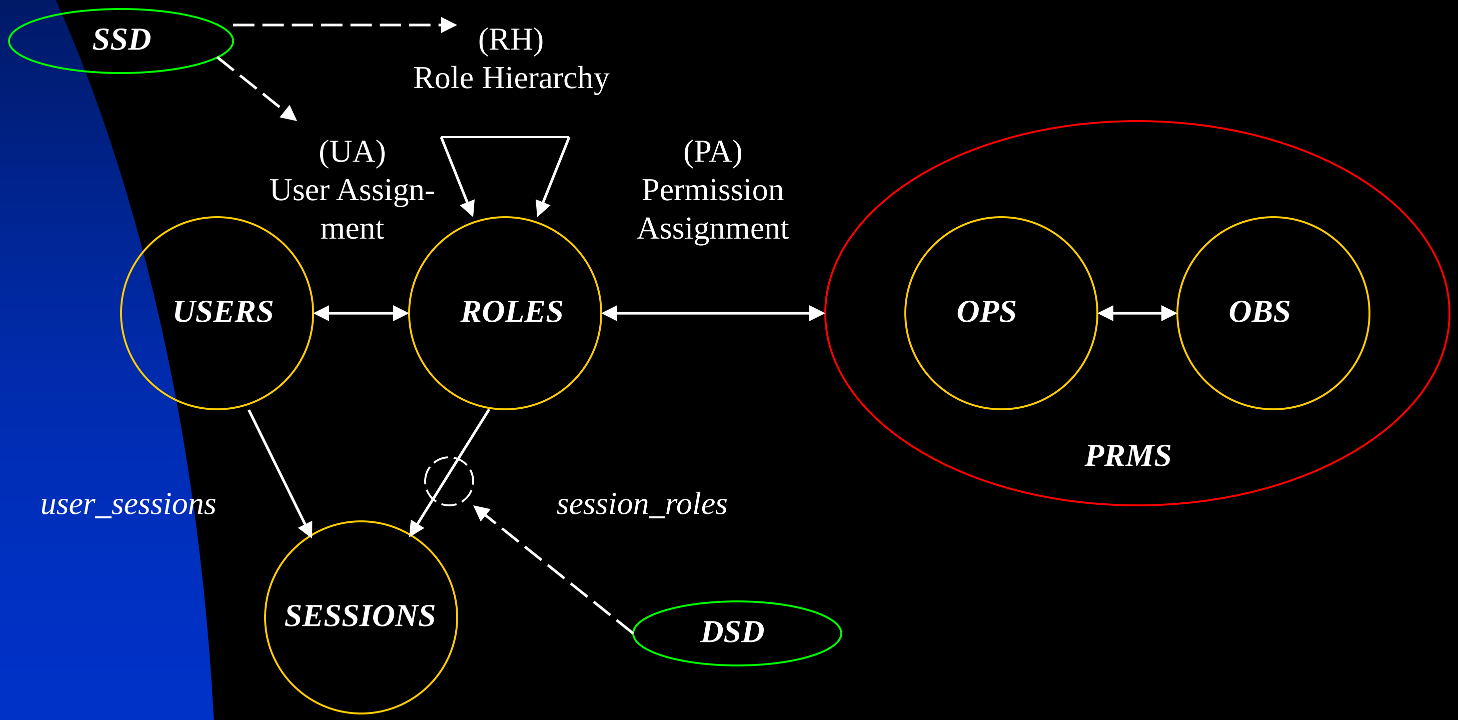
Role-Based Access Control

Overview



Objective

- Establish a common vocabulary for Role Based Access Control for use in SEPM
- Present a Framework for Role Based Access Control for both Physical and Virtual Domains
- Discuss Various AC Models and why RBAC is a must!!!!

Think about this…

- “Although the fundamental concepts of roles are common knowledge, the capability to formalize model specifications needed to implement RBAC models is beyond the knowledge base of existing staff in may software companies”
- “The lack of knowledge and staff expertise in the area of RBAC increases the uncertainty of both the technical feasibility of developing successful RBAC-enabled products and the develop cost and time frame.”

-The Economic Impact of Role-Based Access Control

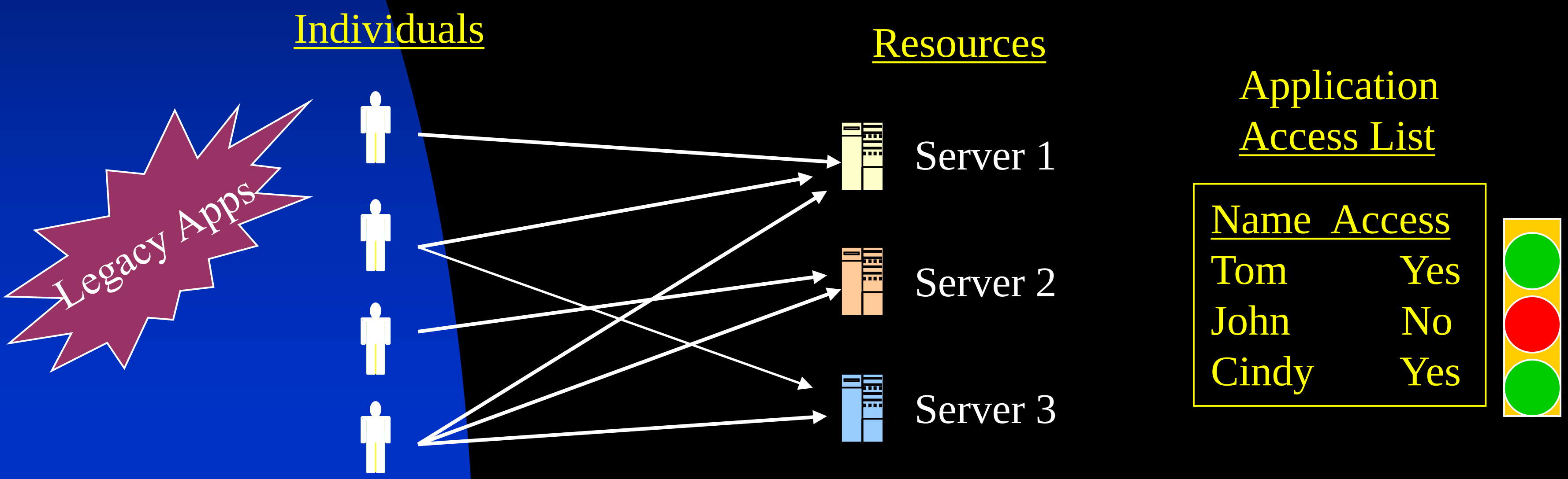
The Time is NOW!

Access Controls Types

- Discretionary Access Control
- Mandatory Access Control
- Role-Based Access Control

Discretionary AC

- Restricts access to objects based solely on the identity of users who are trying to access them.



Name	Access
Tom	Yes
John	No
Cindy	Yes

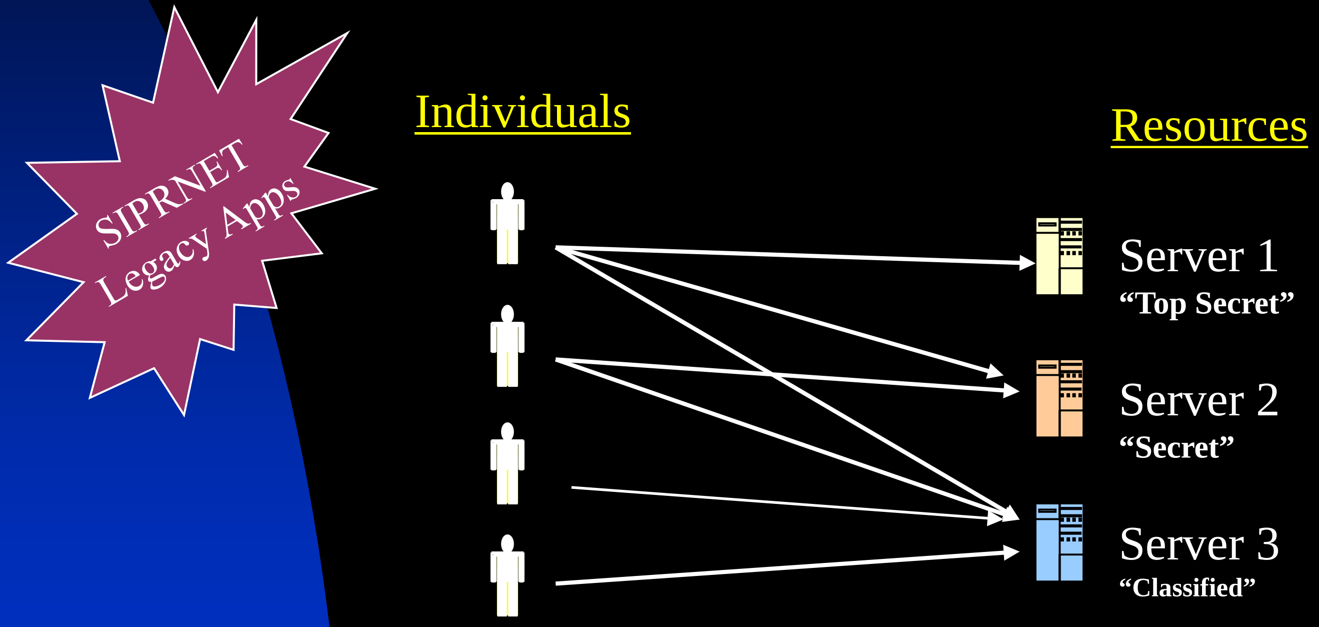
Mandatory AC

- MAC mechanisms assign a security level to all information, assign a security clearance to each user, and ensure that all users only have **access** to that data for which they have a clearance.

