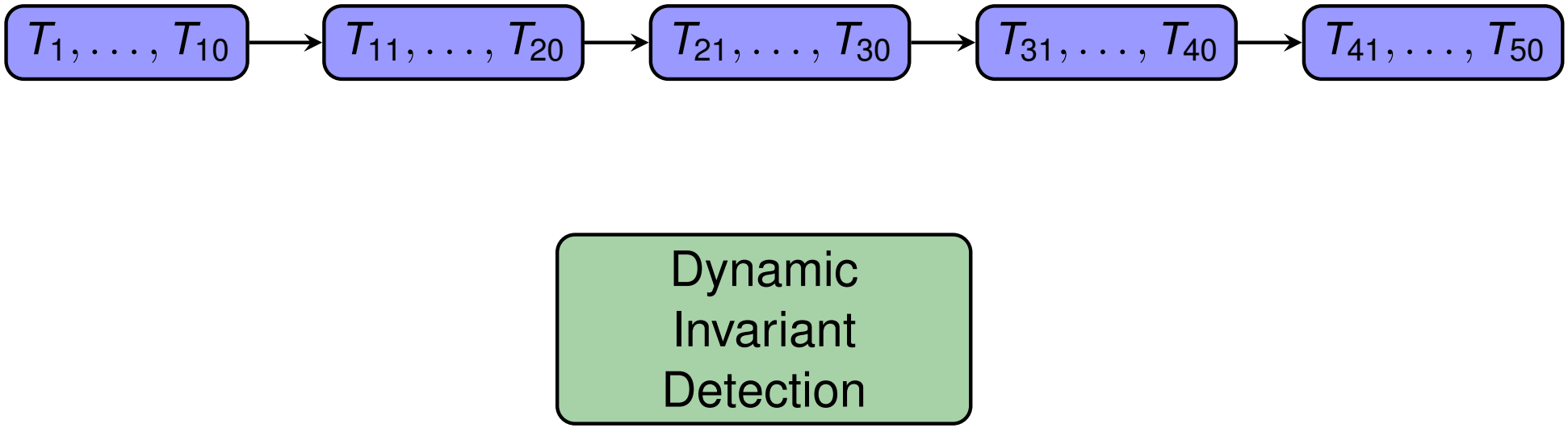
Test Suite Management



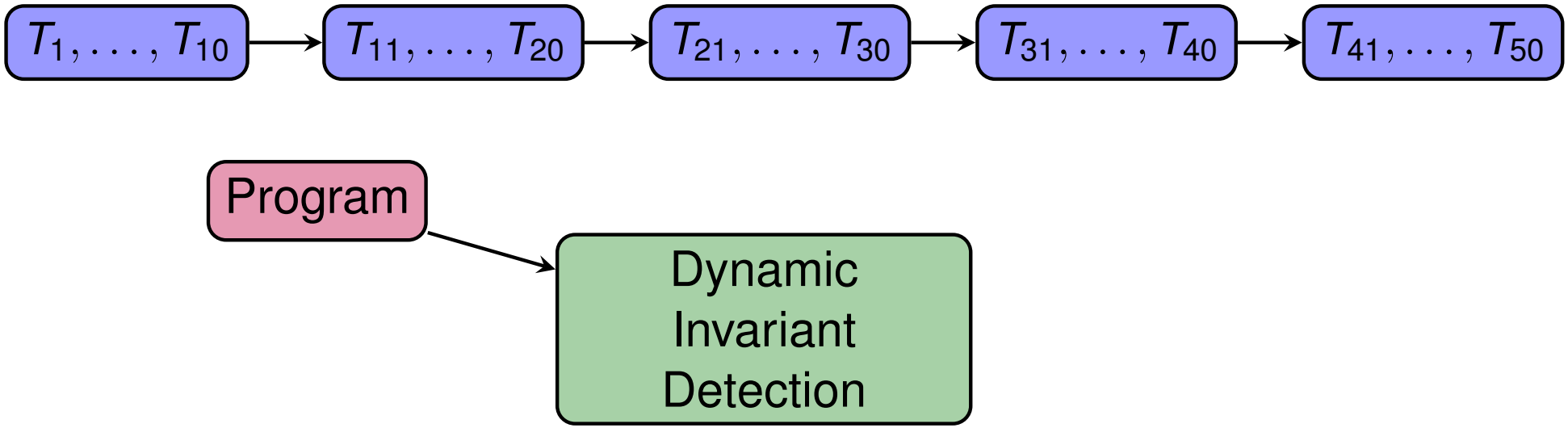
Database-Aware Test Suite Reduction



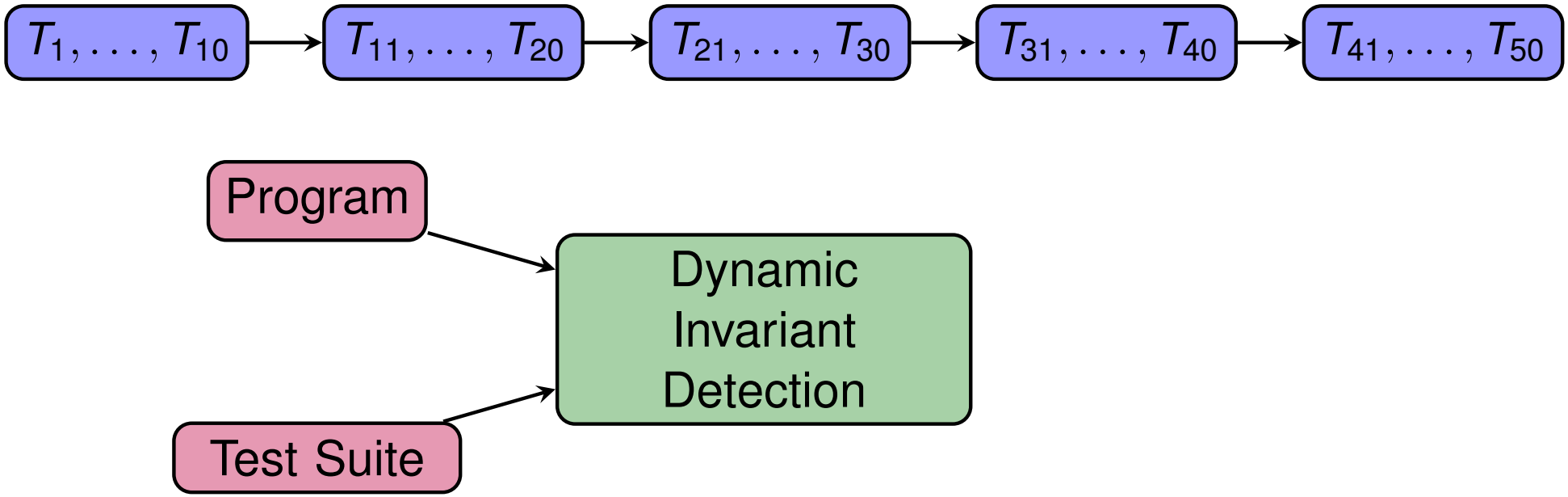
Database-Aware Test Suite Reduction



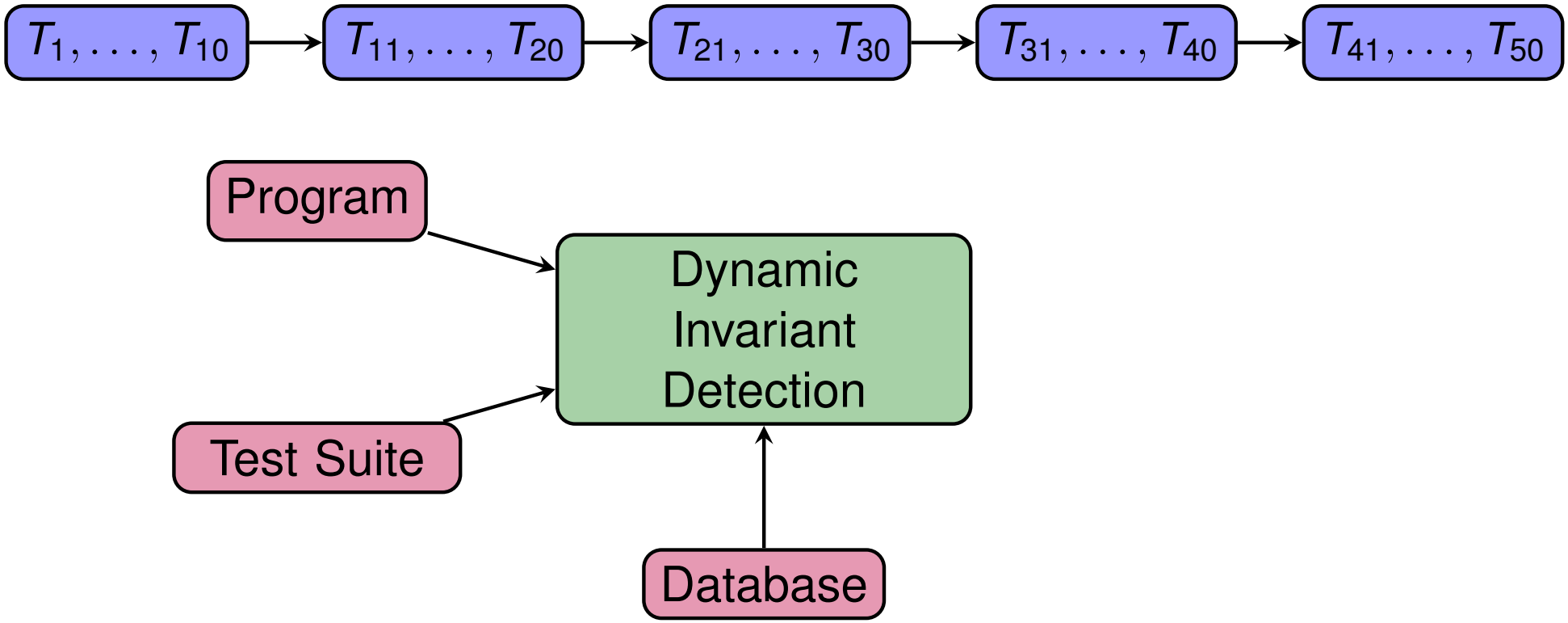
