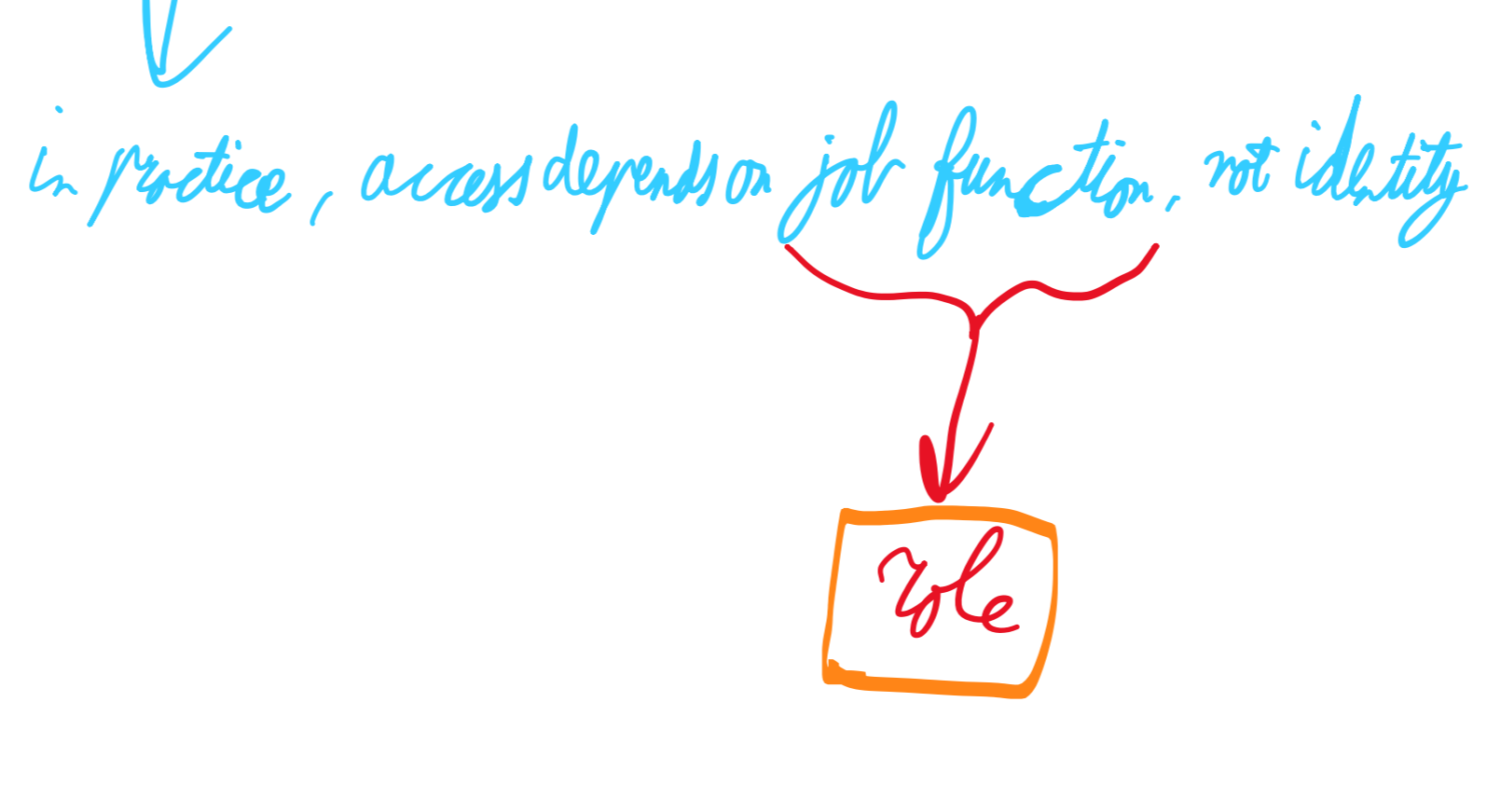


role-based access control:

- organization constantly change
- number of users & apps increases

concern: DAC { capability lists: store perms with users / subjects
access control lists: store perms with files / objects } → not scalable