Principle: Read Down Access
equal or less Clearance
Write Up Access
equal or higher Clearance

Better security than DAC

Mandatory AC (cont)

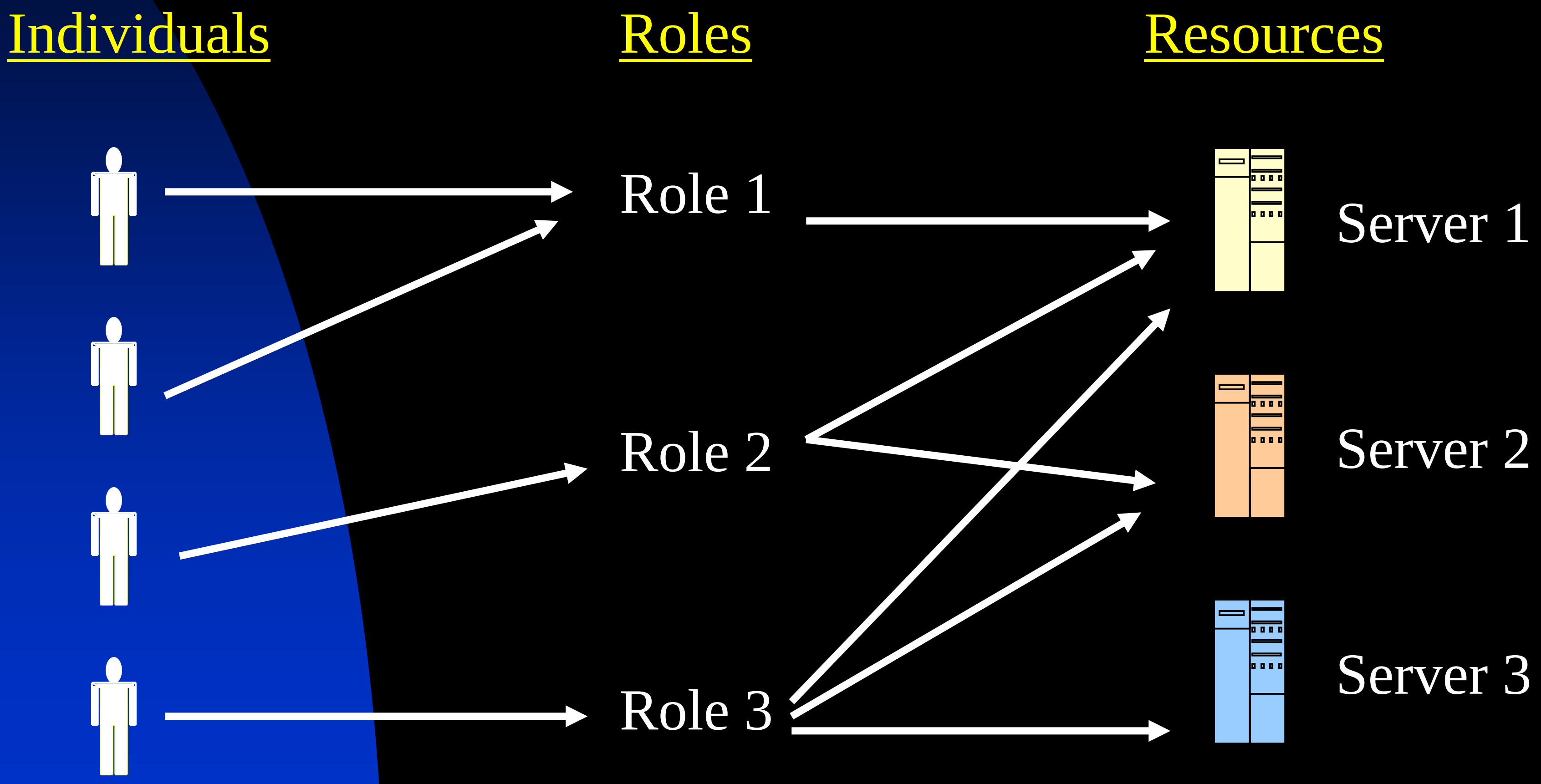


Role-Based AC

“Ideally, the [RBAC] system is clearly defined and agile, making the addition of new applications, roles, and employees as efficient as possible”

- A user has access to an object based on the assigned role.
- Roles are defined based on job functions.
- Permissions are defined based on job authority and responsibilities within a job function.
- Operations on an object are invocated based on the permissions.
- The object is concerned with the user’s role and not the user.

Role-Based AC



Individuals

Roles

Resources

Role 1

Role 2

Role 3

Server 1

Server 2

Server 3

User’s change frequently, Roles don’t

Privilege

- Roles are engineered based on the principle of least privileged .
- A role contains the minimum amount of permissions to instantiate an object.
- A user is assigned to a role that allows him or her to perform only what’s required for that role.
- No single role is given more permission than the same role for another user.

Role-Based AC Framework

- Core Components
- Constraining Components
  - Hierarchical RBAC
    - General
    - Limited
  - Separation of Duty Relations
    - Static
    - Dynamic

Core Components

- Defines:
  - USERS
  - ROLES
  - OPERATIONS (*ops*)
  - OBJECTS (*obs*)
  - User Assignments (*ua*)
    - assigned_users

Core Components (cont)

- Permissions (*prms*)
  - Assigned Permissions
  - Object Permissions
  - Operation Permissions
- Sessions
  - User Sessions
  - Available Session Permissions
  - Session Roles

Constraint Components

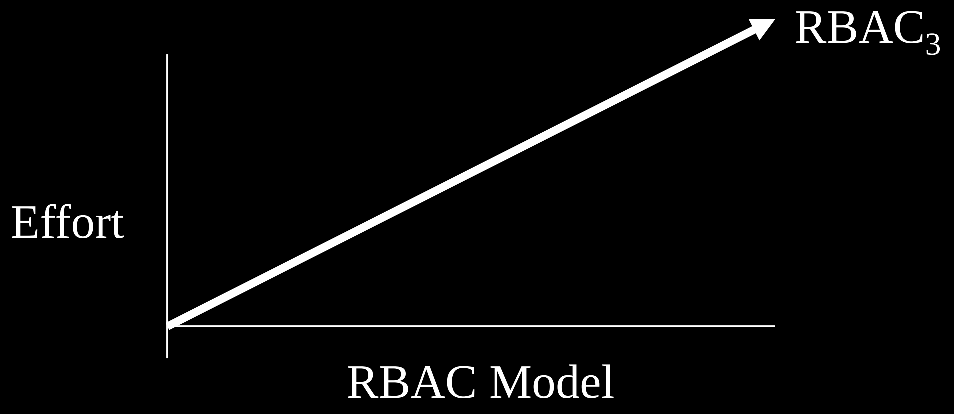
- Role Hierarchies (*rh*)
  - General
  - Limited
- Separation of Duties
  - Static
  - Dynamic

RBAC Transition

Least Privileged Separation of Duties

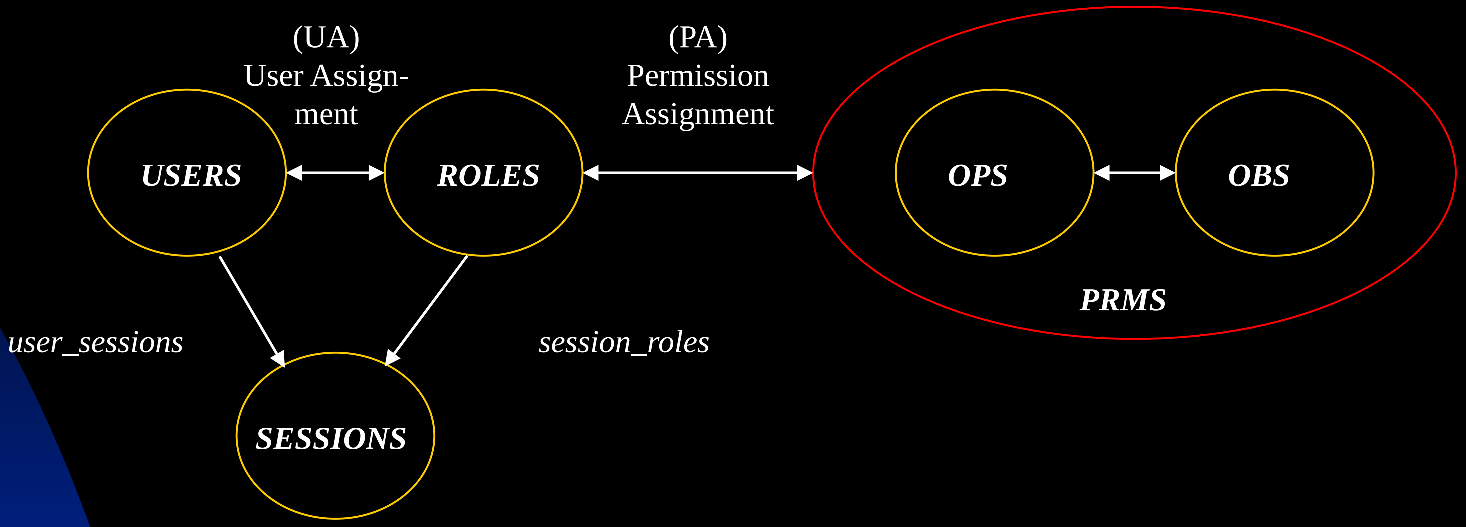
Models	Hierarchies	Constraints
$RBAC_0$	No	No
$RBAC_1$	Yes	No
$RBAC_2$	No	Yes
$RBAC_3$	Yes	Yes

