

DATABASE SECURITY

Sicurezza informatica

Elena Maria Dal Santo

elenamaria.dalsanto@its-ictpiemonte.it

OWASP Top 10 Web security risks (2017)

1. Injection

2. Broken authentication
3. Sensitive Data Exposure
4. XML External Entities
5. Broken Access Control
6. Security Misconfiguration
7. Cross-site scripting
8. Insecure Deserialization
9. Using components with known vulnerabilities
10. Insufficient logging & monitoring

Con “code injection” si intende un tipo di attacco dove si va a “iniettare” all’interno del software alcune righe di codice che non erano inizialmente previste e portano a risultati inattesi. I più comuni sono l’esposizione di dati sensibili, l’eliminazione di tabelle o interi database, la concessione di diritti da admin, ecc

OWASP Top 10 Web security risks (2021)



Freepik, 2020

[Home](#) > [News](#) > [Security](#) > Freepik data breach: Hackers stole 8.3M records via SQL injection

Freepik data breach: Hackers stole 8.3M records via SQL injection

By [Sergiu Gatlan](#)

 August 21, 2020  06:37 PM  0

Freepik says that hackers were able to steal emails and password hashes for 8.3M Freepik and Flaticon users in an SQL injection attack against the company's Flaticon website.

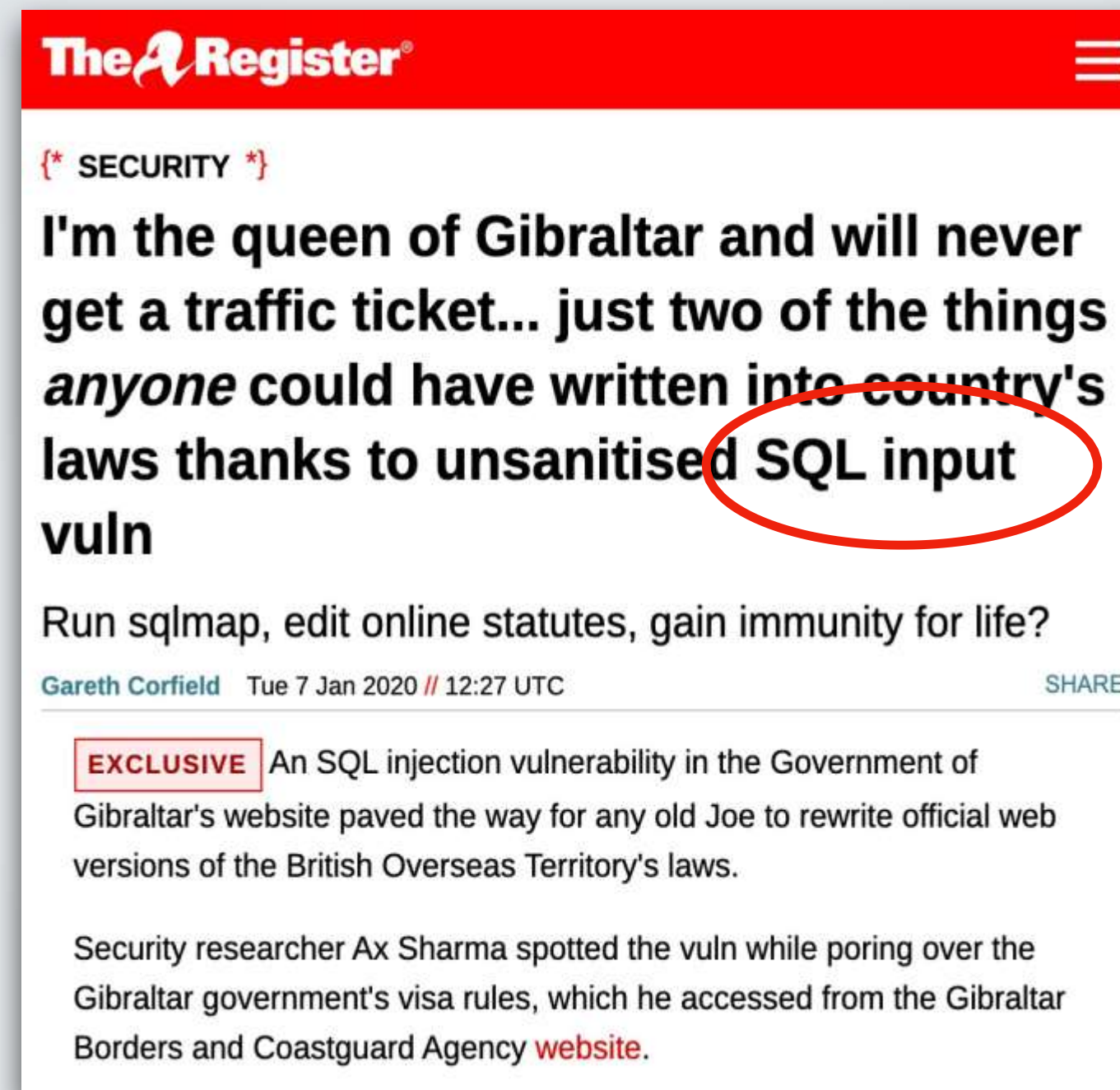
Freepik is the company behind Freepik (one of the largest online graphic resources sites in the world) and Flaticon (an icon database platform) totaling 18 million monthly unique users, 50 million monthly views, and 100 million monthly downloads.

The threat actors behind the Freepik security breach were able to steal the oldest 8.3M users' emails and password hashes, where available.

"To clarify, the hash of the password is not the password, and can not be used to log into your account," Freepik added.

229K MD5 salted passwords reset after breach

Gibilterra, 2020



A Security Vulnerability Let Anyone “Rewrite the Laws” of Gibraltar

Gab, 2021



BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE STORE

TROVE —

Trump's is one of 15,000 Gab accounts that just got hacked

GabLeaks includes 70,000 messages in more than 19,000 chats by over 15,000 users.