Database-Aware Test Suite Reduction



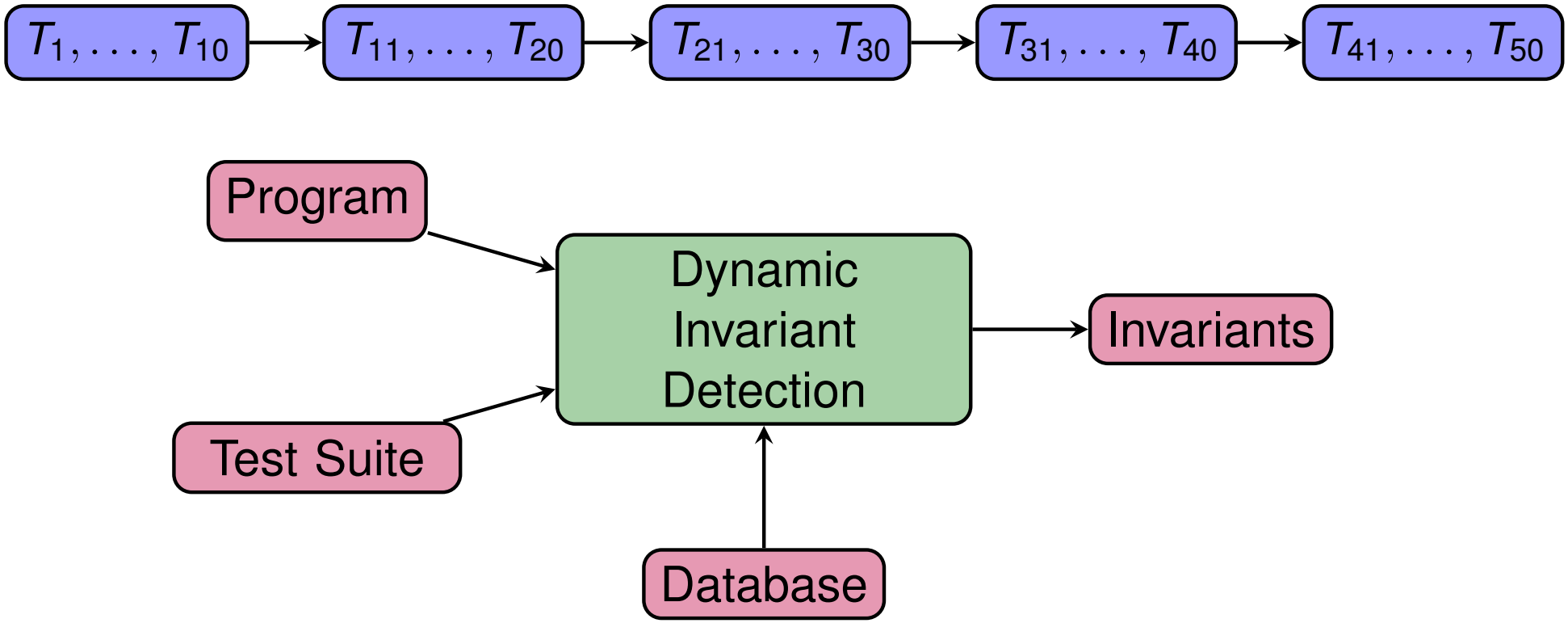
Database-Aware Test Suite Reduction



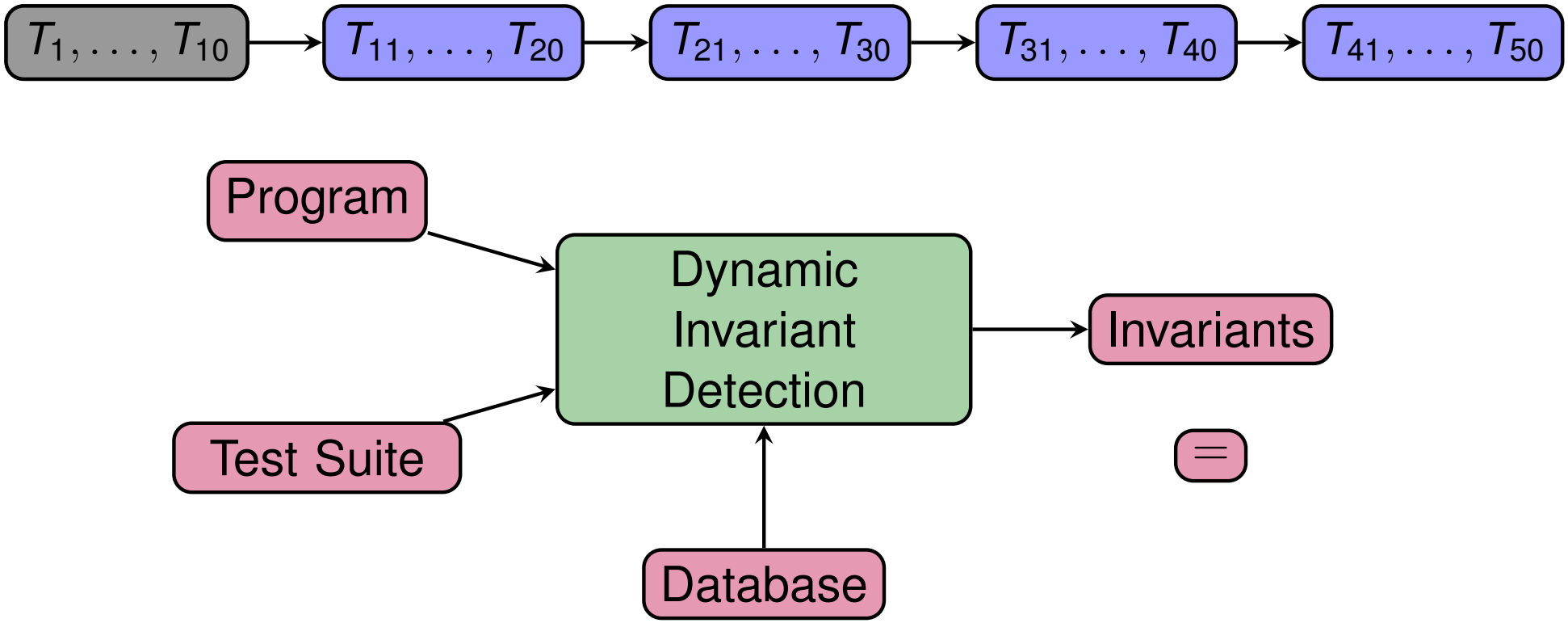
Database-Aware Test Suite Reduction



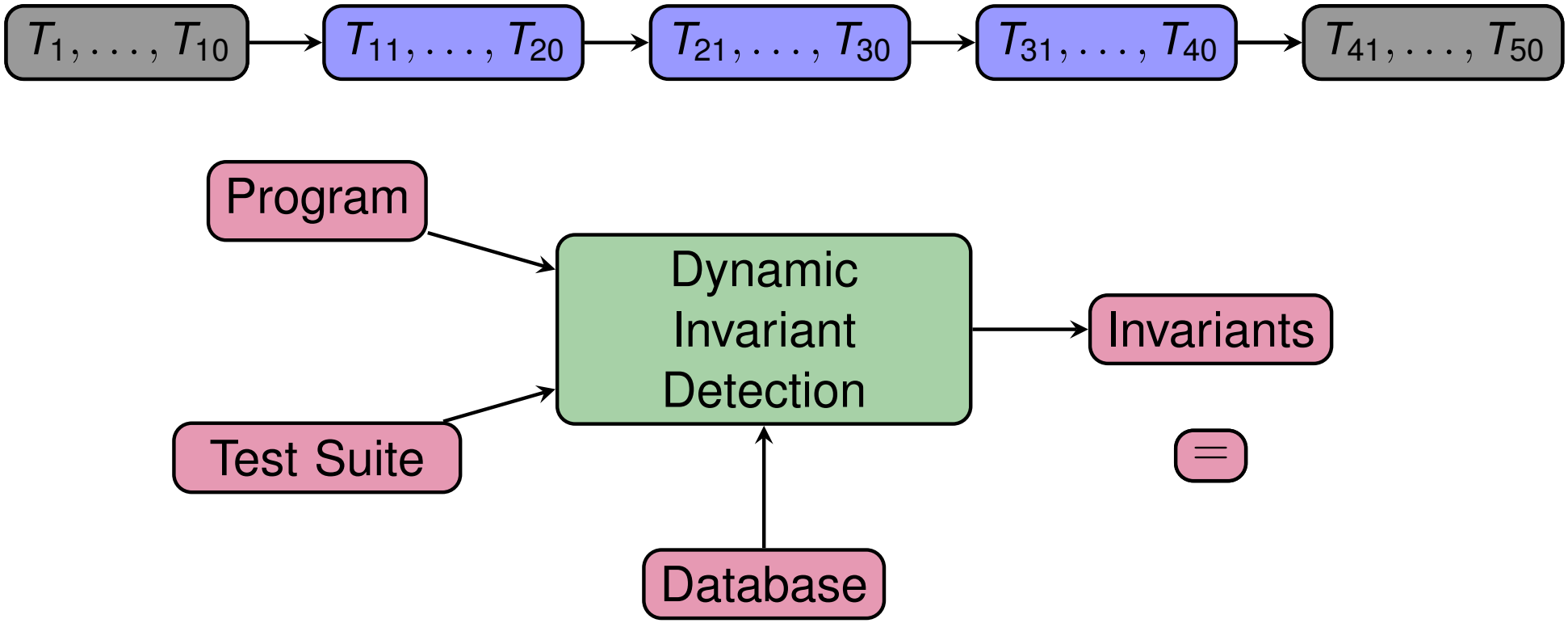
