The Educational Conference For Oracle Techology Users

ODTUG Kscope23

aurora, co june 25-29

Interested in volunteering?
Scan the code to sign up

Don't Forget To
Fill Out Your Evals

The Right API for a PinkDB Application

Philipp Salvisberg
28th June 2023

Philipp Salvisberg

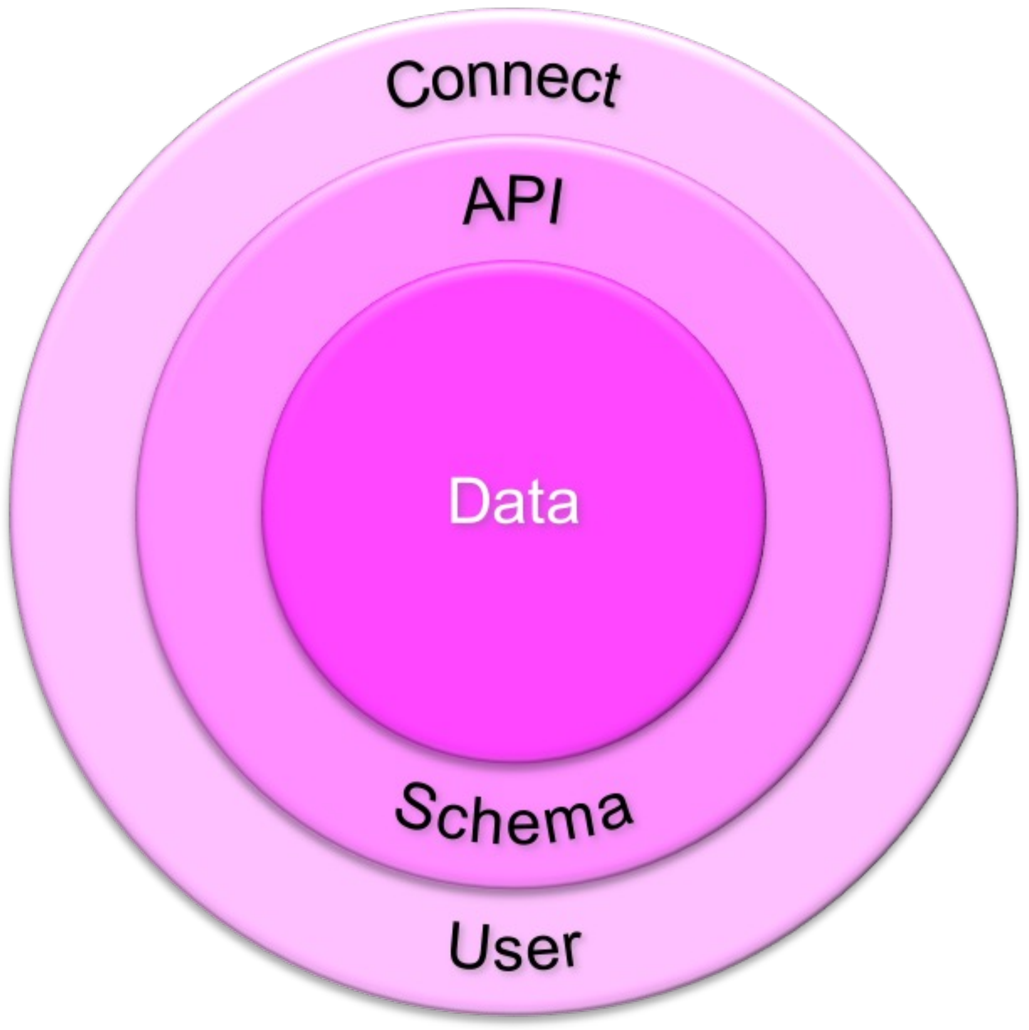
Data Engineering Principal

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

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

