Fighting Bad Database Apps

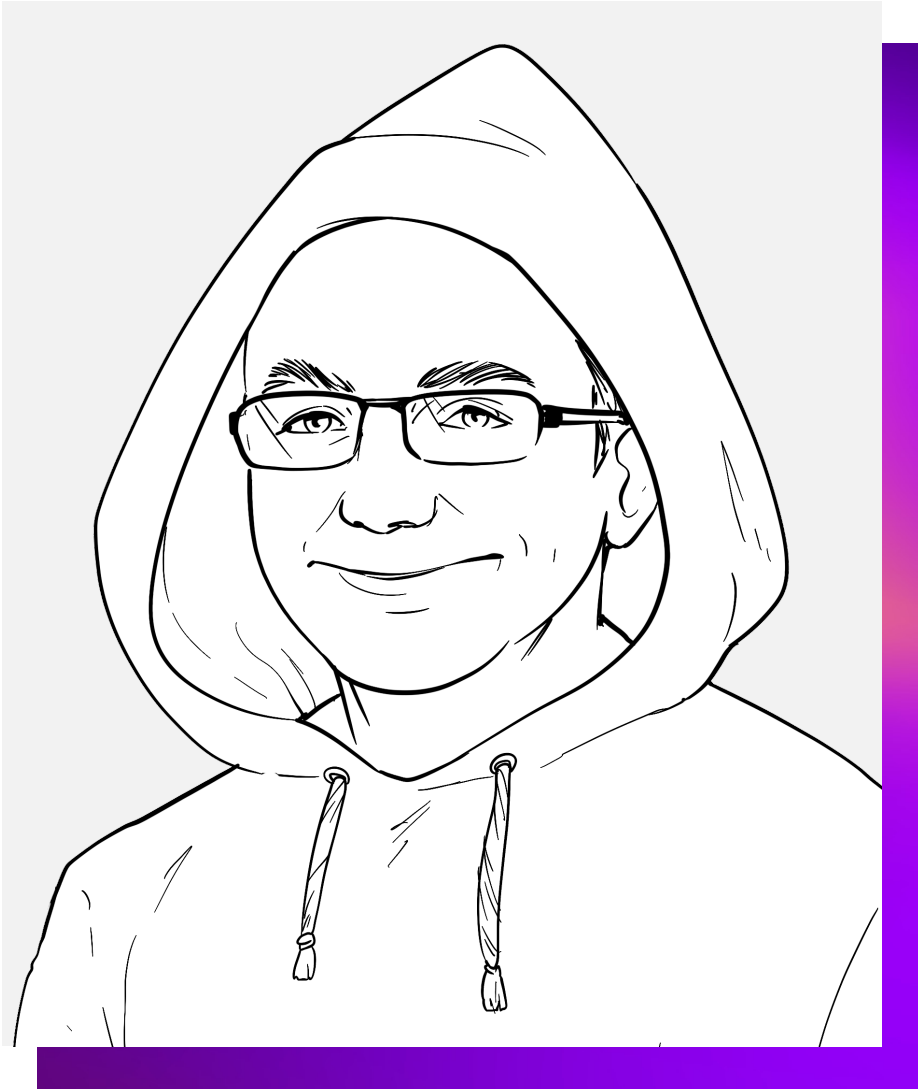
Philipp Salvisberg
12th May 2023

Philipp Salvisberg

Data Engineering Principal

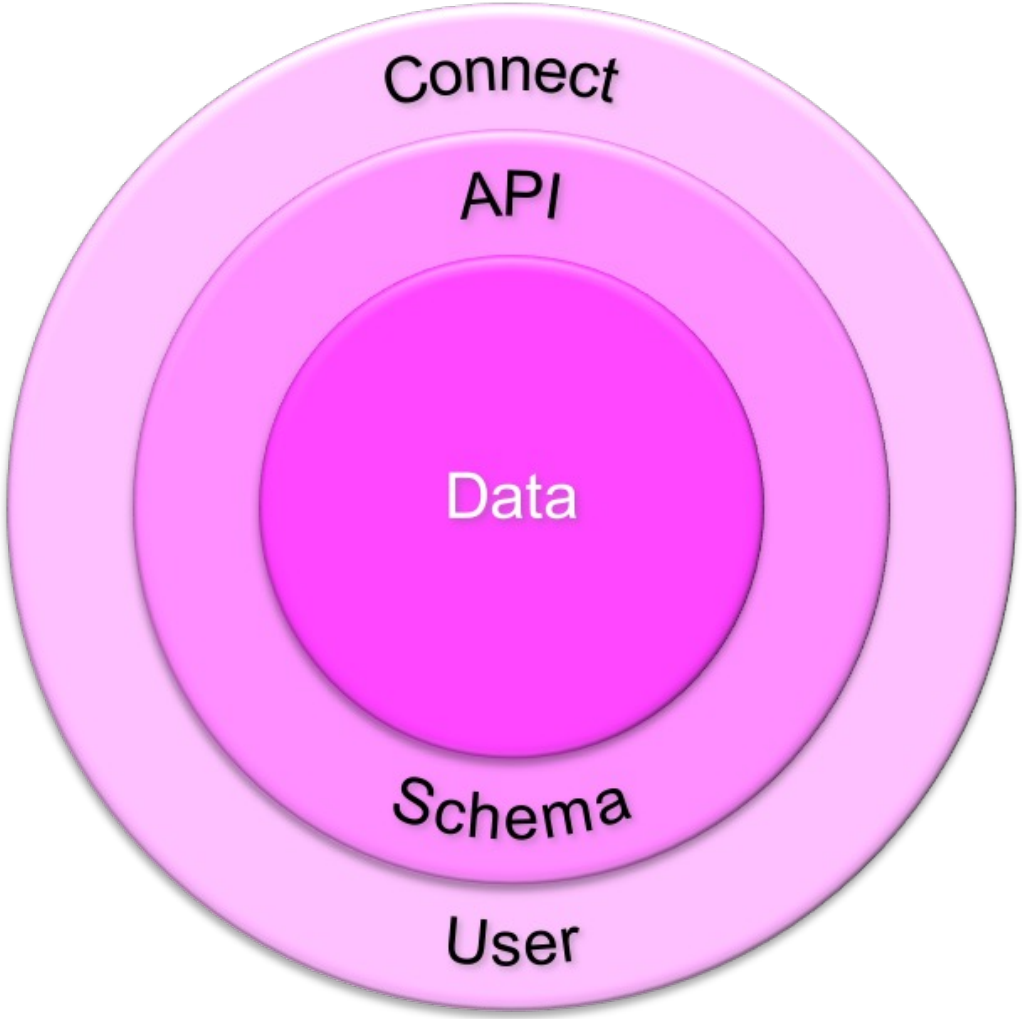
- Database Centric Development