Most Complex



RBAC System and Administrative Functional Specification

- Administrative Operations
  - Create, Delete, Maintain elements and relations
- Administrative Reviews
  - Query operations
- System Level Functions
  - Creation of user sessions
  - Role activation/deactivation
  - Constraint enforcement
  - Access Decision Calculation

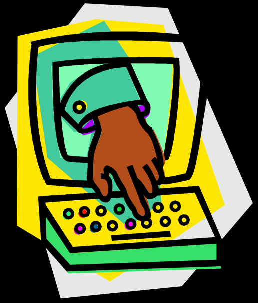

(UA)
User Assign-
ment
(PA)
Permission
Assignment
USERS
ROLES
OPS
OBS
PRMS
user_sessions
session_roles
SESSIONS

Core RBAC

USERS

Proces
s

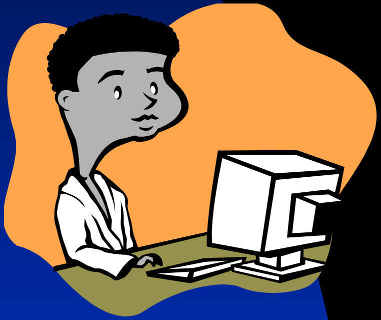
Person



Intelligent Agent

ROLES

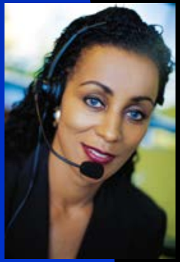
An organizational job function with a clear definition of inherent responsibility and authority (permissions).



Developer

Budget Manager

Director



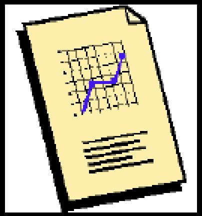
Help Desk Representative

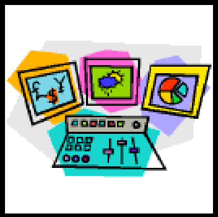
MTM relation between USERS & PRMS

OPS (operations)

An execution of an a program specific function that’s invocated by a user.

- Database – Update Insert Append Delete
- Locks – Open   Close
- Reports – Create View   Print
- Applications - Read Write Execute





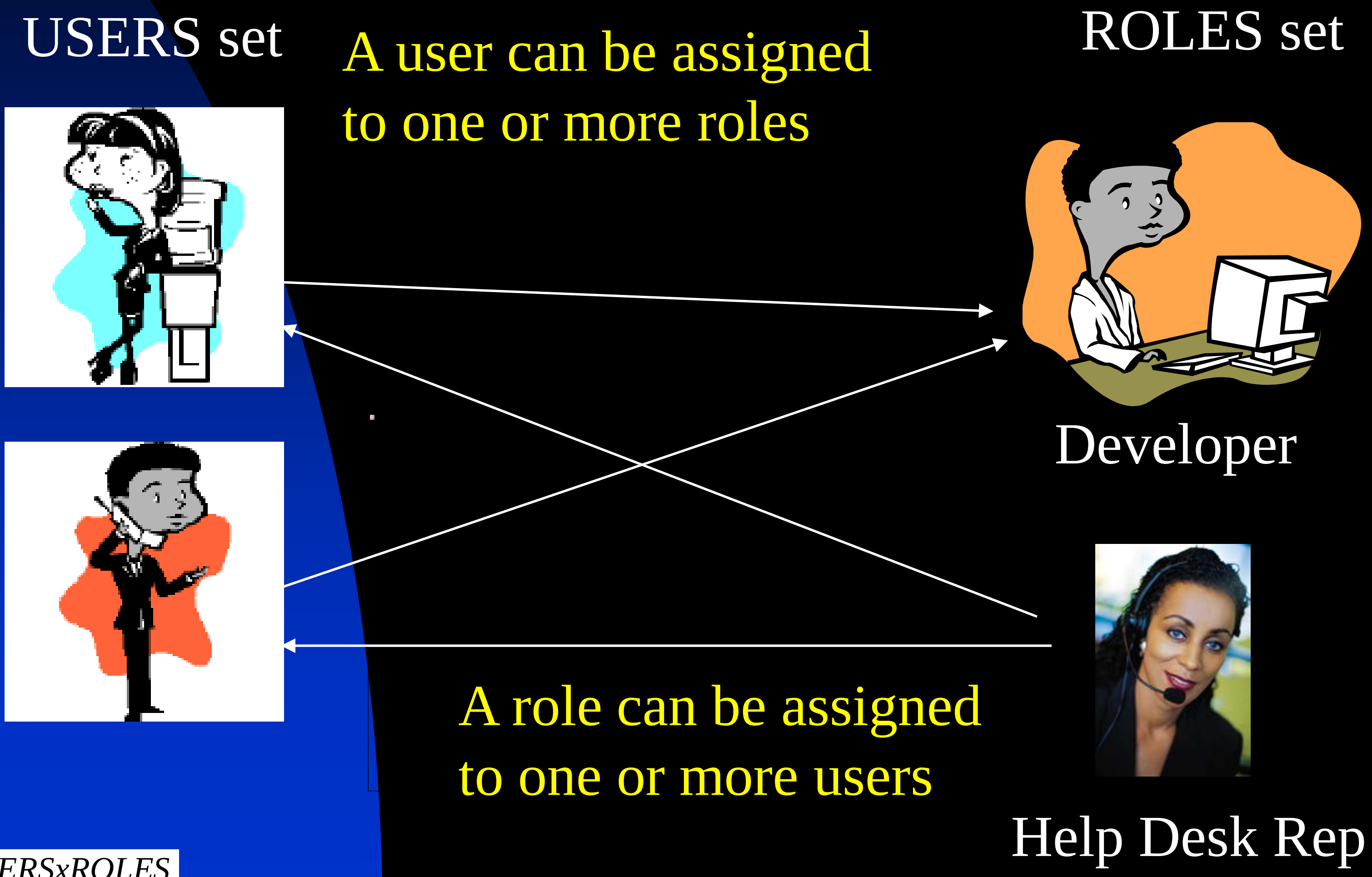
OBS (objects)

An entity that contains or receives information, or has exhaustible system resources.

- OS Files or Directories
- DB Columns, Rows, Tables, or Views
- Printer
- Disk Space
- Lock Mechanisms

RBAC will deal with all the objects listed in the permissions assigned to roles.

UA (user assignment)



USERS set

A user can be assigned to one or more roles

ROLES set

Developer

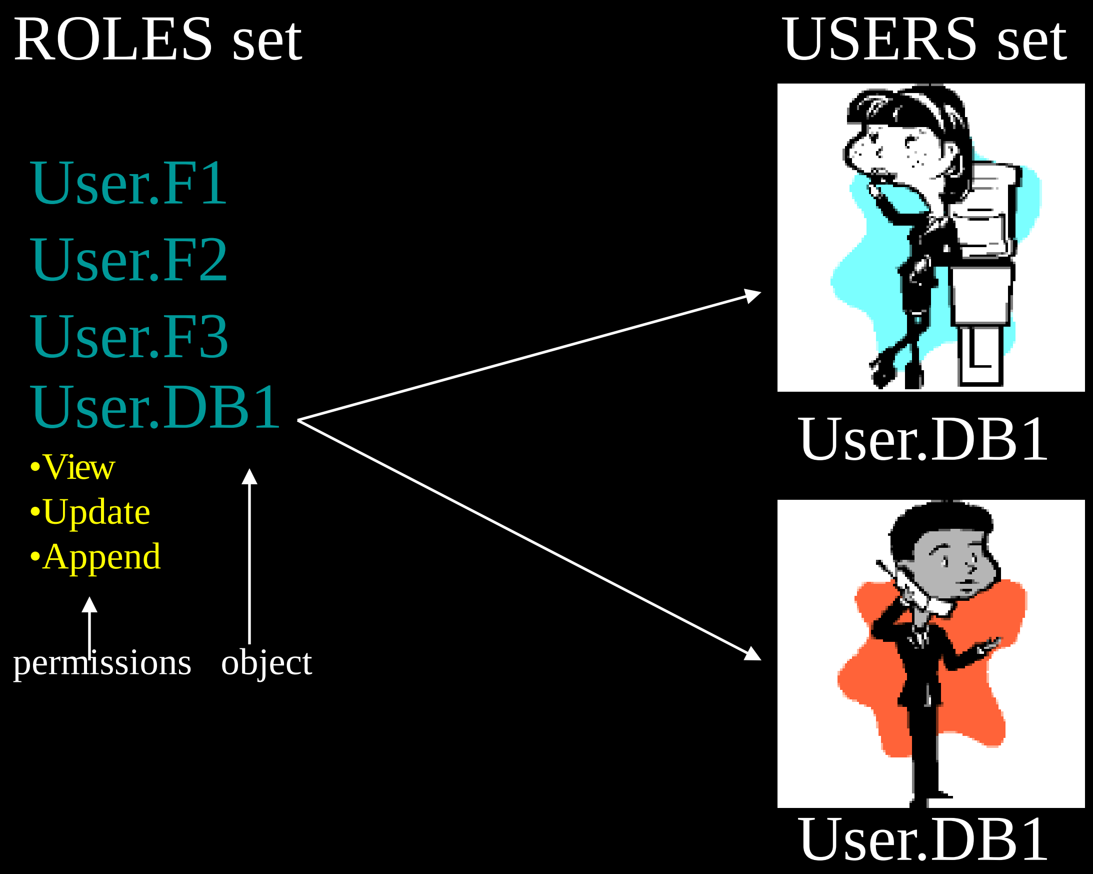
A role can be assigned to one or more users

