



Security for Distributed E-Service Composition

Stefan Seltz Stephan Börzsönyi Alfons Kemper

Universität Passau



Outline

- Motivation
- Security Requirements
- Multilevel Security Architecture
 - Quality Assurance for External Operators
 - Security Measures during Plan Distribution
 - Architecture of the Runtime Security System
- Related Work
- Conclusions



Motivation

- Tomorrow's applications
 - No longer based on monolithic architectures
 - Distributed, dynamically extensible
 - Composed from existing software components/services
- ObjectGlobe
 - Internet query processing engine
 - Extensible by mobile, user-defined operators
 - Implemented in Java 2
 - Currently extended to handle general e-services

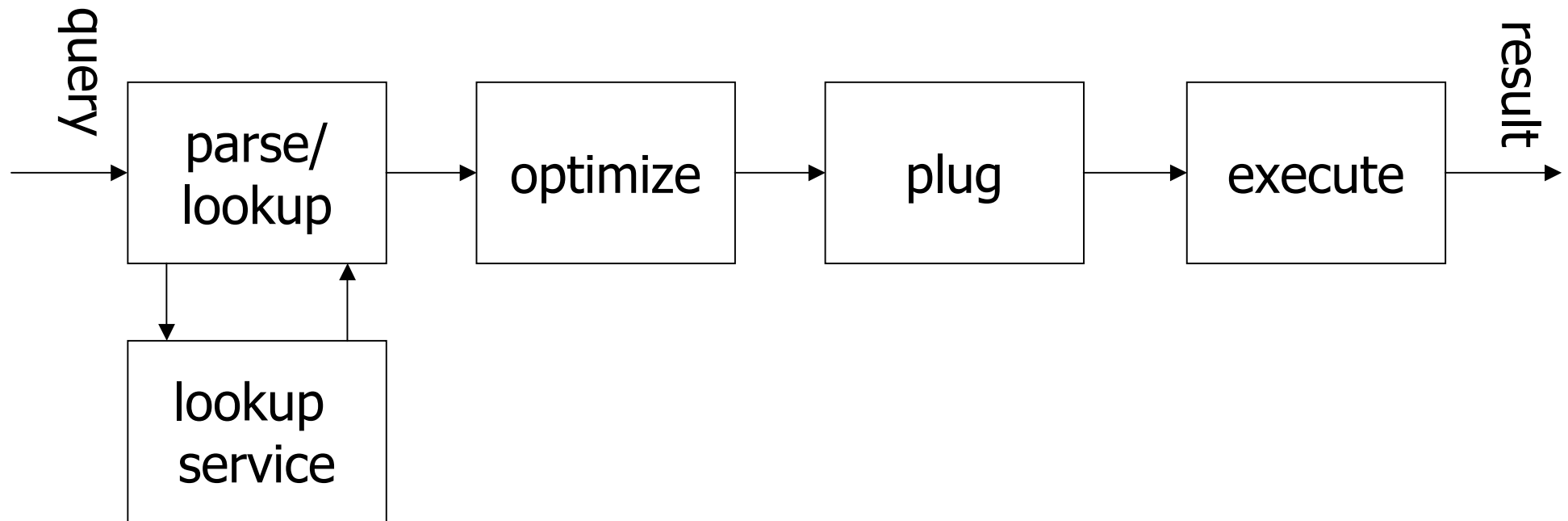


ObjectGlobe - Providers

- Three kinds of service providers:
 - Data providers
 - Function providers
 - Cycle providers
- A single site can comprise all three services

