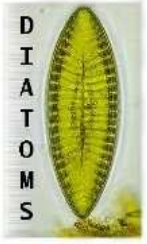
A Primer on Testing Database-Driven Applications

Gregory M. Kapfhammer
Department of Computer Science
Allegheny College

3rd Biennial Intramural Faculty Conference
May 18 - 19, 2004



Outline

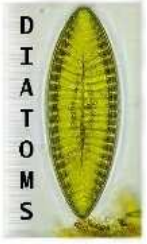
- Introduction and Motivation
- Testing Challenges
- Database-Driven Applications
- A Unified Representation
- Test Adequacy Criteria
- Test Suite Execution
- Test Coverage Monitoring
- Conclusions and Resources

Invocation

Thus spake the master, "*Any program, no matter how small, contains bugs.*"

The novice did not believe the master's words. "*What if the program were so small that it performed a single function?*"

The master thoughtfully responded, "*Such a program would have no meaning. But, if such a one existed, the operating system would fail eventually, producing a bug.*"

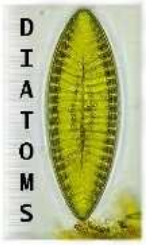


Invocation

But the novice was not satisfied. "*What if the operating system did not fail?*"

The master responded, "*There is no operating system that does not fail. But if such a one existed, the hardware would fail eventually, producing a bug.*"

The novice still was not satisfied. "*What if the hardware did not fail?*"



Invocation

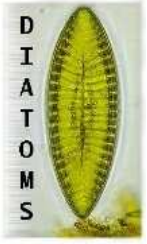
The master gave a great sigh. "*There is no hardware that does not fail. But if such a one existed, the user would want the program to do something different, and this too is a bug.*"

A program without bugs would be an absurdity, a nonesuch. If there were a program without any bugs then the world would cease to exist.

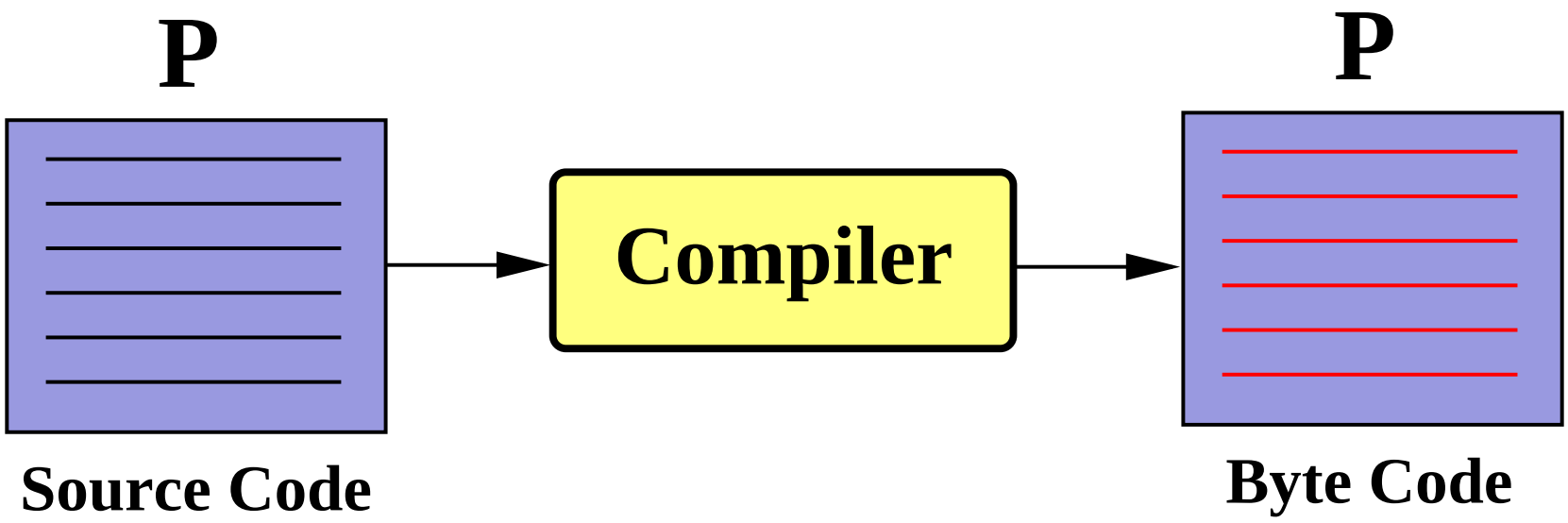
Geoffrey James

The Zen of Programming

(Adaptation)