DAN GOODIN - 3/2/2021, 12:14 AM





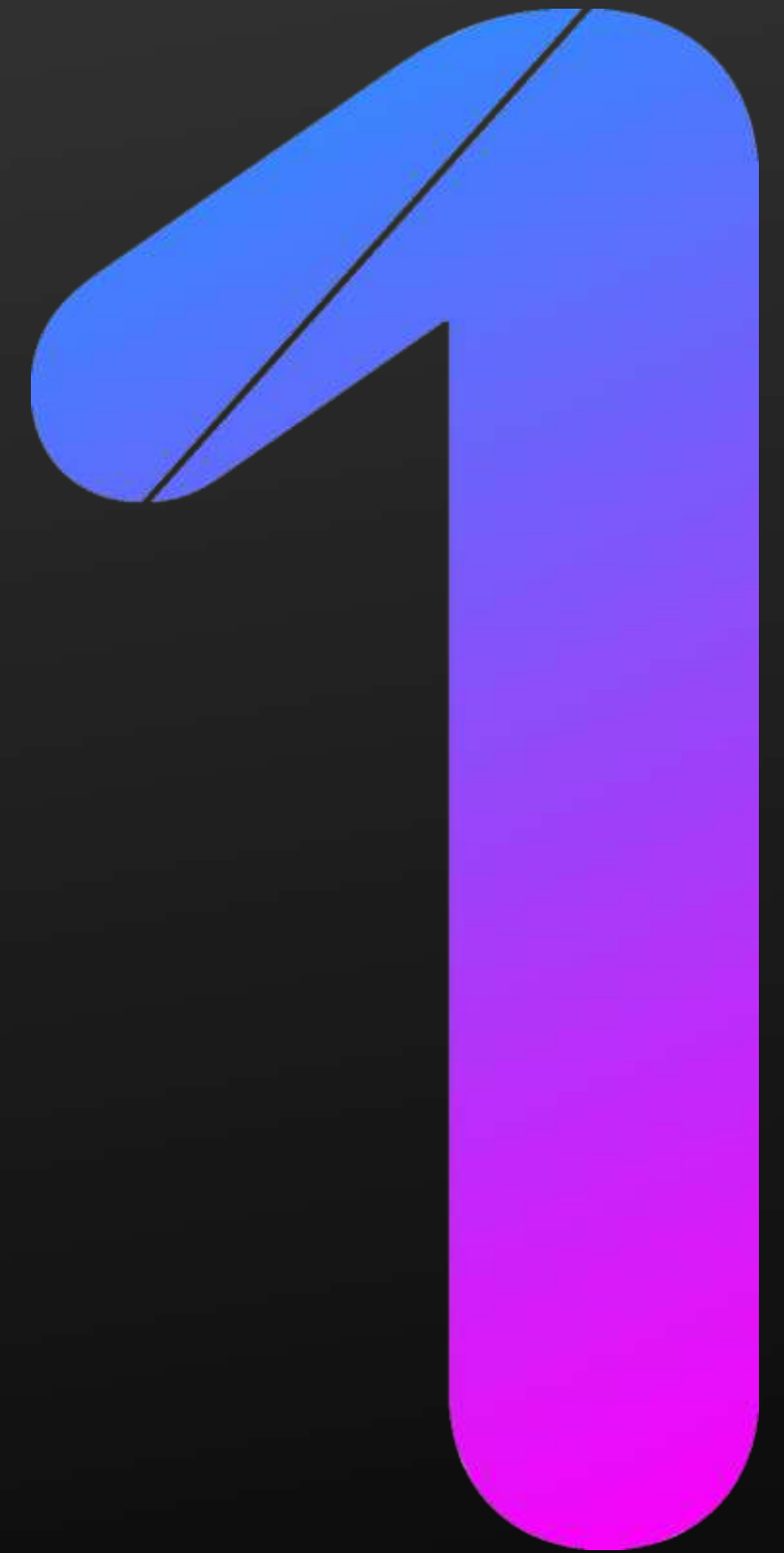
The founder of the far-right social media platform Gab said that the private account of former President Donald Trump was among the data stolen and publicly released by hackers who recently breached the site.



In a statement on Sunday, founder Andrew Torba used a transphobic slur to refer to Emma Best, the co-founder of Distributed Denial of Secrets. The statement confirmed claims the WikiLeaks-style group **made on Monday** that it obtained 70GB of passwords, private posts, and more from Gab and was making them available to select researchers and journalists. The data, Best said, was provided by an unidentified hacker who breached Gab by exploiting a **SQL-injection** vulnerability in its code.

SQLi Hall-of-shame

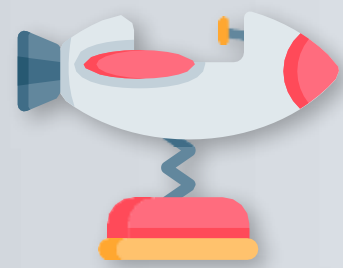
SQLi Attacks			
<div><input type="text" value="Search"/>  </div>			
Company	Date▼	Results	Reference
SonicWall	2021-02	vulnerability in secure access gateway - exploits reported in the wild.	A Vulnerability in SonicWall SMA 100 Series Could Allow for SQL Injection
NIC.lk	2021-02	multiple Sri Lankan domains hacked using data from domain administrator	Hacktivists deface multiple Sri Lankan domains, including Google.lk
Fortinet	2021-02	critical vulnerabilities fixed in multiple products	Fortinet fixes critical vulnerabilities in SSL VPN and web firewall
	2021-	RCE vulnerability in SAP Commerce product	SAP Commerce Critical



SQL Injection

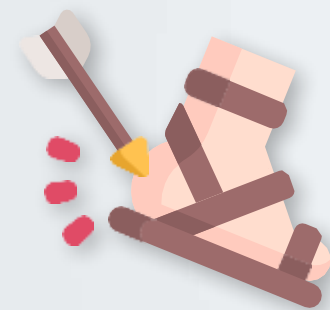
Injection

«Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.»



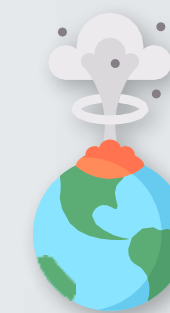
Vettori di attacco

Qualunque sorgente di dati, come variabili d'ambiente, parametri, web services interni ed esterni e tutti i tipi di utenti.



Debolezze

Estremamente diffuse, soprattutto nel codice legacy.
Molto facili da trovare esaminando il codice, anche in modo automatizzato.



Impatto

Perdita, corruzione o rivelazione dei dati a soggetti non autorizzati, negazione dei servizi o completo controllo del bersaglio attaccato.

MySQL: cose da ricordare

Commenti

- `#`
- `/* commento */`
- `--` questo commento richiede uno spazio bianco seguito da qualsiasi carattere per funzionare. La combinazione `-- -` funziona.

Default database

- `mysql` (solo per utenti con privilegi)
- `information_schema` (MySQL 5+)

SQL injection

```
select shape from objects where color = '$color';
```

+

```
<?php
```

```
$color = "red"; drop table objects-- -";
```

```
?>
```

=

```
select shape from objects where color = 'red';
```

```
drop table objects-- -'
```

