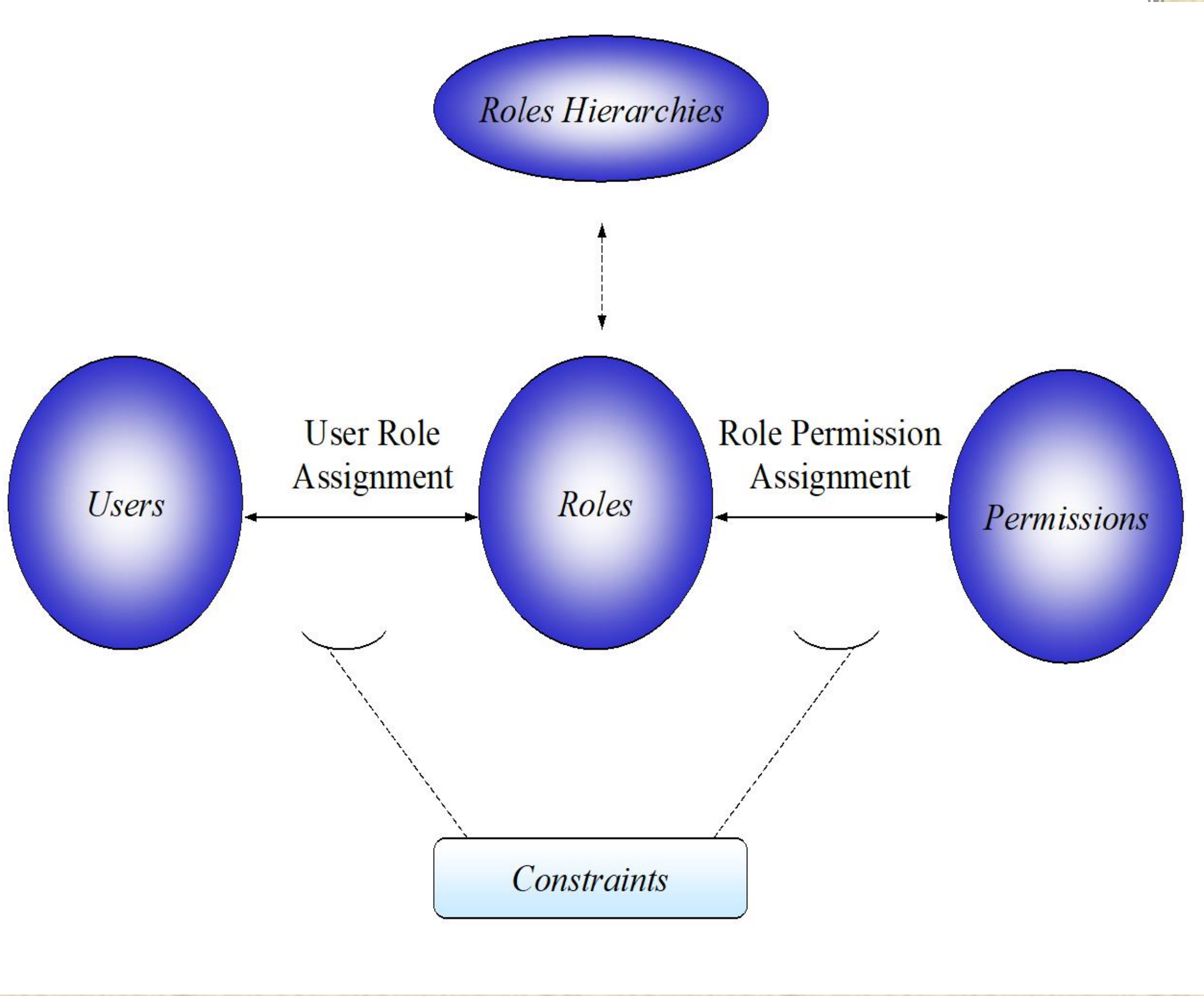
Role Based Access Control

RBAC

- Many organizations base access control decisions on "**the roles** that individual users take on as part of the organization".
- They prefer to <u>centrally control and maintain</u> access rights that reflect the organization's protection guidelines.
- With RBAC, <u>role-permission relationships can be predefined</u>, which makes it simple to assign users to the predefined roles.
- The combination of <u>users and permissions</u> tend to change over time, the permissions associated with a role are more stable.
- RBAC concept supports three well-known security principles:
  - Least privilege
  - Separation of duties
  - Data abstraction

