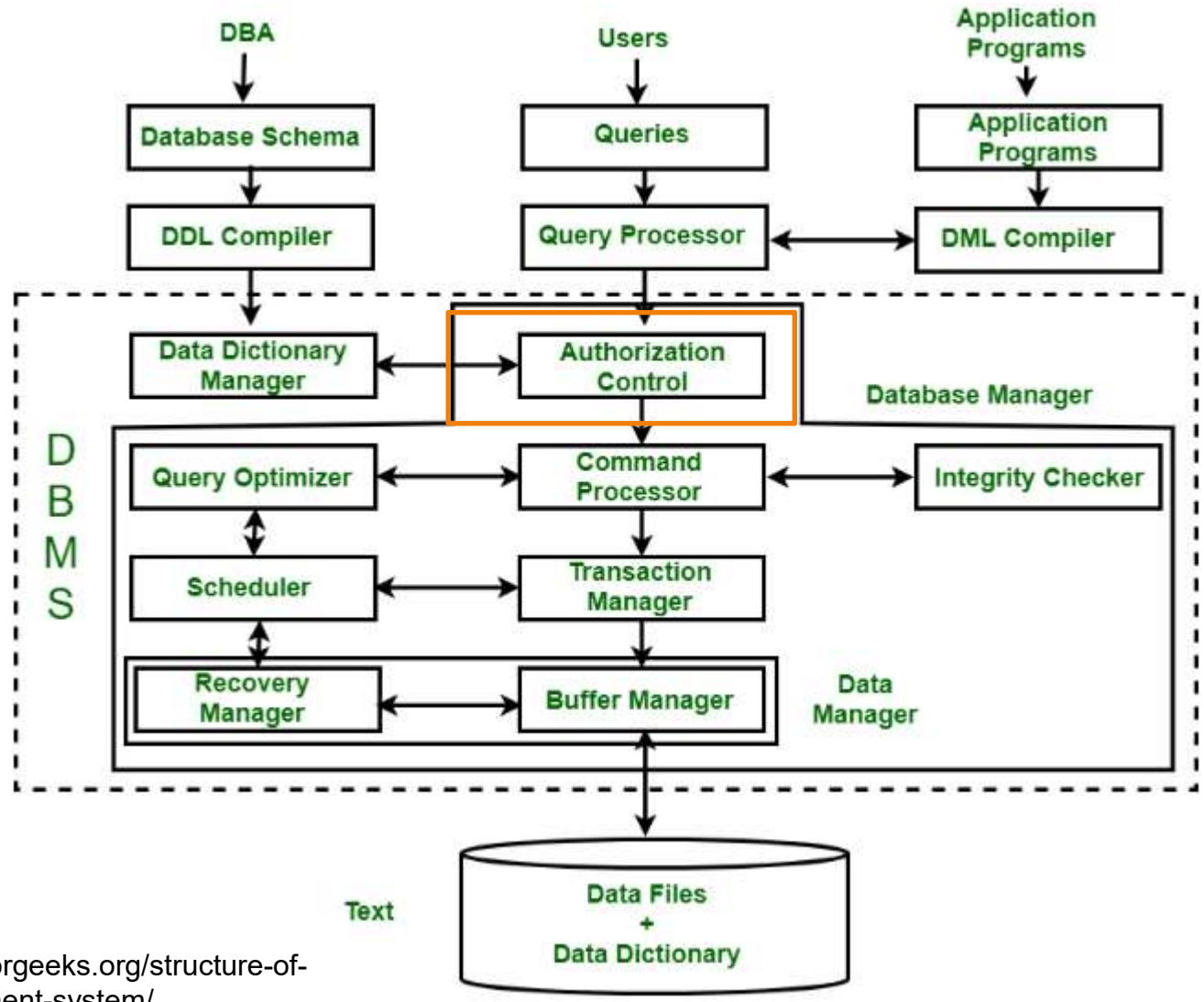
CHAPTER 3:

