

L6

Application Programming

Eugene Wu

Topics

Interfacing with applications

Database APIs (DBAPIS)

Cursors

SQL != Programming Language

Not a general purpose programming language

- Tailored for data access/manipulation

- Easy to optimize and parallelize

- Can't perform “business logic”

Options

- I. Extend SQL, make it Turing Complete

- goes from simple, easy to analyze to complex :(

2. Extend existing languages to understand SQL natively

3. Provide an API between programming languages and DBMSes

Many Database API options

Fully embed into language (embedded SQL)

Low-level library with core database calls (DBAPI)

Object-relational mapping (ORM)

Ruby on rails, django, Hibernate, sqlalchemy, etc

define database-backed classes

magically maps between database rows & objects

magic is a double edged sword

Embedded SQL

Extend host language (python) with SQL syntax

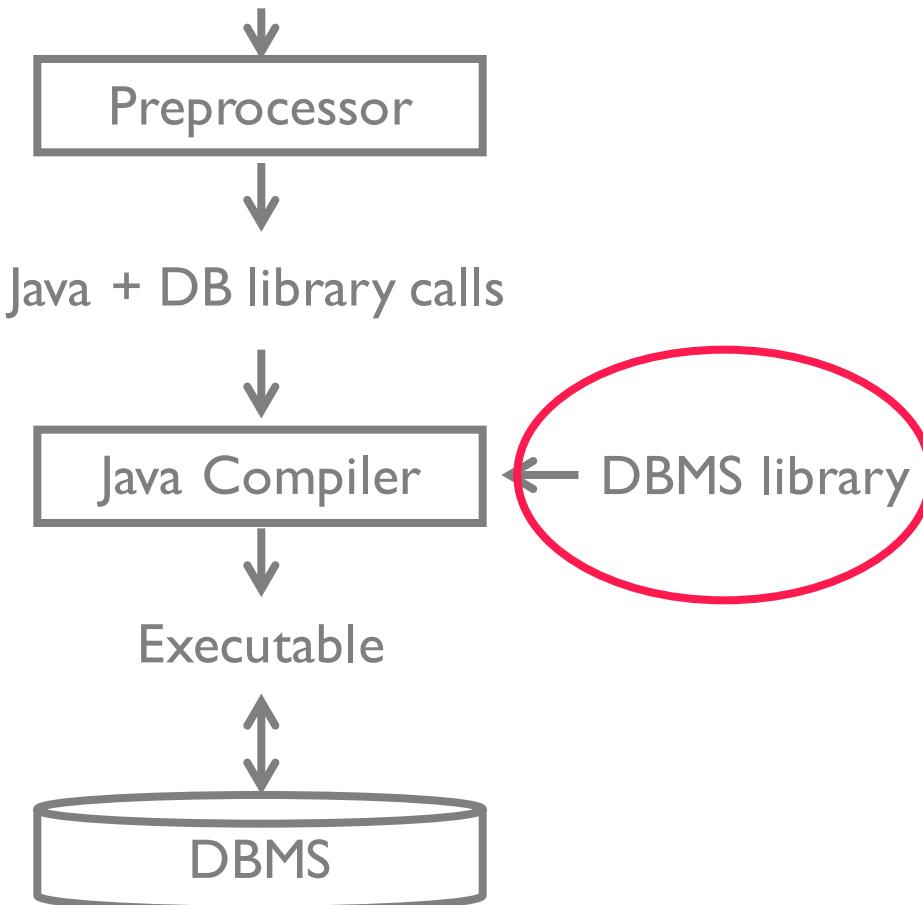
e.g., `EXEC SQL sql-query`

goes through a preprocessor

Compiled into program that interacts with DBMS directly

Embedded SQL

Java + embedded SQL



```
...  
if (user == 'admin') {  
    EXEC SQL select * ...  
}  
else {  
    ...  
}
```

What does a library need to do?