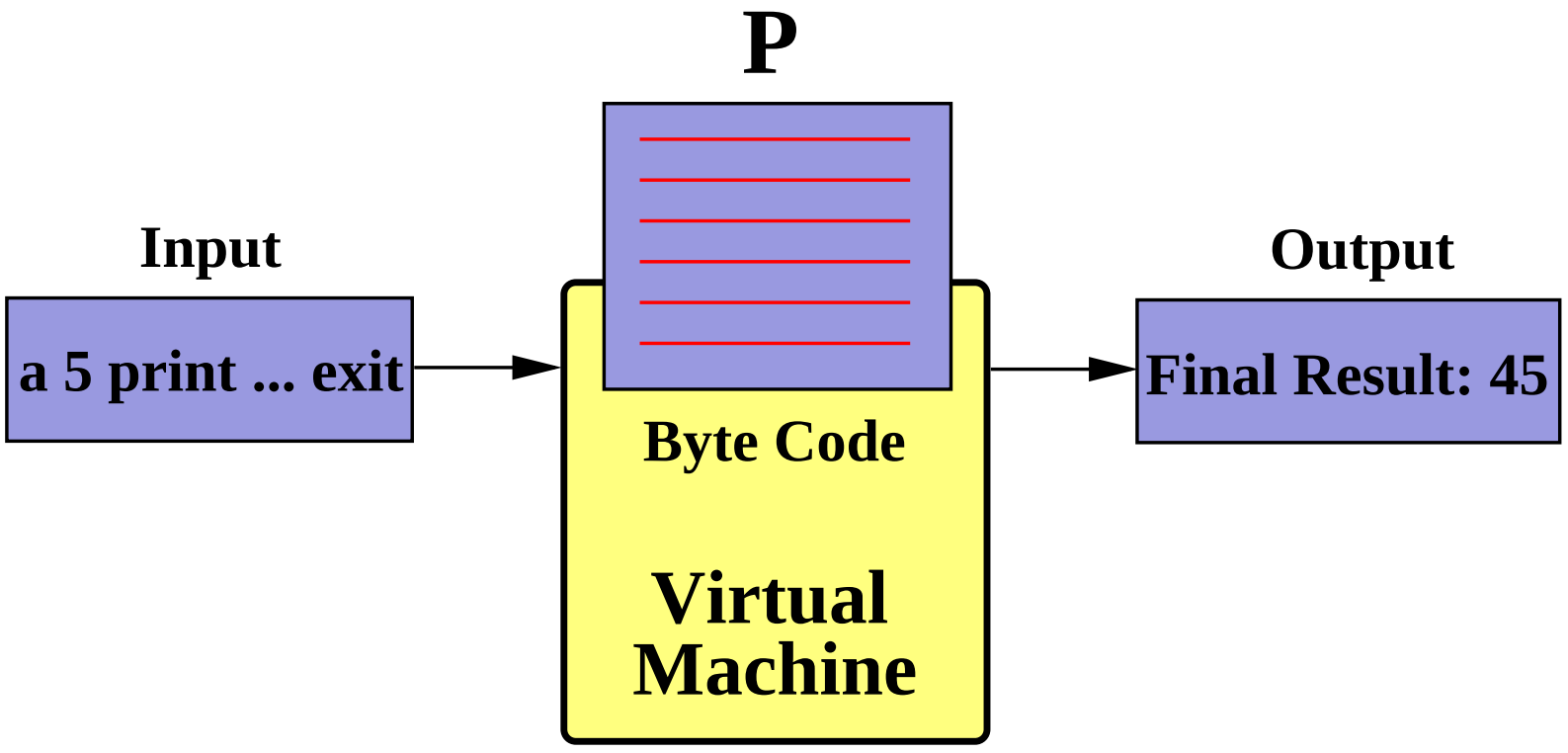


Life of a Program: Compilation



- The programming language compiler produces a representation of a program that can be executed

Life of a Program: Execution

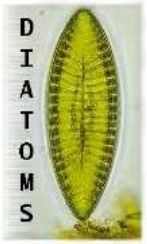


- The virtual machine executes the program's byte codes

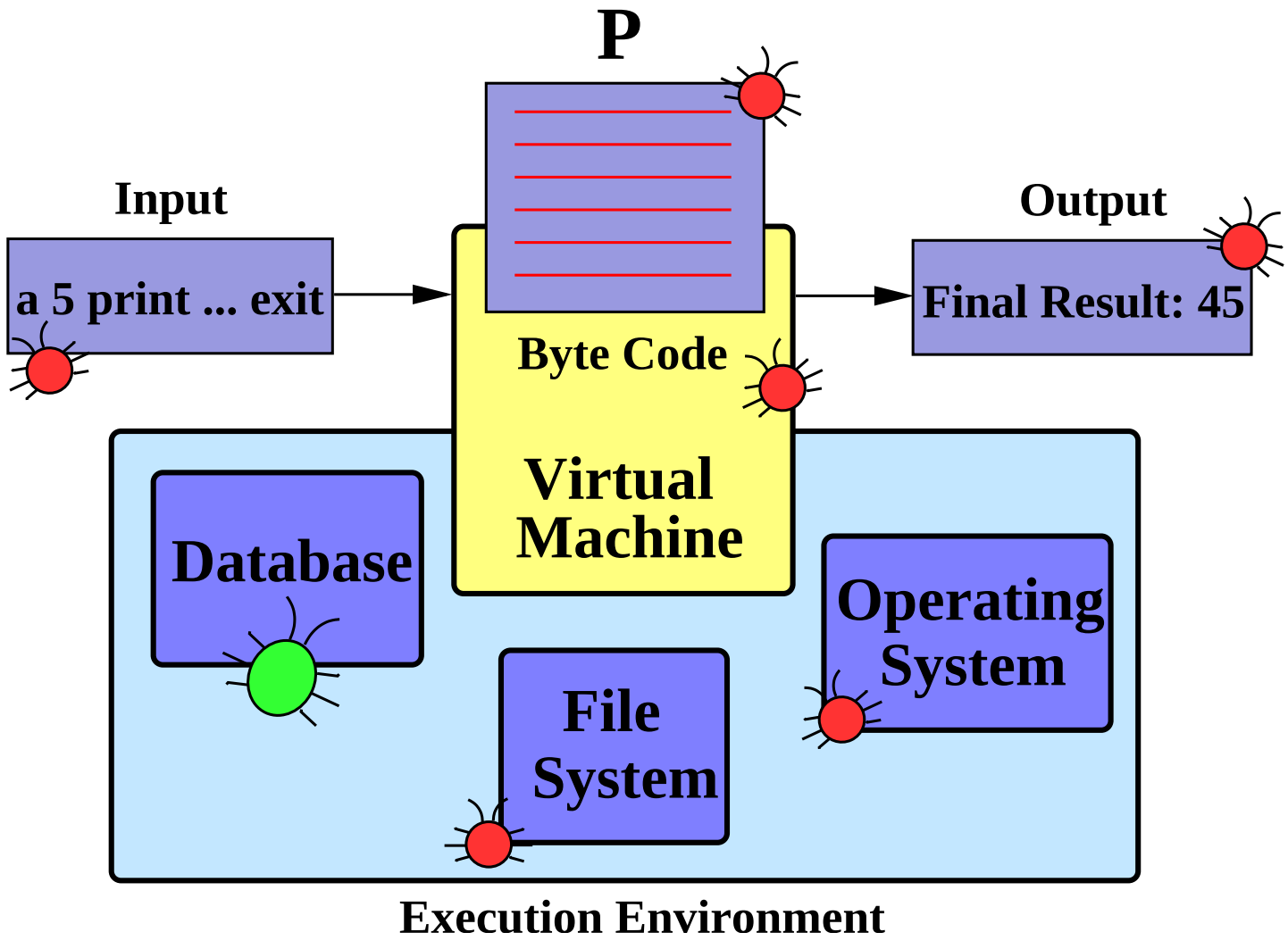
Testing Challenges

I shall not deny that the construction of these testing programs has been a major intellectual effort: to convince oneself that one has not overlooked “a relevant state” and to convince oneself that the testing programs generate them all is no simple matter. The encouraging thing is that (as far as we know!) it could be done.