Role Based Access Control (RBAC)

- Access control in organizations is based on "roles that individual users take on as part of the organization"
- A role is "is a collection of permissions"



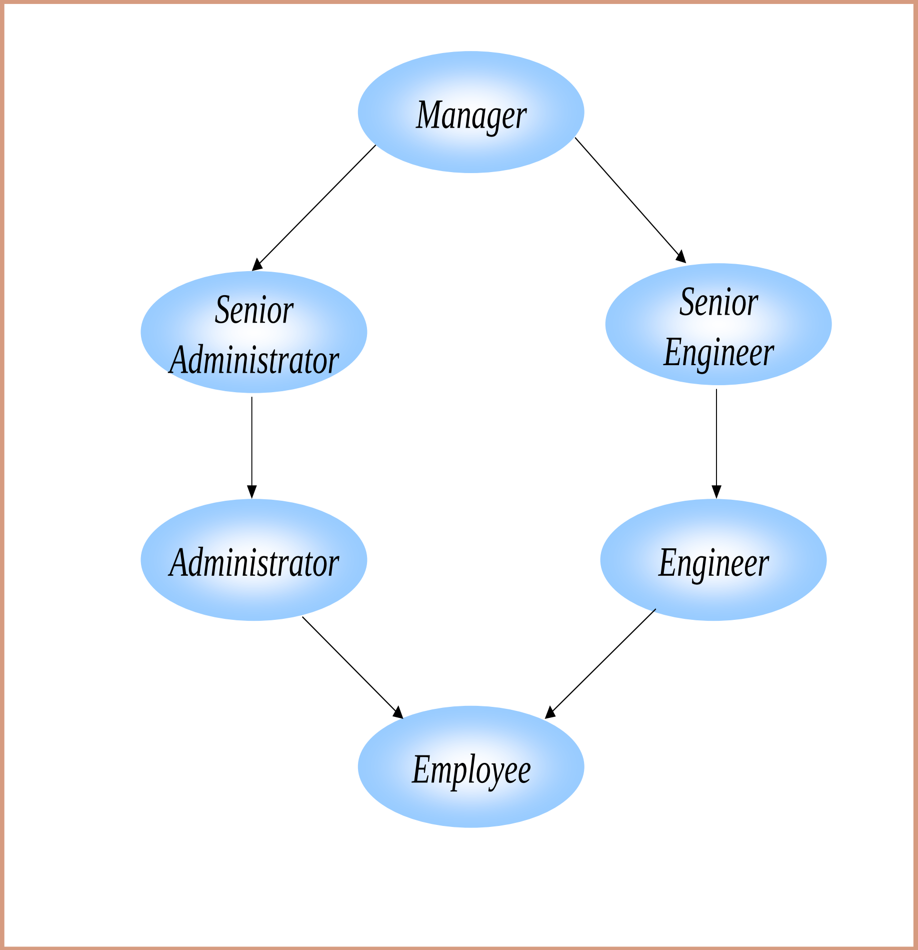
RBAC

- Access depends on role/function, not identity
  - Example: Allison is **bookkeeper** for Math Dept. She has access to financial records. If she leaves and Betty is hired as the new **bookkeeper**, Betty now has access to those records. The role of “bookkeeper” dictates access, not the identity of the individual.