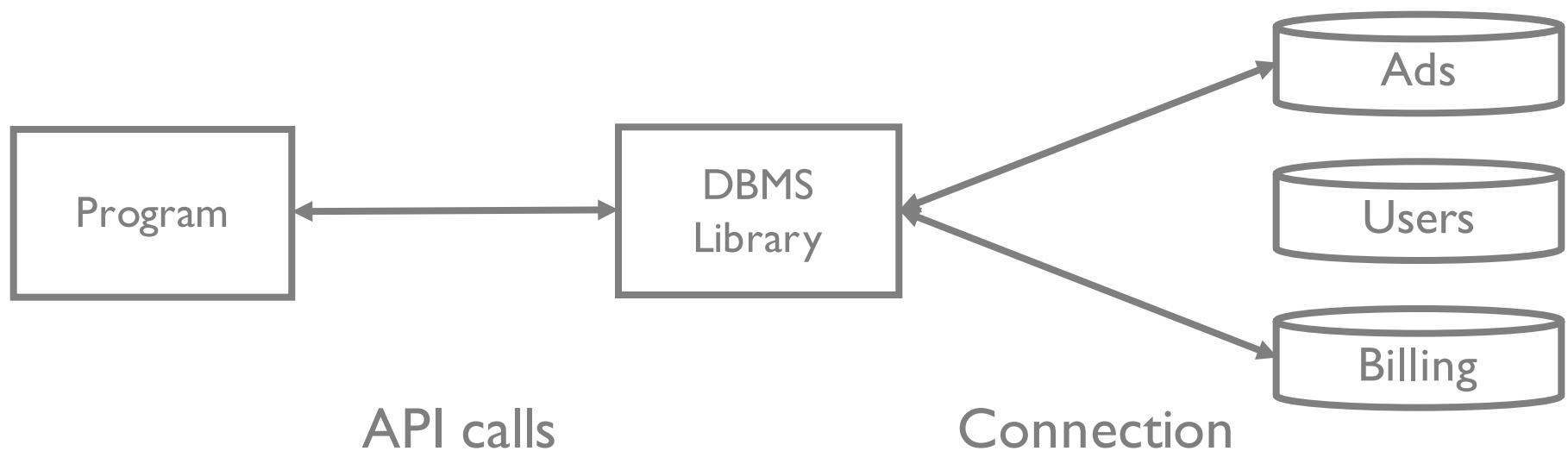
Single interface to possibly multiple DBMS engines

Connect to a database

Manage transactions (later)

Map objects between host language and DBMS

Manage query results



Overview

Library Components

Impedance Mismatches

1. Types
2. Classes/objects
3. Result sets
4. Functions
5. Constraints

Engines

Abstraction for a database engine
tries to hide DBMS language differences

driver://username:password@host:port/database

```
from sqlalchemy import create_engine
db1 = create_engine(
    "postgresql://localhost:5432/testdb"
)

db2 = create_engine("sqlite:///testdb.db")
// note: sqlite has no host name (sqlite:///)
```

Connections

Before running queries need to create a connection

- Tells DBMS to allocate resources for the connection
- Relatively expensive to set up, libraries often cache connections for future use
- Defines scope of a transaction (later)

```
conn1 = db1.connect()  
conn2 = db2.connect()
```

Should close connections when done! Otherwise resource leak.

Query Execution

```
conn1.execute("update table test set a = 1")
conn1.execute("update table test set s = 'wu'")
```

Query Execution

```
foo = conn1.execute("select * from big_table")
```

Challenges

What is the return type of execute()?

Type impedance

How to pass data between DBMS and host language?

Can we only pass data between DBMS and host language?