- Model Driven Software Development
- Open-Source Development

philipp.salvisberg@accenture.com
https://www.salvis.com/blog



Introduction

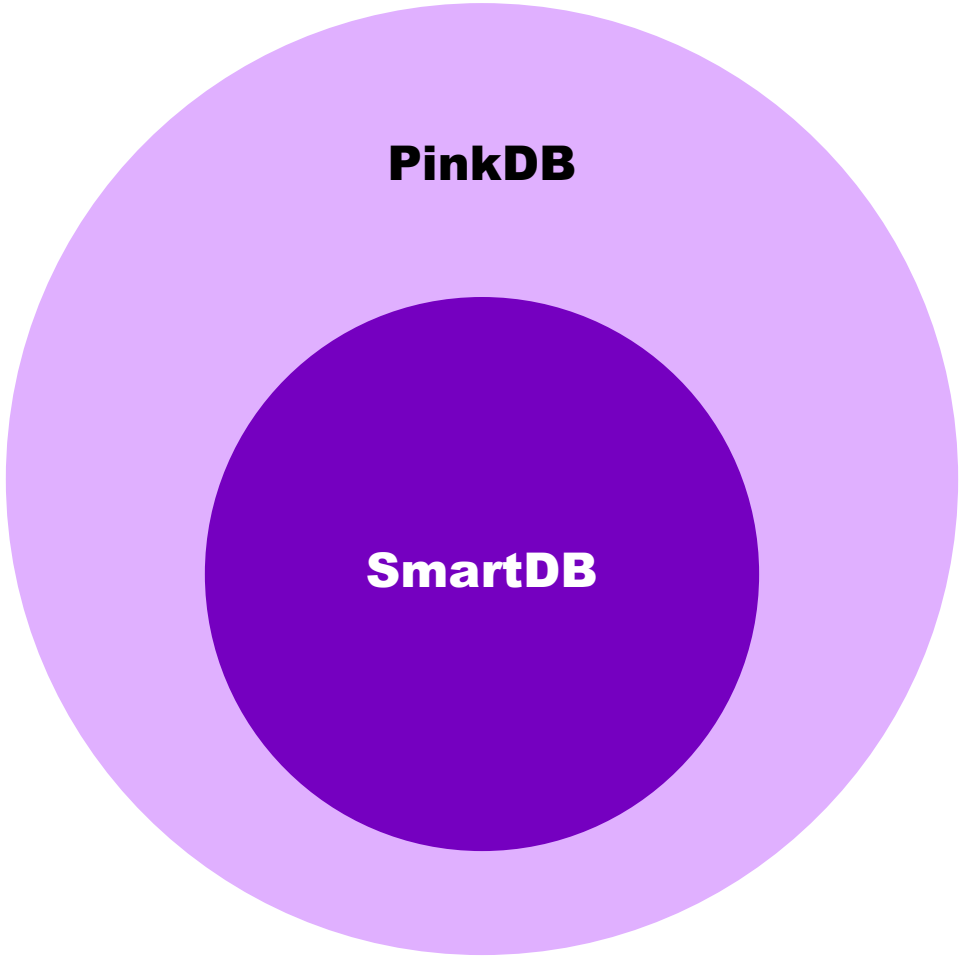
What Is PinkDB?