Login form

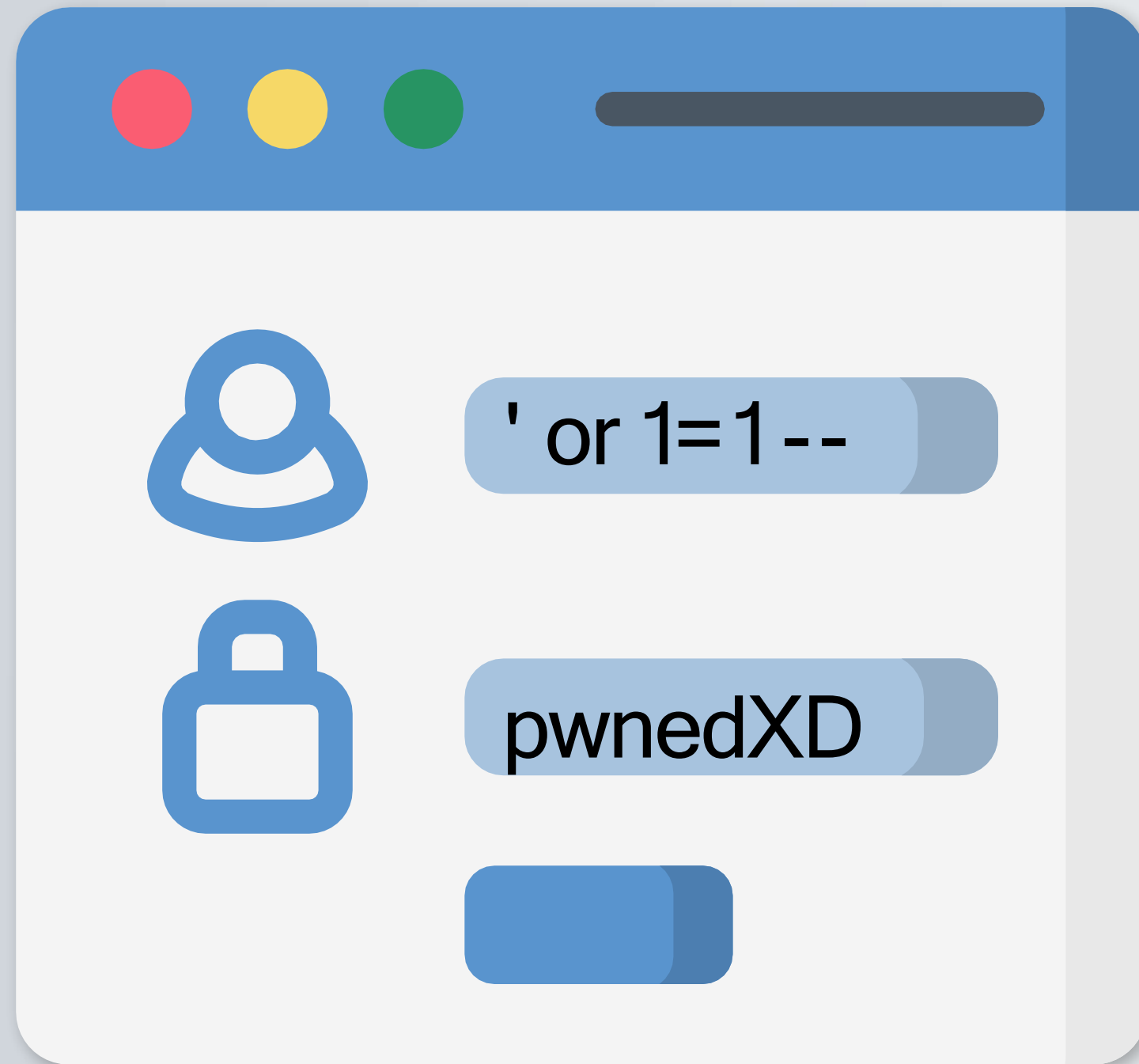


Illustration of a login form in a browser window. The window has a blue header with three colored circles (red, yellow, green) and a dark blue bar. The form has a white background. It contains a blue user icon, a blue padlock icon, and a blue button. The text input fields contain ' or 1=1 --' and 'pwnedXD'.

```
<?php
    $user = $_GET[username];
    $pass = $_GET[password];
    $query = "select * from users
              where username='" + $user + "'
              and password='" + $pass + "'";
?>
```


Login form

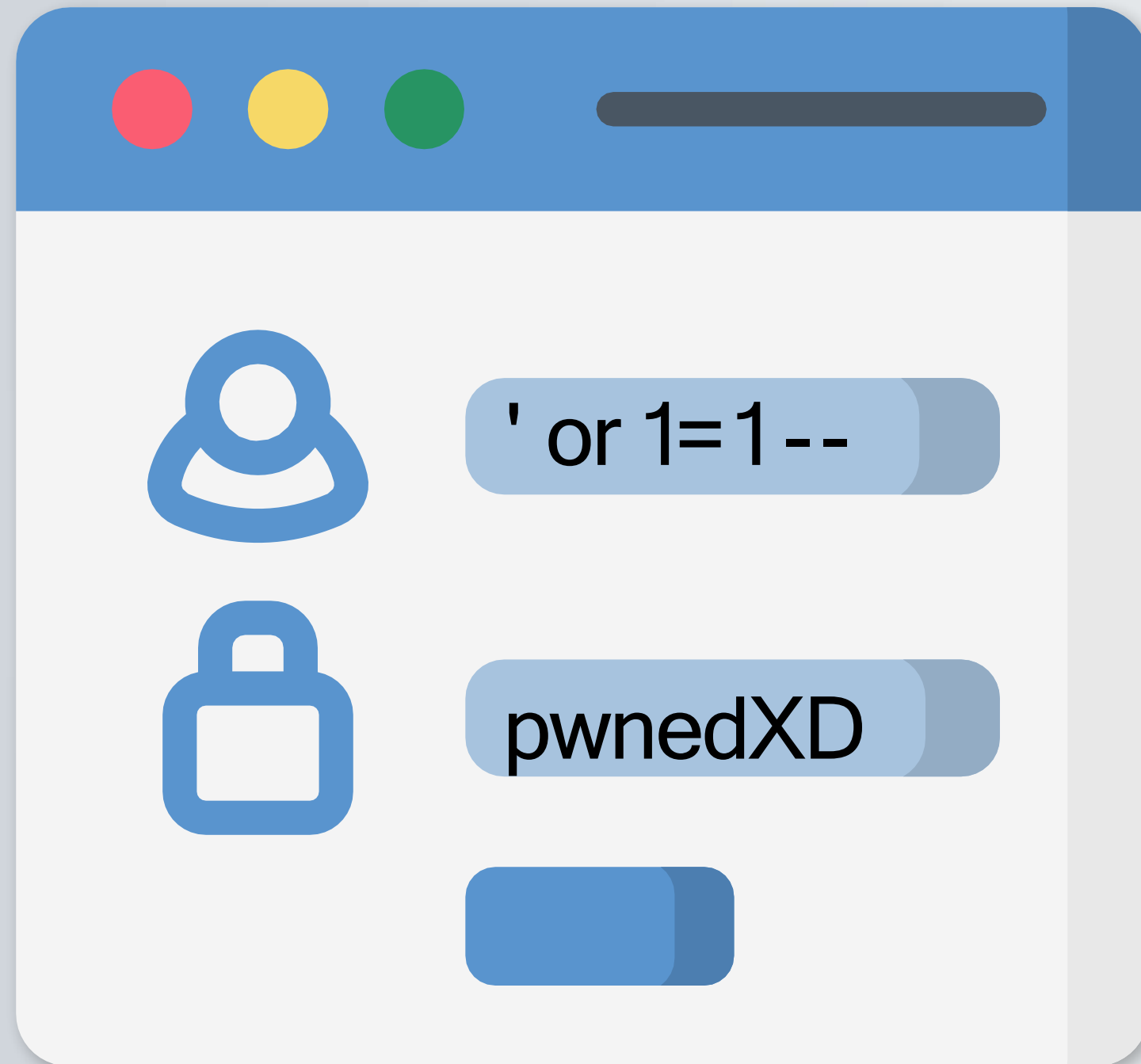
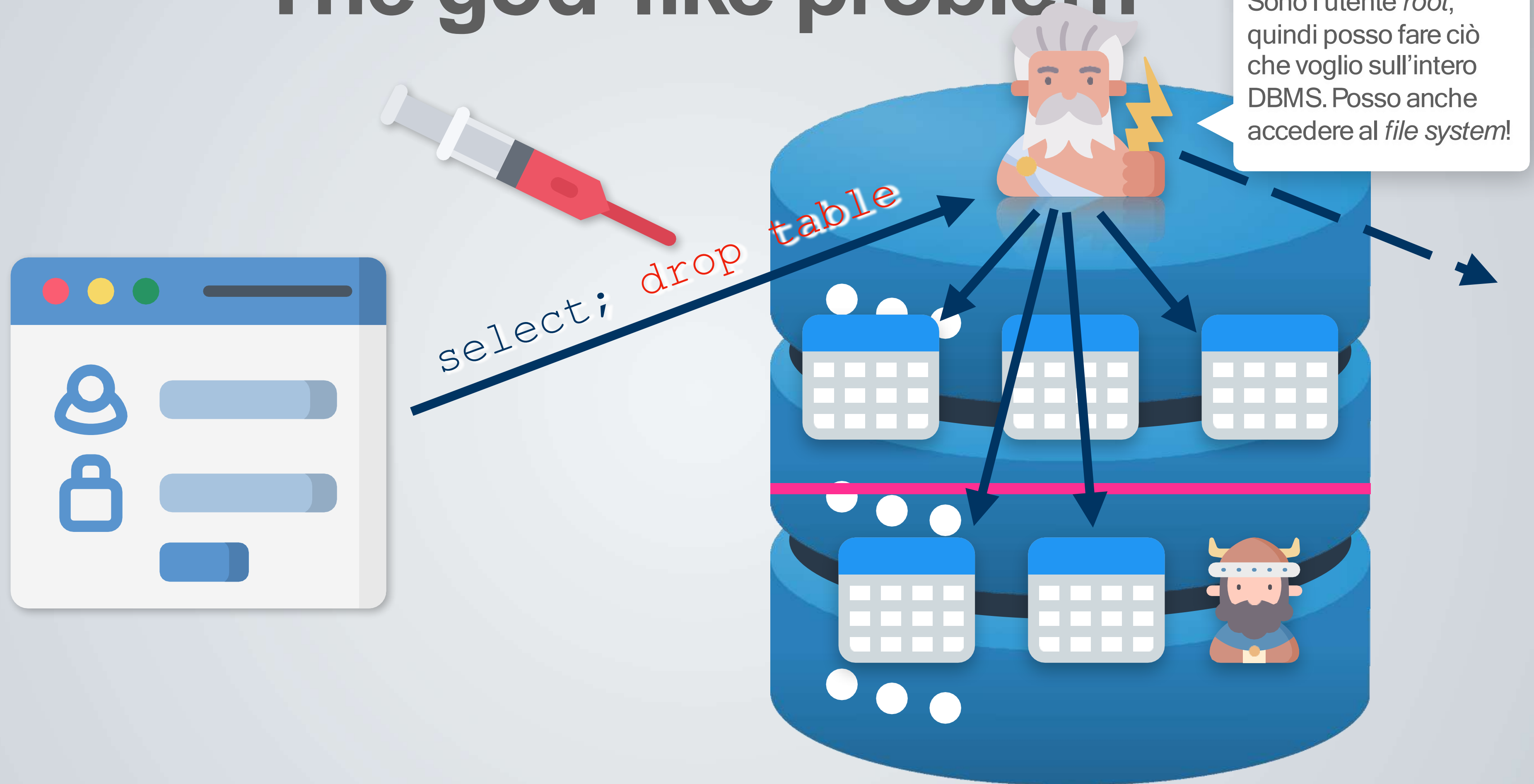