Edsger W. Dijkstra, 1968



Where Bugs Live

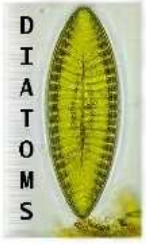


Motivation

The Risks Digest, Volume 22, Issue 64, 2003

Jeppesen reports airspace boundary problems

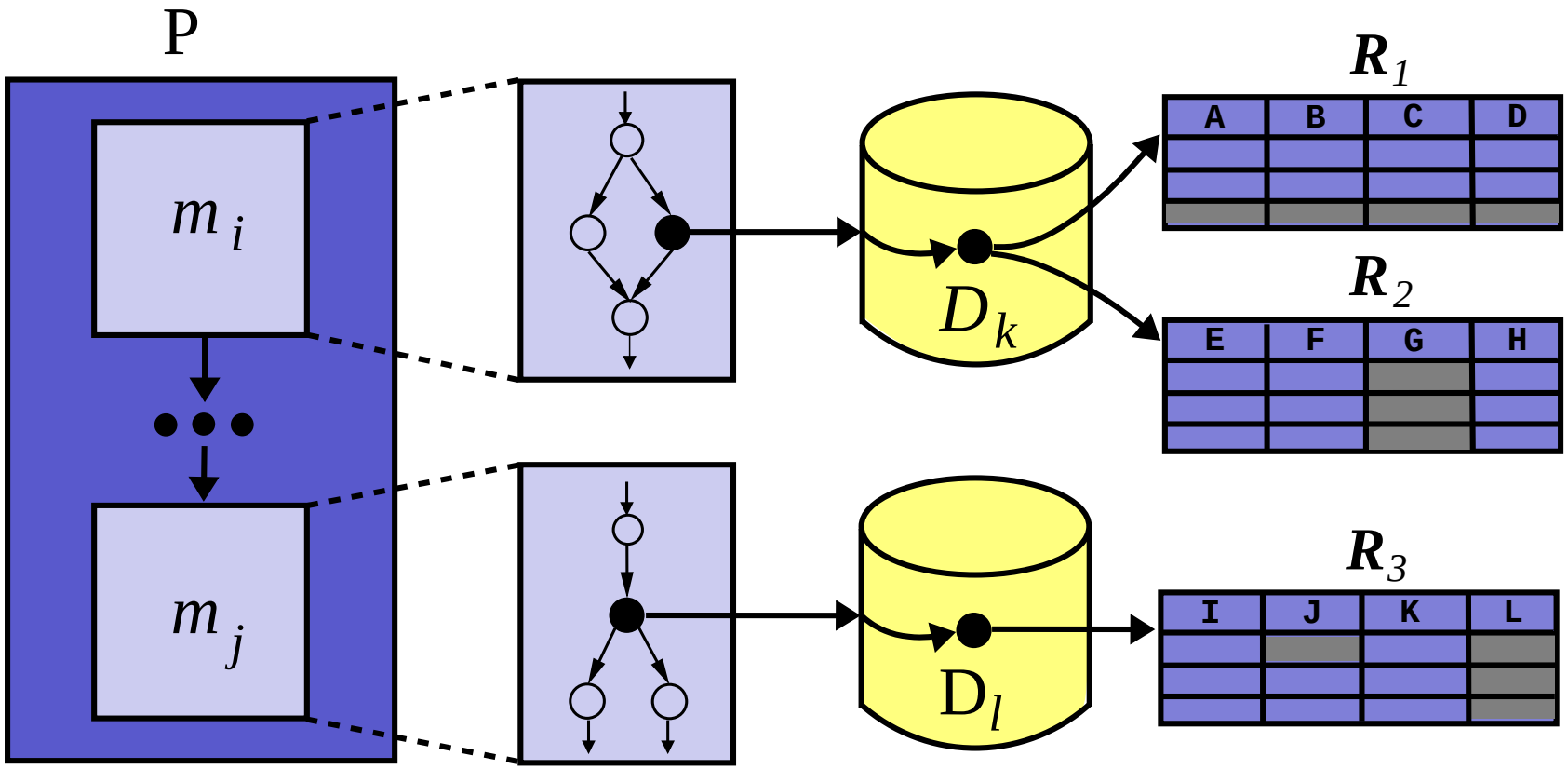
About 350 airspace boundaries contained in Jeppesen NavData are incorrect, the FAA has warned. The error occurred at Jeppesen after a software upgrade when information was pulled from a database containing 20,000 airspace boundaries worldwide for the March NavData update, which takes effect March 20.



More Testing Challenges

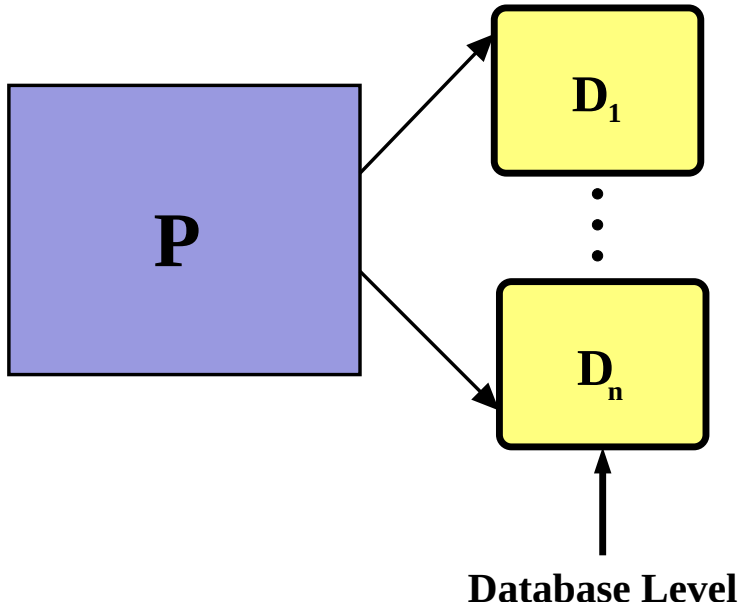
- Should consider the environment in which software applications execute
- Must test a program and its interaction with a database
- Database-driven application's state space is well-structured, but infinite (Chays et al.)
- Need to show program does not violate database integrity, where *integrity* = *consistency* + *validity* (Motro)
- Must locate program and database coupling points that vary in granularity