(Type) Impedance Mismatch

SQL standard defines mappings between SQL and several languages

Most libraries can deal with common types

SQL types	C types	Python types
CHAR(20)	char[20]	str
INTEGER	int	int
SMALLINT	short	int
REAL	float	float

What about complex objects { x:'I', y:'hello' }

(Class) Impedance Mismatch

Programming languages usually have classes
Setting an attribute in User should save it

```
user.name = "Dr Seuss"  
user.job = "writer"
```

```
class User { ... }  
class Employee extends User { ... }  
class Salaries {  
    Employee worker;  
    ...  
}
```

Object Relational Mappings designed to address this

Query Execution

How to pass values into a query?

```
Users(id int serial, name text)
```

```
name = "eugene"
```

```
conn1.execute("""  
    INSERT INTO users(name)  
    VALUES(<what to put here??>)""")
```

Query Execution

How to pass values into a query?

```
Users(id int serial, name text)
```

```
name = "eugene"
```

```
conn1.execute("""  
    INSERT INTO users(name)  
    VALUES(''{name}'')""".format(name=name))
```

Why is this a *really* bad idea?

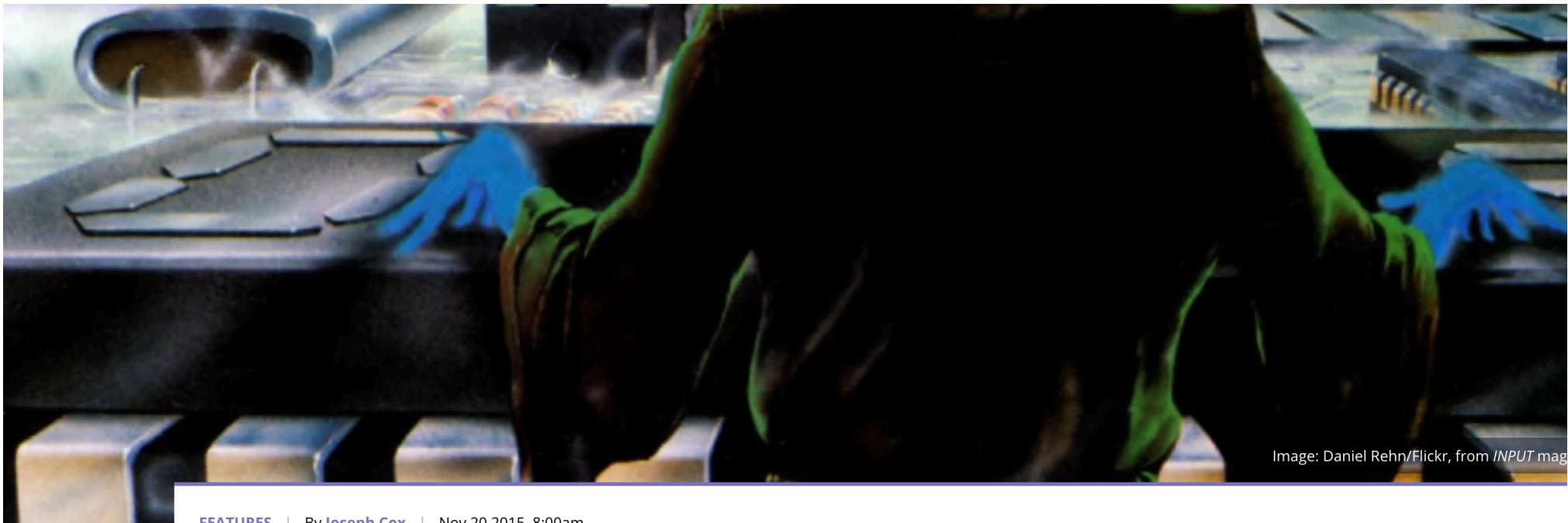


Image: Daniel Rehn/Flickr, from INPUT mag

FEATURES | By Joseph Cox | Nov 20 2015, 8:00am

The History of SQL Injection, the Hack That Will Never Go Away

Over 15 years after it was first publicly disclosed, SQL injection is still the number one threat to websites.

https://motherboard.vice.com/en_us/article/aekzez/the-history-of-sql-injection-the-hack-that-will-never-go-away

FILTER



Running an SQL Injection Attack - Computerphile

Computerphile 1.6M views • 2 years ago

Just how bad is it if your site is vulnerable to an **SQL Injection**? Dr Mike Pound shows us how they work.
Cookie Stealing: ...