AUTHORIZATION / ACCESS CONTROL

Zakia Challal
zakiachallal@gmail.com

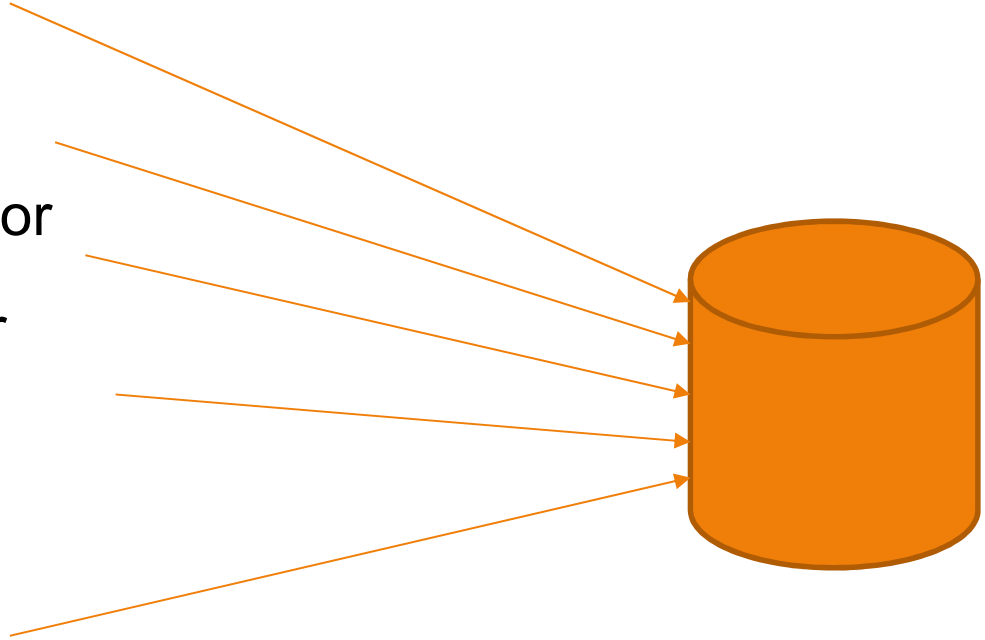
DBMS internal architecture



- Source: https://www.geeksforgeeks.org/structure-of-database-management-system/

Database users

- Database designer
- Database developer
- Database administrator
- Application developer
- End users
  - Naïve users
  - Specialized users



They have different roles and different tasks

=> Need different access authorization

Access control

- Access control is a method that determines who has access to which resources
- Access control methods include:
  - **Creating accounts** for users → Authentification
  - **Granting privileges** to users who need to use the system.
  - **Revoking privileges** from users
  - **Defining security levels** by classifying users and data in accordance with the policy of the organization.

Users accounts

- **Whenever a user need to access a database system, he must first apply for a user account.**
  - The DBA create a new **account id** and **password** for the user if he/she has a legitimate need to access the database
- **The user must log in to the DBMS by entering account id and password whenever database access is needed.**

Granting priveleges

- Granting privileges is giving the righ to some **users** to do some **operations** on some **objects** of the database.

- Granting privileges is the responsability of the DBA; a surper user who has the privilige to give privileges.

- Syntaxe:

GRANT privilege [ON table/view] TO user [WITH GRANT/ADMIN OPTION]

- Example:

GRANT select, delete, update ON employee TO administrateur;

Granting priveleges

- Granting all the privileges to a user:

Syntaxe:

GRANT ALL PRIVILEGES TO user;

- Granting a privilege to all users:

Syntaxe:

GRANT privileges TO PUBLIC;

GRANT/ADMIN OPTION

- The GRANT/ADMIN OPTION is used to propagate the privileges to other users.
- A user granted a privilege with GRANT/ADMIN OPTION, can grant that privilege to other accounts.
- We use [with Grant option] for **Object privileges** operations like: select, insert, …on table
- We use [with Admin option] for **System privileges** like: create user, create table…

Note:

Suppose that A grants privileges on R to B with GRANT OPTION and then B grants these privileges on R to a third account C, also with GRANT OPTION. In this way, privileges on R can propagate to other accounts without the knowledge of the owner of R.

Revoking priveleges

- Revoking privileges is retrieving the righ to some **users** from doing some **operations** on some **objects** of the database.

- Revoking privileges is the responsability of the DBA; a surper user who has the privilige to revoke privileges.

- Syntaxe:

REVOKE privilege [ON table/view] FROM user;

- Example:

REVOKE delete ON employee FROM developer;

Creating roles

- **Roles** are used to group together privileges or other roles. They are a means of facilitating the granting of multiple privileges or roles to users.
- Roles are defined upon the entreprise policy which classify users according the their role (function/activity) in the system and grant the privileges accordingly.