Database-Driven Applications

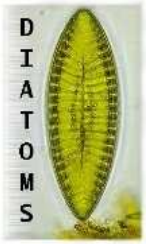


- Program P interacts with two relational databases

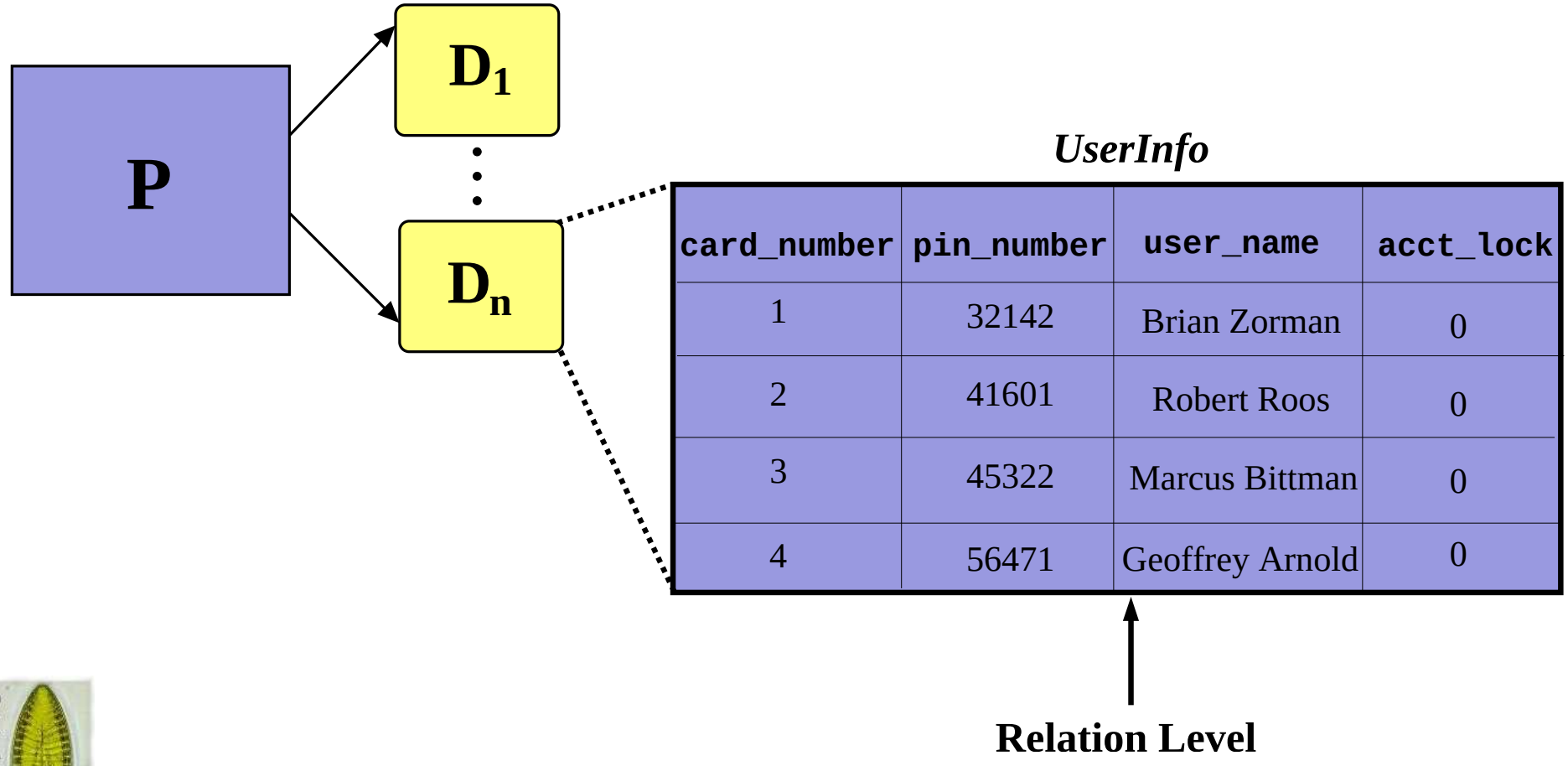
Database Interaction Levels



- A program can interact with a database at different levels of granularity

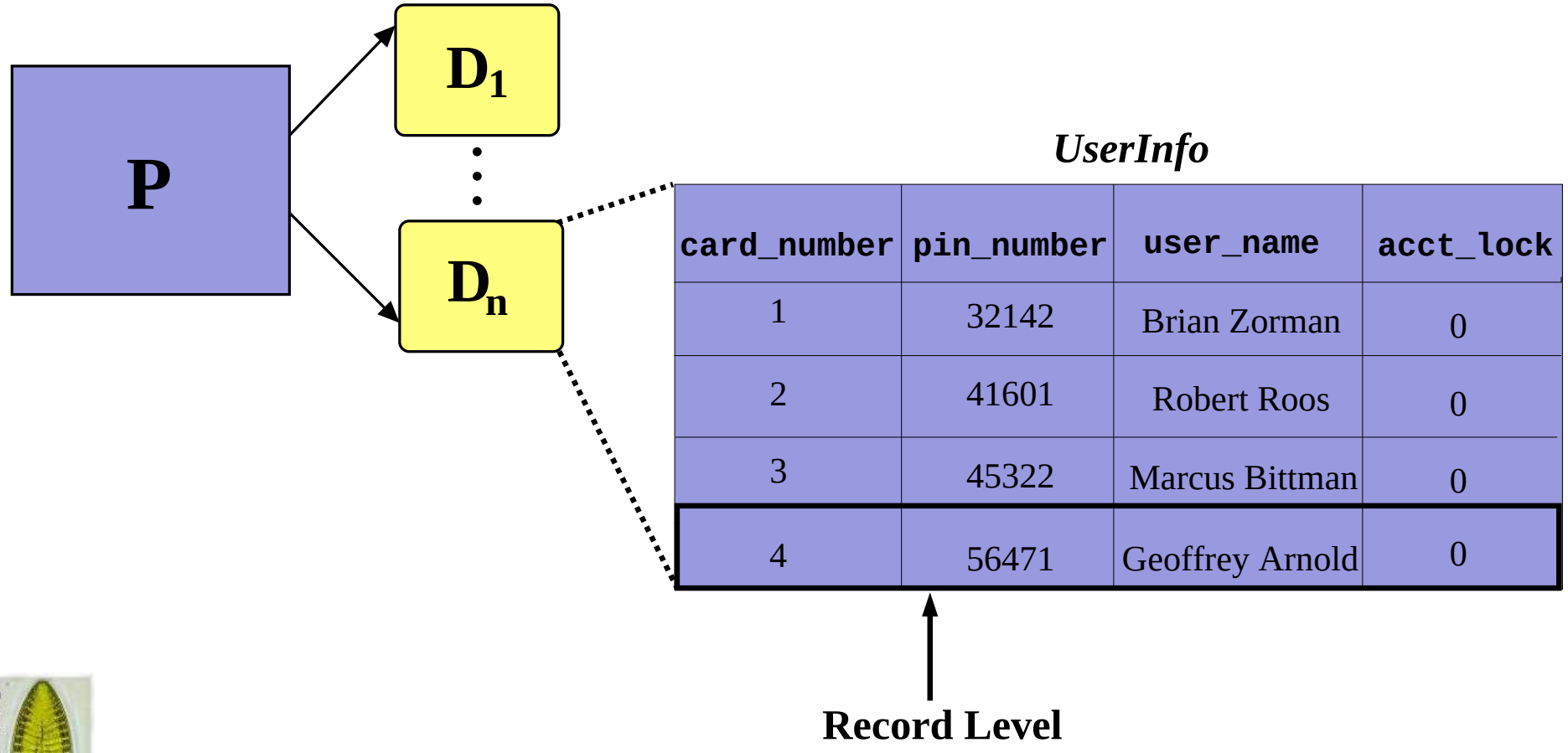


Database Interaction Levels

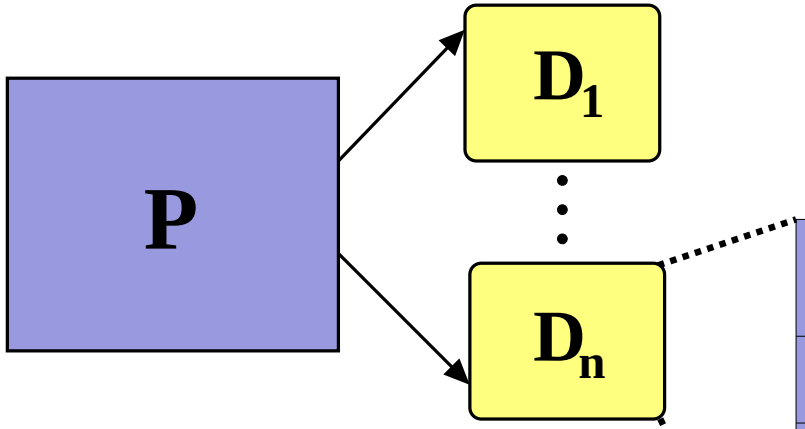


card_number	pin_number	user_name	acct_lock
1	32142	Brian Zorman	0
2	41601	Robert Roos	0