SQL Injection Attack Tutorial (2019)

HackHappy 76K views • 8 months ago

SQL Injection attacks are still as common today as they were ten years ago. Today I'll discuss what are SQLi and how you can ...

CC



SQL Injection Basics Demonstration

Imperva 360K views • 9 years ago

Imperva presents an educational video series on Application and Database Attacks in High Definition (HD)

6:45



Hacking Websites with SQL Injection - Computerphile

Computerphile 1.4M views • 5 years ago

Websites can still be hacked using **SQL injection** - Tom explains how sites written in PHP (and other languages too) can be ...

CC

⋮



DEFCON 17: Advanced SQL Injection

Christiaan008 220K views • 8 years ago

Speaker: Joseph McCray Founder of Learn Security Online **SQL Injection** is a vulnerability that is often missed by web application ...

Detour: SQL Injections

<http://tinyurl.com/w4111injection>

code on github:

w4111/w4111.github.io/src/injection/

bad form

Add your name

1 eugene
2 wu

```
@app.route('/', methods=["POST", "GET"])
def index():
    if request.method == "POST":
        name = request.form['name']
        q = "INSERT INTO bad_table(name) VALUES('%s');" % name
        print q
        g.conn.execute(q)
```

Detour: SQL Injections

If we submit:

'); DELETE FROM bad_table; --

Query is

INSERT INTO bad_table(name) VALUES('');
DELETE FROM bad_table; -- '');

```
@app.route('/', methods=["POST", "GET"])
def index():
    if request.method == "POST":
        name = request.form['name']
        q = "INSERT INTO bad_table(name) VALUES(%s);" % name
        print q
        g.conn.execute(q)
```