Help Desk Rep

$UA \subseteq USERS x ROLES$

UA (user assignment)

Mapping of role *r* onto a set of users



ROLES set

User.F1

User.F2

User.F3

User.DB1

- View
- Update
- Append

permissions object

USERS set

User.DB1

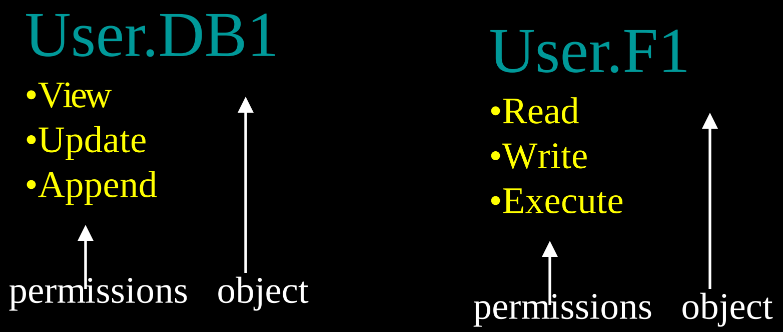
User.DB1

$$assigned\_user : (r : ROLES) \to 2^{users}$$

$$assigned\_user(r) = \{u \in USERS \mid (u,r) \in UA\}$$

PRMS (permissions)

The set of permissions that each grant the approval to perform an operation on a protected object.



User.DB1

- View
- Update
- Append

permissions object

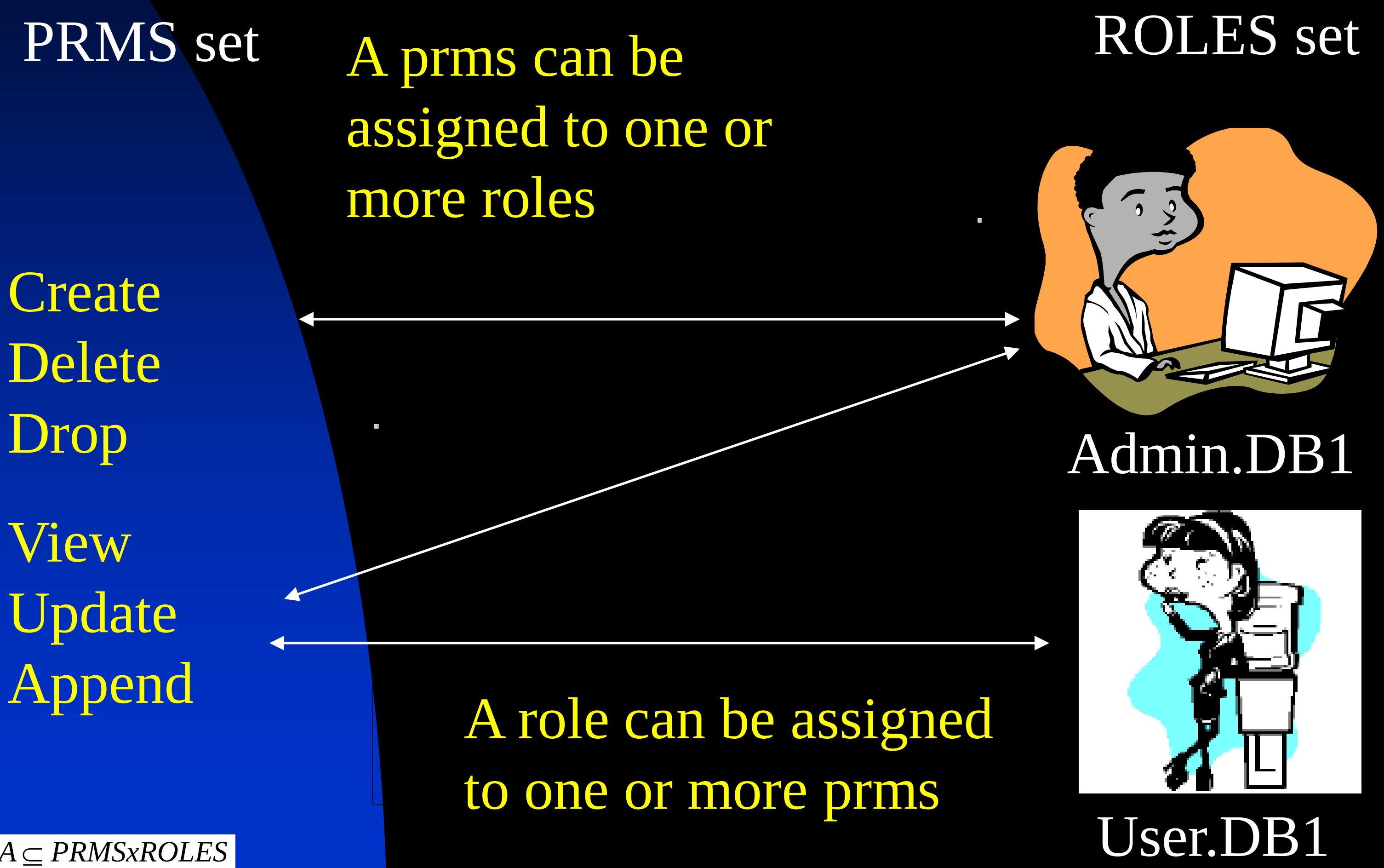
User.F1

- Read
- Write
- Execute

permissions object

$PRMS = 2^{(OPSxOBS)}$

PA (prms assignment)



PRMS set

A prms can be assigned to one or more roles

ROLES set

Create
Delete
Drop

View
Update
Append

Admin.DB1

A role can be assigned to one or more prms

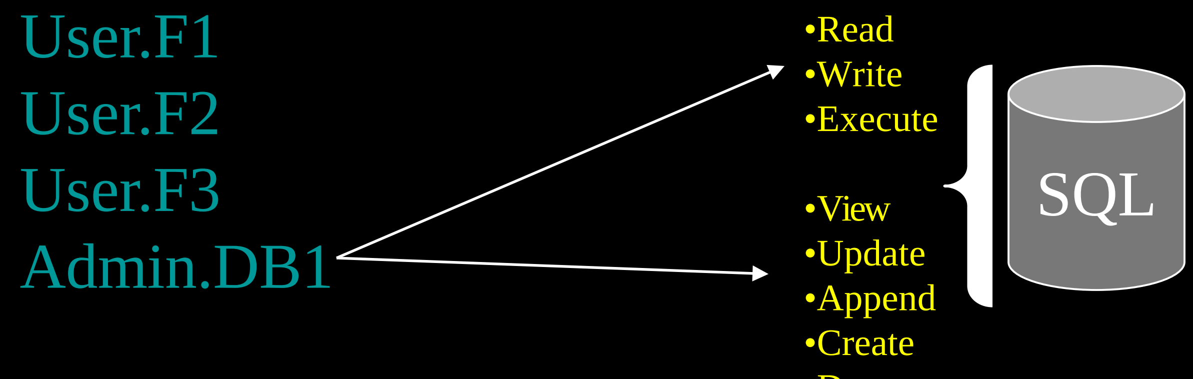
User.DB1

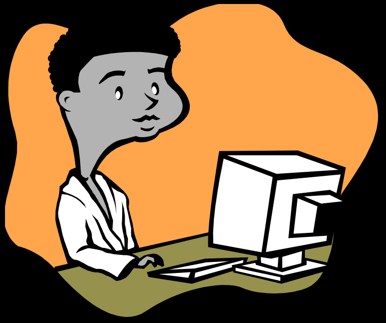
PA $\subseteq$ *PRMSxROLES*

PA (prms assignment)

Mapping of role *r* onto a.set of permissions

ROLES set PRMS set





$assigned\_permissions(r : ROLES) \rightarrow 2^{PRMS}$

$assigned\_permissions(r) = \{ p \in PRMS \mid (p, r) \in PA \}$

PA (prms assignment)

Mapping of operations to permissions

OPS set

PRMS set

public int read(byteBuffer dst)
throws IOException

Inherited methods from java.nio.channls
close()
isOpen()