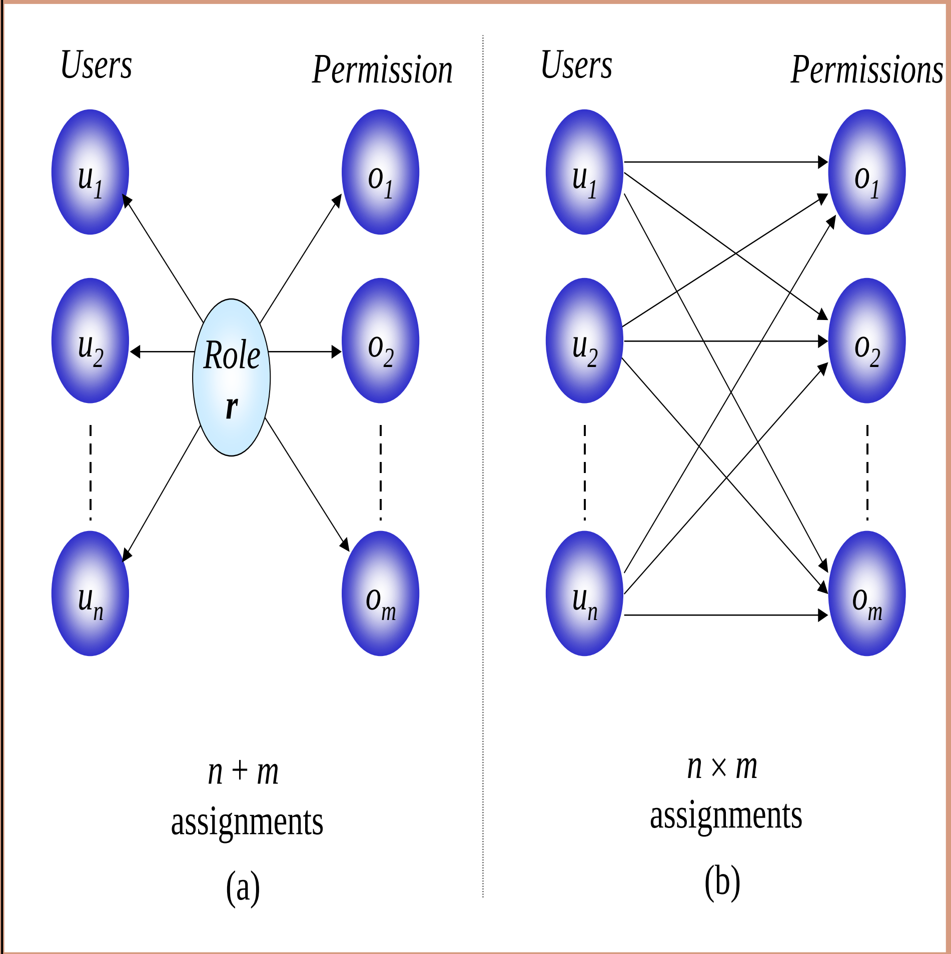
Advantages of RBAC

- Allows Efficient Security Management
  - Administrative roles, Role hierarchy
- Principle of least privilege allows minimizing damage
- Separation of Duties constraints to prevent fraud
- Allows grouping of objects
- Policy-neutral - Provides generality