Detour: SQL Injections

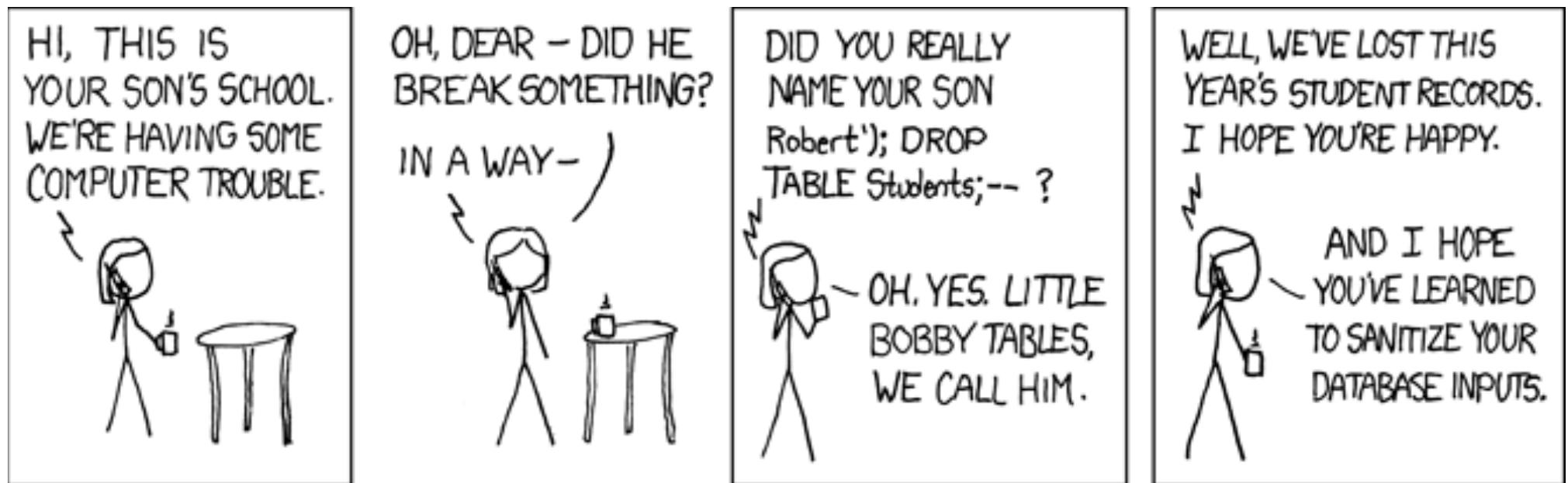
Safe implementation

Pass form values as arguments to the execute() function

Library sanitizes inputs automatically (and correctly!)

```
@app.route('/safe/', methods=["POST", "GET"])
def safe_index():
    if request.method == "POST":
        name = request.form['name']
        q = "INSERT INTO bad_table(name) VALUES(%s);"
        print q
        g.conn.execute(q, (name,))
```

Detour: SQL Injections



Project: You'll need to protect against simple SQL injections

More examples:<https://corenumb.wordpress.com/2016/05/14/mr-robot-blind-sql-injection-vulnerability/>

Query Execution

Pass *sanitized* values to the database

```
args = ('Dr Seuss', '40')
conn1.execute(
    "INSERT INTO users(name, age) VALUES(%s, %s)",
    args)
```