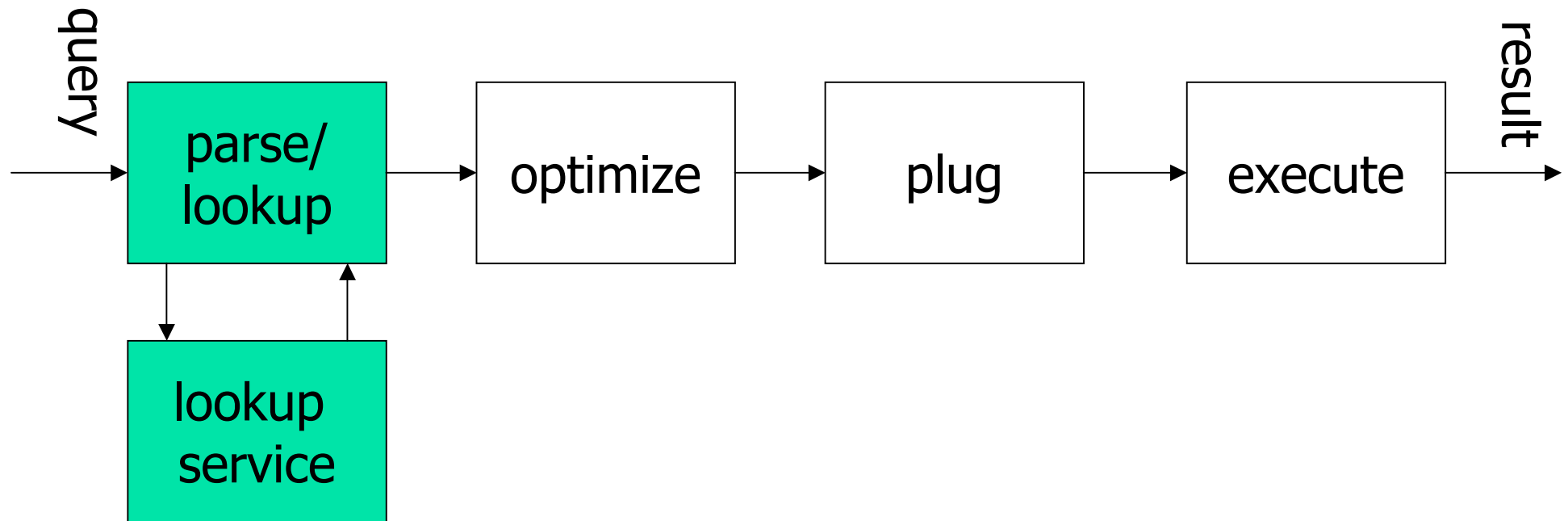


ObjectGlobe – Query Processing



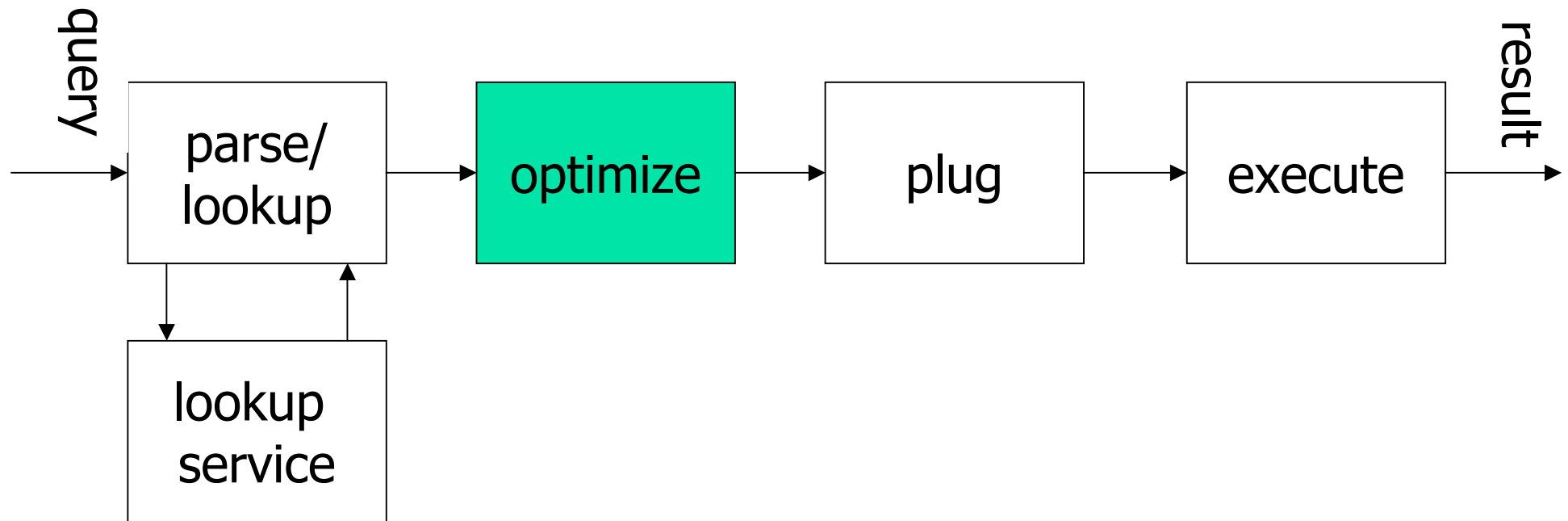


ObjectGlobe – Query Processing



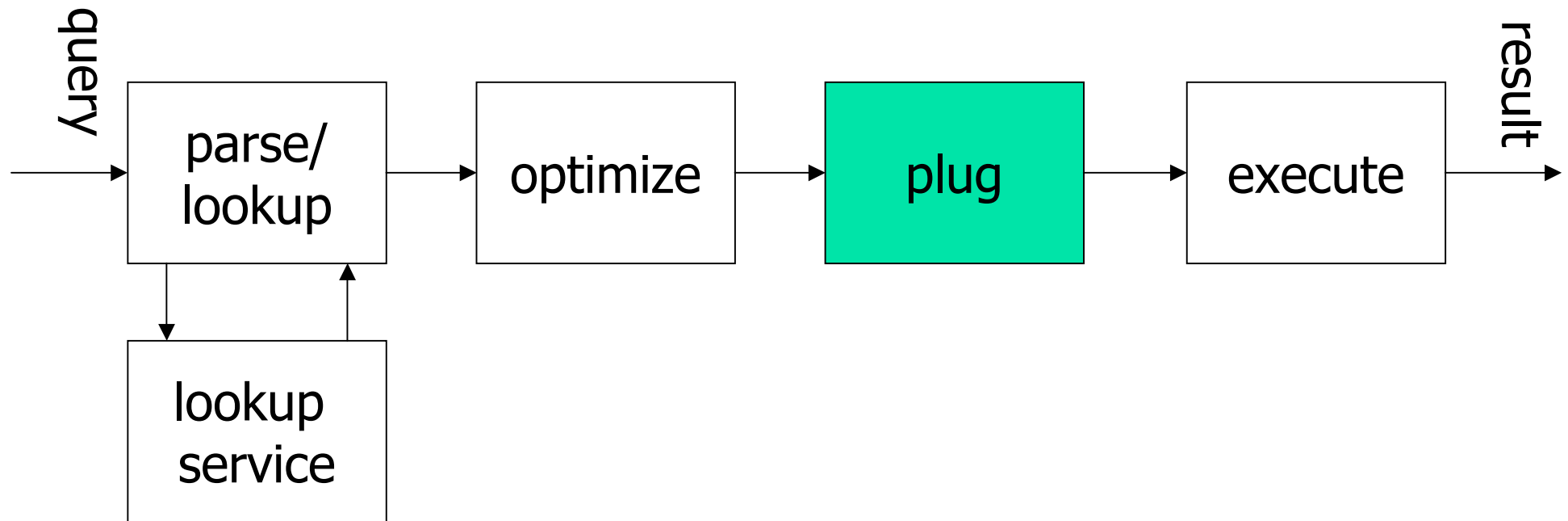


ObjectGlobe – Query Processing



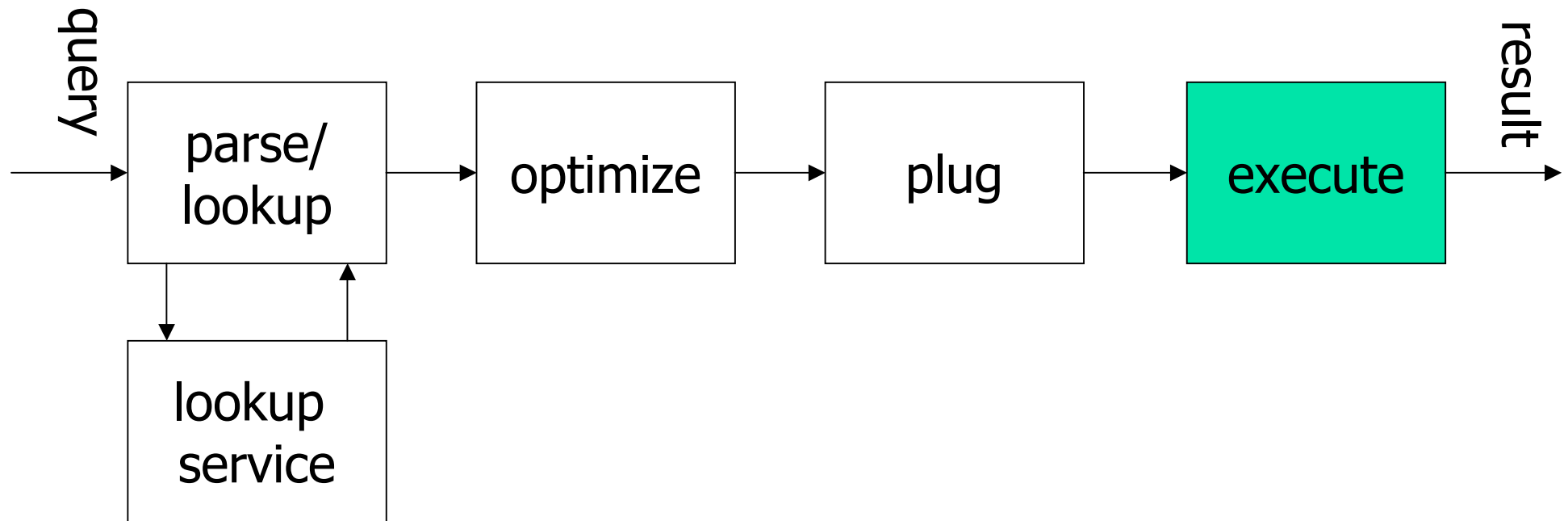


ObjectGlobe – Query Processing





ObjectGlobe – Query Processing

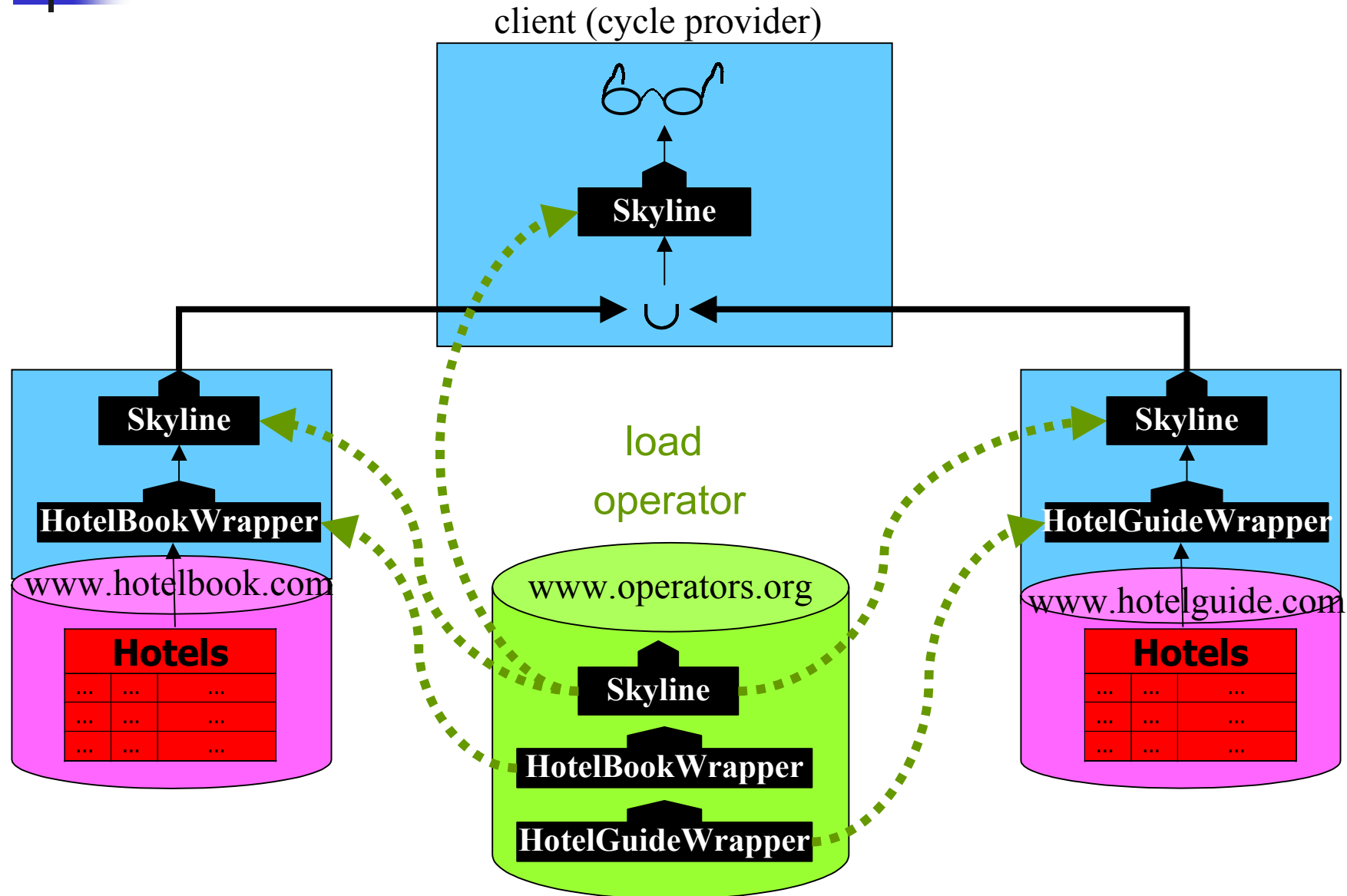




ObjectGlobe – Example Query

- “Find a hotel that is cheap and close to the beach in Nassau, Bahamas”
- User-defined operator “Skyline” to find all relevant hotels
[K. Stocker et.al.: The Skyline Operator, ICDE 2001]
- Skyline = all hotels where no other exists, which is closer to the beach and cheaper

ObjectGlobe – Example Query





Security Requirements

- Basic assumptions
 - Trustworthy cycle providers
 - Unmodified code of ObjectGlobe and Java
 - Security System of Java 2 works as designed
- Security concerns of ObjectGlobe
 - Common security concerns of distributed systems
 - Mobile code introduces specific security concerns



Common Security Concerns

- Authentication and authorization
- Anonymity
- Secure communication channels
- Admission control



Concerns by User-Defined Operators

- Protection of cycle providers against
 - Resource monopolization
 - Unauthorized resource access (e.g., file system)
 - Manipulation of ObjectGlobe components
- Users are concerned about
 - semantics of user-defined operators
 - privacy of the processed data



Example attack

- Example attack: resource monopolization

```
public class Skyline extends IteratorClass {
    public TypeSpec open() throws Exception {
        List l = new LinkedList();
        while(true)