Illustration of a login form in a browser window. The window has a blue header with three colored circles (red, yellow, green) and a dark blue bar. The form has a white background. It contains a blue user icon, a blue padlock icon, and a blue button. The text input fields contain the strings ' or 1=1 --' and 'pwnedXD'.

```
<?php
    $user = $_GET[username];
    $pass = $_GET[password];
    $query = "select * from users
              where username='' or 1=1 --
              and password='pwnedXD'";
?>
```

Sempre vera!

The god-like problem



Concatenare è male.

```
String query =  
    select * from users  
    where username = ' " + user + " '  
    and password = ' " + pass + " ' ;
```

È sbagliato il modo in cui viene costruita la query. Non deve essere una concatenazione di stringhe.

Concatenare è male.

```
select * from users  
where username = 'overlord'  
and password = 'ov3rl0rd';
```

L'input dell'utente viene passato e processato dal DBMS così come viene fornito.

Concatenare è male.

```
select * from users
where username = ' ' or 1=1 --
and password = 'ov3r10rd';
```

Non ci si può mai fidare
di ciò che gli utenti
inseriscono nelle
stringhe.

Concatenare è male.

```
select * from users
where username = 'overlord'
and password = '' union all
select owner || '.' || table_name
from all_tables where 1='1';
```

Si possono concatenare query con `union all` per ottenere il contenuto di tutte le tabelle per cui l'utente ha accesso nel database.



Difendersi dall'SQL injection

Forse ti hanno detto che...

per evitare l'SQL injection bisogna

sanificare l'input



NO(N SOLO)! La validazione e la sanificazione dell'input non sono mai sufficienti per mettere al sicuro il database, men che mai se fatti lato client!