- Syntaxe:

```
CREATE ROLE rolename;
GRANT privileges to rolename;
GRANT rolename to username;
```

Creating roles / Example

- End users have just the right to **read** data about **products**.
- Creating a role:

```
> create role EndUser;
```

- Granting specific privileges the role

```
> grant select on Products to EndUser;
```

- Assign users to the role:

```
> grant EndUser to user1;
> grant EndUser to user2;
...
```

Profile

- A user profile is a **set of limits** on the database **resources** and the user **password**.
- A user assigned to a Profile cannot exceed the database resource and password limits defined in that Profile.
- Parameters defined in a Profile:
  - Resource parameters
  - Password parameters

- **Syntaxe:**

  CREATE PROFILE profile_name **LIMIT** { resource_parameters | password_parameters};

Profile: resource parameters

- **SESSIONS_PER_USER:** specify the number of concurrent sessions that a user can have when connecting to the Oracle database.
- **CPU_PER_SESSION**: specify the CPU time limit for a user session, represented in hundredth of seconds.
- **CPU_PER_CALL:** specify the CPU time limit for a call such as a parse, execute, or fetch, expressed in hundredths of seconds.
- **CONNECT_TIME:** specify the total elapsed time limit for a user session, expressed in minutes.
- **IDLE_TIME**: specify the number of minutes allowed for periods of continuous inactive time during a user session.
- **LOGICAL_READS_PER_SESSION**: specify the allowed number of data blocks read in a user session, including blocks read from both memory and disk.
- **LOGICAL_READS_PER_CALL**: specify the allowed number of data blocks read for a call to process a SQL statement.
- **PRIVATE_SGA:** specify the amount of private memory space that a session can allocate in the shared pool of the system's global area (SGA).
- **COMPOSITE_LIMIT**: specify the total resource cost for a session, expressed in service units. The total service units are calculated as a weighted sum of CPU_PER_SESSION, CONNECT_TIME, LOGICAL_READS_PER_SESSION, and PRIVATE_SGA.

Profile: password parameters

- **FAILED_LOGIN_ATTEMPTS**: Specify the number of consecutive failed login attempts before the user is locked. The default is 10 times.
- **PASSWORD_LIFE_TIME**: specify the number of days that a user can use the same password for authentication. The default value is 180 days.
- **PASSWORD_REUSE_TIME**: specify the number of days before a user can reuse a password.
- **PASSWORD_REUSE_MAX**: specify the number of password changes required before the current password can be reused.
- **PASSWORD_LOCK_TIME**: specify the number of days that Oracle will lock an account after a specified number of consecutive failed logins. The default is 1 day if you omit this clause.
- **PASSWORD_GRACE_TIME** : specify the number of days after the grace period starts during which a warning is issued and login is allowed. The default is 7 days when you omit this clause.

Profile: Example

- Create a profile

```
create profile Profil1 limit
sessions_per_user 3
cpu_per_call 2000
connect_time 60
logical_reads_per_session 1200
private_sga 40K
idle_time 15
failed_login_attempts 5
password_lock_time 1
password_life_time 70
password_reuse_time 50
password_reuse_max unlimited
password_grace_time 5;
```

- Assign the profile to a user

```
alter user User1  profile Profil1;
```

Access control Using Views

- The mechanism of views is an important authorization mechanism.
- Example1: **Limiting access to certain attributes**

If the owner A of a relation R wants another account B to be able to retrieve only some fields of R, then A can create a view V of R that includes only those attributes and then grant SELECT on V to B.