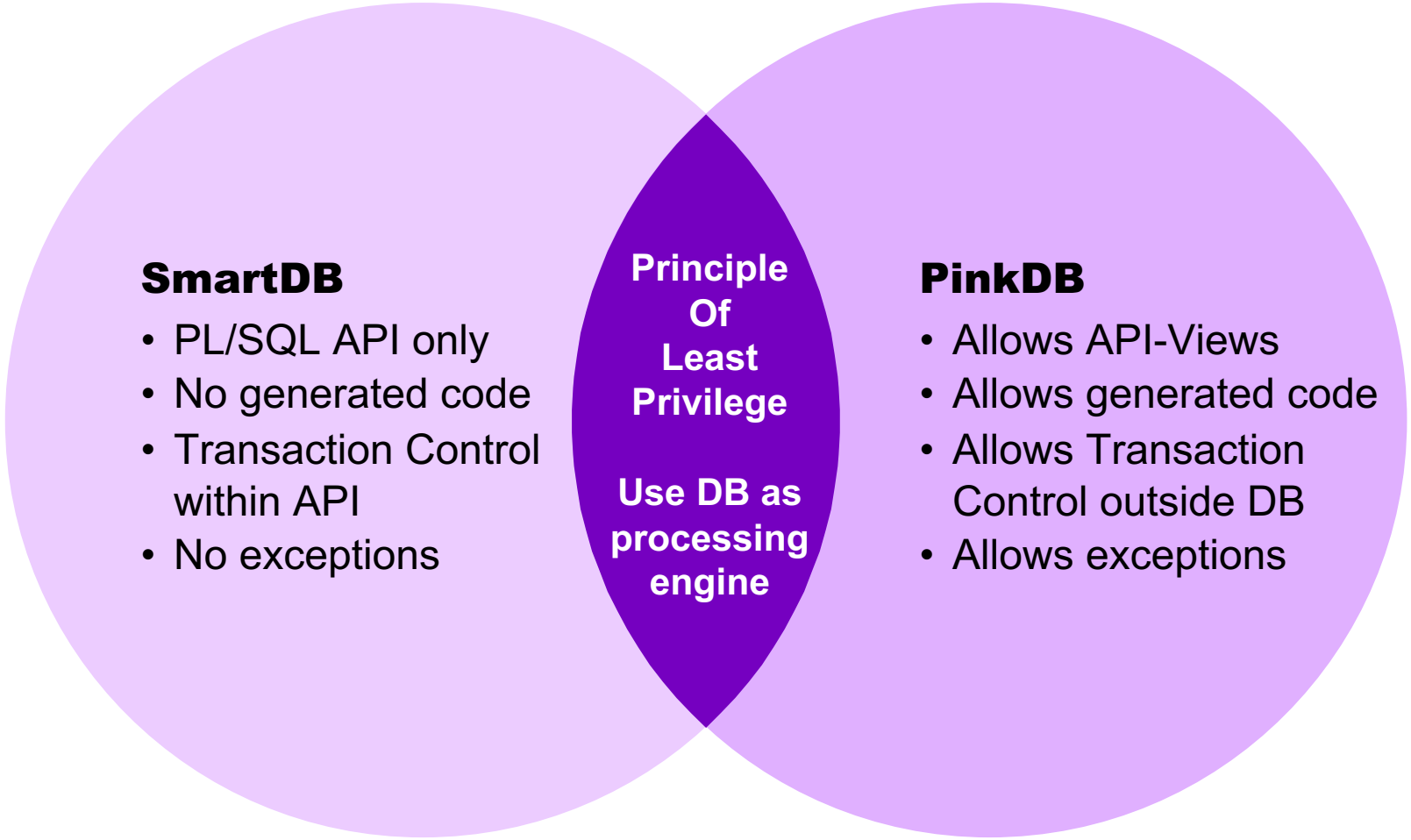
"(…) **application architecture** for database centric applications. It is focusing on relational database systems and is vendor neutral. The principles are **based on** the ideas of **SmartDB**, with some adaptions that make PinkDB easier to apply in existing development environments. (…)"