**STEAL WHAT'S
NOT THERE YOU CAN'T**



LEAST PRIVILEGE

Regola n° 1

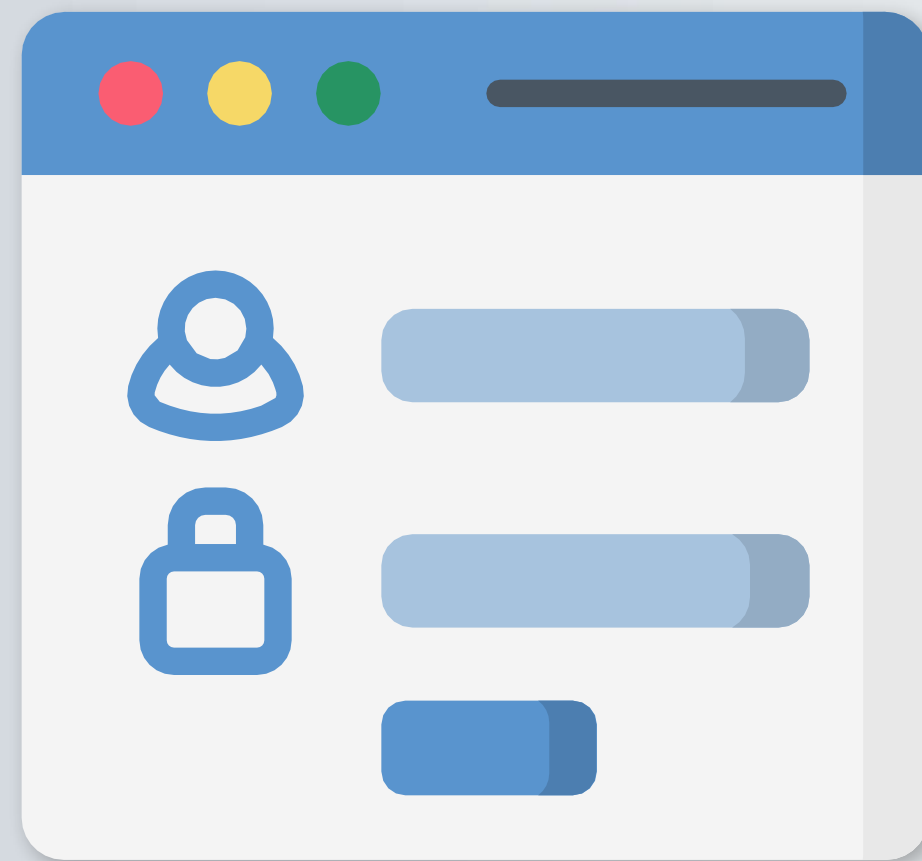
Principio del

least privilege

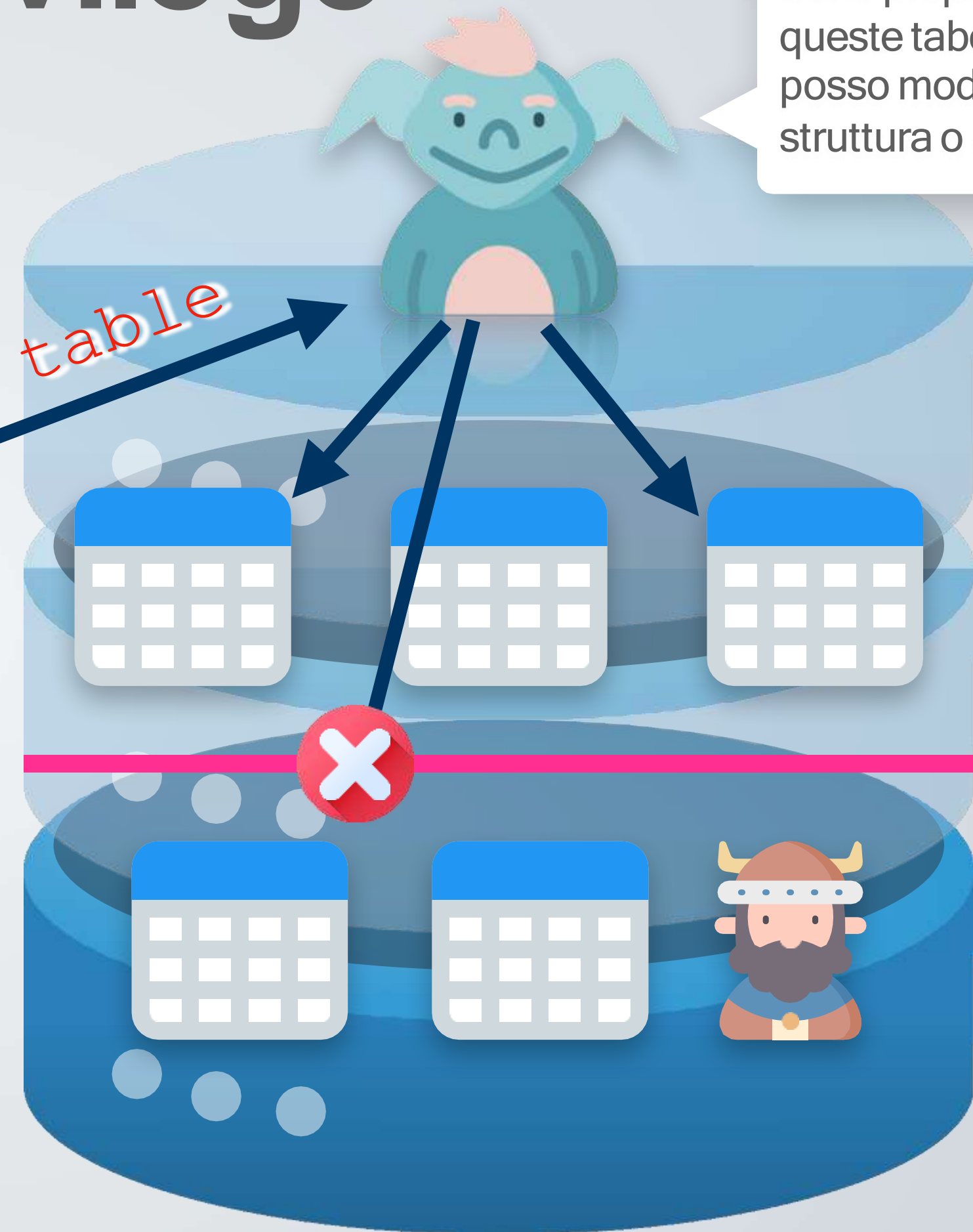
Adottare il principio del privilegio minimo o least privilege significa che ogni utente, programma, processo o dispositivo connesso abbia solo i privilegi minimi necessari per eseguire la sua mansione o la sua funzione.

Least privilege

Sono proprietario di queste tabelle, quindi posso modificarne la struttura o cancellarle.