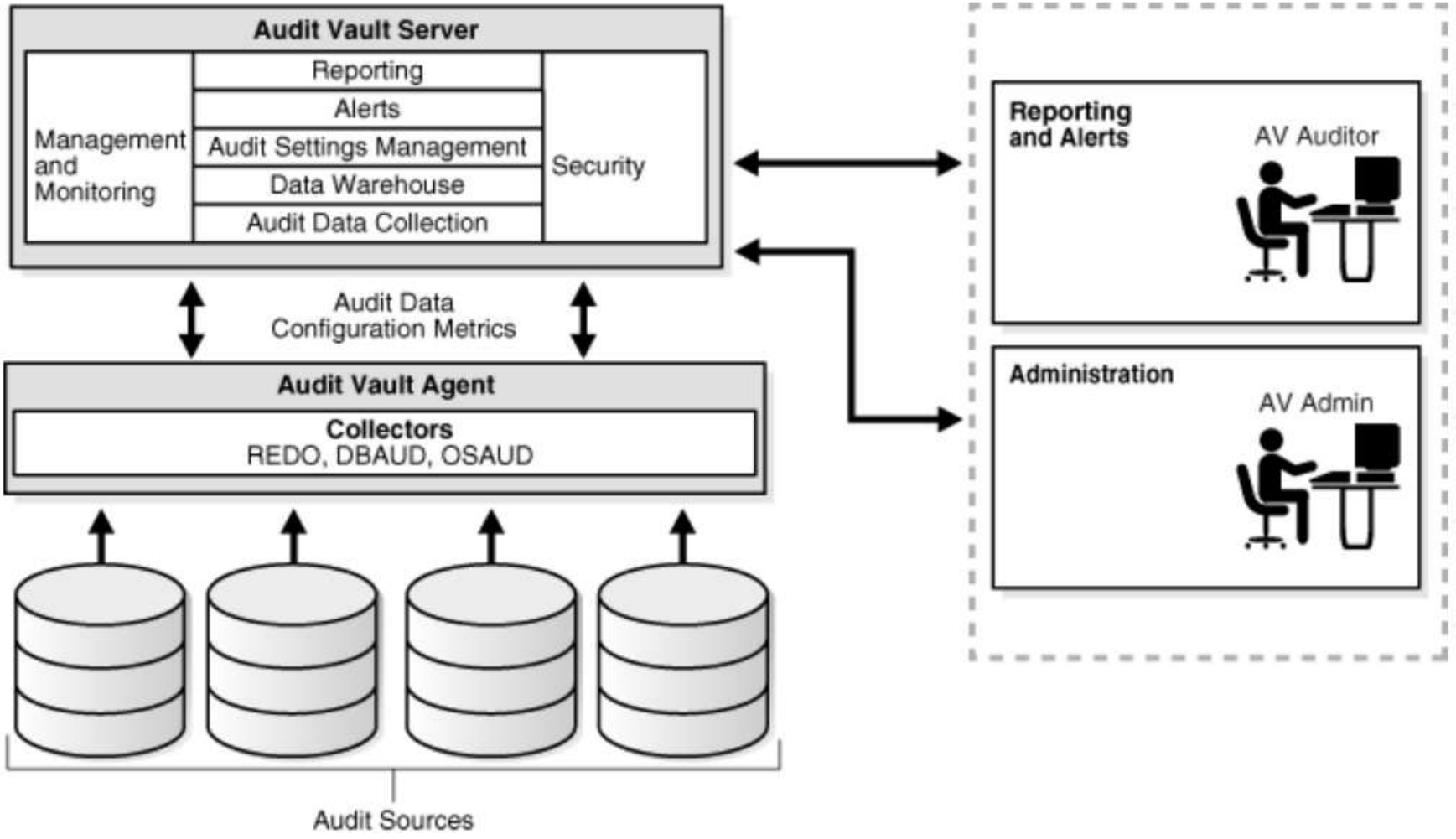
- Example 2: **Limiting access to certain tuples**

If A wants limiting B to retrieving only certain tuples of R; a view V' can be created by defining the view by means of a query that selects only those tuples from R that A wants to allow B to access.