            l.add(new Object());
        ...
    }
    ...
}
```



Example attack

- Example attack: resource monopolization

```
public class Skyline extends IteratorClass {  
    public TypeSpec open() throws Exception {  
        List l = new LinkedList();  
        while(true)  
            l.add(new Object());  
        ...  
    }  
    ...  
}
```




Example attack (2)

- Example attack: wrong semantics

```
public class Skyline extends IteratorClass {
    private ElementDescriptor currElem = null;
    private PredicateFunctionInterface eliminationPredicate =
        FunctionConstructor.construct(inputTypes[0],
                                     "name=\"Sheraton\"");
    public ElementDescriptor next() throws Exception {
        ...
        do {
            currElem = inputIterators[0].next();
        } while (currElem != null &&
                eliminationPredicate.test(currElem));
        ... /* skyline code */ ...
    } ... }
```



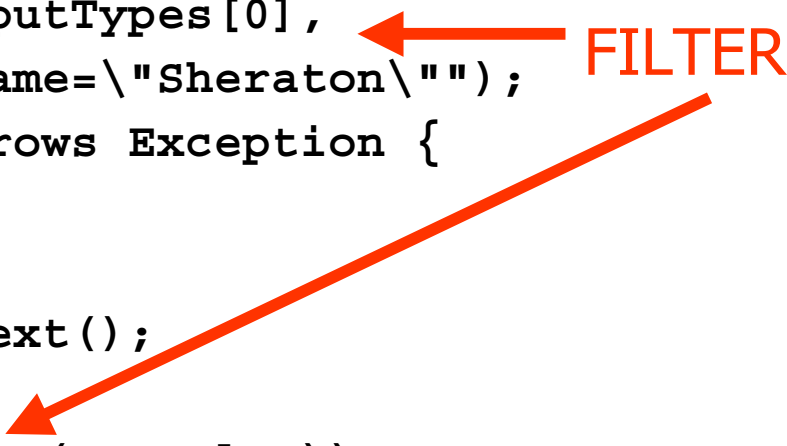
Example attack (2)

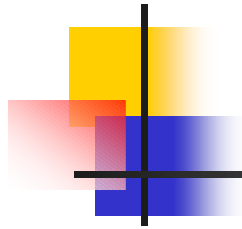
- Example attack: wrong semantics

```
public class Skyline extends IteratorClass {
    private ElementDescriptor currElem = null;
    private PredicateFunctionInterface eliminationPredicate =
        FunctionConstructor.construct(inputTypes[0],
                                     "name=\"Sheraton\"");
    public ElementDescriptor next() throws Exception {
        ...
        do {
            currElem = inputIterators[0].next();
        } while (currElem != null &&
                eliminationPredicate.test(currElem));
        ... /* skyline code */ ...
    } ... }

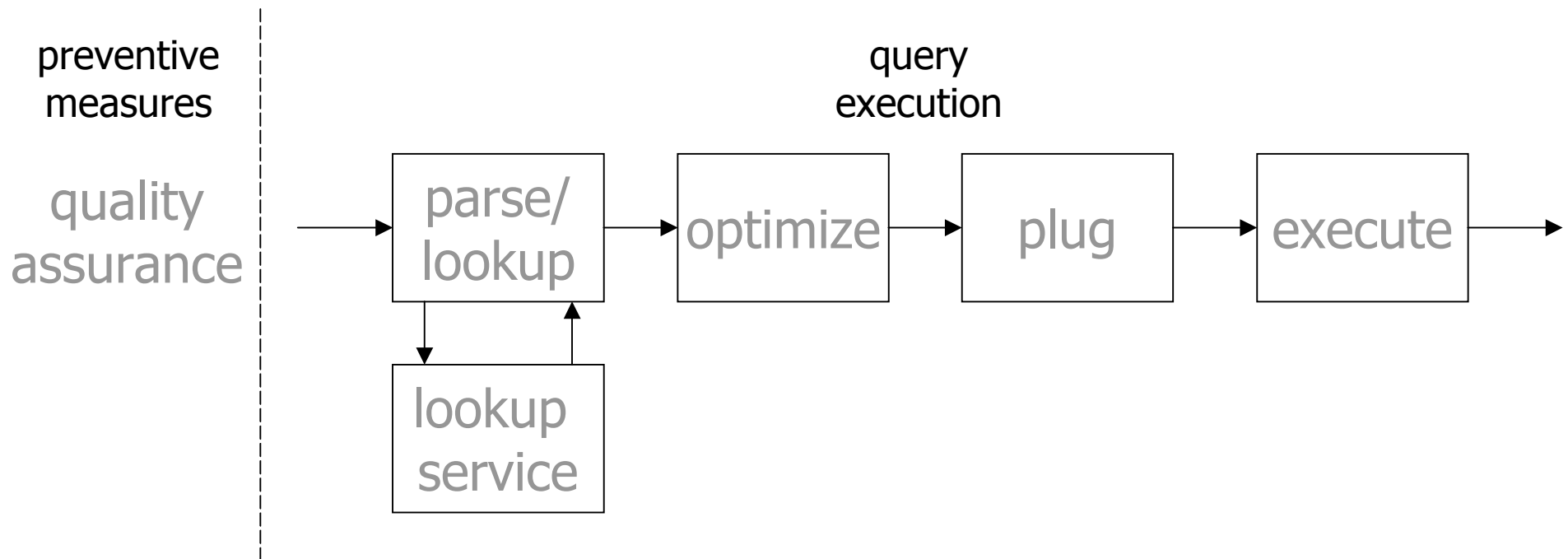
```