Introduction

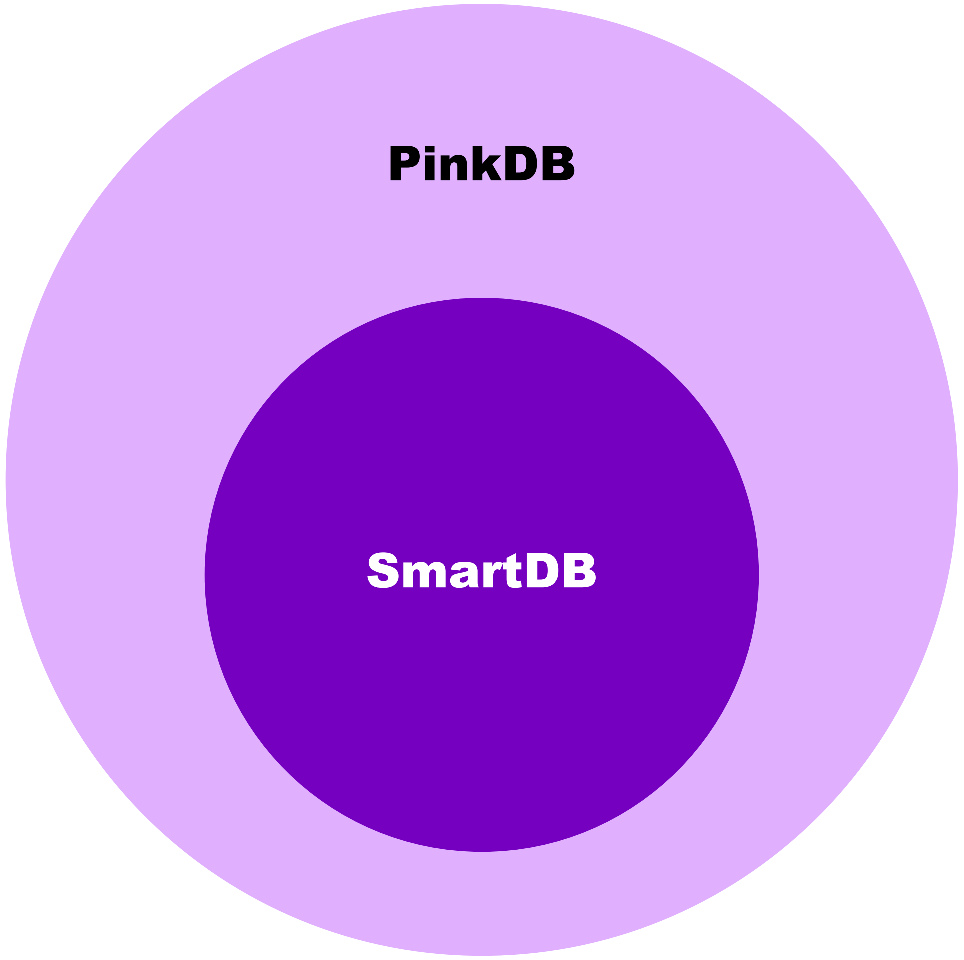
What Is PinkDB?



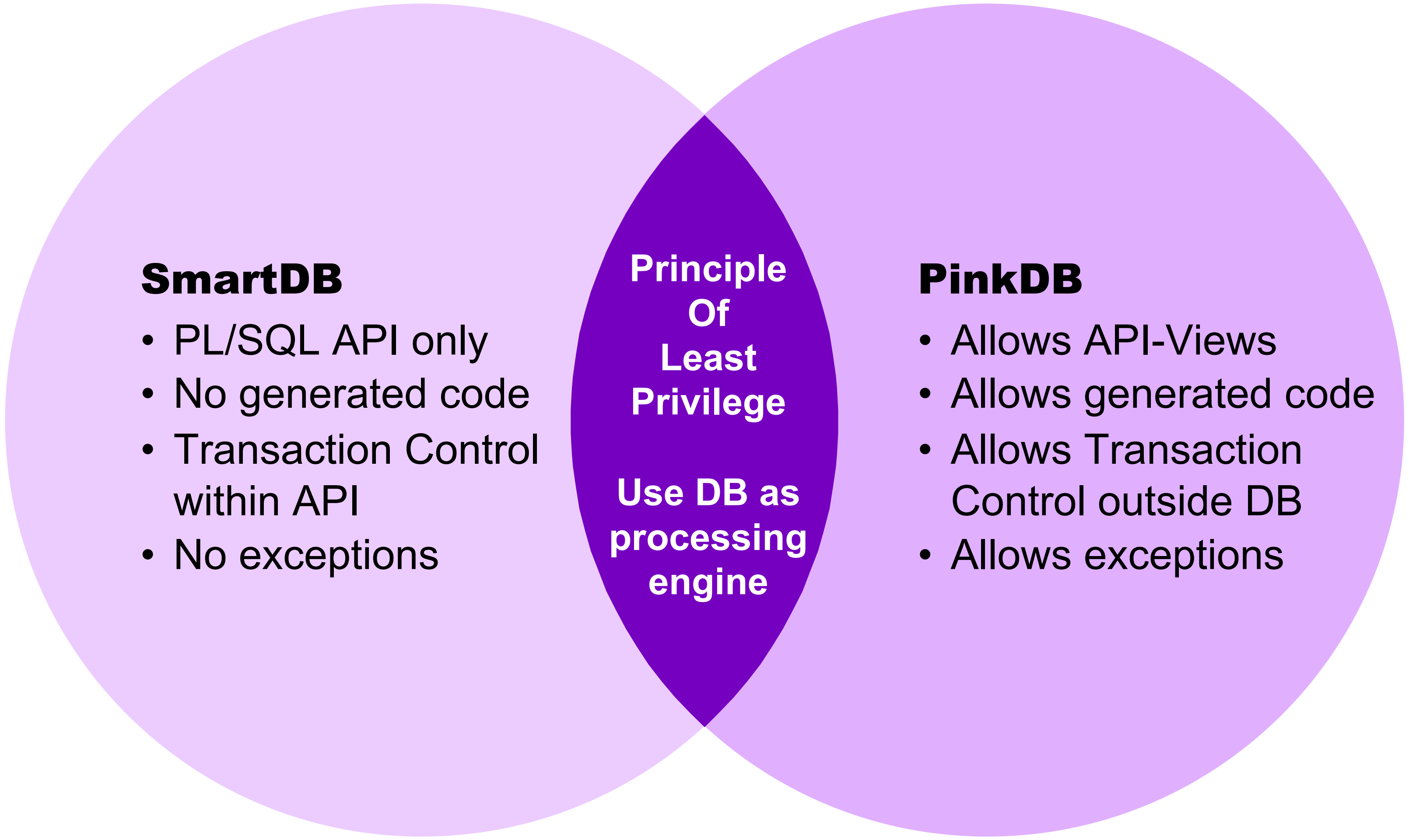
"(…) **application architecture** for database centric applications. It is focusing on relational database systems and is vendor neutral. The principles are **based on** the ideas of **SmartDB**, with some adaptions that make PinkDB easier to apply in existing development environments. (…)"