Pass in a tuple of query arguments

DBAPI library will *properly escape* input values

Most libraries support this

Never construct raw SQL strings

(results) Impedance Mismatch

SQL relations and results are sets of records

What is the type of table?

```
table = execute("SELECT * FROM big_table")
```

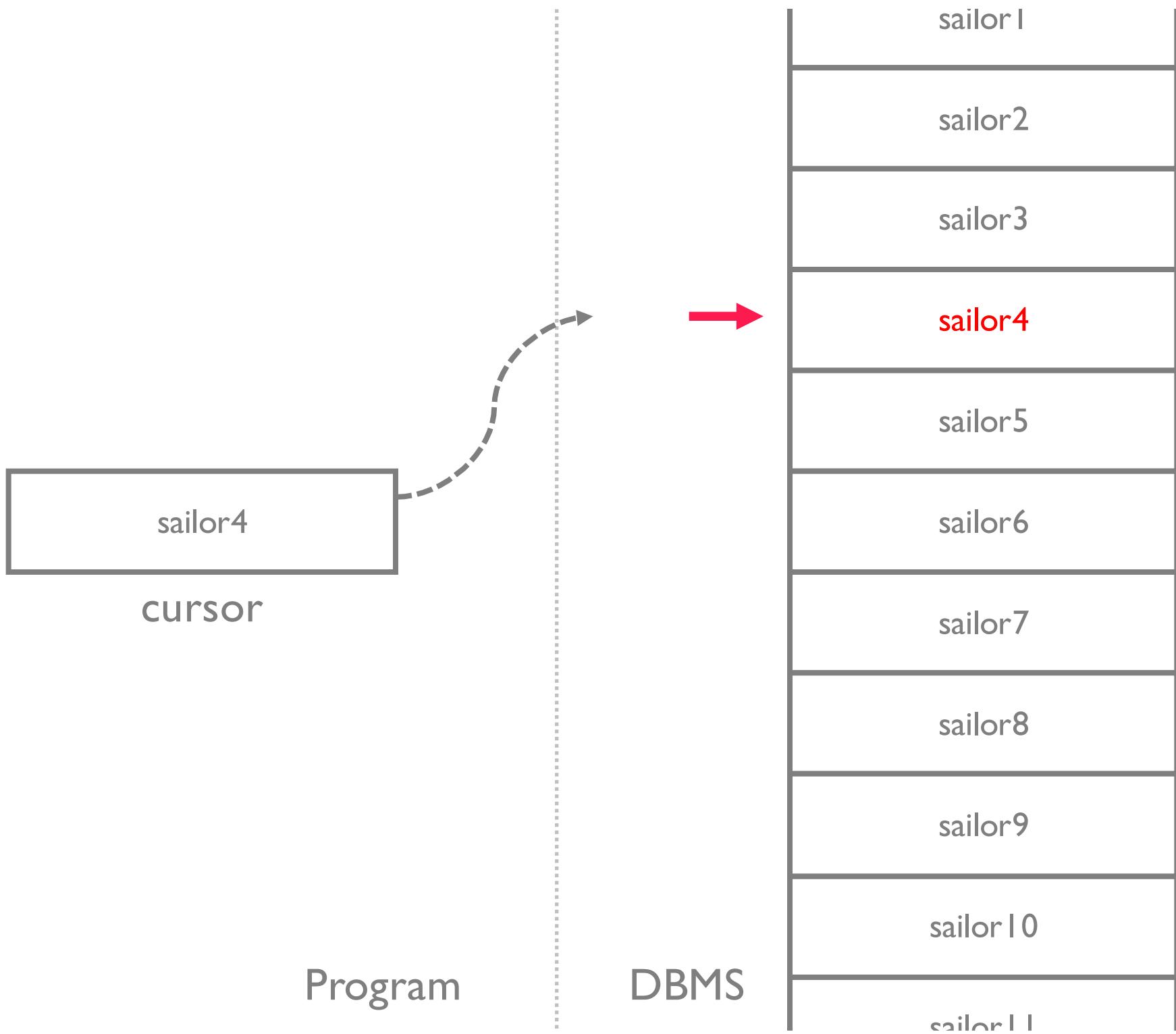
Cursor over the Result Set

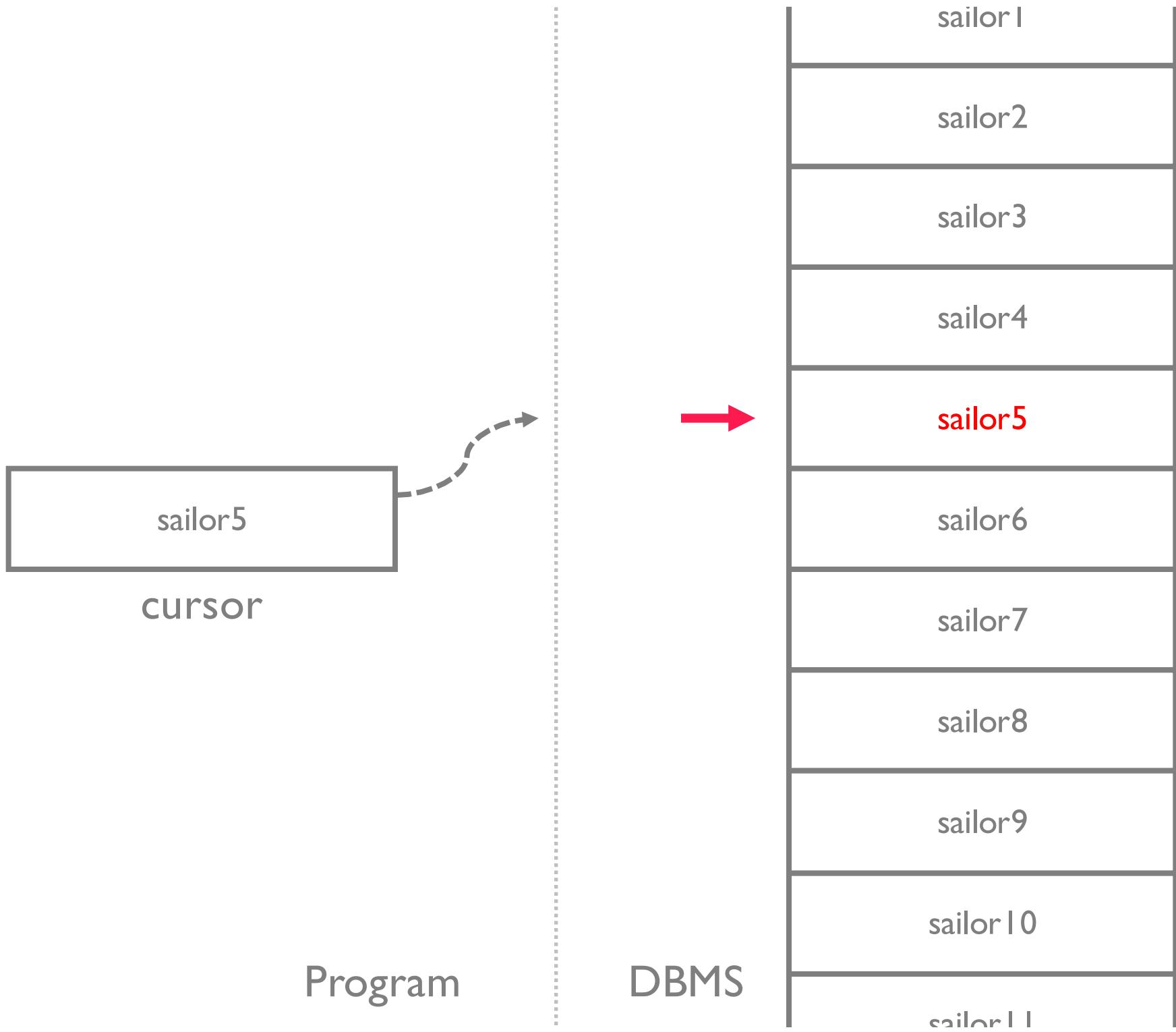
similar to an iterator interface

Note: relations are unordered!

Cursors have no ordering guarantees

Use ORDER BY to ensure an ordering





(results) Impedance Mismatch

Cursor similar to an iterator (next() calls)

```
cursor = execute("SELECT * FROM bigtable")
```

Cursor attributes/methods (logical)

rowcount

keys()

previous()

next()

get(idx)

(results) Impedance Mismatch

Cursor similar to an iterator (next() calls)

```
cursor = execute("SELECT * FROM bigtable")
cursor.rowcount() # 1000000
cursor.fetchone() # (0, 'foo', ...)
for row in cursor: # iterate over the rest
    print row
```

Actual Cursor methods vary depending on implementation

(functions) Impedance Mismatch

What about functions?

```
def add_one(val):  
    return val + 1
```

```
conn1.execute("SELECT add_one(1)")
```

Would need to embed a language runtime into DBMS

Many DBMSes support runtimes e.g., python

Can register User Defined Functions (UDFs)

(constraints) Impedance Mismatch

DB-style constraints often as conditionals or exceptions
Constraints often duplicated throughout program