RBAC



Manager
Senior Administrator
Senior Engineer
Administrator
Engineer
Employee

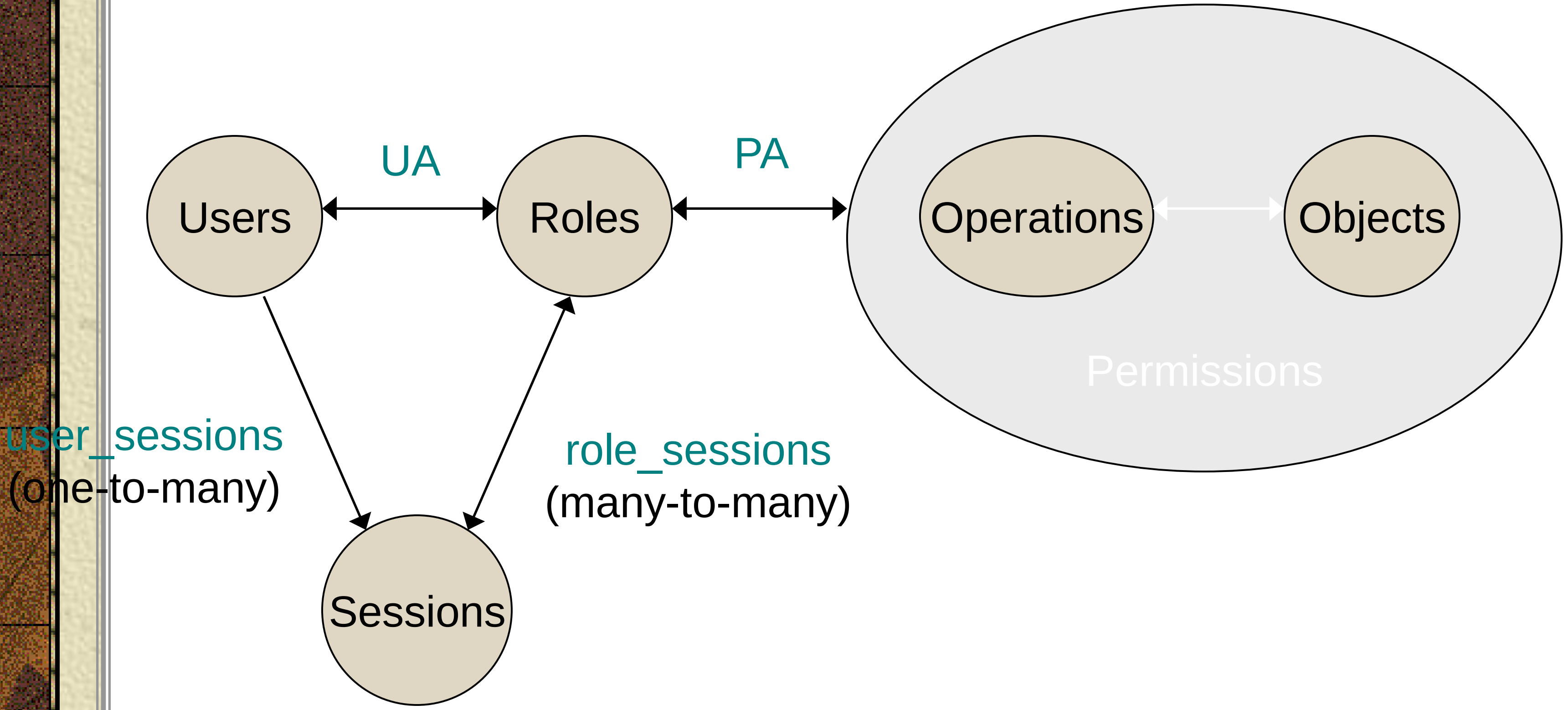


Users
Permission
u_1
u_2
u_n
Role
r
o_1
o_2
o_m
$n + m$ assignments
(a)
Users
Permissions
u_1
u_2
u_n
o_1
o_2
o_m
$n \times m$ assignments
(b)

Core RBAC (relations)

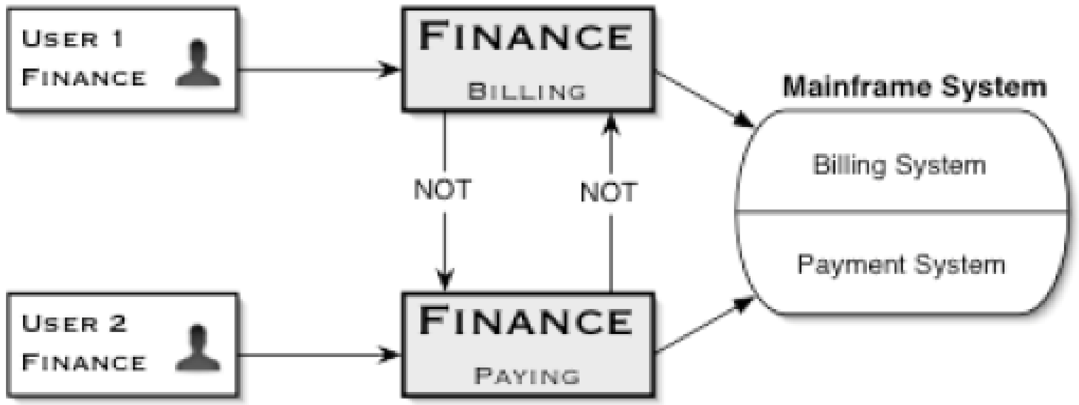
- Permissions = $2^{\text{Operations x Objects}}$
- UA $\subseteq$ Users x Roles
- PA $\subseteq$ Permissions x Roles
- *assigned_users*: Roles $\rightarrow 2^{\text{Users}}$
- *assigned_permissions*: Roles $\rightarrow 2^{\text{Permissions}}$
- *Op*(p): set of operations associated with permission p
- *Ob*(p): set of objects associated with permission p
- *user_sessions*: Users $\rightarrow 2^{\text{Sessions}}$
- *session_user*: Sessions $\rightarrow$ Users
- *session_roles*: Sessions $\rightarrow 2^{\text{Roles}}$
  - *session_roles*(s) = $\{r \mid (\text{session\_user}(s), r) \in \text{UA}\}$
- *avail_session_perms*: Sessions $\rightarrow 2^{\text{Permissions}}$

RBAC (NIST Standard)



Separation of Duties

- No user should be given enough privileges to misuse the system on their own.
- Statically: defining the conflicting roles
- Dynamically: Enforcing the control at access time


User 1
Finance
Finance
Billing
User 2
Finance
Finance
Paying
NOT
NOT
Mainframe System
Billing System
Payment System