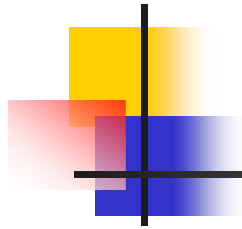
FILTER





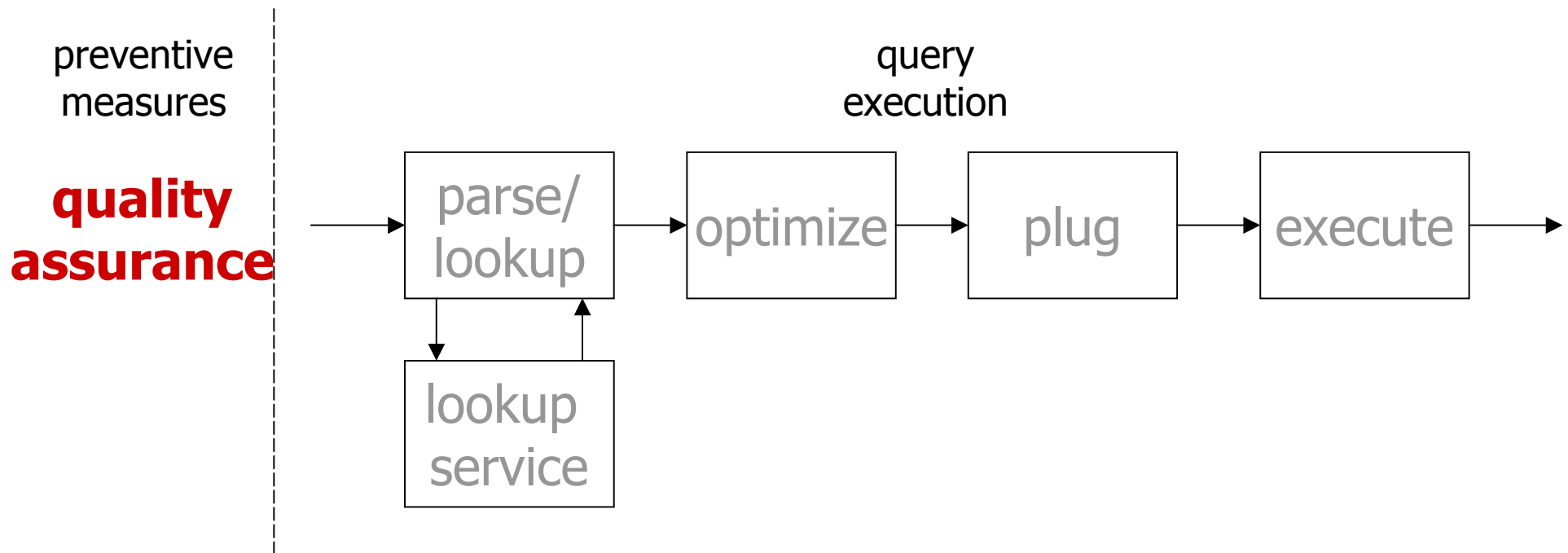
Multilevel Security Architecture





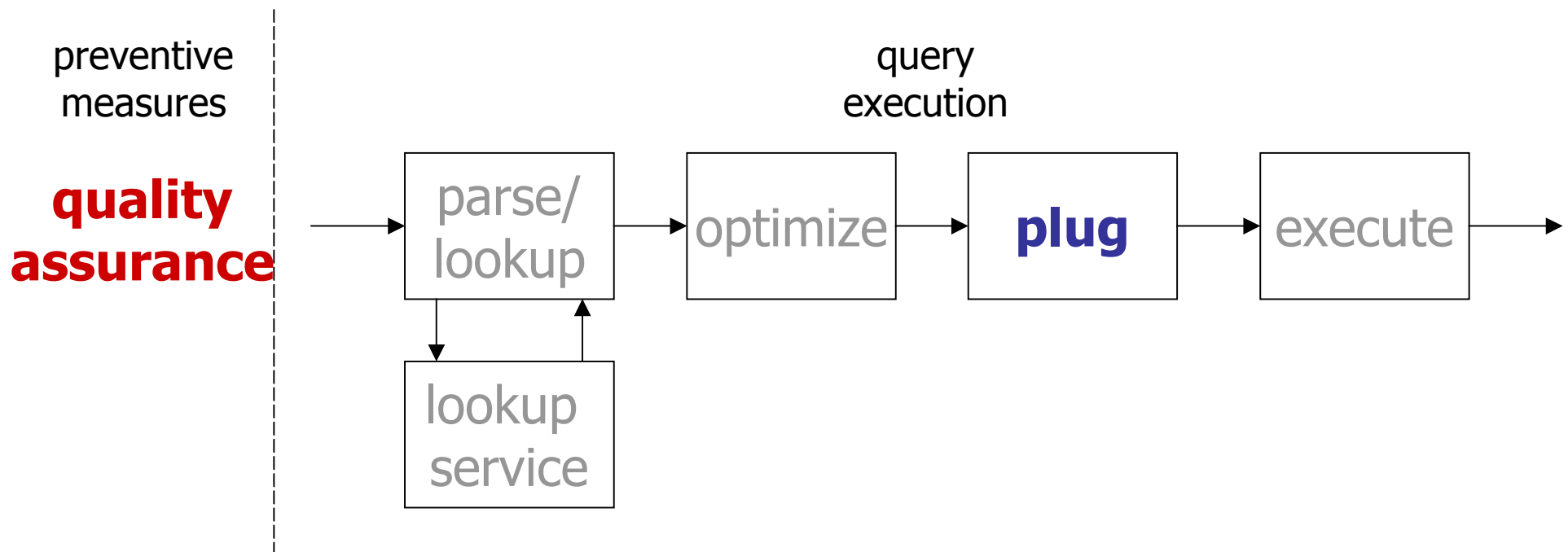
Multilevel Security Architecture

- Preventive measures



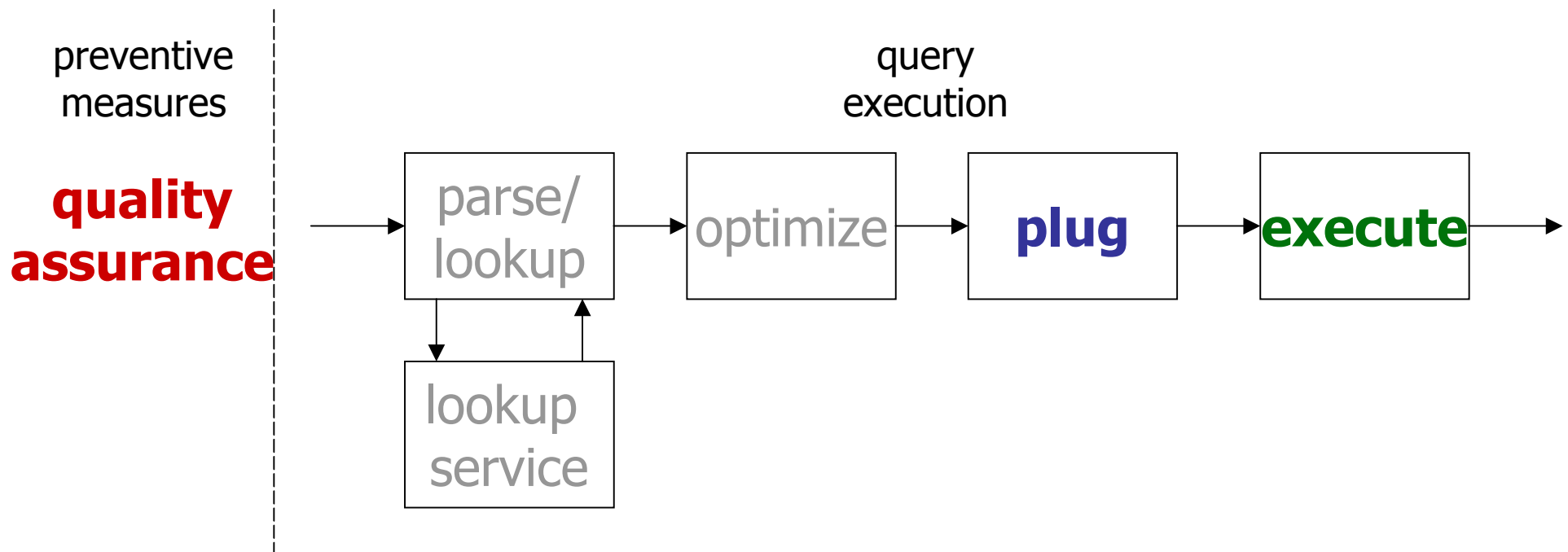
Multilevel Security Architecture

- Preventive measures
- Security measures during plan distribution



Multilevel Security Architecture

- Preventive measures
- Security measures during plan distribution
- Runtime security system