RBA C: describe organizational AC policies based on job functions

- roles for job function
- permission for roles
- users have access to objects based on roles

RBA C principles:

- least privilege → give no more permission than necessary; calculate otherwise...
- separation of duties → users can abuse their position within an organization
↓
require more than 1 user to perform critical tasks

summary → base policies on duties and responsibilities of job functions

advantage: changing user's job function is possible by updating ^{assigned} roles instead of individual permissions

RBA C₀: main concepts

- user
- role
- permission
- session → roles permission "wise" at a given time

Exercise

	reservation	history	image data
Alice	read write	read	insert
Bob	read write	read	
Charlie		read	insert
David	read		

RBA C₁: **role hierarchy**

roles subsume others

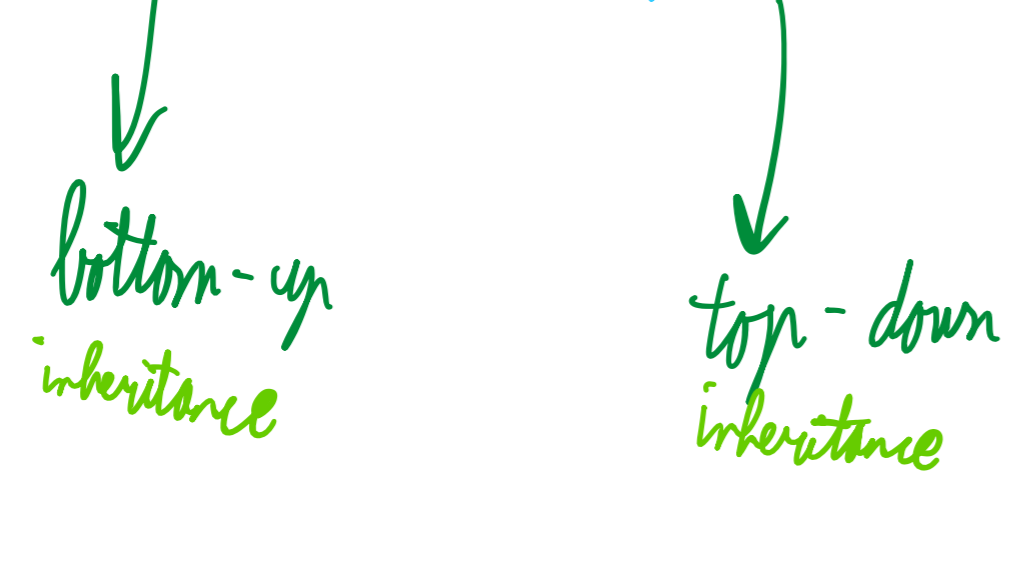
inheritance of roles → granting perm to role ⇒ grant perm to specialization as well

Exercise

	p ₁	p ₂	p ₃	p ₄	p ₅
u ₁	✓	✓			
u ₂	✓	✓	✓		
u ₃	✓	✓		✓	
u ₄	✓	✓	✓	✓	✓

in green: from own role
in blue: inherited permission

Sometimes, we use dominance instead of specialization



RBA C₂: constraints

- mutually exclusive roles for separation of duties → ensure two ^{same} users are involved in performing tasks → ensure certain perm combinations do not exist
↓
e.g. ordering materials & approving payments
- cardinality constraints → maximum number of users per role or roles per user
- prerequisite roles

static separation of duty: restricted permissions that can be assigned
dynamic separation of duty: restricted permissions that can be exercised by a user

$$SSOD(\{perm_1, perm_2, \dots, perm_n\}, m)$$

at least m users are required to perform all n permissions

SSOD can be too restrictive; e.g. you might want only that someone cannot approve their own orders

- dynamic separation of duty (dsod)
- object-based dsod → may not act on object they have already acted on
- history-based dsod

Static mutually exclusive roles: restrictions on role assignment

$SMR(r, s, n) \rightarrow$ user cannot have more than n roles from set r's

dynamic mutually exclusive roles: restriction on role activation

$DMR(r, s, n) \rightarrow$ user cannot activate more than n roles from set r's

cardinality constraints → at most / at least / exactly k users should belong to role

prerequisite constraints → user can only be assigned to a role if they are already assigned to another role