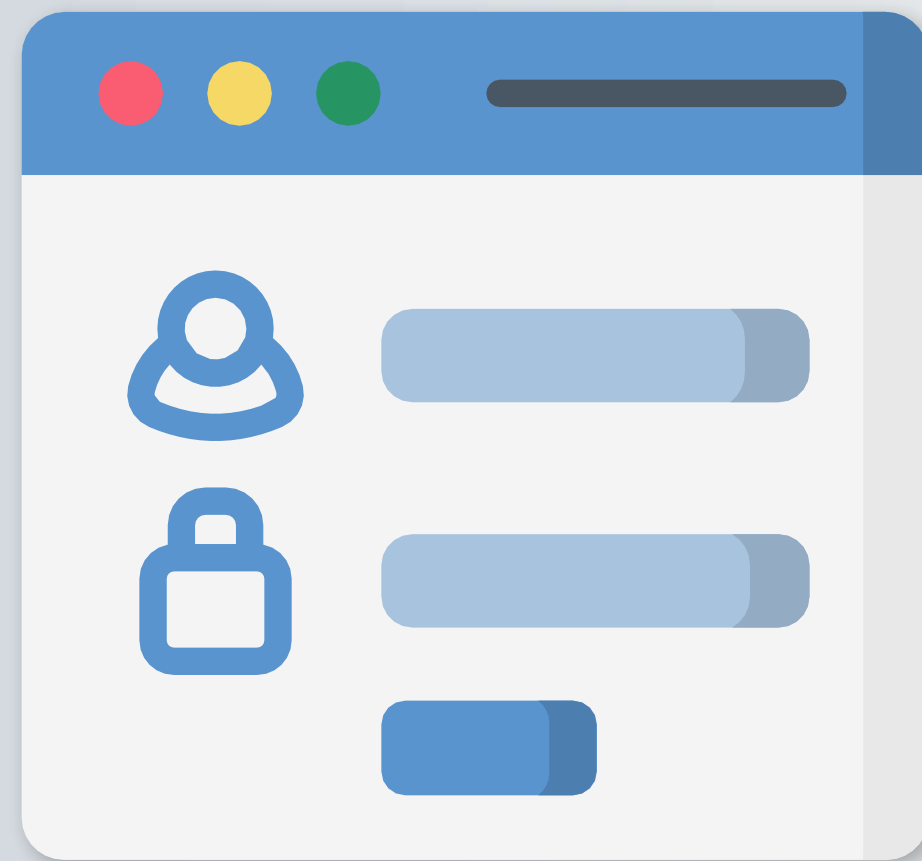


`select;` `drop table`

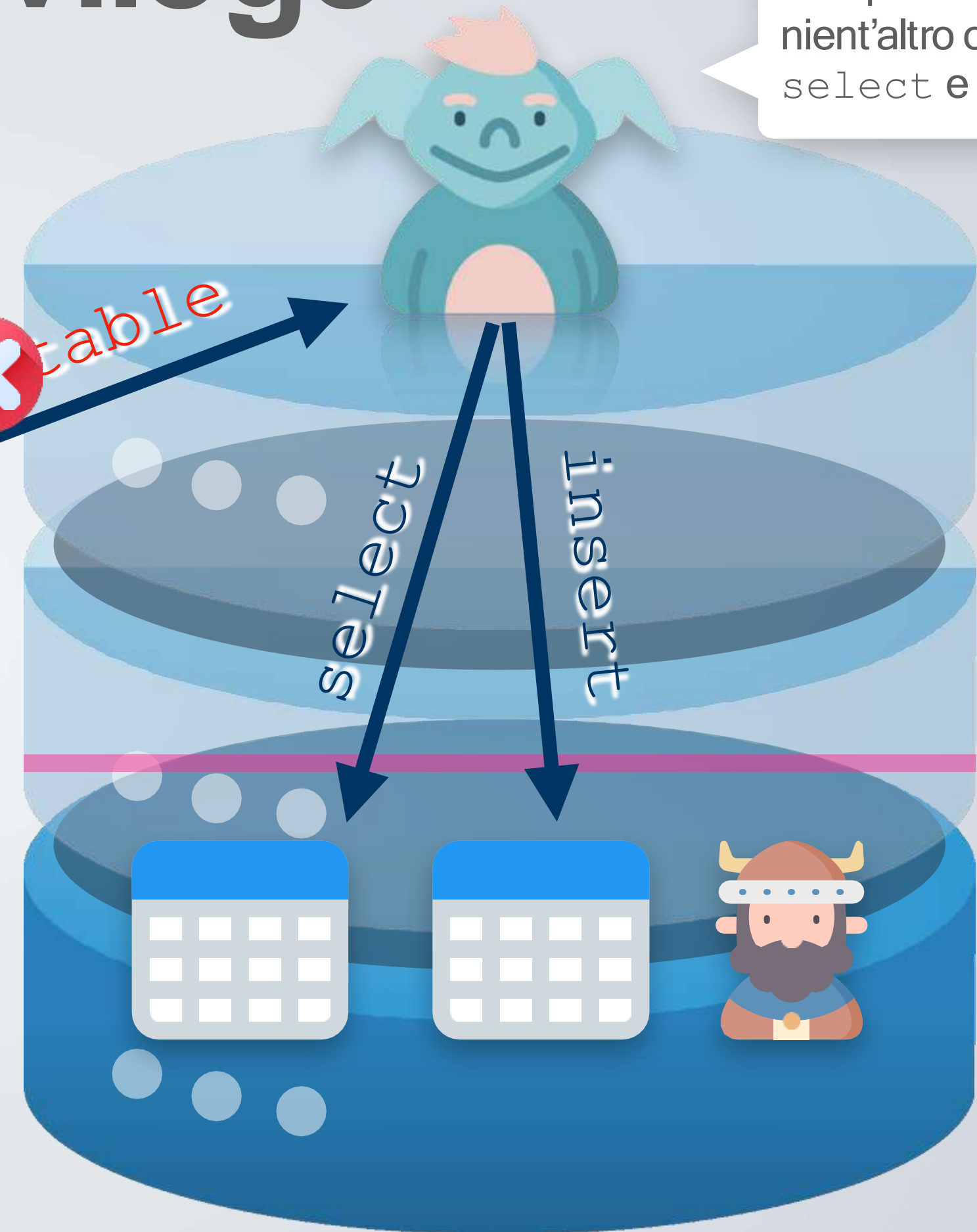


Least privilege

Non posso fare
nient'altro oltre
`select` e `insert`.



`select; drop table`



Least privilege

Magari bastasse questo...

Cosa succede se riesco ad iniettare questa query?

```
select * from users join users join users  
join users join users join users  
order by 1
```

Denial of service!

Le singole regole da sole non bastano!



Regola n° 2

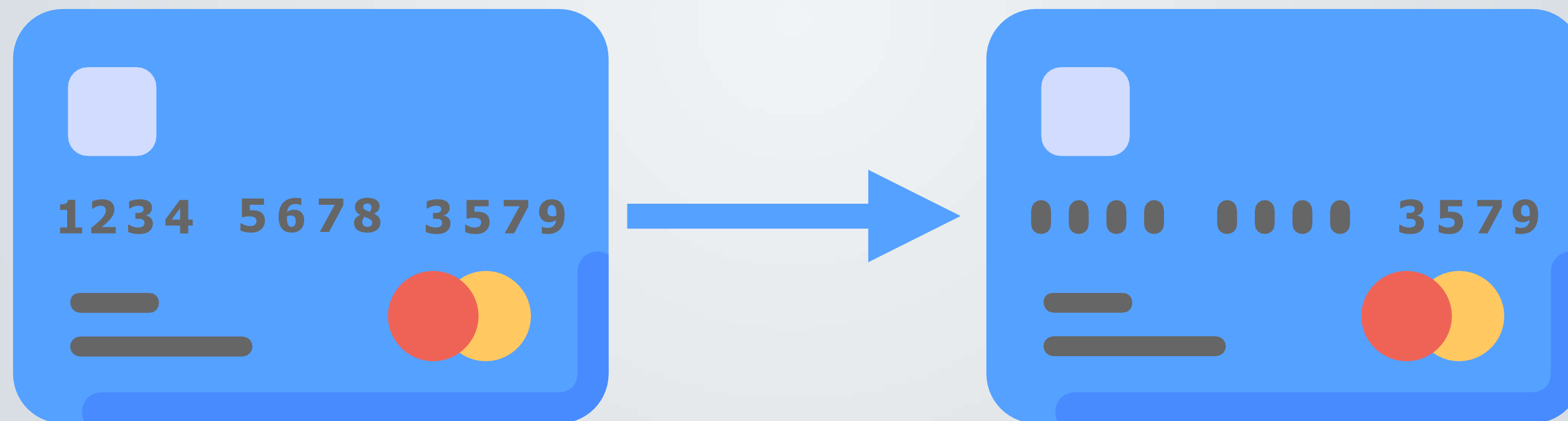
Principio del

Data redu***n**

Salvo meno dati → occupo meno spazio, ottimizzo i tempi di ricerca
e espongo molti meno dati a rischi

Regola n° 2

Data reduction



Regola n° 3

Usare le

bind variables

Bind variables

```
String query =  
    select * from users  
    where username = ?  
    and password = ?;
```

I ? sono dei *value placeholder*. Il loro valore viene impostato prima di procedere con l'esecuzione della query.

Usando le *bind variables* l'input non diventa mai parte dell'SQL.

Dynamic SQL

It's trickier, but still doable.

Attributi dinamici

```
String sql = "select * from users where 1=1 ";  
if (parameter1 != null) {  
    sql += "and c1 = ? ";  
}  
if (parameter2 != null) {  
    sql += "and c2 = ? ";  
}
```

Dynamic SQL

It's trickier, but still doable.