https://www.salvis.com/blog/2018/07/18/the-pink-database-paradigm-pinkdb/

SmartDB vs. PinkDB – Used DB Features



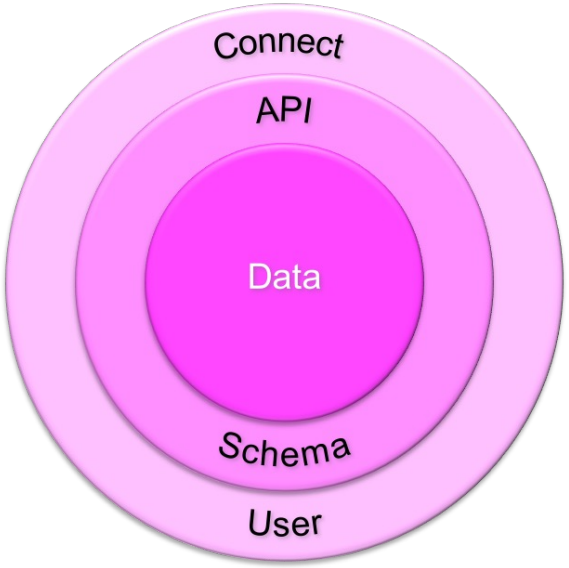
SmartDB vs. PinkDB – Enforced Principles



Principle of Least Privilege

"The principle means **giving** a user account or process **only those privileges** which are essential **to perform** its **intended function**."