READ

Gives the set of ops associated with the permission

$Ob(p : PRMS) \rightarrow \{op \subseteq OPS\}$

PA (prms assignment)

Mapping of permissions to objects

PRMS set

Objects

- Open
- Close

BLD1.door2

- View
- Update
- Append
- Create
- Drop

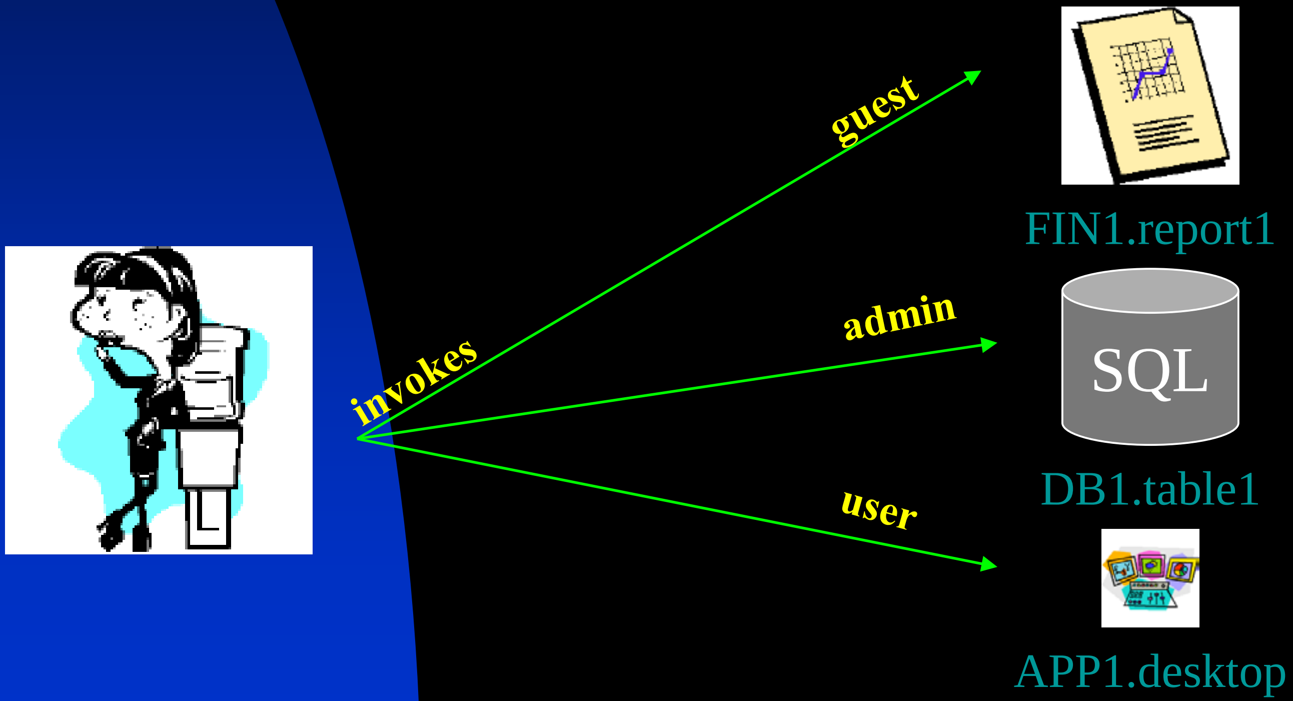
DB1.table1

Gives the set of objects associated with the prms

$Ob(p:PRMS) \rightarrow \{ob \subseteq OBS\}$

SESSIONS

The set of sessions that each user invokes.



USER

SESSION

invokes

guest

admin

user

FIN1.report1

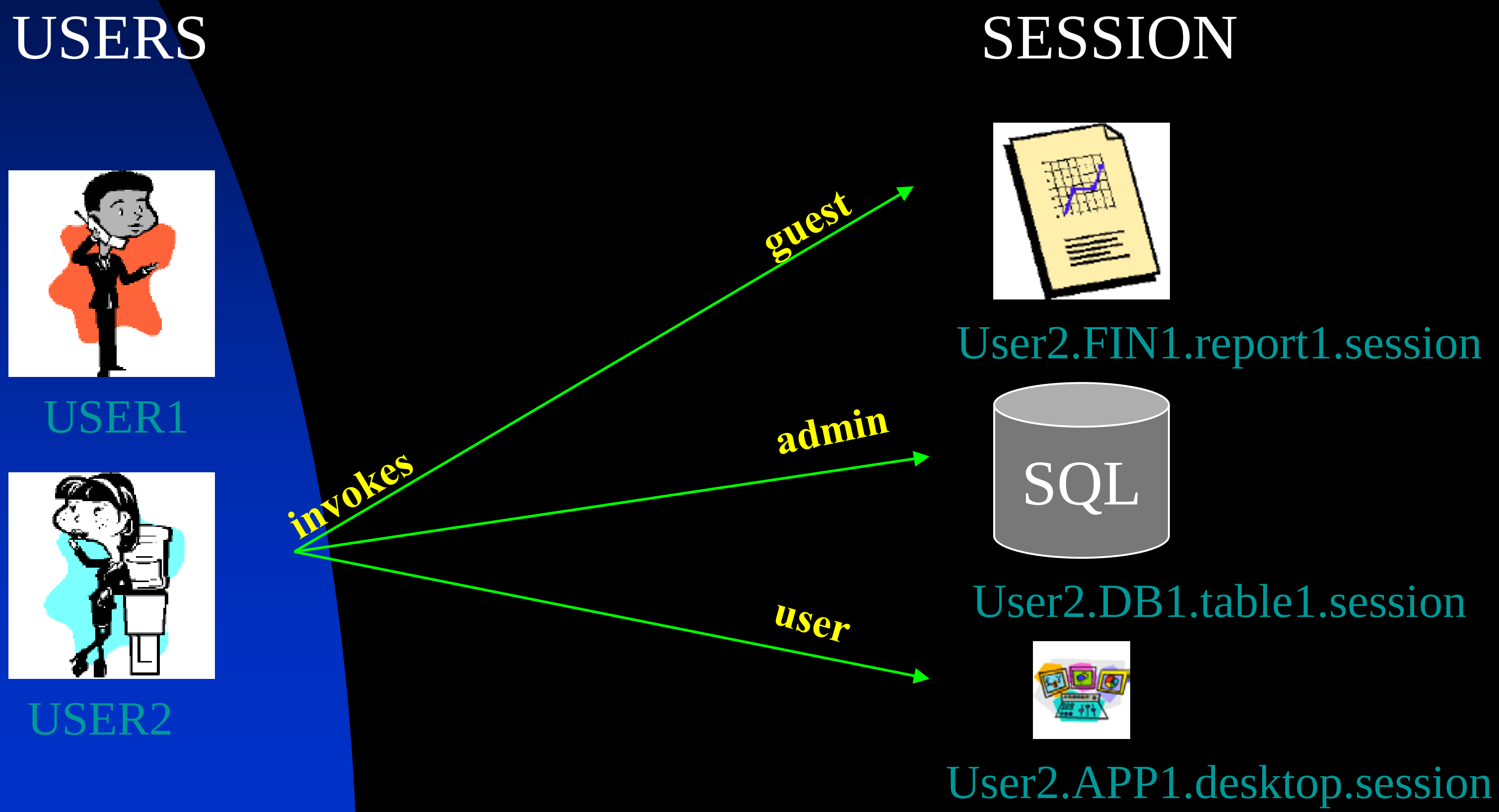
SQL

DB1.table1

APP1.desktop

SESSIONS

The mapping of user *u* onto a set of sessions.



USERS

SESSION

USER1

USER2

invokes

guest

admin

user

User2.FIN1.report1.session

SQL

User2.DB1.table1.session

User2.APP1.desktop.session

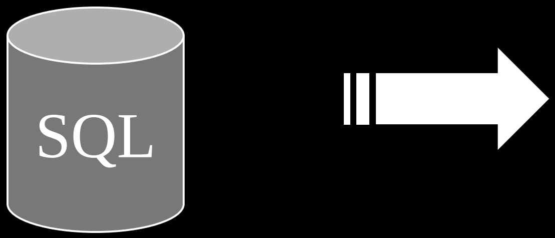
$user\_sessions(u:USERS) \rightarrow 2^{SESSIONS}$

SESSIONS

The mapping of session *s* onto a set of roles

SESSION

ROLES



- Admin
- User
- Guest

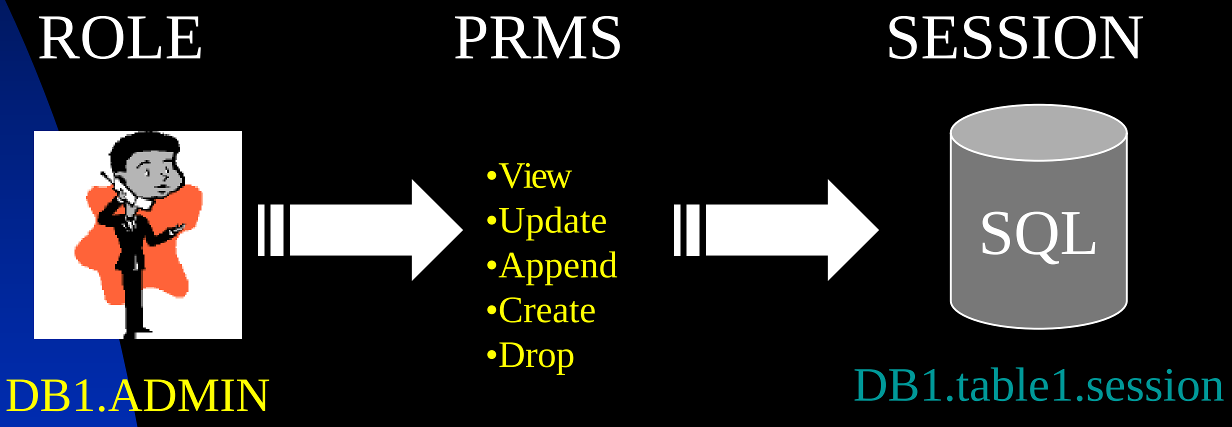
DB1.table1.session

$$session\_roles(s : SESSIONS) \rightarrow 2^{ROLES}$$

$$session\_roles(s_i) \subseteq \{r \in ROLES \mid (session\_users(s_i), r \in UA\}$$

SESSIONS

Permissions available to a user in a session.