Preventive Measures

- Optional, preventive step
- Goals – Quality assurance
 - Verification of the semantics of the operator
 - Compare resource consumption with given cost models
 - Stress testing
- Results are digitally signed



Methods of Formal Specification

- Skyline - Mathematical Formula

$$\{s \mid s \in S \wedge \neg \exists t \in S: t \neq s \wedge t \geq s\}$$

- Skyline - Haskell

```
skyline :: [α] -> [α]
skyline ss = skyline' ss ss
skyline' [] ts = []
skyline' (s:ss) ts =
  if dominated s ts
  then skyline' ss ts
  else s:skyline' ss ts
dominated s [] = False
dominated s (t:ts) =
  dominance t s || dominated s ts
dominance t s = (t≠s && t≥s)
```




Test Data Generation

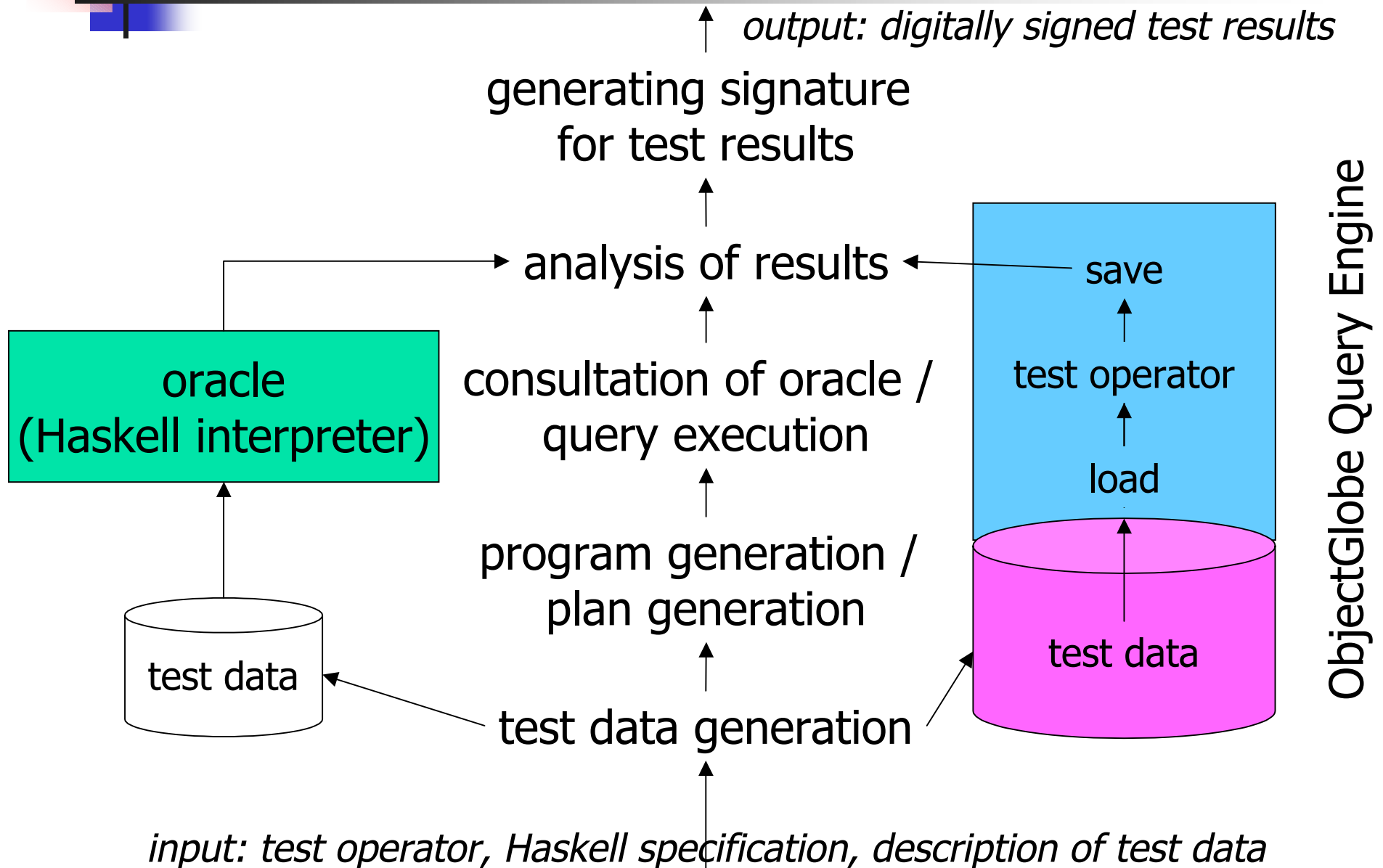
- User-directed
 - Test data fulfill preconditions of operators
 - Test data meet the testers' strategies
- Features
 - Specification of attribute values
 - Functional dependencies between attributes
 - Relationships between relations
 - Control on the order of the tuples



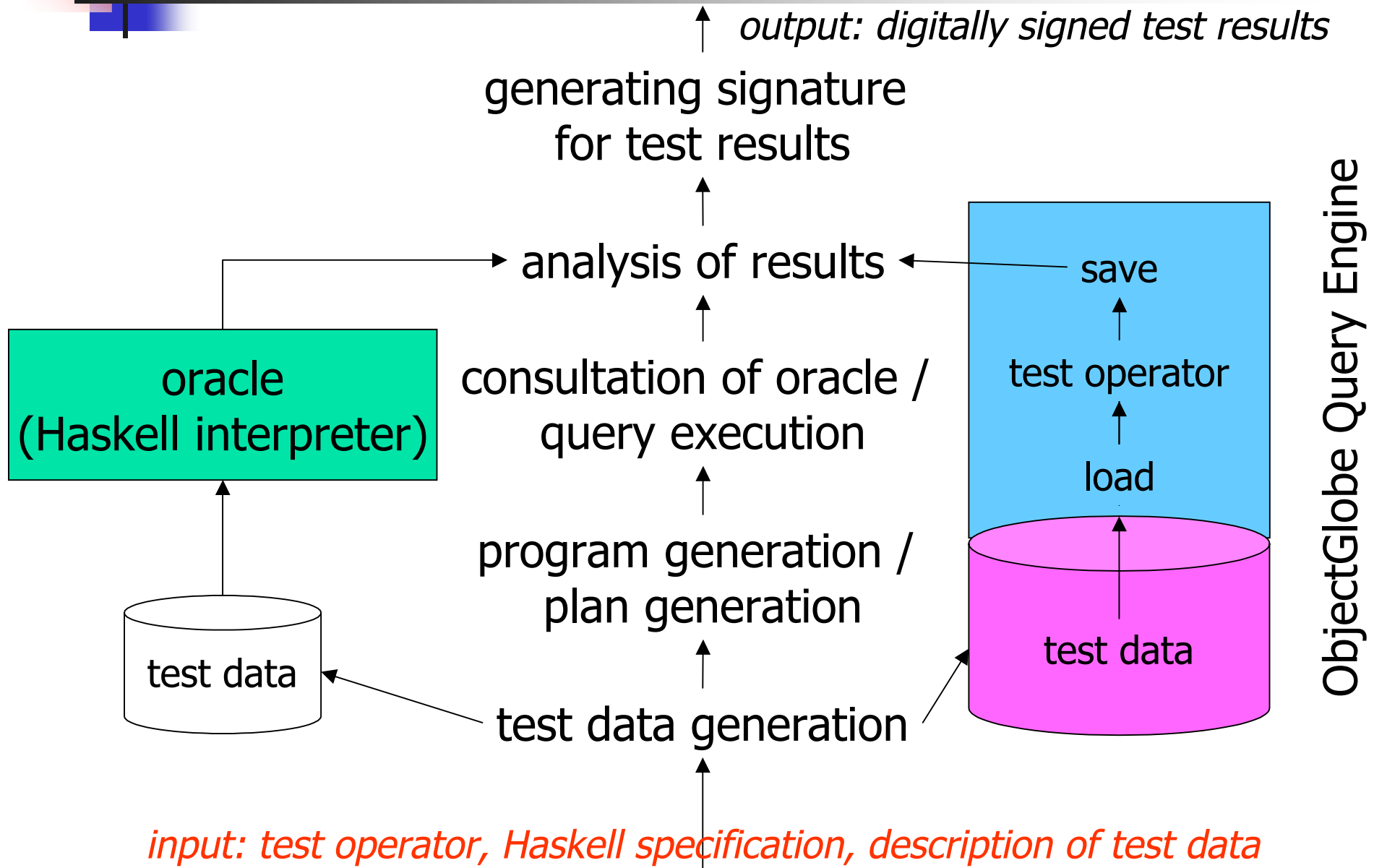
The OperatorCheck Server

- Benchmark test
 - Different sizes of input data
 - Resource consumption is measured
 - Results are compared to cost models (MathML)
- Correctness test
 - Verifies the semantics of operators
 - Black box testing
 - Haskell program as oracle
 - Different result comparison semantics