Database audits

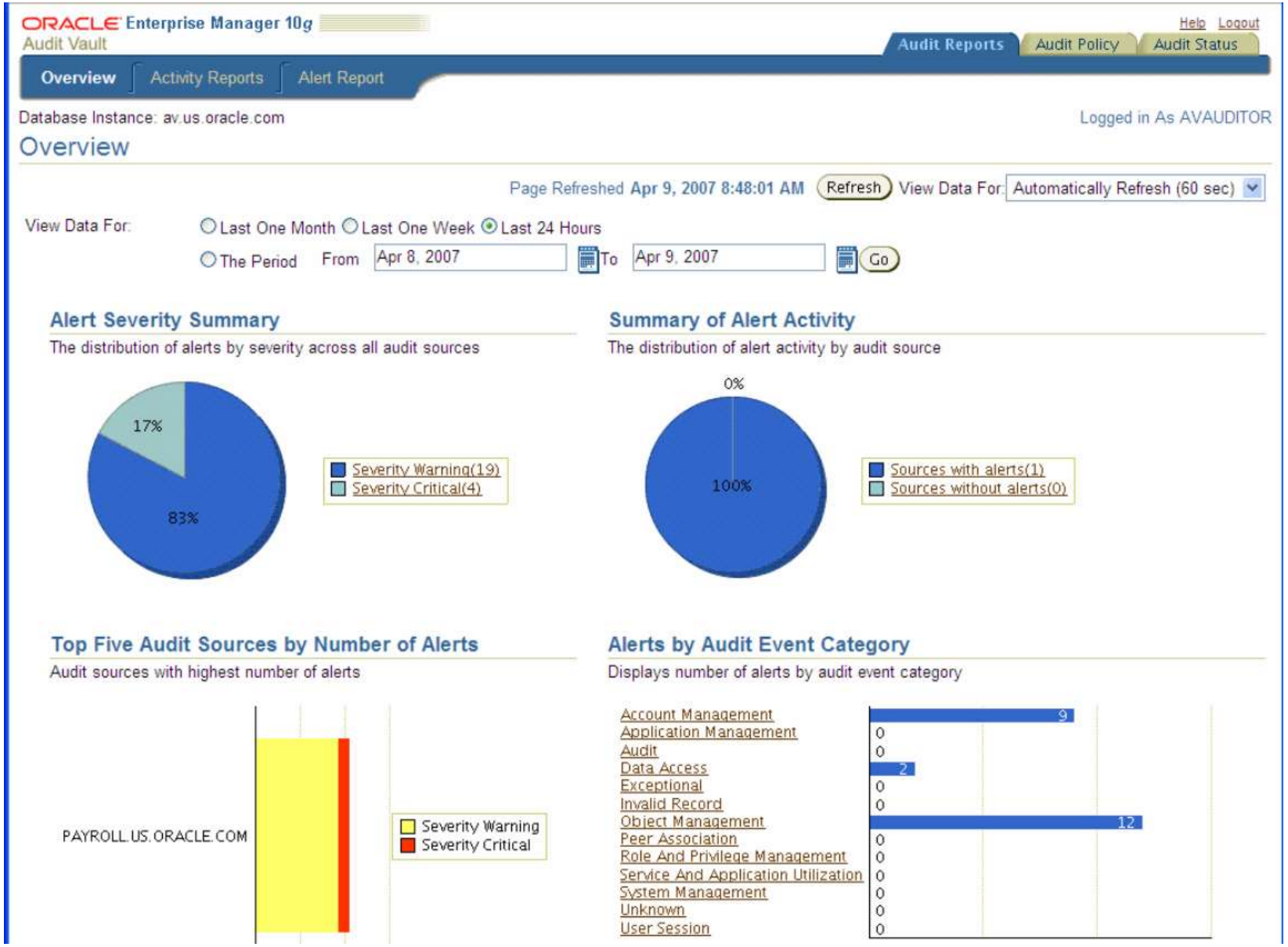
- The database system must **keep track of all operations on the database**
- A database audit consists of reviewing the **log files** to examine all accesses and operations applied to the database during a certain time period.
  - **which database object or data record was accessed, who did the action, and when it occurred**.
- Auditing enables the **detection of unauthorized actions**, as well as the actions performed by authorized users.
- To analyse the vast quantities of audit log data, we need audit tools.
  - Ex: **Oracle Audit Vault**

Oracle Audit Vault Architecture



Source: https://docs.oracle.com/cd/B25369_01/server.102/b28853/avusr_overview.htm

Oracle Audit Vault Dashboard



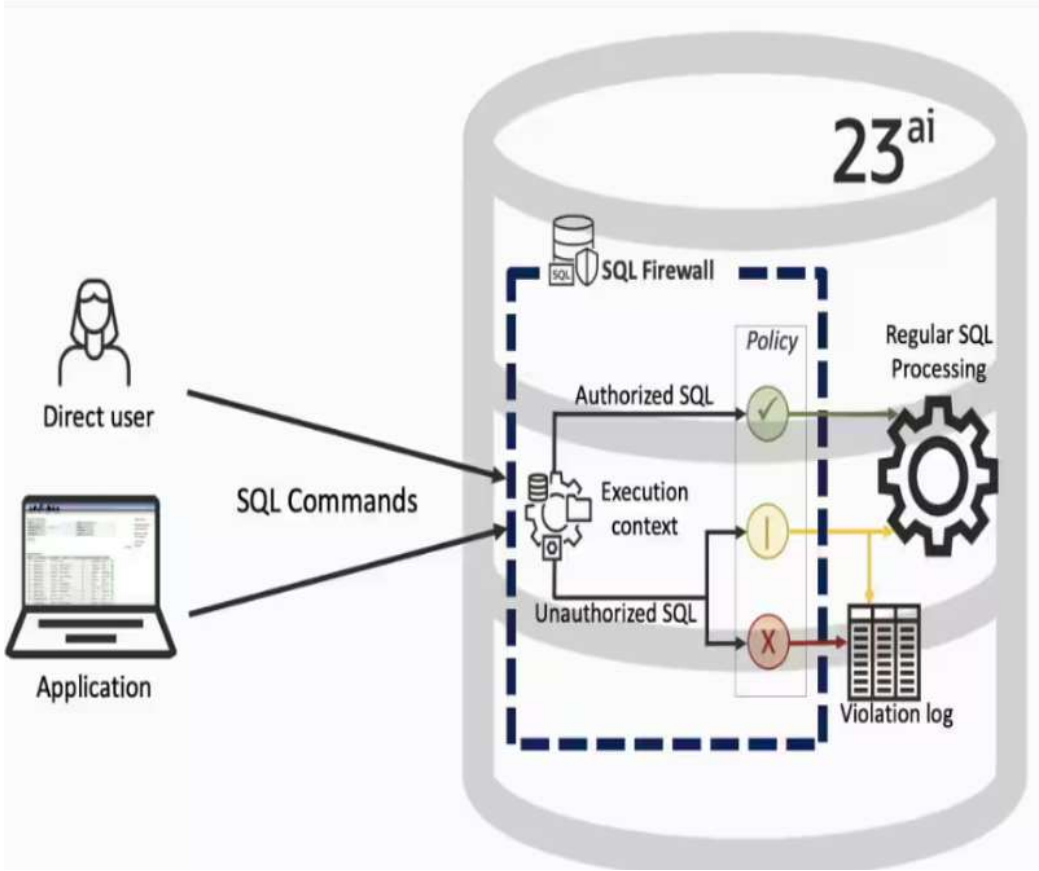
Audit with SQL

```
-- Enable auditing
AUDIT SESSION;
AUDIT SELECT TABLE, INSERT TABLE, UPDATE TABLE,
DELETE TABLE BY ACCESS;

-- Check audit trail
SELECT * FROM DBA_AUDIT_TRAIL;
```