3	45322	Marcus Bittman	0
4	56471	Geoffrey Arnold	0

Database Interaction Levels



Database Interaction Levels

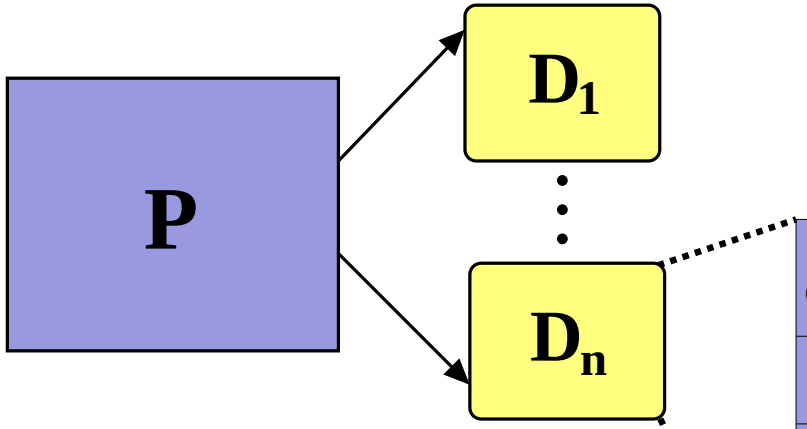


UserInfo

card_number	pin_number	user_name	acct_lock
1	32142	Brian Zorman	0
2	41601	Robert Roos	0
3	45322	Marcus Bittman	0
4	56471	Geoffrey Arnold	0

Attribute Level

Database Interaction Levels



UserInfo

card_number	pin_number	user_name	acct_lock
1	32142	Brian Zorman	0
2	41601	Robert Roos	0
3	45322	Marcus Bittman	0
4	56471	Geoffrey Arnold	0