ROLE

PRMS

SESSION

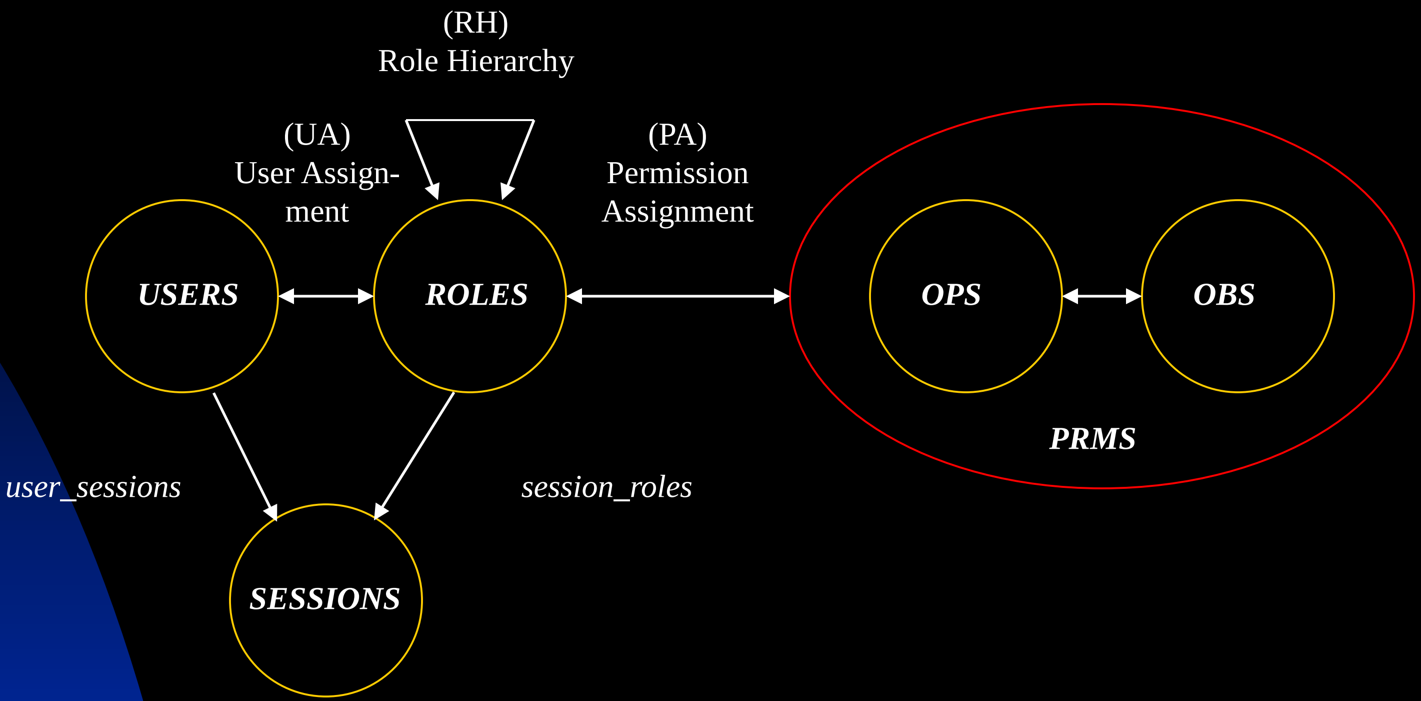
- View
- Update
- Append
- Create
- Drop

DB1.ADMIN

DB1.table1.session

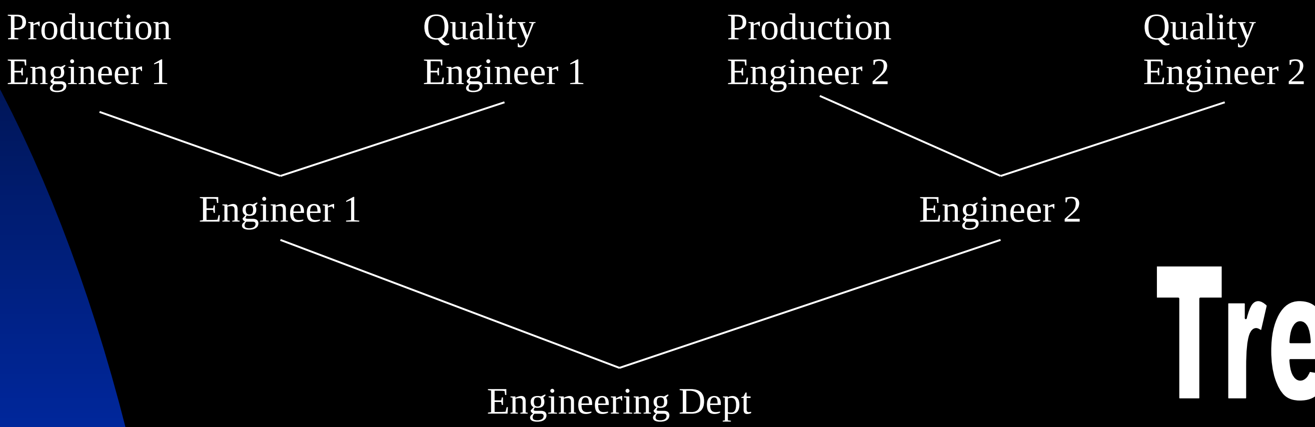
$$avail\_session\_persm(s : SESSIONS) \rightarrow 2^{PRMS}$$