SQL firewall

- SQL Firewall **inspects** all incoming database connections and SQL statements, whether local or over the network, encrypted or clear text.
- It only allows explicitly authorized SQL and can log or **block** SQL statements and connections that do not fall within the SQL Firewall allowlists.




```
SQL> exec dbms_sql_firewall.enable; PL/SQL procedure successfully
completed. SQL> SQL> select status from dba_sql_firewall_status; STATUS -
------- ENABLED SQL>
```

SQL Firewall: How it works?

1. Learning Mode

The firewall observes SQL commands executed by an application or user and builds a **“known good” model** (approved list).

```
EXEC DBMS_SQL_FIREWALL.CREATE_LEARNING_SESSION('APP_USER');
```

2. Approve Learned SQLs

This stores the learned patterns as **authorized SQL statements**.

```
EXEC DBMS_SQL_FIREWALL.END_LEARNING_SESSION('APP_USER');
EXEC DBMS_SQL_FIREWALL.APPROVE_LEARNING('APP_USER');
```

3. Enforcement Mode

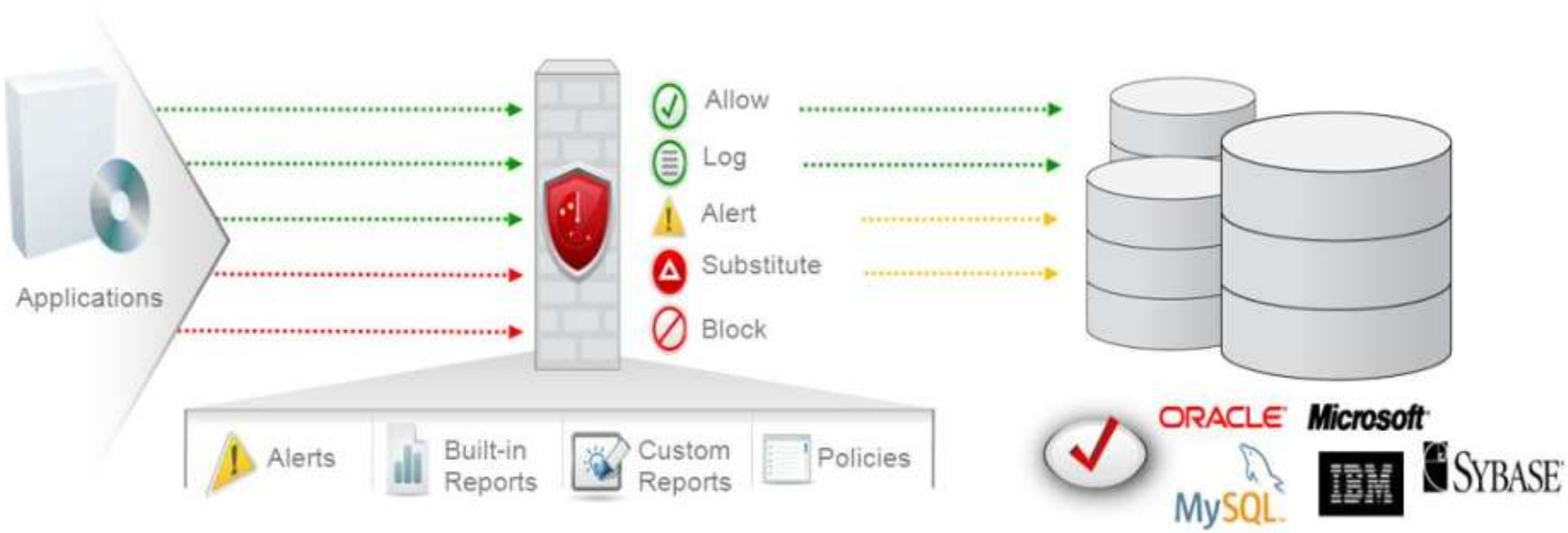
Every SQL that the user tries to execute is compared against the approved list

```
EXEC DBMS_SQL_FIREWALL.ENFORCE_USER('APP_USER', TRUE);
```