Attribute Value Level

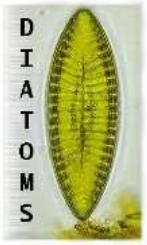
Database Interaction Points

select $A_1, A_2, \ldots, A_q$
from $r_1, r_2, \ldots, r_m$
where Q

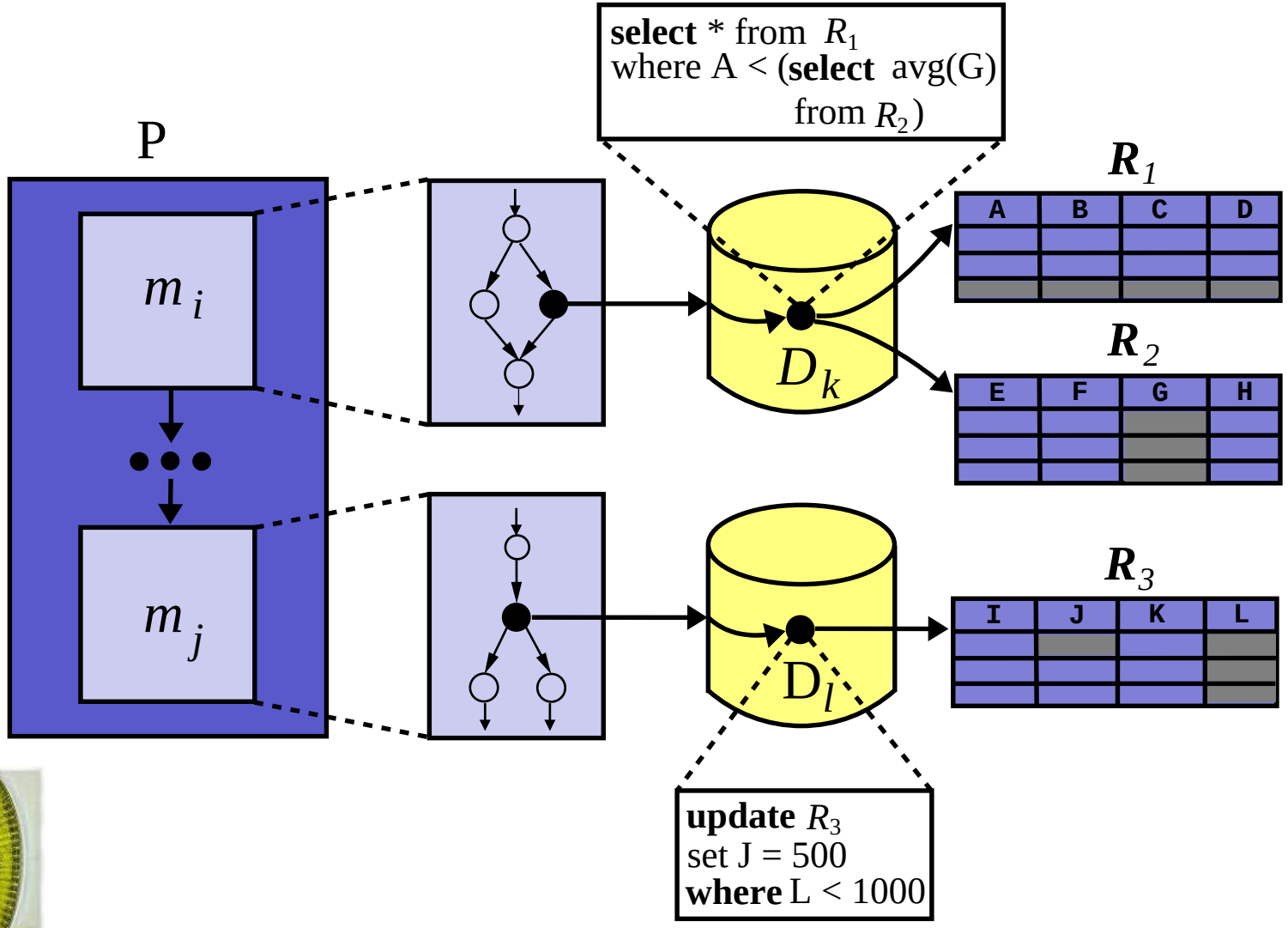
delete from r
where Q

insert into $r(A_1, A_2, \ldots, A_q)$
values$(v_1, v_2, \ldots, v_q)$

update r
set $A_l = F(A_l)$
where Q

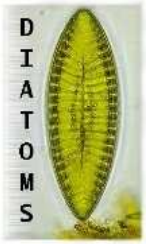


Refined Database-Driven Application

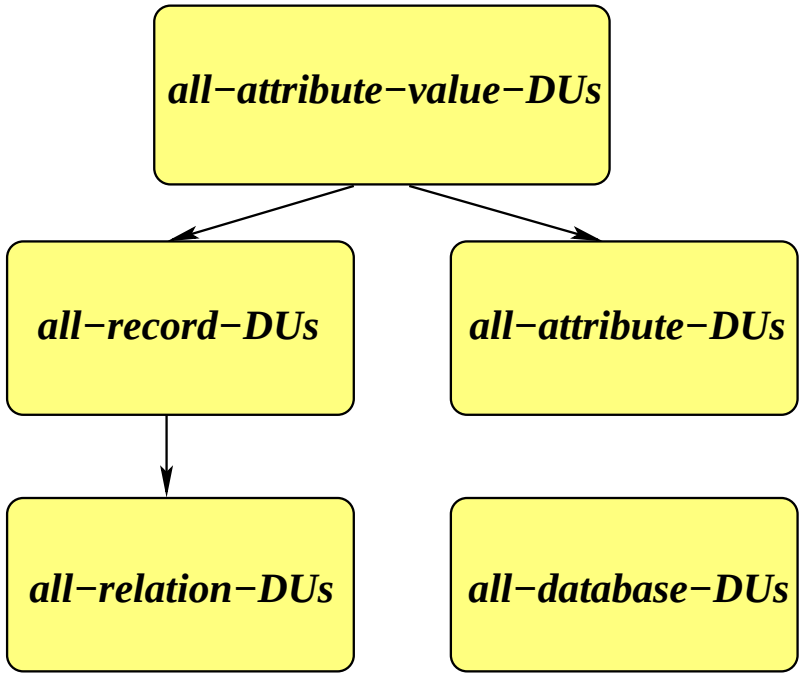


Data Flow Information

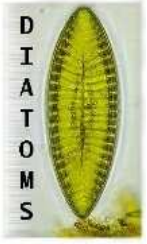
- Interaction point: `‘‘UPDATE UserInfo SET acct_lock=1 WHERE card_number=’’ + card_number + ‘‘;’’;`
  - Database Level: *define(BankDB)*
  - Attribute Level: *define(acct_lock)* and *use(card_number)*
- Data flow information varies with respect to the granularity of the database interaction