$$\bigcup_{r \in session\_roles(s)} assigned\_permissions(r)$$



Hierarchal RBAC

Tree Hierarchies



Production Engineer 1

Quality Engineer 1

Production Engineer 2

Quality Engineer 2

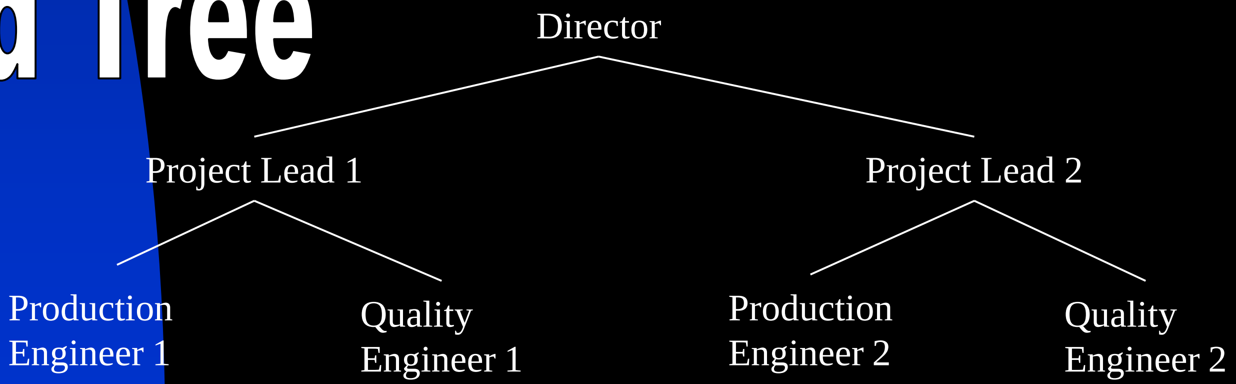
Engineer 1

Engineer 2

Engineering Dept

Tree

Inverted Tree



Director

Project Lead 1

Project Lead 2

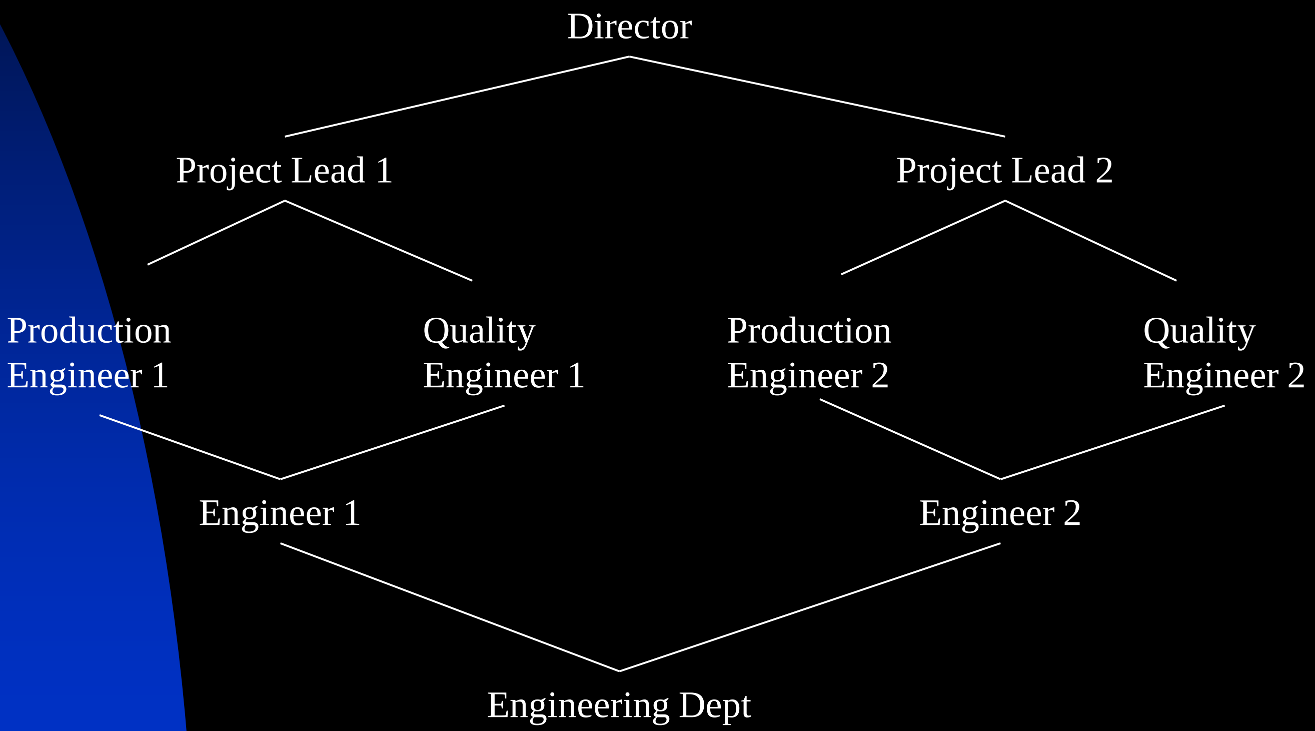
Production Engineer 1

Quality Engineer 1

Production Engineer 2

Quality Engineer 2

Lattice Hierarchy



Director

Project Lead 1

Project Lead 2

Production Engineer 1

Quality Engineer 1

Production Engineer 2

Quality Engineer 2

Engineer 1

Engineer 2

Engineering Dept

RH (Role Hierarchies)

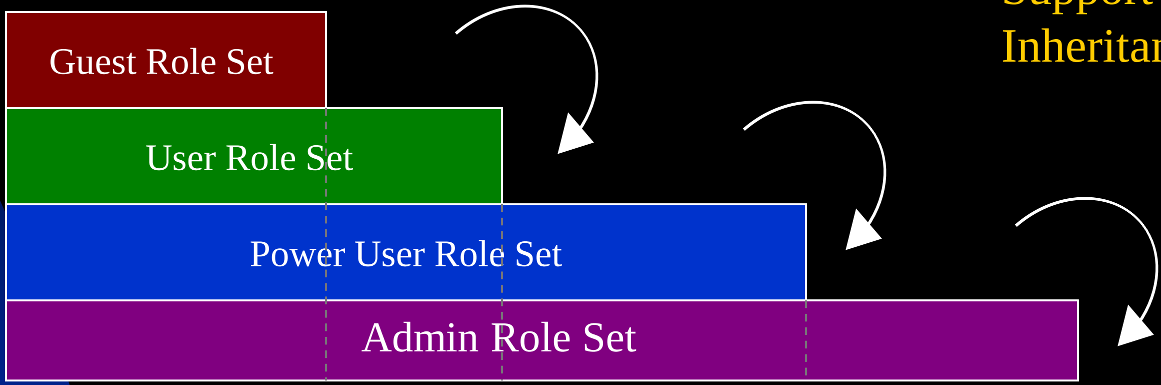
- Natural means of structuring roles to reflect organizational lines of authority and responsibilities
- General and Limited
- Define the inheritance relation among roles

  i.e. r_1 *inherits* r_2

User	Guest
r-w-h	-r-

$RH \subseteq ROLES x ROLES$

General RH



Support Multiple Inheritance

Only if all permissions of r_1 are also permissions of r_2

i.e. r1 *inherits* r2

Only if all users of r_1 are also users of r_2

User Guest

r-w-h -r-

$$r_1 \succeq r_2 \Rightarrow authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1)$$
$$\wedge\, authorized\_users(r_1) \subseteq authorized\_users(r_2)$$

authorized users

Mapping of a role onto a set of users in the presence of a role hierarchy

ROLES set

First Tier USERS set

Admin.DB1

User.DB2

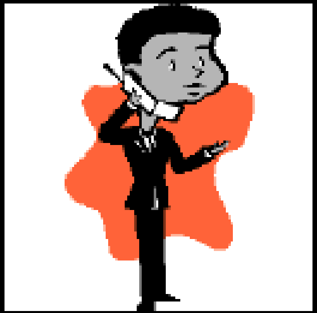
User.DB3

User.DB1

- View
- Update
- Append

permissions object

User.DB1



User.DB1

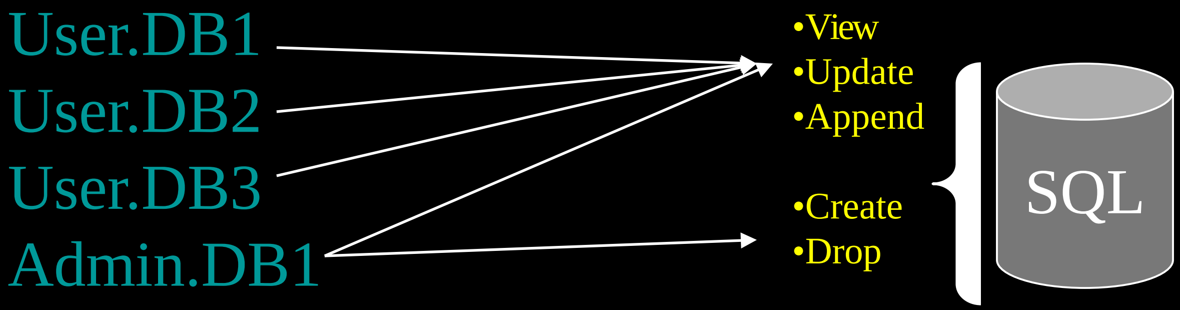
$$authorized\_users(r) = \{u \in USERS \mid r' \succeq r(u, r') \in UA\}$$