https://en.wikipedia.org/wiki/Principle_of_least_privilege

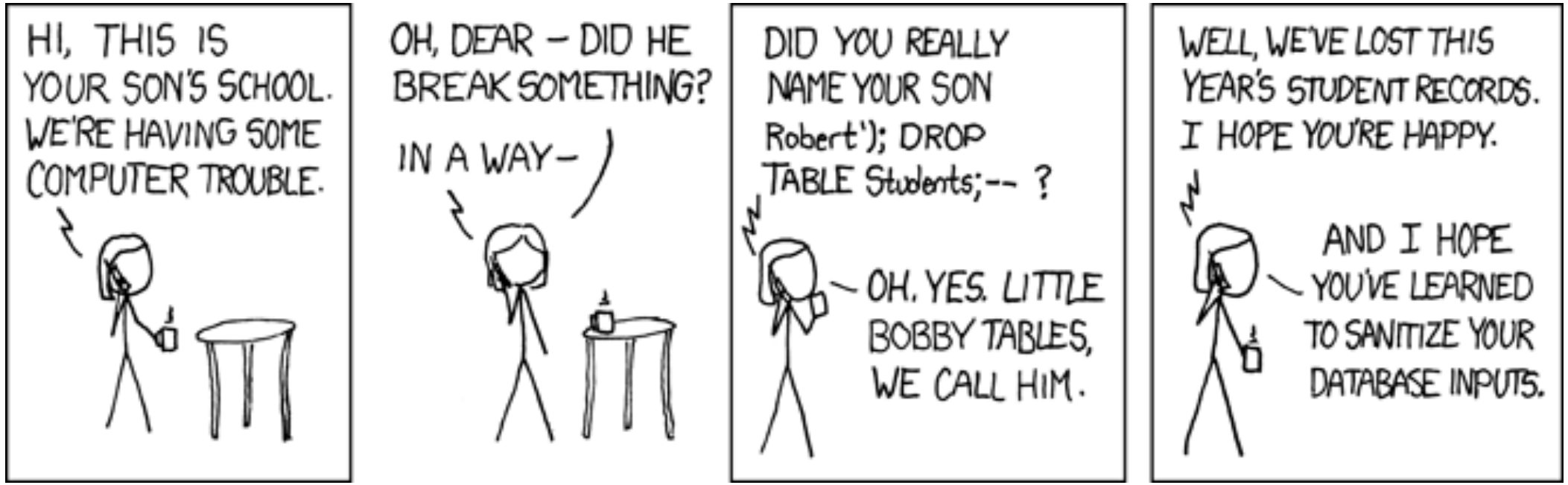


"**Minimizes** the **attack surface** (…)
Reduces malware propagation (…)
Improves operational performance (…)
Safeguards against human error (…)"

https://www.paloaltonetworks.com/cyberpedia/what-is-the-principle-of-least-privilege

SQL Injection

Little Bobby Tables



Source: https://xkcd.com/327, https://www.explainxkcd.com/wiki/index.php/Little_Bobby_Tables

Dynamic SQL – Flexibility vs. Security

```
create or replace package body pkg is
   procedure exec_sql(in_sql in varchar2) is
   begin
      execute immediate in_sql;
   end exec_sql;
end pkg;
```

Dynamic SQL – Unasserted Input

```
create or replace package body pkg is
    function f (in_table_name in varchar2) return boolean as
        co_templ      constant varchar2(4000 byte) :=
                          'DROP TABLE #in_table_name# PURGE';
        l_table_name varchar2(128 byte);
        l_sql        varchar2(4000 byte);
    begin
        l_table_name := in_table_name;
        l_sql := replace(co_templ, '#in_table_name#', l_table_name);
        execute immediate l_sql;
        return true;
    end f;
end pkg;
```

Dynamic SQL – Assert Input

```
create or replace package body pkg is
    function f (in_table_name in varchar2) return boolean as
        co_templ      constant varchar2(4000 byte) :=
                          'DROP TABLE #in_table_name# PURGE';
        l_table_name varchar2(128 byte);
        l_sql        varchar2(4000 byte);
    begin
        l_table_name := sys.dbms_assert.enquote_name(in_table_name);
        l_sql := replace(co_templ, '#in_table_name#', l_table_name);
        execute immediate l_sql;
        return true;
    end f;
end pkg;
```

Dynamic SQL – Without Binds

```
create or replace package body pkg is
   function sum_sal(in_dname in varchar2) return number is
      co_sql   constant clob             := q'[
         select sum(emp.sal)
           from emp join dept on emp.deptno = dept.deptno
          where dept.dname = '#dname#'
      ]';
      co_dname constant dept.dname%type := in_dname;
      l_sql    clob;
      l_result number;
   begin
      l_sql := replace(co_sql, '#dname#', co_dname);
      execute immediate l_sql into l_result;
      return l_result;
   end sum_sal;
end pkg;
```