JS age = get_age_input();
 if (age > 100 or age < 18)
 show_error("age should be 18 - 100");

DBMS CREATE TABLE Users (
 ...
 age int CHECK(age >= 18 and age <= 100)
 ...
)

(constraints) Impedance Mismatch

Some ORMs try to have one place to define constraints

```
class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name  = models.CharField(max_length=30, null=True)

CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name"  varchar(30)
);
```

Some Useful Names

DBMS vendors provide libraries for most libraries

Two heavyweights in enterprise world

ODBC Open DataBase Connectivity

Microsoft defined for Windows libraries

JDBC Java DataBase Connectivity

Sun developed as set of Java interfaces

`java.sql.*`

`javax.sql.*` (recommended)

Modern Database APIs

DryadLinq, SparkSQL

DBMS executor in same language (dotNET, Spark) as app code

what happens to language impedance?

what happens to exception handling?

what happens to host language functions?

```
val lines = spark.textFile("logfile.log")
val errors = lines.filter(_ startsWith "Error")
val msgs = errors.map(_.split("\t")(2))

msgs.filter(_ contains "foo").count()
```

Summary

DBAPIs

- Impedance mismatch

- Cursors

- SQL injection

Some hard queries

- More in the HW

Windows are optional material

SQL Injection: only what's in slides

What to Understand

Impedance mismatch
examples, and possible solutions

SQL injection and how to protect

The different uses of a DBAPI

Why Embedded SQL is no good

What good are cursors?