Database-Aware Test Suite Reduction



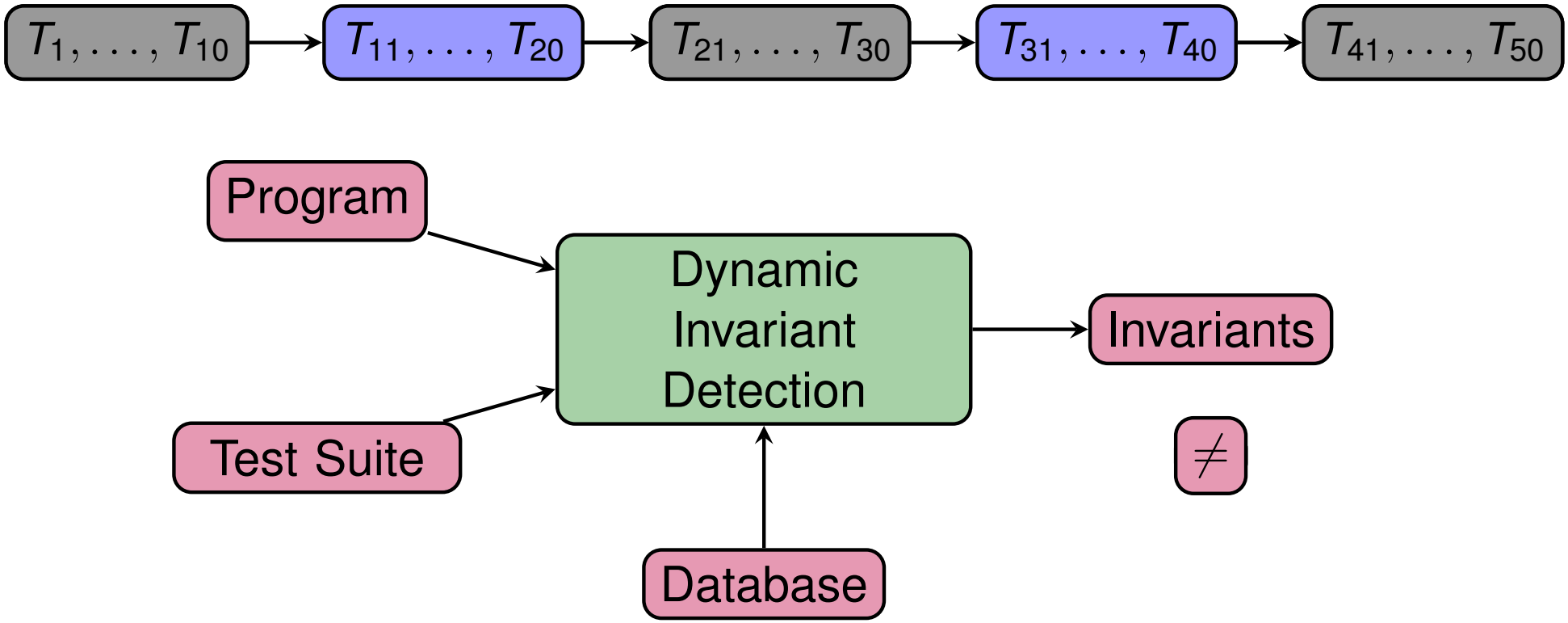
Database-Aware Test Suite Reduction



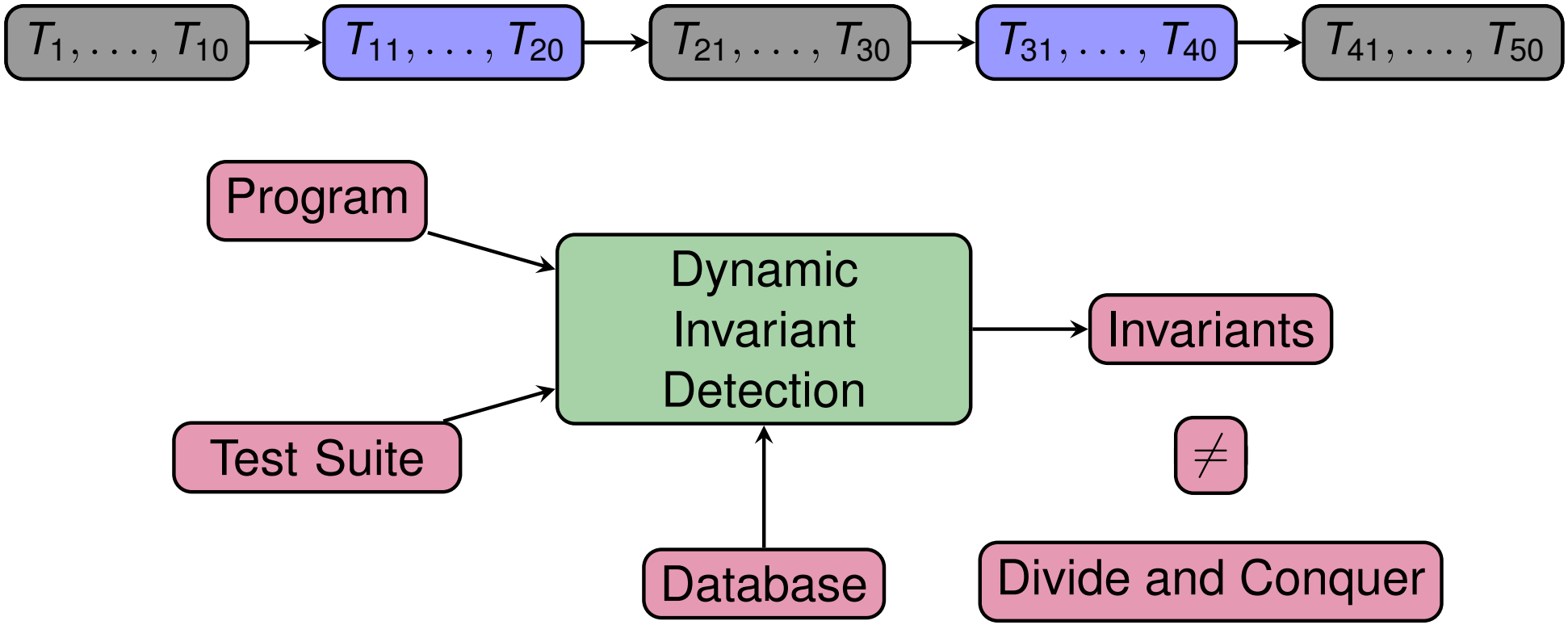
Database-Aware Test Suite Reduction



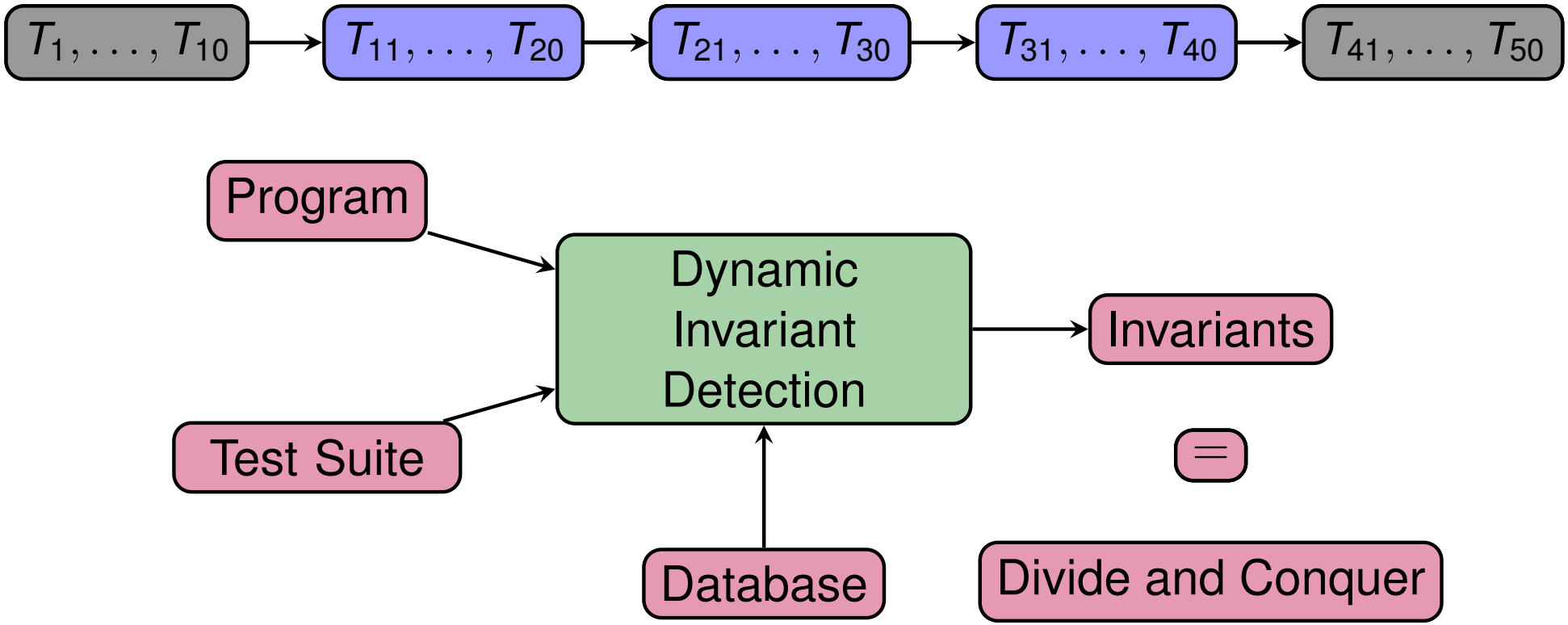
Database-Aware Test Suite Reduction



Database-Aware Test Suite Reduction



Database-Aware Test Suite Reduction



Conclusion

Conclusion

- Databases are widely used in real-world applications
- Database applications have complex state and structure
- Programmers often encode constraints in program source
- Dynamic invariant detection reverse engineers constraints
- Detected invariants are meaningful and enforceable

Future Work

- Further empirical studies of dynamic invariants
- Implement and evaluate several client applications

Conclusion

Conclusion

- Databases are widely used in real-world applications
- Database applications have complex state and structure
- Programmers often encode constraints in program source
- Dynamic invariant detection reverse engineers constraints
- Detected invariants are meaningful and enforceable

Future Work

- Further empirical studies of dynamic invariants
- Implement and evaluate several client applications

Using Dynamic Invariant Detection to Support the Testing and Analysis of Database Applications

Gregory M. Kapfhammer

Department of Computer Science
Allegheny College
http://www.cs.allegheny.edu/~gkapfham/

Thank you for your attention!
I welcome your questions and comments.

ALLEGHENY COLLEGE