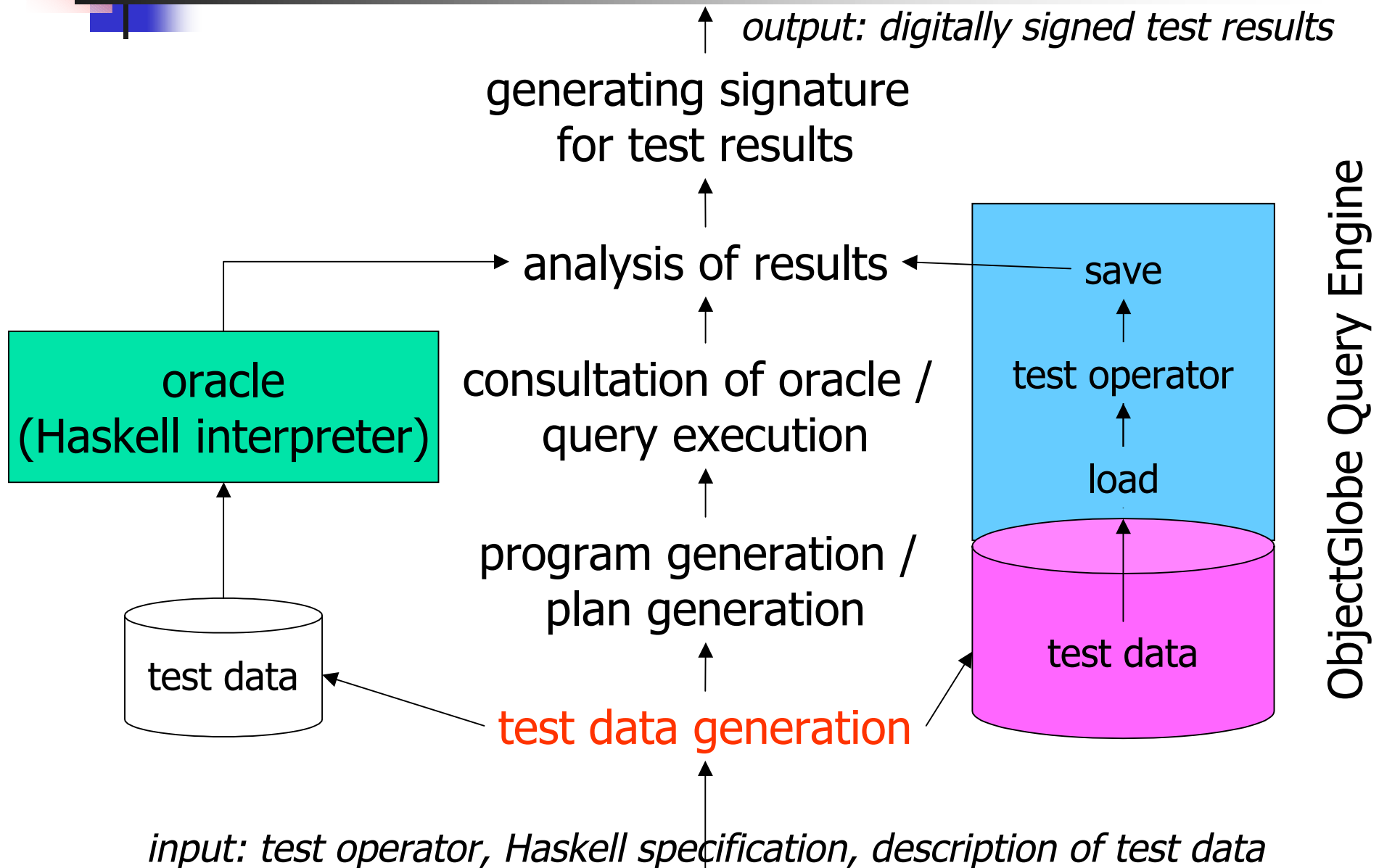
Architecture of OperatorCheck



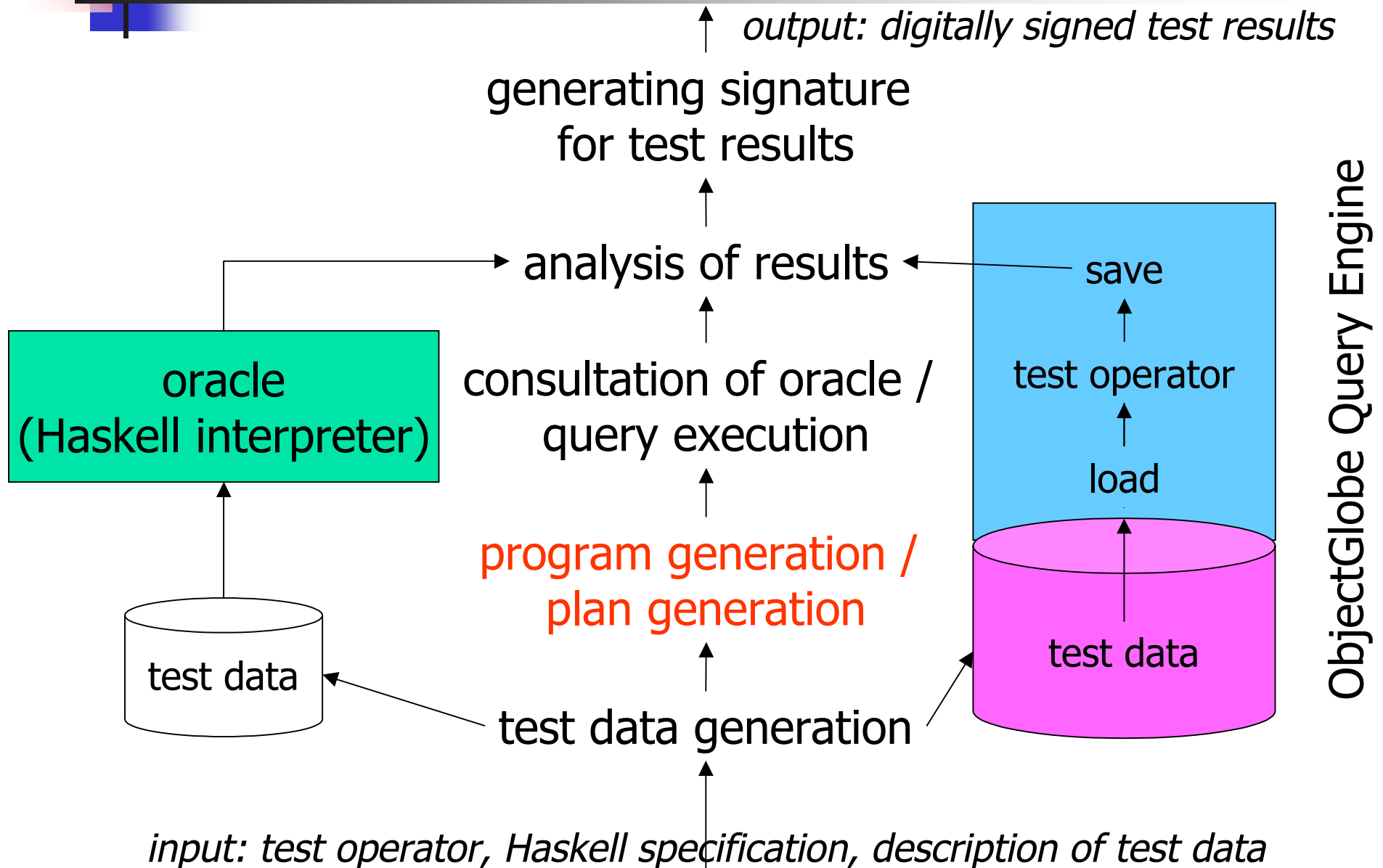
Architecture of OperatorCheck



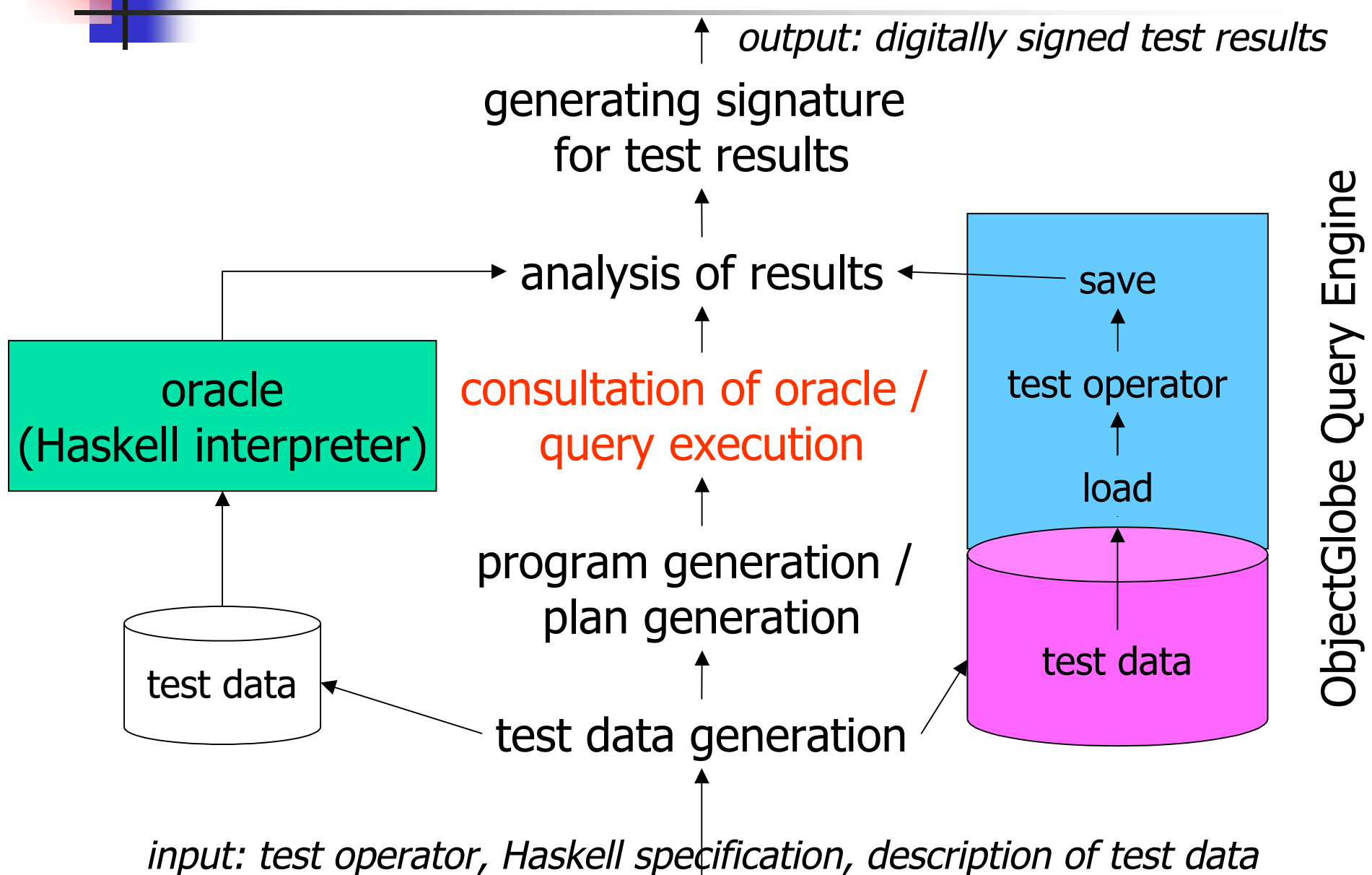
Architecture of OperatorCheck



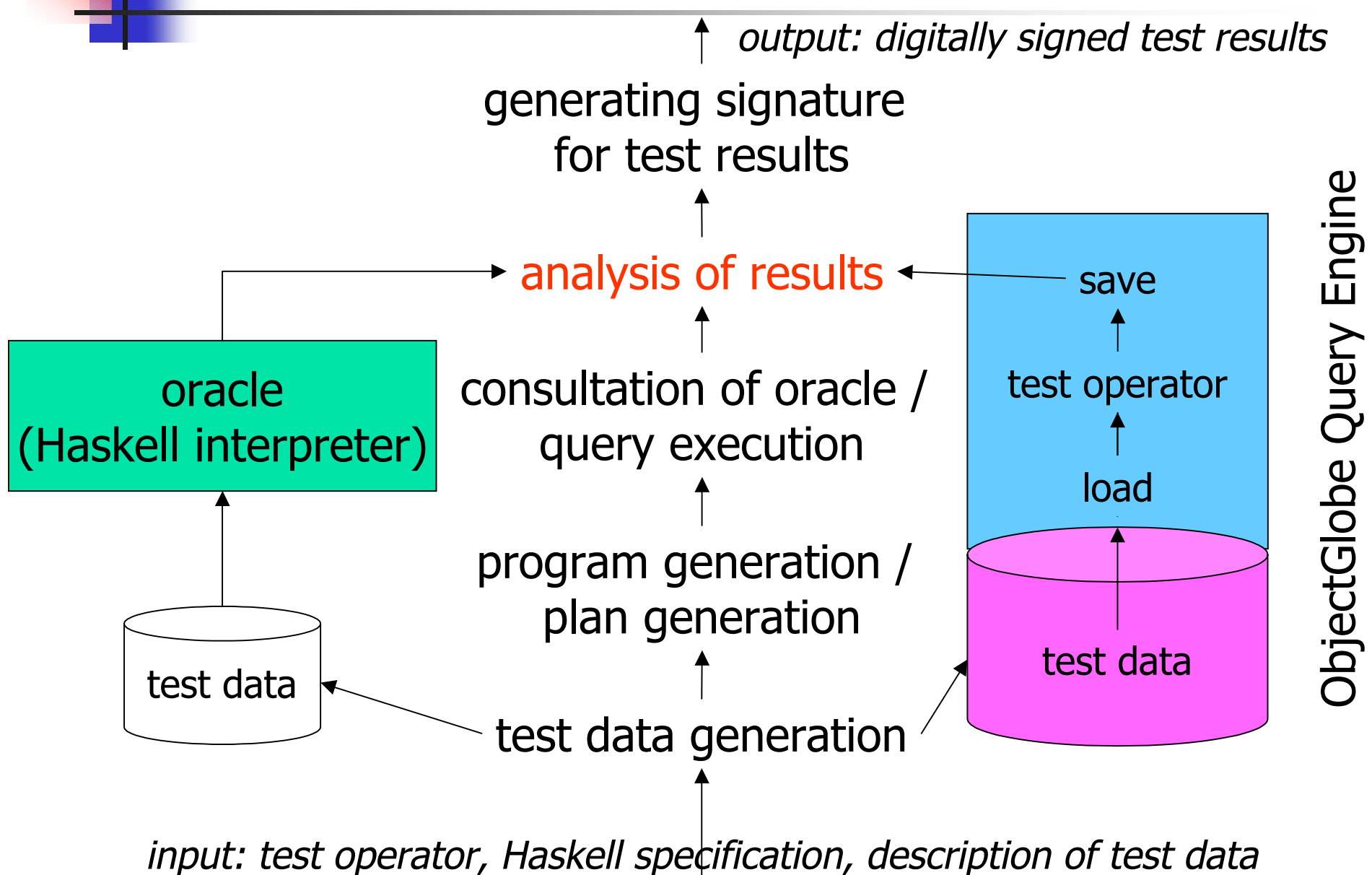
Architecture of OperatorCheck



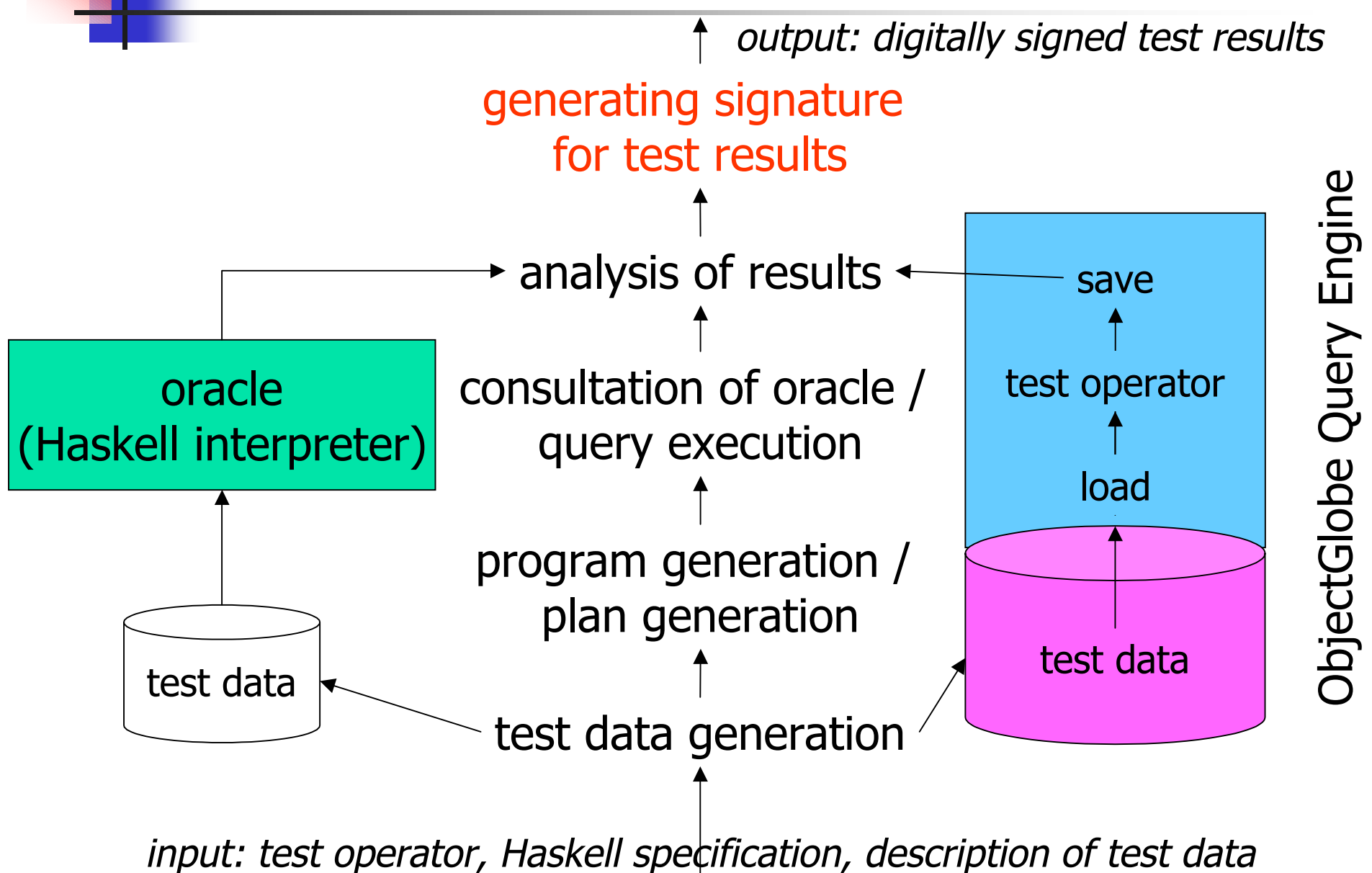
Architecture of OperatorCheck



Architecture of OperatorCheck



Architecture of OperatorCheck



http://www.db.fmi.uni-passau.de/projects/OG/OnlineDemo/operatorcheck.phtml

Homepage der ObjectGlobe-Entwickler - Microsoft Internet Explorer

Adresse http://www.db.fmi.uni-passau.de/projects/OG/OnlineDemo/operatorcheck_skyline.phtml Wechseln zu

 **ObjectGlobe**
Ubiquitous Query Processing on the Internet

Operator Check

Name:

ClassName:

FunctionProvider:

InputRelations:

OutputRelation:

Parameters:

TestdataDirectives:

Test: Correctness Test
 Reference Test
 Benchmark Test

Fertig Internet

<http://www.db.fmi.uni-passau.de/projects/OG/OnlineDemo/operatorcheck.phtml>



Homepage der ObjectGlobe-Entwickler - Microsoft Internet Explorer

Datei Bearbeiten Ansicht Favoriten Extras ?

Adresse <http://www.db.fmi.uni-passau.de/servlets-operatorcheck/operatorcheck?action=start> Wechseln zu

ObjectGlobe

Ubiquitous Query Processing on the Internet

Operator Check

Name: Skyline

Specification:

```
import System
data R = R Double Double deriving (Read, Show)

skyline :: [R] -> [R]
```

```
skyline ss = skyline' ss ss