Tabelle dinamiche

```
String sql = "select * from ";  
if (tab == "CARD") {  
    sql += "card where ... ";  
}  
if (tab == "CASH") {  
    sql += "cash where ... ";  
}
```


Bind variables

Perché le bind variables non vengono utilizzate?

- Perché non vengono spiegate nei corsi di database
- Perché sono più difficili da usare della concatenazione

```
public void getUser(String username, String password) throws SQLException {  
    String query = "select * from users where username = ? and password = ?";  
    try (PreparedStatement getUser = connection.prepareStatement(query)) {  
        getUser.setString(1, username);  
        getUser.setString(2, password);  
        ResultSet rows = getUser.executeQuery();  
    } catch (SQLException e) {  
        // catch the exception! Never shut'em up!  
    }  
}
```

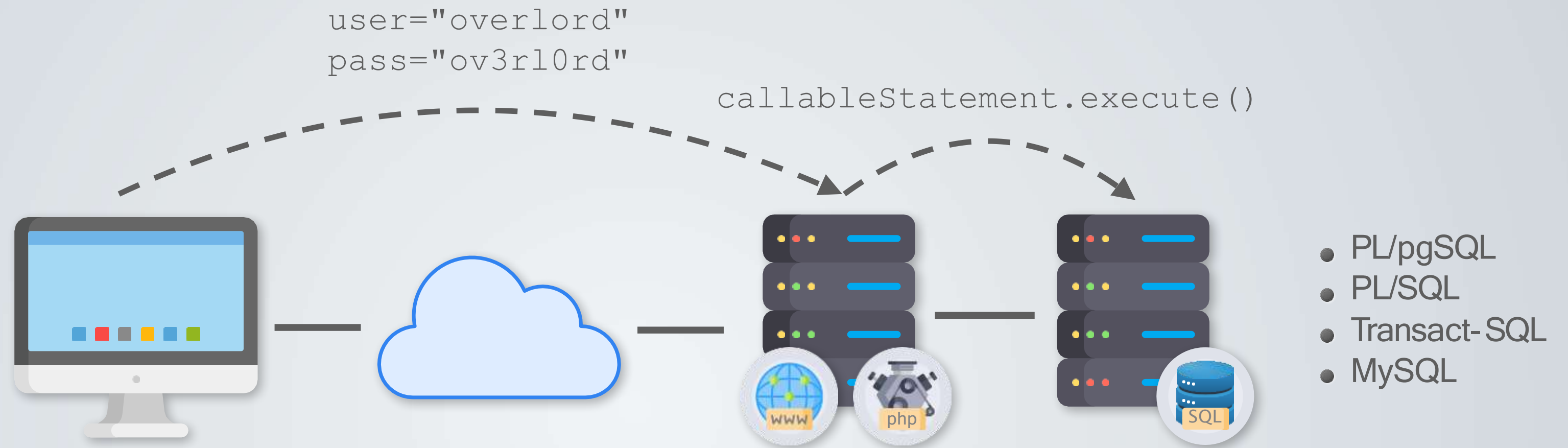
One rule to rule them all.

Usare le

stored procedure

Una stored procedure è un programma scritto in SQL o altri linguaggi e mantenuto nel database stesso. Lo potete immaginare come un insieme di query, da eseguire in un ordine ben preciso.

Stored procedure



- Chiamare una procedura invece di riscrivere query complesse.
- Poche informazioni scambiate tra client e server a vantaggio delle prestazioni.
- Elaborazioni complesse non altrimenti realizzabili usando unicamente query SQL.
- Compile una volta sola, eseguite a piacimento, migliorando le prestazioni.
- Creazione di librerie di funzioni da usare nel database.
- L'admin evita di concedere permessi non necessari, concedendo solo il permesso di eseguire determinate stored procedure.



Backend security

Dance like nobody's watching...

encrypt like everyone is!



I dati salvati nel database vengono criptati, per poi essere decriptati in modo trasparente per utenti autorizzati. Se anche venissero rubate copie fisiche del DB, sarebbero comunque criptate.

Virtual Private Database

Un database privato virtuale o VPD maschera i dati in un database più grande in modo che sembri esistere solo un sottoinsieme di dati, senza effettivamente separare i dati in tabelle, schemi o database diversi.

I vantaggi:

- protezione dei dati
- facilità di accesso per gli utenti autorizzati
- possibilità di stabilire regole semplici e facilmente modificabili
- difficoltà di bypassare le restrizioni (che sono legate direttamente ai dati)

Virtual Private Database

Serve a controllare l'accesso ai dati a livello di record o di colonna.

Quando un utente accede direttamente o indirettamente a una tabella o una vista protetta, la VPD policy modifica dinamicamente l'istruzione SQL dell'utente.

```
select * from orders
```



```
select * from orders  
where customer = 106
```

CUST_FIRST_NAME	CUST_LAST_NAME	CUSTOMER_ID
Matthias	Hannah	106

ORDER_DATE	CUSTOMER_ID	ORDER_TOTAL
31-AUG-99 09:19:37.811132 AM	105	22150.1
20-MAR-96 05:18:21.862632 PM	106	5546.6
01-AUG-00 10:22:48.734526 AM	106	2075.2
31-AUG-99 08:53:06.008765 PM	107	70576.9



VPD: column relevance

Il VPD può essere configurato in modo che la policy venga attivata solo quando una colonna critica viene selezionata.

```
select cust_last_name, cust_first_name, account_mgr_id  
from customers
```

CUST_LAST_NAME	CUST_FIRST_NAME	ACCOUNT_MGR_ID
Roberts	Ishwarya	145
Steenburgen	Gustav	145
Olin	Hal	147
Kanth	Hannah	147
Seignier	Blake	149
Powell	Claude	149

```
select cust_last_name, cust_first_name, credit_limit, account_mgr_id  
from customers
```

CUST_LAST_NAME	CUST_FIRST_NAME	CREDIT_LIMIT	ACCOUNT_MGR_ID
Seignier	Blake	1200	149
Powell	Claude	1200	149