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

SmartDB vs. PinkDB – Used DB Features



SmartDB vs. PinkDB – Enforced Principles



SmartDB

- PL/SQL API only
- No generated code
- Transaction Control within API
- No exceptions

Principle Of Least Privilege

Use DB as processing engine

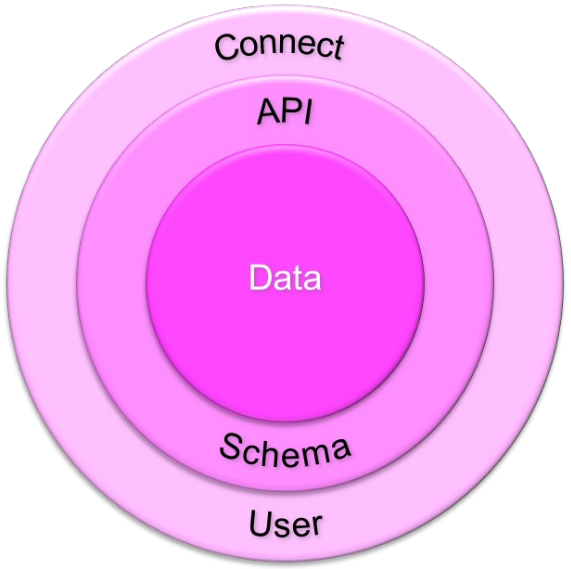
PinkDB

- Allows API-Views
- Allows generated code
- Allows Transaction Control outside DB
- Allows exceptions

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 (…)"

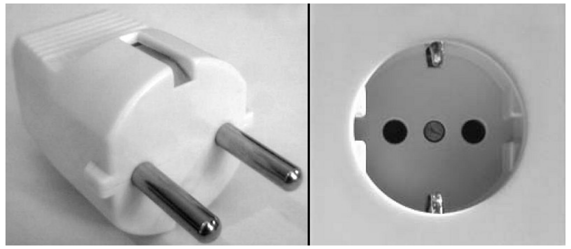
https://www.paloaltonetworks.com/cyberpedia/what-is-the-principle-of-least-privilege

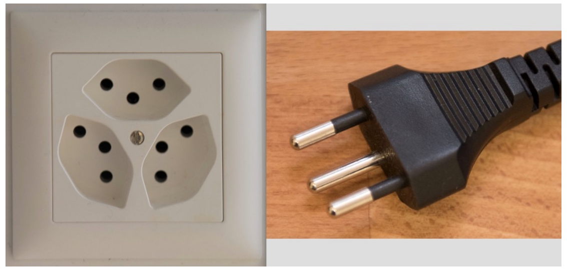
Value of an API

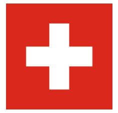
Abstraction – Simplified Usage

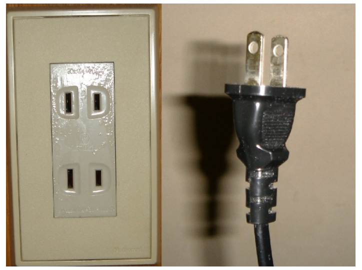
Source: https://de.wikipedia.org/wiki/Schuko#Spannung_und_Strom

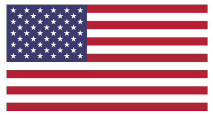
Standard – Stability, Interoperability





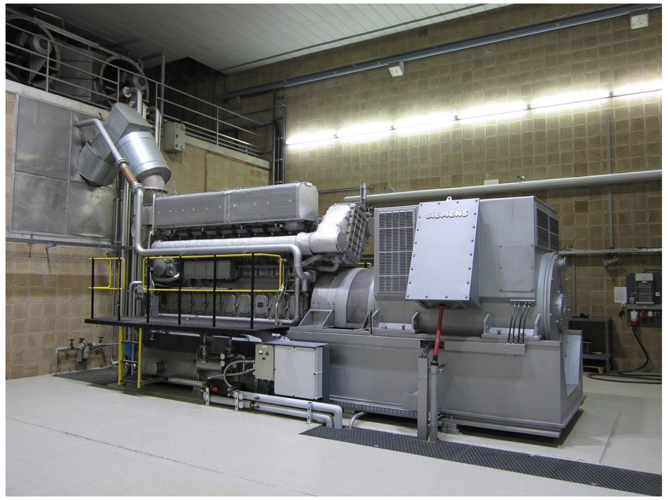






Source: https://de.wikipedia.org/wiki/Länderübersicht_Steckertypen,_Netzspannungen_und_-frequenzen

Hide Implementation Details



Minimalistic View API

1:1 Views

```
create or replace view countries_v   as
   select * from countries;
create or replace view departments_v as
   select * from departments;
create or replace view employees_v   as
   select * from employees;
create or replace view job_history_v as
   select * from job_history;
create or replace view jobs_v        as
   select * from jobs;
create or replace view locations_v   as
   select * from locations;
create or replace view regions_v     as
   select * from regions;
```

SELECT * ?