skyline' [] ts = []
skyline' (s:ss) ts = if dominated s ts
  then skyline' ss ts
  else s:skyline' ss ts
```

main :: IO()
main = do
 [finR, fout] <- getArgs
 fileR <- readFile finR
 relationR <- return ((fst.head.reads) fileR)
 writeFile fout (show (skyline relationR))

Comparison: List (order and count matter)
 Multiset (order ignored)
 Set (order and count ignored)

Options: Show test data and result

Start Test

Fertig Internet

http://www.db.fmi.uni-passau.de/projects/OG/OnlineDemo/operatorcheck.phtml

Homepage der ObjectGlobe-Entwickler - Microsoft Internet Explorer

Datei Bearbeiten Ansicht Favoriten Extras ?

Adresse <http://www.db.fmi.uni-passau.de/servlets-operatorcheck/operatorcheck?action=correctnesstest> Wechseln zu

[Home](#) - [Lokales](#) - [Personen](#)

 **ObjectGlobe**
Ubiquitous Query Processing on the Internet

Operator Check

Generating Haskell program...
Generating ObjectGlobe plan...
Generating test cases...
Running test with Haskell interpreter...
Running test with ObjectGlobe...
Comparing results...

Results match!

[Home](#) -- [\[Informatik\]](#) - [\[Universität Passau\]](#)

Fertig Internet



Advantages/Limitations

- Advantages
 - Improvement of trust
 - Resource stability
 - More reliable query execution
 - Continuously available cycle providers
 - Better result quality
 - ObjectGlobe can renounce runtime monitoring
- Limitations
 - Correctness can not be proved
 - Results depend on intuition of testers