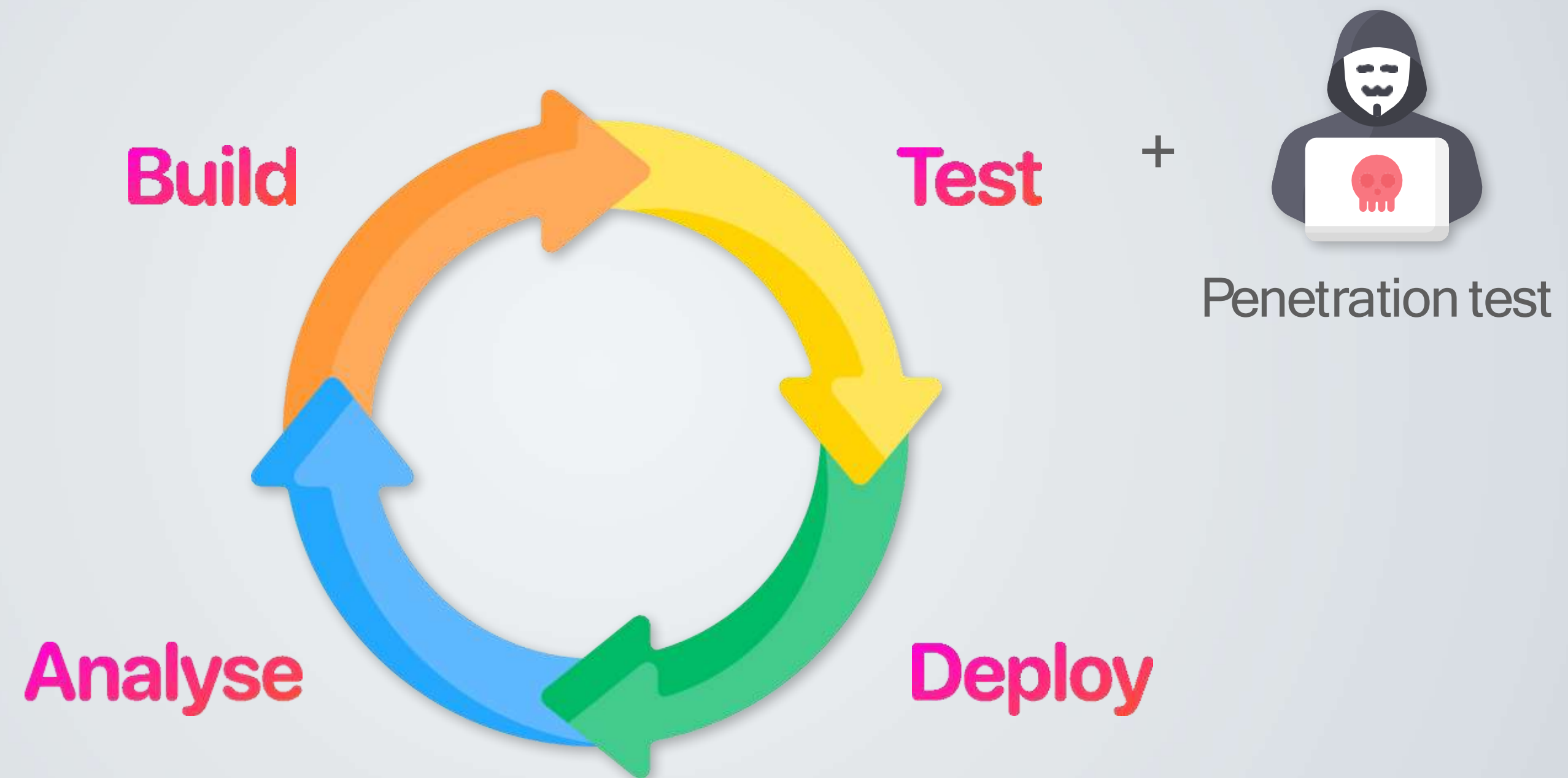
VPD: column hiding

La configurazione *column hiding* permette di avere accesso a tutti i record, ma le informazioni confidenziali restano nascoste.

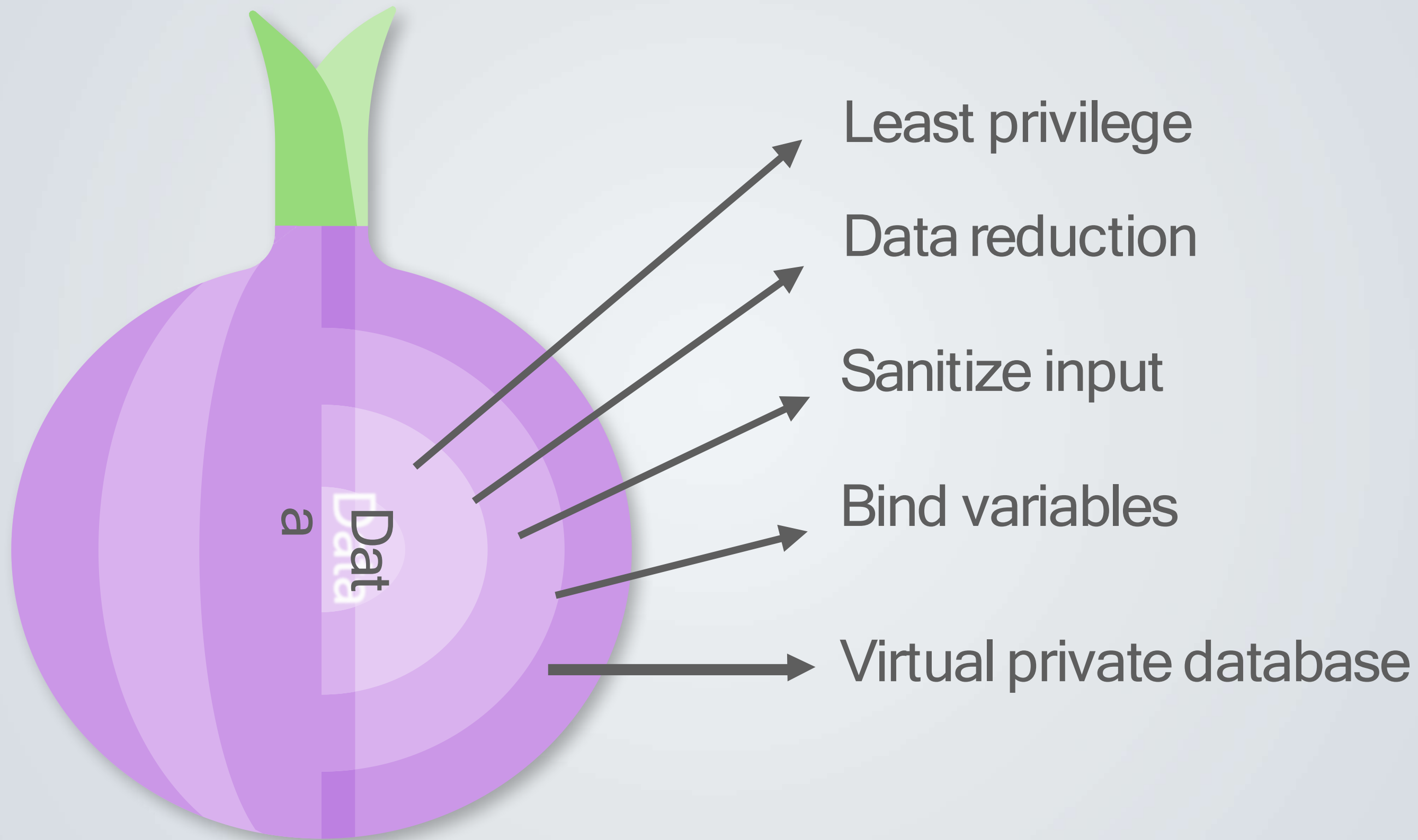
```
select cust_last_name, cust_first_name, credit_limit, account_mgr_id  
from customers
```

CUST_LAST_NAME	CUST_FIRST_NAME	CREDIT_LIMIT	ACCOUNT_MGR_ID
Edwards	Guillaume		145
Mahoney	Maurice		145
Warden	Maria		147
Landis	Marilou		147
Dvrrie	Rufus		148
Belushi	Rufus		148
Seignier	Blake	1200	149
Powell	Claude	1200	149

Attaccate il vostro backend



La risposta è “la cipolla”.



Link utili

- OWASP Top Ten - <https://owasp.org/www-project-top-ten/>