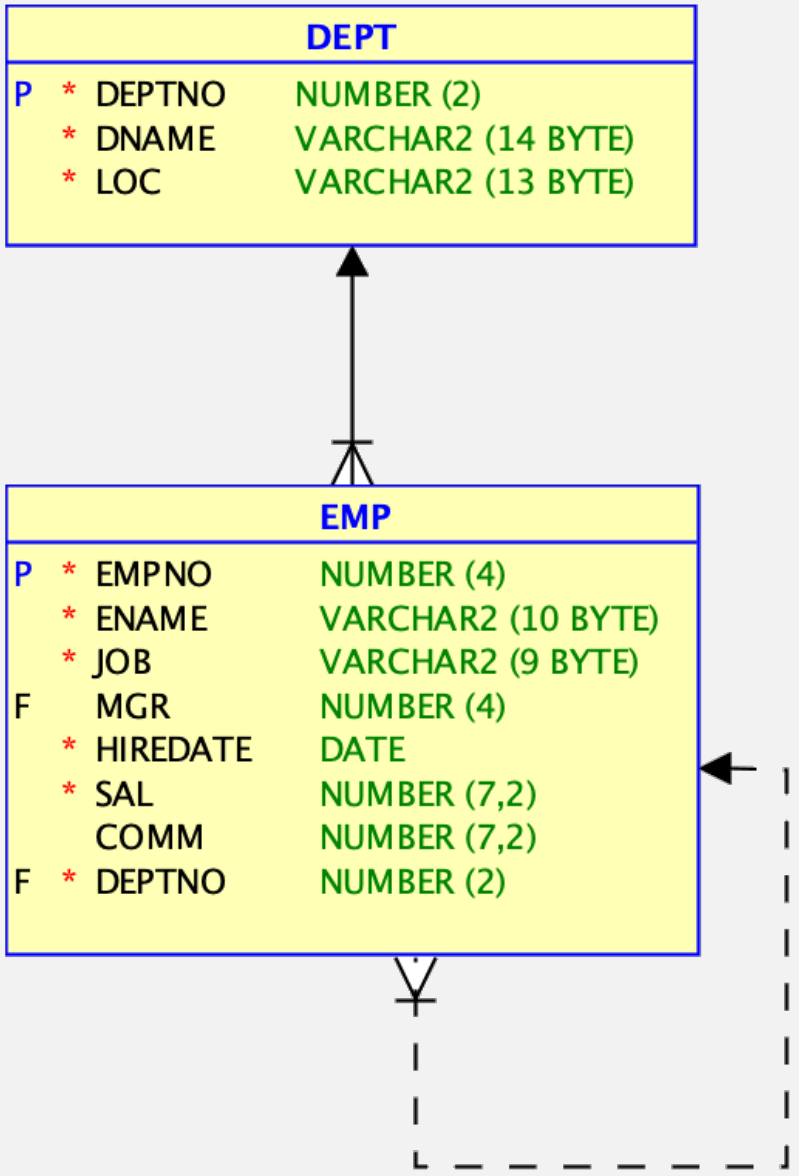
Dynamic SQL – With Binds

```
create or replace package body pkg is
   function sum_sal(in_dname in varchar2) return number is
      co_sql   constant clob             := q'[
         select sum(emp.sal)
           from emp join dept on emp.deptno = dept.deptno
          where dept.dname = :dname
      ]';
      co_dname constant dept.dname%type := in_dname;
      l_result number;
   begin
      execute immediate co_sql into l_result using co_dname;
      return l_result;
   end sum_sal;
end pkg;
```

Static SQL – Auto-Binds

```
create or replace package body pkg is
   function sum_sal(in_dname in varchar2) return number is
      co_dname constant dept.dname%type := in_dname;
      l_result number;
   begin
      select sum(emp.sal)
        into l_result
        from emp join dept on emp.deptno = dept.deptno
       where dept.dname = co_dname;
      return l_result;
   end sum_sal;
end pkg;
```