Where is the value of a view then?

We define the complete column list as soon as we make an incompatible change

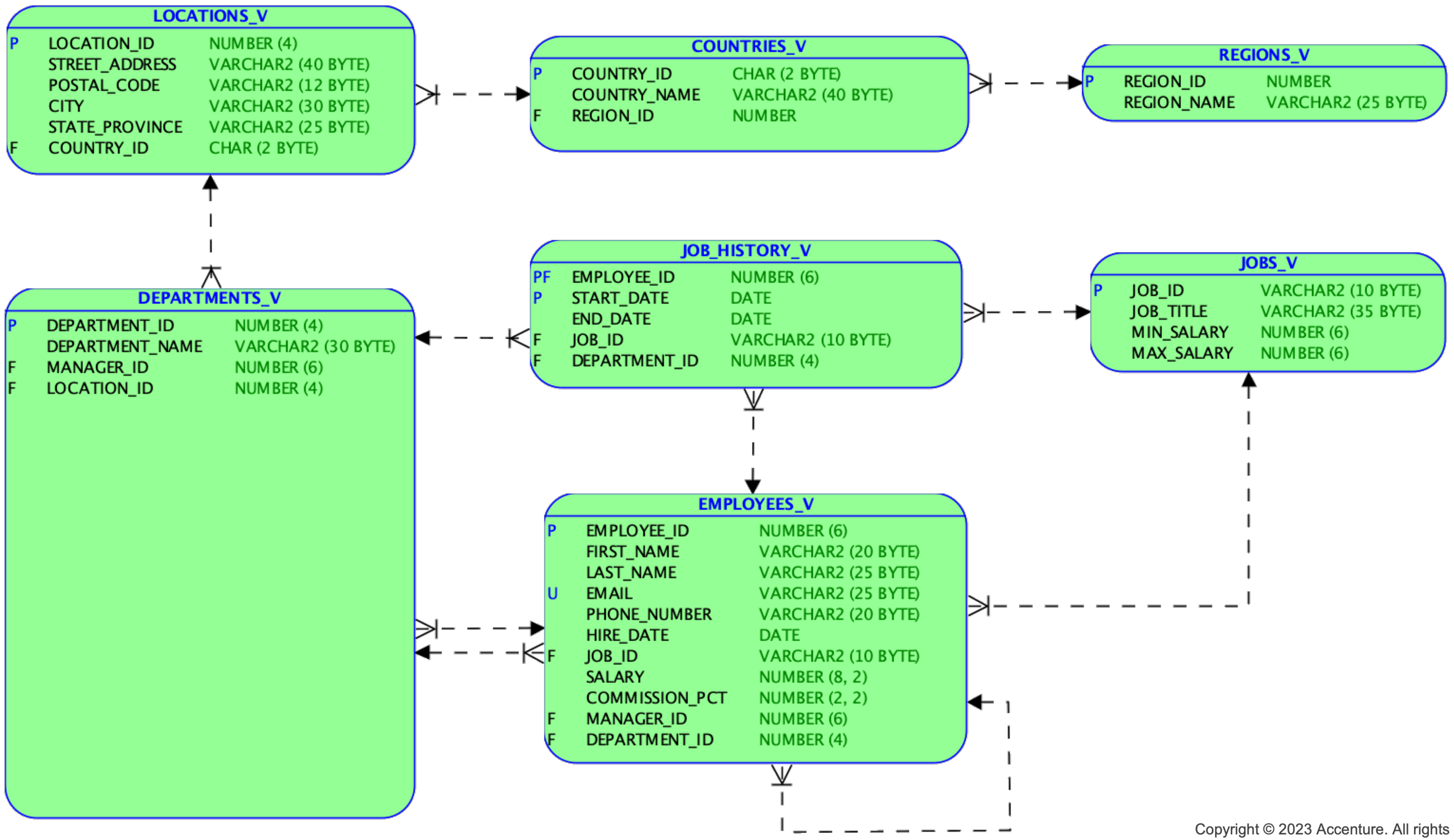
Bonus – PK/UK Constraints

```
alter view countries_v
   add primary key (country_id)                   disable novalidate;
alter view departments_v
   add primary key (department_id)                disable novalidate;
alter view employees_v
   add primary key (employee_id)                  disable novalidate;
alter view employees_v
   add unique      (email)                        disable novalidate;
alter view job_history_v
   add primary key (employee_id, start_date) disable novalidate;
alter view jobs_v
   add primary key (job_id)                       disable novalidate;
alter view locations_v
   add primary key (location_id)                  disable novalidate;
alter view regions_v
   add primary key (region_id)                    disable novalidate;
```