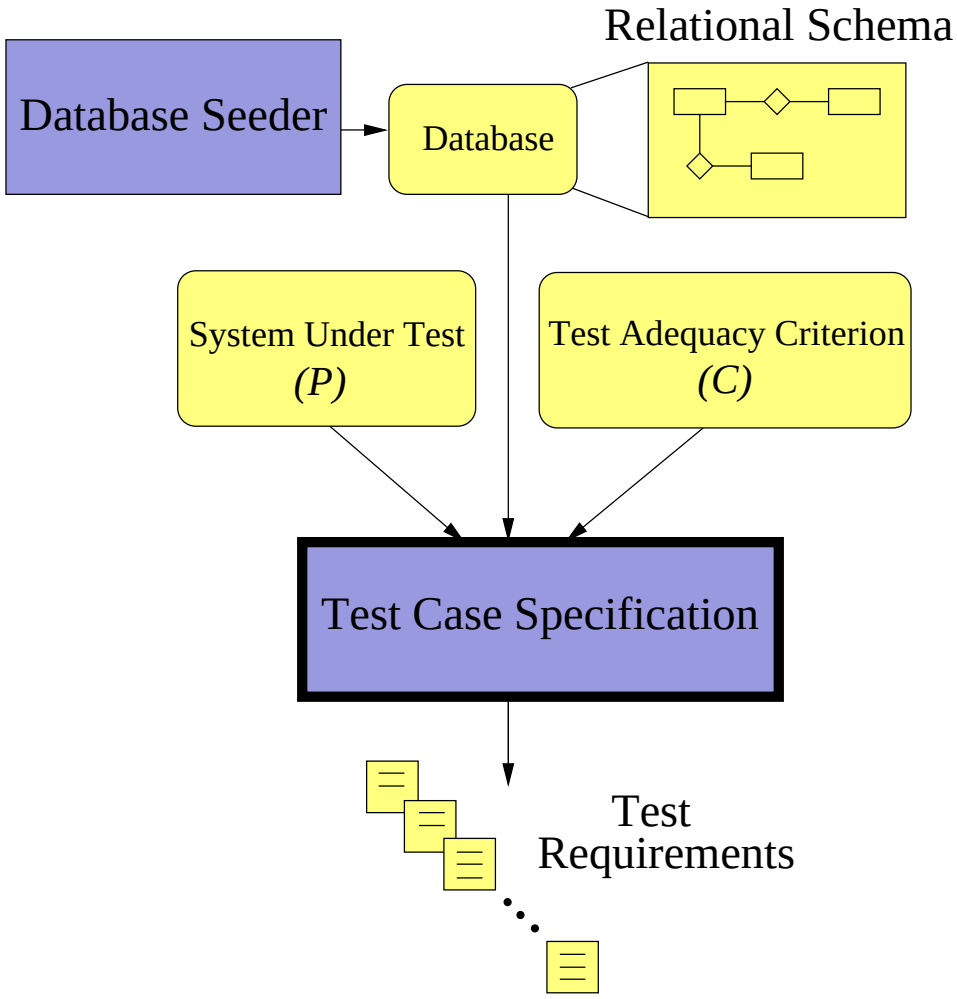
Test Adequacy Criteria



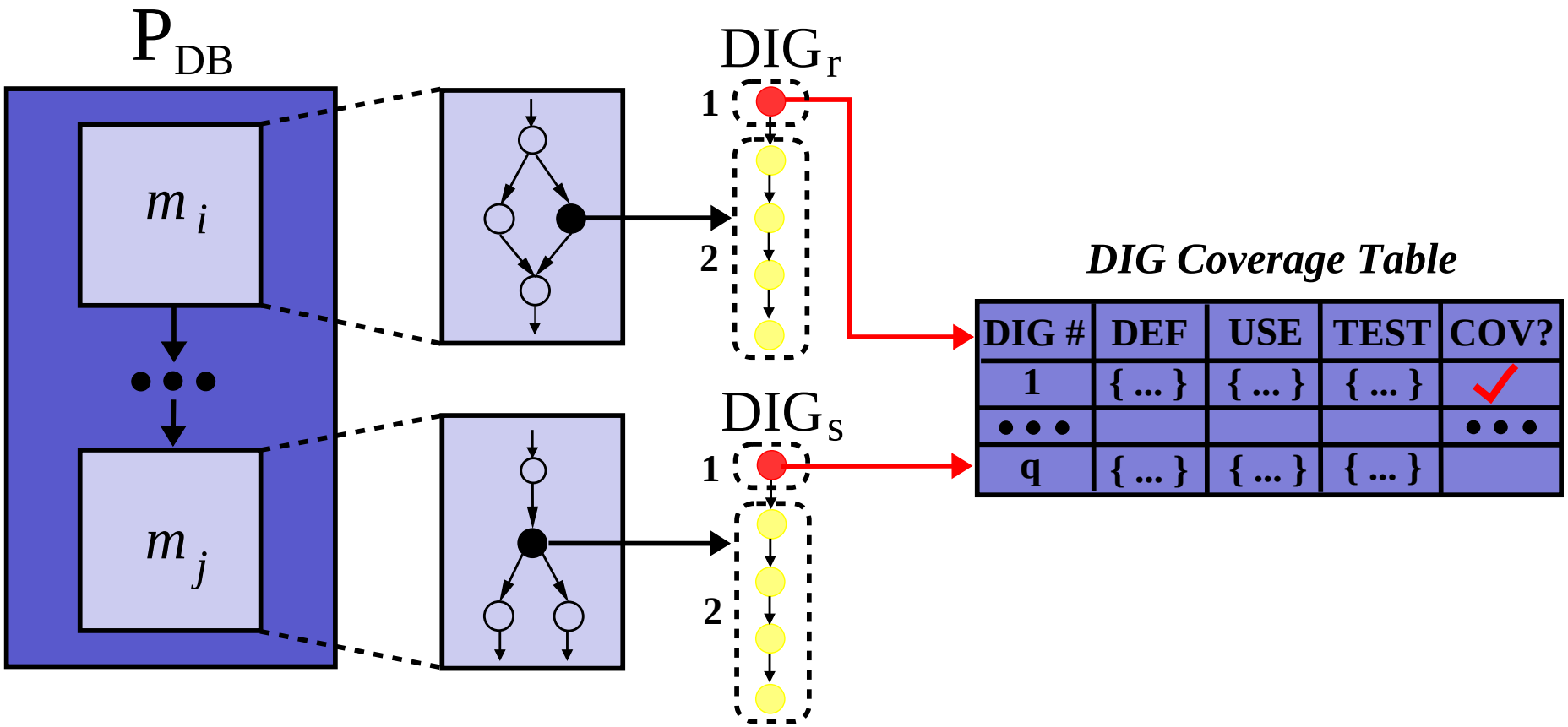
- Database interaction association (DIA) involves the *def* and *use* of a database entity
- DIAs can be located in the DICFG with data flow analysis
- *all-database-DUs* requires tests to exercise all DIAs for all of the accessed databases