Database firewalls

- A Database Firewall is a specialized firewall product that monitors and **tracks all connections** made to a database engine.
- It can **detect** and take proactive action against various types of **attacks**, such as SQL injection, buffer overflow, and denial of service attacks.
- It can monitor network connections remotely or install an agent on the database server to inspect all database calls.
- Allows the **creation of policies** to define which statements can be run, trigger alerts, or be blocked.

Database Firewall



Source: https://www.oracle.com/technetwork/database/security/ovw-oracle-database-firewall-1447166.pdf

Oracle Database Firewall Report

Alerted Policy Anomalies by Client IP

ORACLE Database Firewall

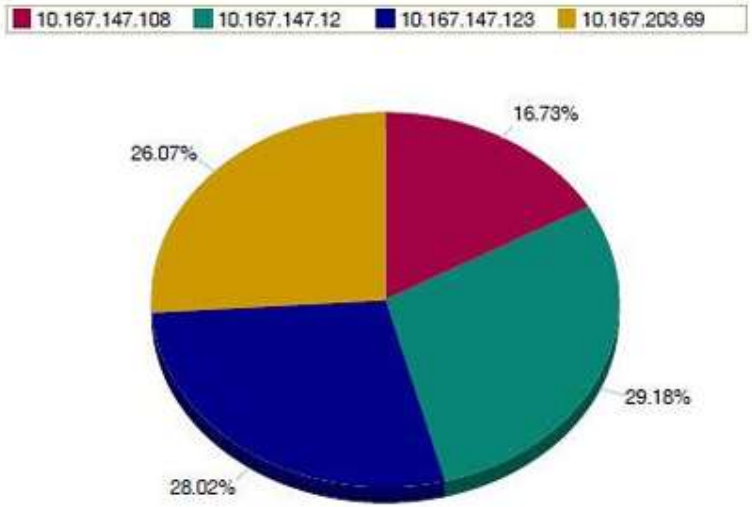
Report period: 28-NOV-2011 18:28:34 to 05-DEC-2011 18:28:34
Run by: Admin Account

Report records limit: 20000

Report Filters

Database Server IP Address: 10.167.147.104
Client IP: ALL Database User: ALL Event Action: ALL

Policy Anomaly Alerts by Client IP



Client IP: 10.167.147.12

Count / Threat	Client IP / Database IP	DB User Name	Event Action / Cluster ID
25	10.167.147.12 10.167.147.104	ura_user_1	anomaly warned 327911410
	SELECT client, proxy FROM sys.proxy_users		
25	10.167.147.12 10.167.147.104	ura_user_1	anomaly warned 3604051212
	SELECT username, grantee, granted_role, type, MAX(lvl) max_level FROM (SELECT REPLACE(SYS_CONNECT_BY_PATH(DECODE(LEVEL, 0, grantee), '#'), '#') username, grantee, granted_role, type, LEVEL lvl FROM (SELECT '####' type, grantee, granted_role FROM sys.dba_role_privs UNION ALL SELECT '#########' type, grantee, privilege FROM		

Audit and Database firewall together

- **Database Firewall** protects *before* execution (preventive).
- **Audit** investigates *after* execution (detective).