authorized permissions

Mapping of a role onto a set of permissions in the presence of a role hierarchy

ROLES set

PRMS set



User.DB1

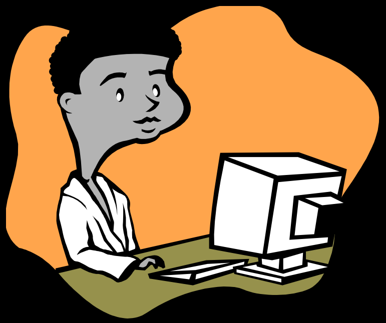
User.DB2

User.DB3

Admin.DB1

- View
- Update
- Append

- Create
- Drop

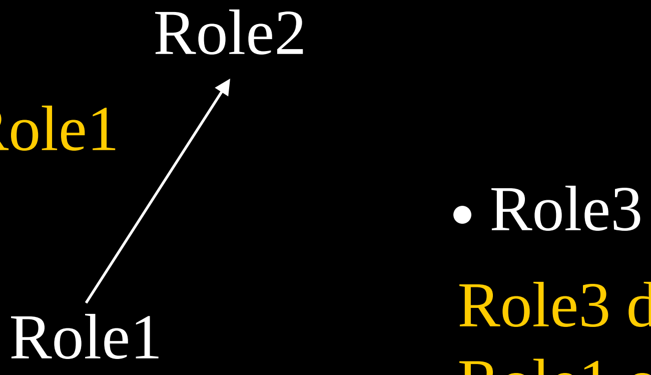


$authorized\_permissions(r : ROLES) \rightarrow 2^{PRMS}$

$authorized\_permissions(r) = \{ p \in PRMS \mid r' \succeq r, (p, r') \in PA$

Limited RH

A restriction on the immediate descendants of the general role hierarchy

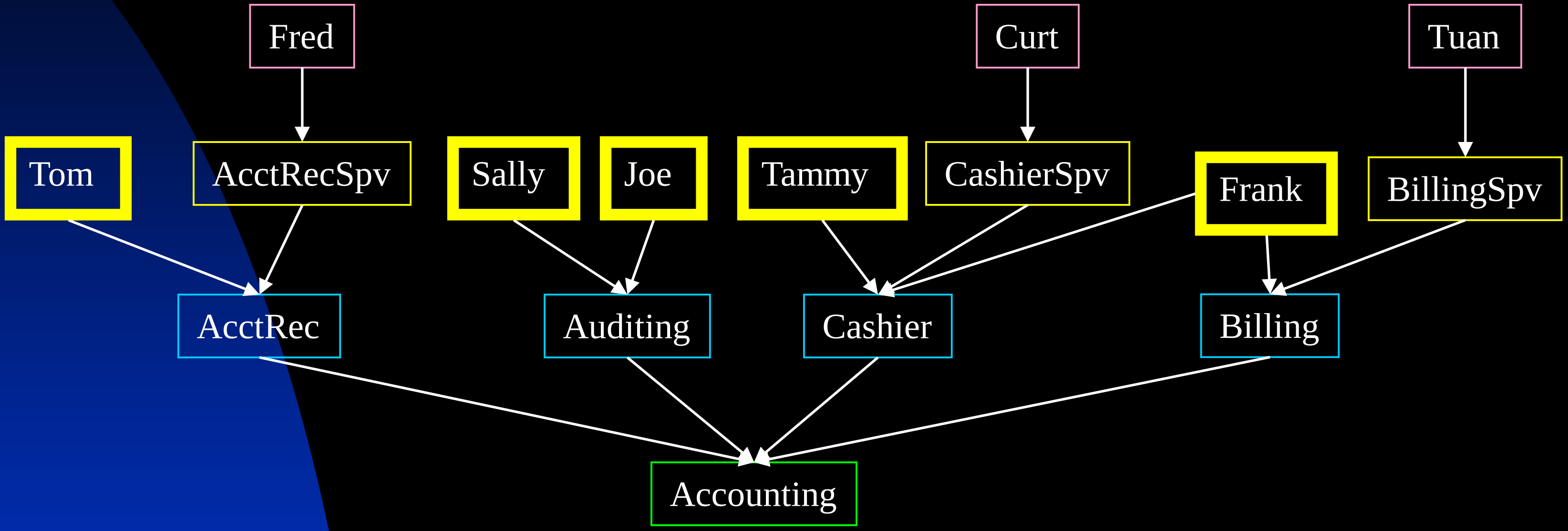


Role2 inherits from Role1

Role3 does not inherit from Role1 or Role2

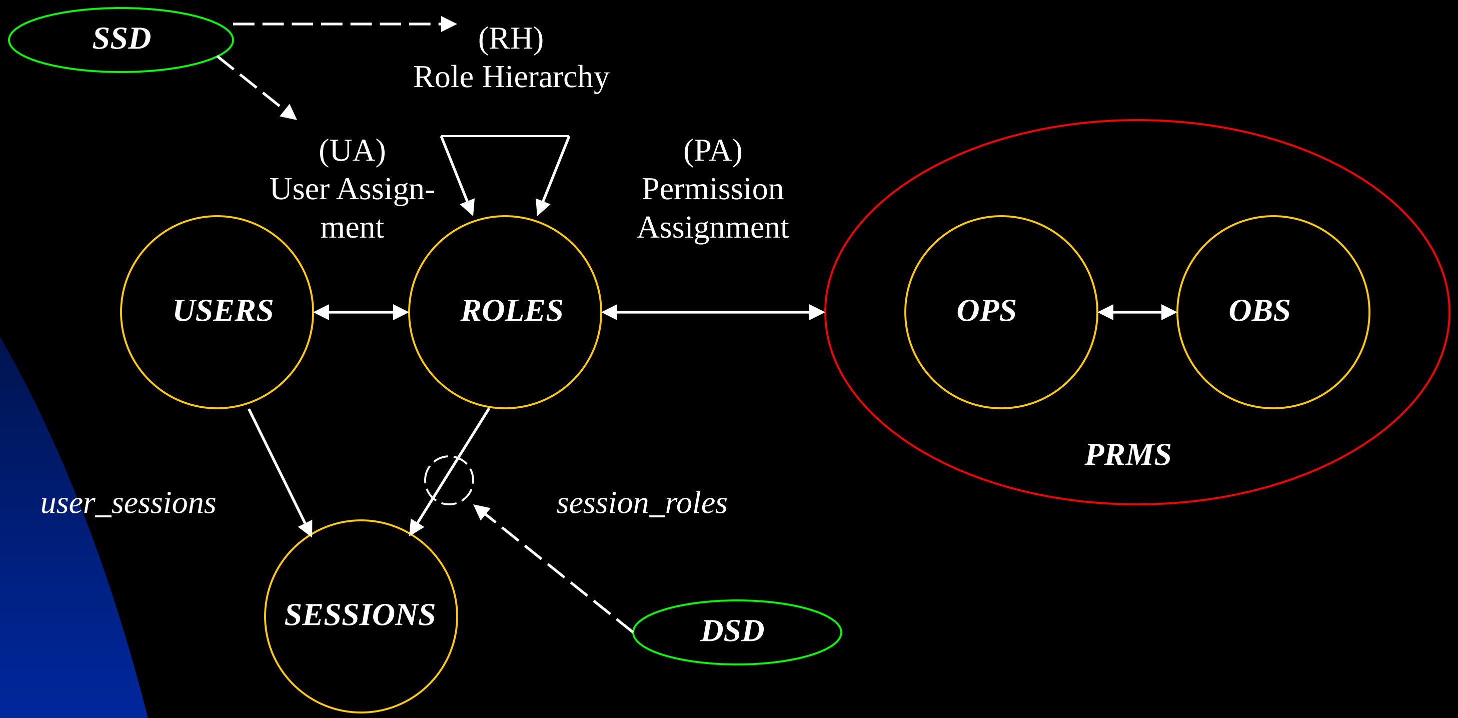
$\forall r, r_1, r_2 \in ROLES, r \succeq r_1 \wedge r \succeq r_2 \Rightarrow r_1 = r_2$

Limited RH (cont)



Accounting Role

Notice that Frank has two roles: Billing and Cashier
This requires the union of two distinct roles and prevents
Frank from being a node to others



Constrained RBAC

Separation of Duties

- Enforces conflict of interest policies employed to prevent users from exceeding a reasonable level of authority for their position.
- Ensures that failures of omission or commission within an organization can be caused only as a result of collusion among individuals.
- Two Types:
  - Static Separation of Duties (SSD)
  - Dynamic Separation of Duties (DSD)

SSD

- SSD places restrictions on the set of roles and in particular on their ability to form *UA* relations.
- No user is assigned to *n* or more roles from the same role set, where *n* or more roles conflict with each other.
- A user may be in one role, but not in another—mutually exclusive.
- Prevents a person from submitting and approving their own request.

$$SSD \subseteq (2^{ROLES} xN)$$

$$\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} assigned\_users(r) = \varnothing$$

SSD in Presence of RH

- A constraint on the authorized users of the roles that have an SSD relation.
- Based on the authorized users rather than assigned users.
- Ensures that inheritance does not undermine SSD policies.
- Reduce the number of potential permissions that can be made available to a user by placing constraints on the users that can be assigned to a set of roles.

$$\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} authorized\_users(r) = \varnothing$$

DSD

- Places constraints on the users that can be assigned to a set of roles, thereby reducing the number of potential prms that can be made available to a user.
- Constraints are across or within a user's session.
- No user may activate *n* or more roles from the roles set in each user session.
- *Timely Revocation of Trust* ensures that prms do not persist beyond the time that they are required for performance of duty.

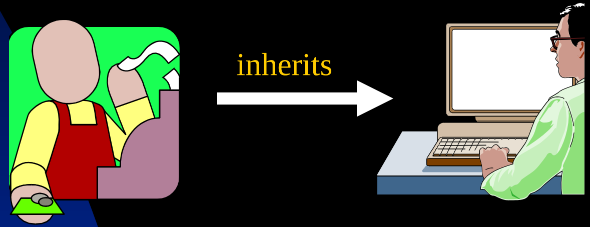
$$DSD \subseteq (2^{ROLESxN})$$

$$\forall rs \in 2^{ROLES}, n \in N, (rs, n) \in DSD \Rightarrow n \geq 2 \wedge |rs| \geq n, and$$

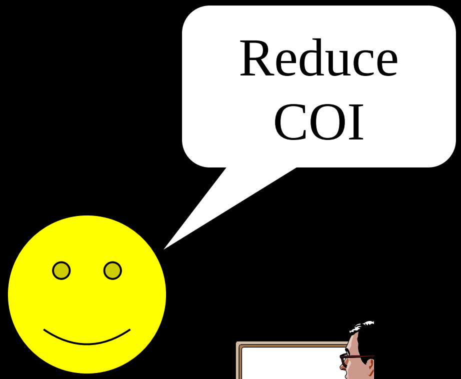
$$\forall s \in SESSIONS, \forall rs \in 2^{ROLES}, \forall role\_subset \in 2^{ROLES}, \forall n \in N, (rs, n) \in DSD, role\_subset \subseteq rs, role\_subset \subseteq session\_role(s) \Rightarrow |role\_subset| < n$$

DSD (cont)

Roles



Cashier Supervisor



Cashier

Accounting Error

Closes Cashier Role session

Close Cash Drawer

Opens Supv Role session

Open Cash Drawer

Supervisor

Correct Error

QUESTIONS…COMMENTS??