Finding Violations

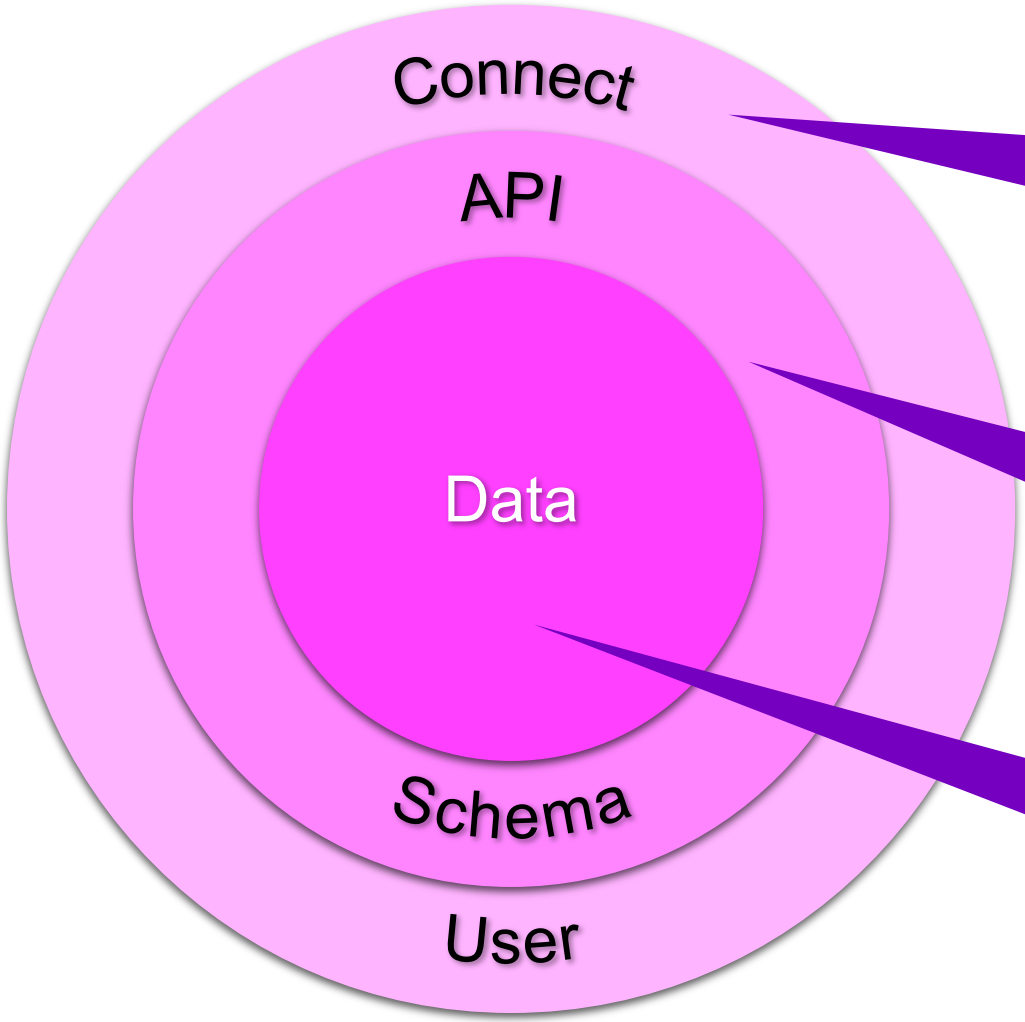


Demo App

- Based on Scott's dept/emp
- Converted to a PinkDB app
- PinkDB and PoLP tests

Source: https://github.com/PhilippSalvisberg/fighting-bad-db-apps

Database Objects



User **app_connect**:

No objects, only access to API

Schema **app**:

API-Views departments, employees, department_salaries
API-Packages department_api, employee_api

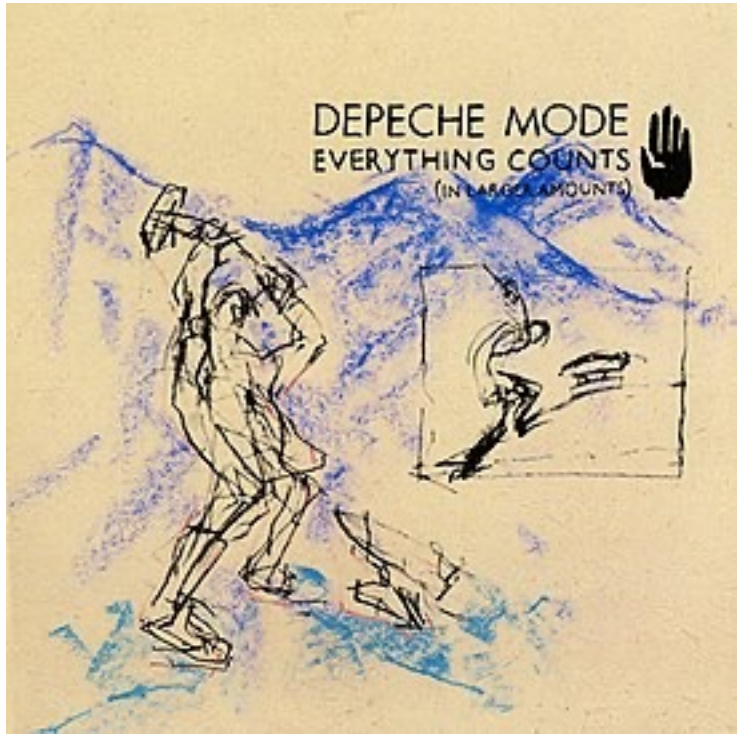
Schema **app**:

Tables dept, emp

Key Messages

Everything Counts

- **Performance**
  - "Everything counts in large amounts"
  - When optimizing the runtime
  - Data, sessions, calls, …
- **Security**
  - "Everything counts"
  - When minimizing the attack surface
  - Access, privileges, objects, data, …
  - Different needs for dev, test, prod
  - Follow Principle of Least Privileges



Source: https://en.wikipedia.org/wiki/Everything_Counts

Thank You