 - Further security measures necessary



Measures during Plan Distribution

- Setup of secure communication channels using SSL and/or TLS
- Authentication of communication partners
- Authentication of users
- Authorization
- Admission control



Runtime Security System

- Based on
 - Java's security architecture
 - Native library
- Tasks
 - Guarantee privacy
 - Protection of cycle providers
 - Guarding
 - Monitoring



Guarding

- Prevention of unauthorized resource access
- Access to temporary memory
- Prevention of access to ObjectGlobe components
- Isolation of user-defined operators



Monitoring

- Monitored resources
 - CPU
 - Primary and secondary memory
 - Data volume produced by operators
 - Number of temporary files
- Dynamically adapted limits
- Operators are terminated upon limit violations



Related Work

- Extensible database systems:
 - POSTGRES, Predator, Jaguar
 - Oracle, DB2
- Braumandl et.al.: ObjectGlobe: Ubiquitous Query Processing on the Internet, VLDBJ 2001
- Seshadri et.al.: Secure and Portable Database Extensibility, SIGMOD 1998
- Dalton et.al.: An Operating System Approach to Securing E-Services, Communications of the ACM, 2001
- Weikum: The Web in 2010: Challenges and Opportunities for Database Research, Springer, 2001



Conclusions

- Security requirements of cycle providers and users
- ObjectGlobe as an Example
- Multilevel security architecture
 - OperatorCheck server
 - Measures during plan distribution
 - Runtime security system