Bonus – Foreign Key Constraints

```
alter view countries_v
   add foreign key (region_id)     references hr.regions_v     disable novalidate;
alter view departments_v
   add foreign key (location_id)   references hr.locations_v   disable novalidate;
alter view departments_v
   add foreign key (manager_id)    references hr.employees_v   disable novalidate;
alter view employees_v
   add foreign key (department_id) references hr.departments_v disable novalidate;
alter view employees_v
   add foreign key (job_id)        references hr.jobs_v        disable novalidate;
alter view employees_v
   add foreign key (manager_id)    references hr.employees_v   disable novalidate;
alter view job_history_v
   add foreign key (department_id) references hr.departments_v disable novalidate;
alter view job_history_v
   add foreign key (employee_id)   references hr.employees_v   disable novalidate;
alter view job_history_v
   add foreign key (job_id)        references hr.jobs_v        disable novalidate;
alter view locations_v
   add foreign key (country_id)    references hr.countries_v   disable novalidate;
```

View-API Model



LOCATIONS_V

P	LOCATION_ID	NUMBER (4)
	STREET_ADDRESS	VARCHAR2 (40 BYTE)
	POSTAL_CODE	VARCHAR2 (12 BYTE)
	CITY	VARCHAR2 (30 BYTE)
	STATE_PROVINCE	VARCHAR2 (25 BYTE)
F	COUNTRY_ID	CHAR (2 BYTE)

COUNTRIES_V

P	COUNTRY_ID	CHAR (2 BYTE)
	COUNTRY_NAME	VARCHAR2 (40 BYTE)
F	REGION_ID	NUMBER

REGIONS_V

P	REGION_ID	NUMBER
	REGION_NAME	VARCHAR2 (25 BYTE)

JOB_HISTORY_V

PF	EMPLOYEE_ID	NUMBER (6)
P	START_DATE	DATE
	END_DATE	DATE
F	JOB_ID	VARCHAR2 (10 BYTE)
F	DEPARTMENT_ID	NUMBER (4)

JOBS_V

P	JOB_ID	VARCHAR2 (10 BYTE)
	JOB_TITLE	VARCHAR2 (35 BYTE)
	MIN_SALARY	NUMBER (6)
	MAX_SALARY	NUMBER (6)

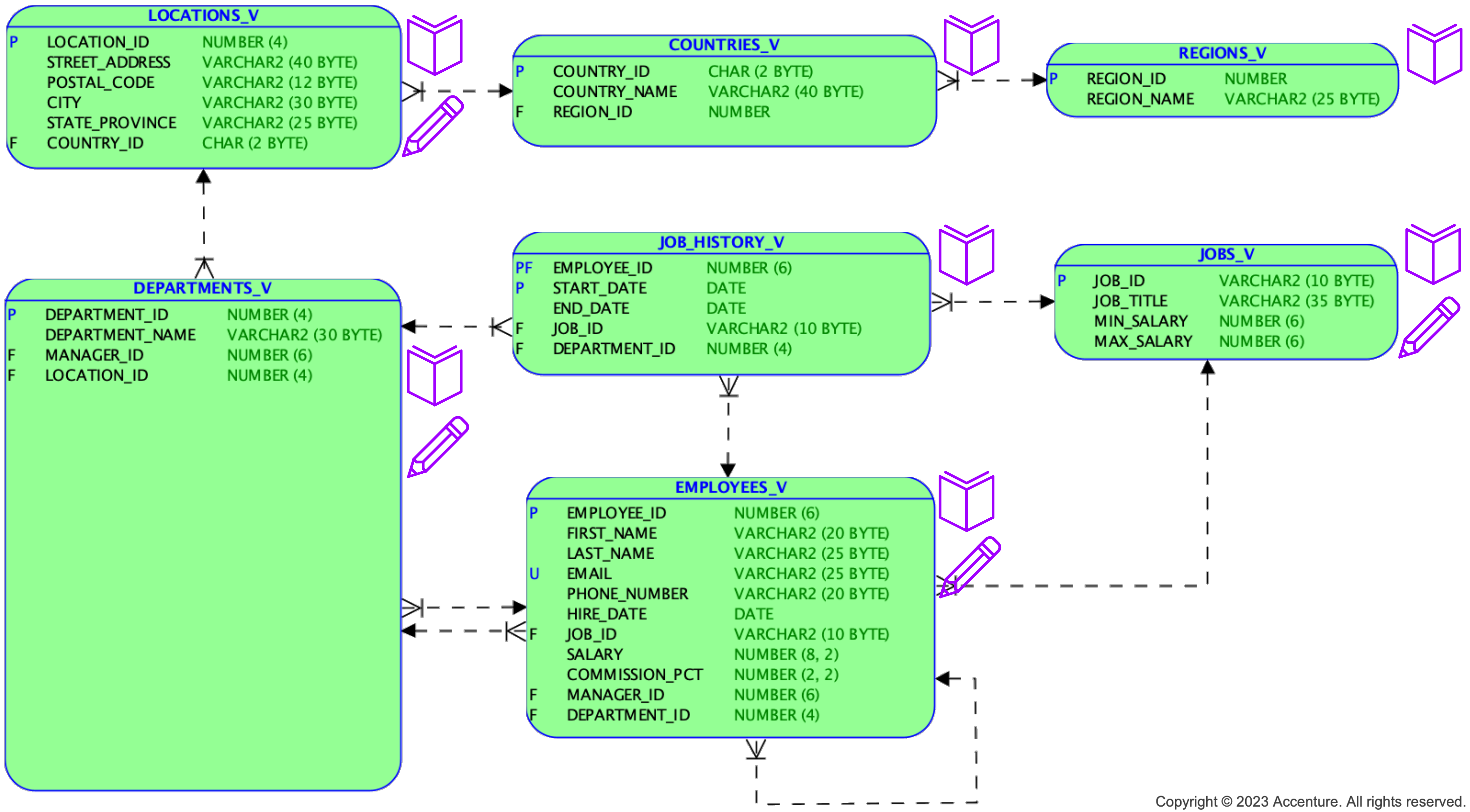
DEPARTMENTS_V

P	DEPARTMENT_ID	NUMBER (4)
	DEPARTMENT_NAME	VARCHAR2 (30 BYTE)
F	MANAGER_ID	NUMBER (6)
F	LOCATION_ID	NUMBER (4)