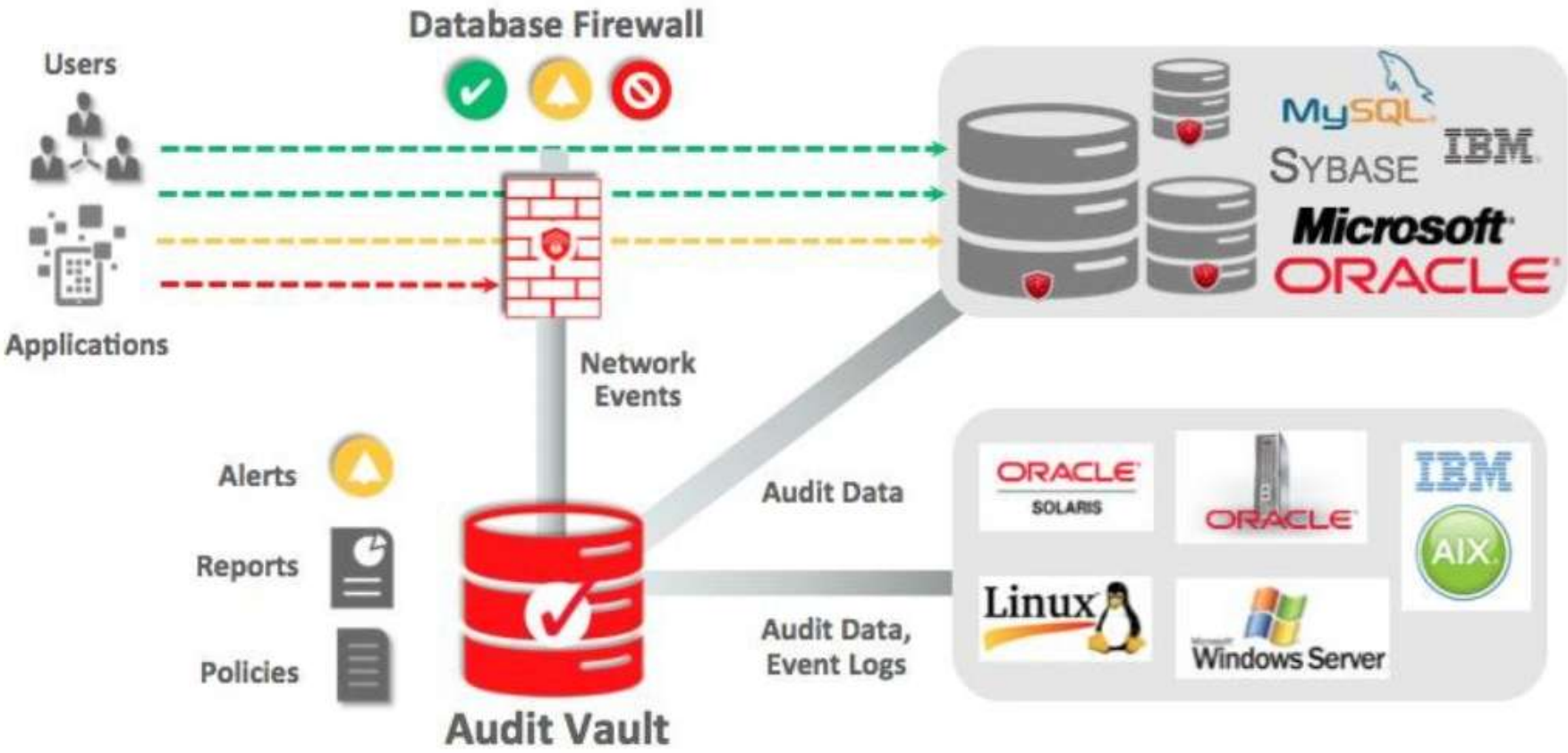
Database Firewall

- Learns and enforces **SQL whitelists** (per user/application).
- Blocks or logs **unauthorized SQLs**.
- Detects **SQL injection** and **policy violations**.
- **Sends its logs to Audit for central analysis**.

Audit Vault

- Collects audit data from:
  - Databases (Oracle, MySQL, SQL Server, etc.)
  - Operating Systems (Linux, Windows)
  - **Database Firewall**
- Stores all audit records securely.
- Offers **dashboards, reports, alerts**

Oracle Audit Vault and Database Firewall



Sourece: https://www.oracle.com/technetwork/products/audit-vault/downloads/owp-audit-vault-db-firewall-122-2844505.pdf

Questions?

Quiz

1. How to control access to a DBMS? Give different ways.
2. What is a role in a DBMS?
3. What is a profile in a DBMS?
4. How to use views to control access to a database?
5. Give a strategy to control access to a database where the main users are:
   - Database designers
   - Database developers
   - Database administrators
   - Application developers
   - End users