Generating Test Requirements

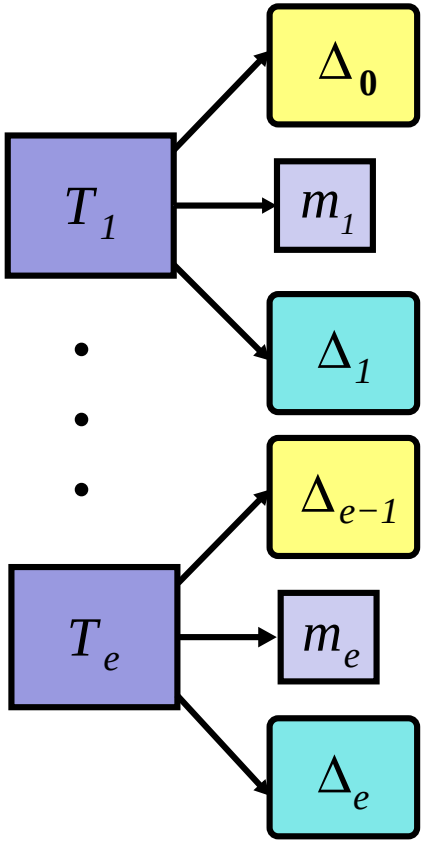


Tracking Covered DIGs and DIAs

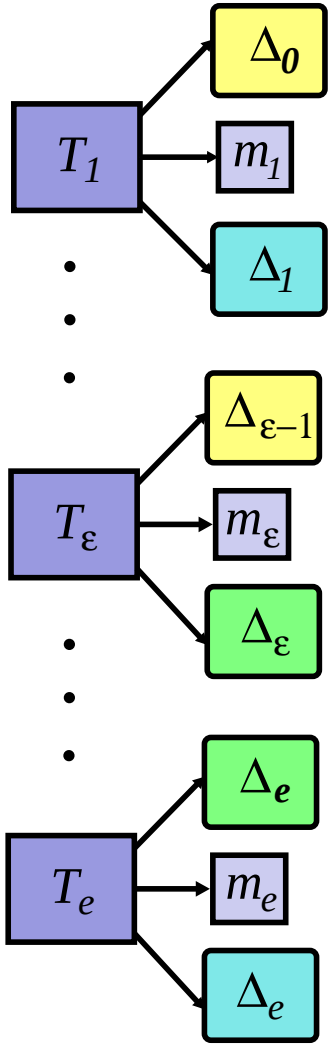


- DIA coverage can be tracked by recording which DIGs within a DICFG were executed during testing

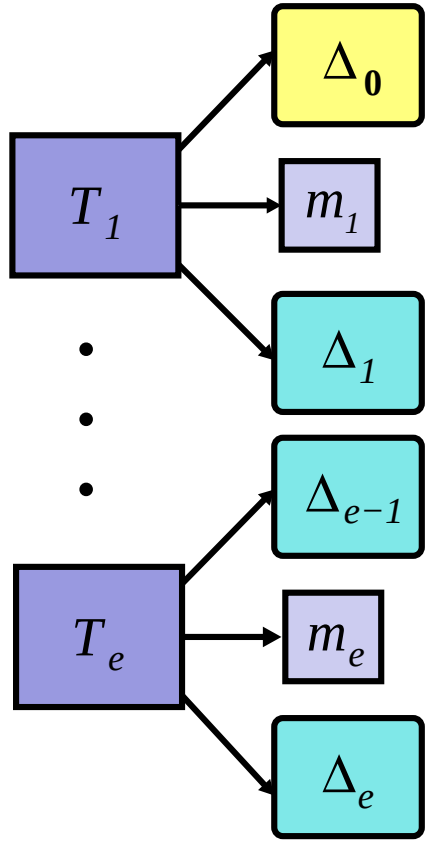
Types of Test Suites



Independent

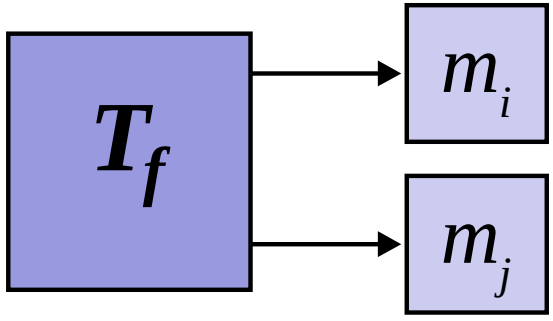


Partially Independent



Non-restricted

Calculating Adequacy

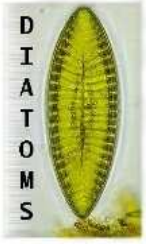


Test Requirements m_i

DIA	COV?
<def(e1), use(e1)>	✓
<def(e2), use(e2)>	
<def(e3), use(e3)>	
<def(e4), use(e4)>	✓

Test Requirements m_j

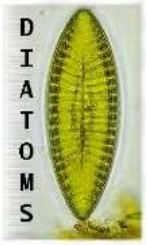
DIA	COV?
<def(e5), use(e5)>	✓
<def(e6), use(e6)>	
<def(e7), use(e7)>	✓
<def(e8), use(e8)>	✓
<def(e9), use(e9)>	
<def(e10), use(e10)>	✓



$$cov(m_i) = \frac{2}{4} \quad cov(m_j) = \frac{4}{6} \quad cov(T_f) = \frac{6}{10}$$

Conclusions

- Software testing is hard, especially when interaction with the application's environment is considered
- Must test the program's interaction with the database
- Many challenges associated with (1) unified program representation, (2) test adequacy criteria, (3) test coverage monitoring, (4) test suite execution
- Unique family of test adequacy criteria to ensure that test suites detect violations of database validity and completeness



Resources

Gregory M. Kapfhammer and Mary Lou Soffa. A Family of Test Adequacy Criteria for Database-Driven Applications. In Proceedings of the ACM SIGSOFT International Conference on the Foundations of Software Engineering, 2003.

Gregory M. Kapfhammer. Software Testing. CRC Press Computer Science Handbook. June, 2004.

`http://cs.allegheny.edu/~gkapfham/research/diatoms/`