EMPLOYEES_V

P	EMPLOYEE_ID	NUMBER (6)
	FIRST_NAME	VARCHAR2 (20 BYTE)
	LAST_NAME	VARCHAR2 (25 BYTE)
U	EMAIL	VARCHAR2 (25 BYTE)
	PHONE_NUMBER	VARCHAR2 (20 BYTE)
	HIRE_DATE	DATE
F	JOB_ID	VARCHAR2 (10 BYTE)
	SALARY	NUMBER (8, 2)
	COMMISSION_PCT	NUMBER (2, 2)
F	MANAGER_ID	NUMBER (6)
F	DEPARTMENT_ID	NUMBER (4)

Read or Write Access?



LOCATIONS_V
P LOCATION_ID NUMBER (4)
STREET_ADDRESS VARCHAR2 (40 BYTE)
POSTAL_CODE VARCHAR2 (12 BYTE)
CITY VARCHAR2 (30 BYTE)
STATE_PROVINCE VARCHAR2 (25 BYTE)
F COUNTRY_ID CHAR (2 BYTE)
COUNTRIES_V
P COUNTRY_ID CHAR (2 BYTE)
COUNTRY_NAME VARCHAR2 (40 BYTE)
F REGION_ID NUMBER
REGIONS_V
P REGION_ID NUMBER
REGION_NAME VARCHAR2 (25 BYTE)
DEPARTMENTS_V
P DEPARTMENT_ID NUMBER (4)
DEPARTMENT_NAME VARCHAR2 (30 BYTE)
F MANAGER_ID NUMBER (6)
F LOCATION_ID NUMBER (4)
JOB_HISTORY_V
PF EMPLOYEE_ID NUMBER (6)
P START_DATE DATE
END_DATE DATE
F JOB_ID VARCHAR2 (10 BYTE)
F DEPARTMENT_ID NUMBER (4)
JOBS_V
P JOB_ID VARCHAR2 (10 BYTE)
JOB_TITLE VARCHAR2 (35 BYTE)
MIN_SALARY NUMBER (6)
MAX_SALARY NUMBER (6)
EMPLOYEES_V
P EMPLOYEE_ID NUMBER (6)
FIRST_NAME VARCHAR2 (20 BYTE)
LAST_NAME VARCHAR2 (25 BYTE)
U EMAIL VARCHAR2 (25 BYTE)
PHONE_NUMBER VARCHAR2 (20 BYTE)
HIRE_DATE DATE
F JOB_ID VARCHAR2 (10 BYTE)
SALARY NUMBER (8, 2)
COMMISSION_PCT NUMBER (2, 2)
F MANAGER_ID NUMBER (6)
F DEPARTMENT_ID NUMBER (4)

API to Change Data

Options

CRUD View

- Instead of trigger for complex views or business logic
- Integral part of the TAPI
- Can be generated

TAPI

- CRUD pattern
- PL/SQL Package
- Business logic hooks before/after insert, update, delete
- Handles optimistic locking
- Handles temporal data
- Can be generated
- Business logic needs to be implemented manually

XAPI

- Transaction
- One call for a complex process
- PL/SQL Package
- Cannot be generated
- Can use TAPIs

CRUD – Updateable Views / TAPI

Advantages		Disadvantages
• Easy interface • Fast development of RESTful API and UI • Can be generated using metadata/conventions • Row(s) representation • Audit columns • Optimistic locking • Temporal data • Other business logic via hooks	vs	• Technical Interface • Exposes internals (model, column names) • Caller is responsible for the transaction • Hides business process, e.g. • Exit of an employee • Salary increase • Instead of triggers and business logic at row level leads to row-by-row processing. • Row based API processes all columns for select and update • Incomplete generators can hinder evolution and versioning of models and the API

XAPI – API for Transactional Business Funcs

Advantages		Disadvantages
• Easy interface • Business Language • Procedure/function names • Parameter names • Payload structures (e.g., JSON, XML) • Minimize / optimize interaction with the DB • Transactions are automatically handled correctly • Explicit, controlled API evolution • Easier to move to another UI framework	vs	• Manually crafted code (no generators) • No default processing logic in the UI • Designing a good initial version of an API is hard

Philipp's Going-in Position for PinkDB APIs

API to Query Data

Manually Crafted Code

General

- 1:1 views based on tables
- Include surrogate key
- Business names
    - View name
    - Column name
- JSON for complex structures

Access

- Grant READ to role
- All columns
- VPD/RAS for
    - Row-level access
    - Masking sensitive columns

Abstractions

- SQL Macros
    - Wrapped in views (etag)
    - Exposed package functions (temporal joins)

Views & Packages

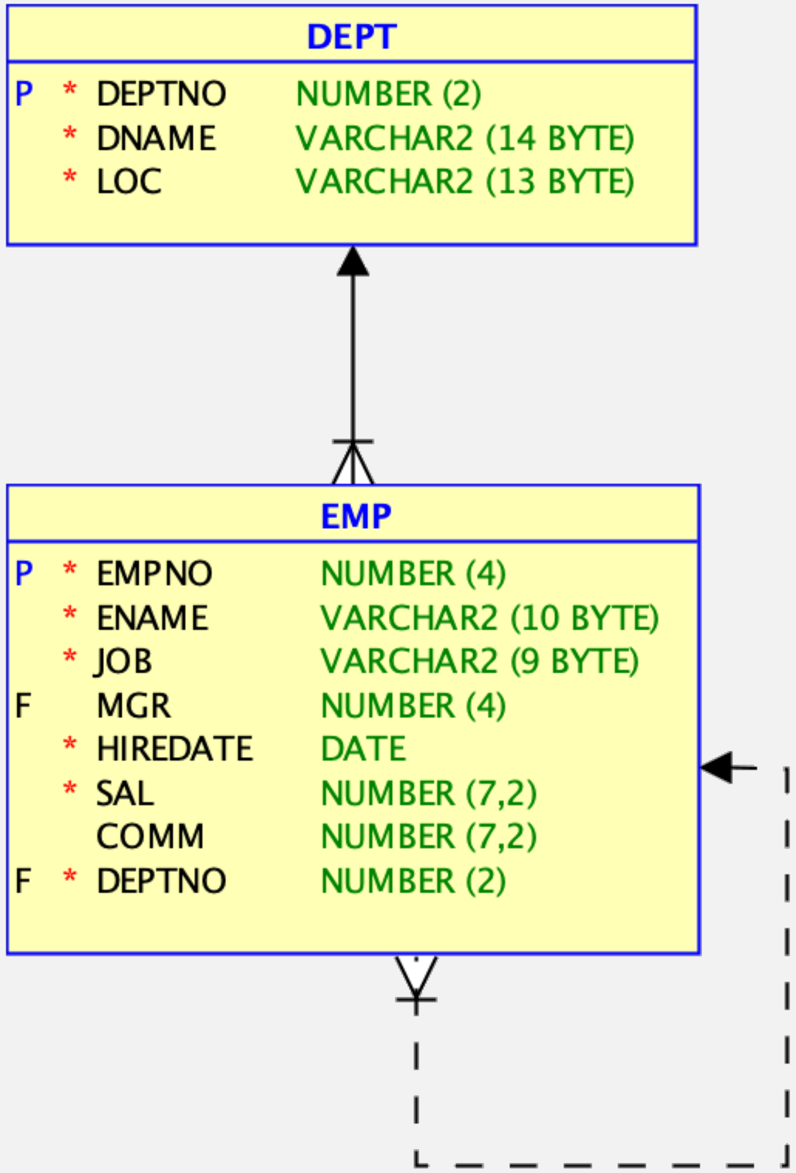
API to Change Data

Manually Crafted Code

General	Access	Abstractions
• XAPI • Scalars data types, if feasible • JSON for everything else • No XAPI for reference data and alike (using load scripts instead)	• Grant EXECUTE to role • Split package to match role-level access • Row-level access via views • Dedicated procedures/functions for sensitive columns	• SQL Macros • Polymorphic Table Functions • Dynamic SQL (ad-hoc generators)

Packages

Finding PinkDB Violations

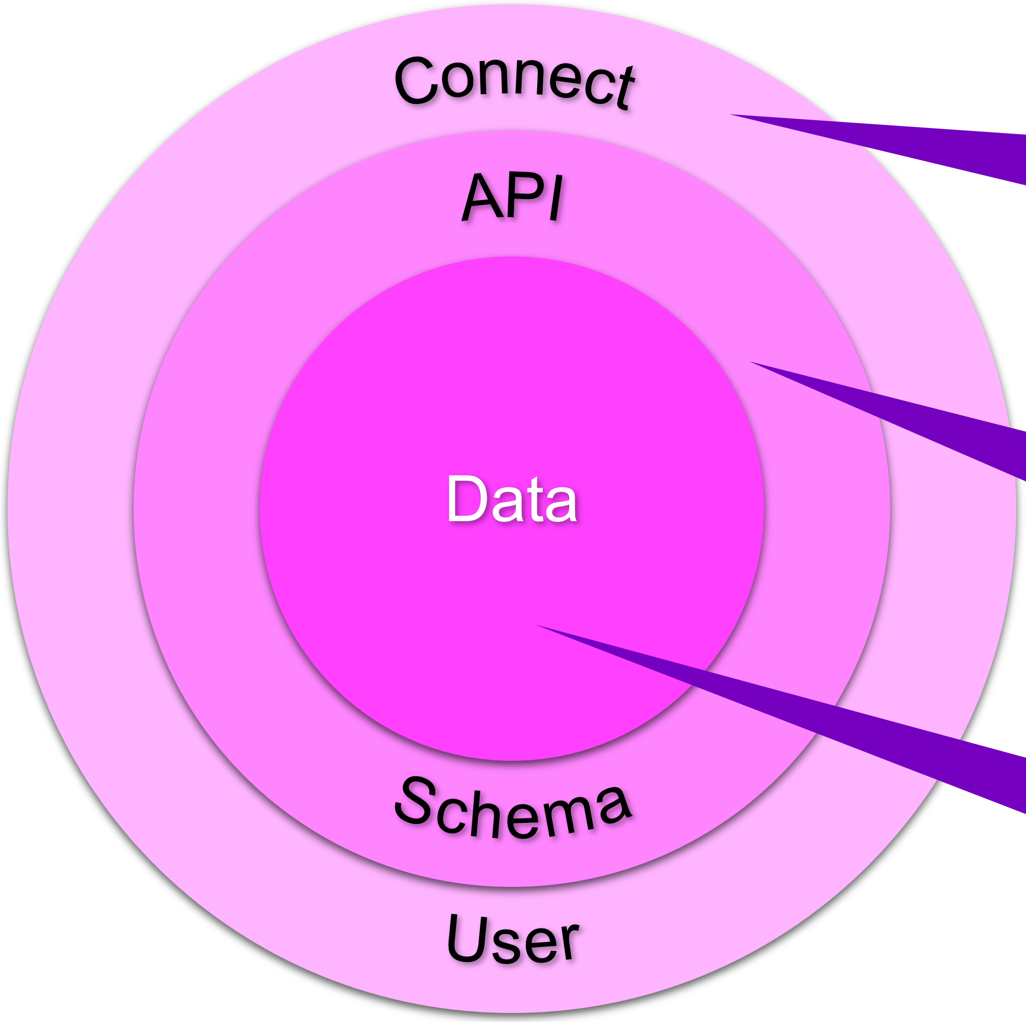


Demo App

- 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

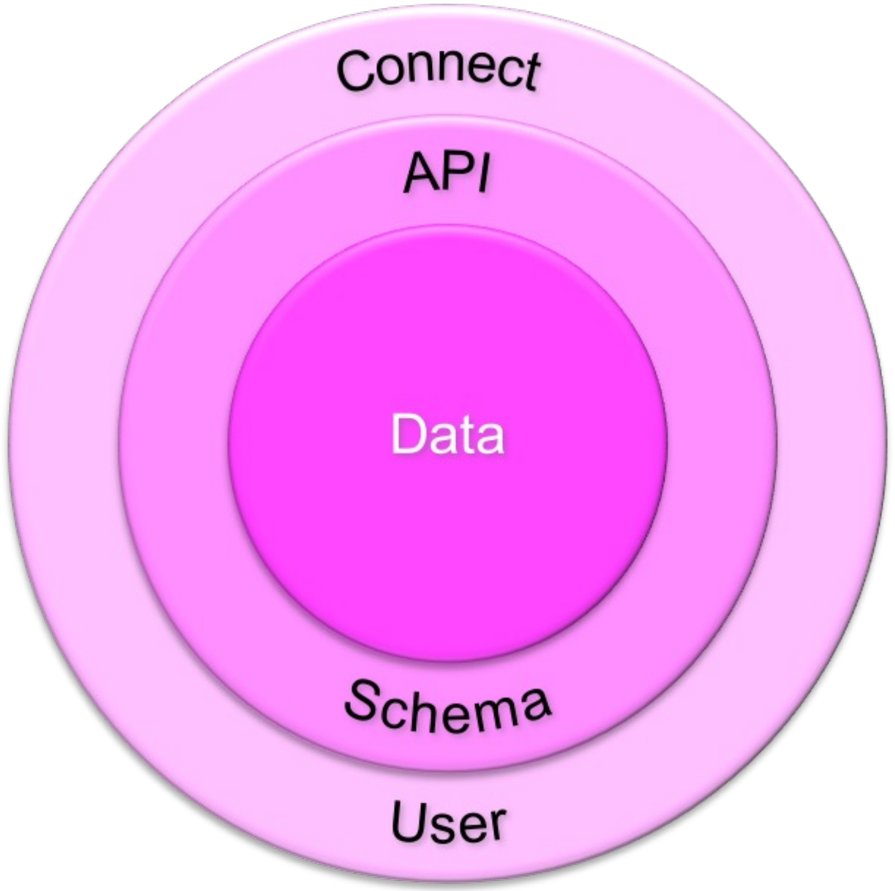
Value of an API

- **Binding Contract on Abstraction**
  - Separation of concerns
  - Simplified usage
  - Stability
  - Any API is better than none

- **Simplified Change**
  - Implementation details are hidden
  - Freedom to change as long as
    the existing API is not affected
  - Independent release cycle of DB app

Value of a PinkDB Application

- **Security**
  - – Connect user minimizes the attack surface
  - – Follow principle of least privileges
- **Performance**
  - – Uses the database as a processing engine
  - – Minimizes network roundtrips
- **Maintainability**
  - – Adding new features
  - – Changes behind the API
  - – Changes before the API (e.g. UI framework)



Thank You

The Educational Conference For Oracle Techology Users

ODTUG

Kscope23

aurora, co june 25-29

Interested in volunteering?
Scan the code to sign up

Don't Forget To
Fill Out Your Evals