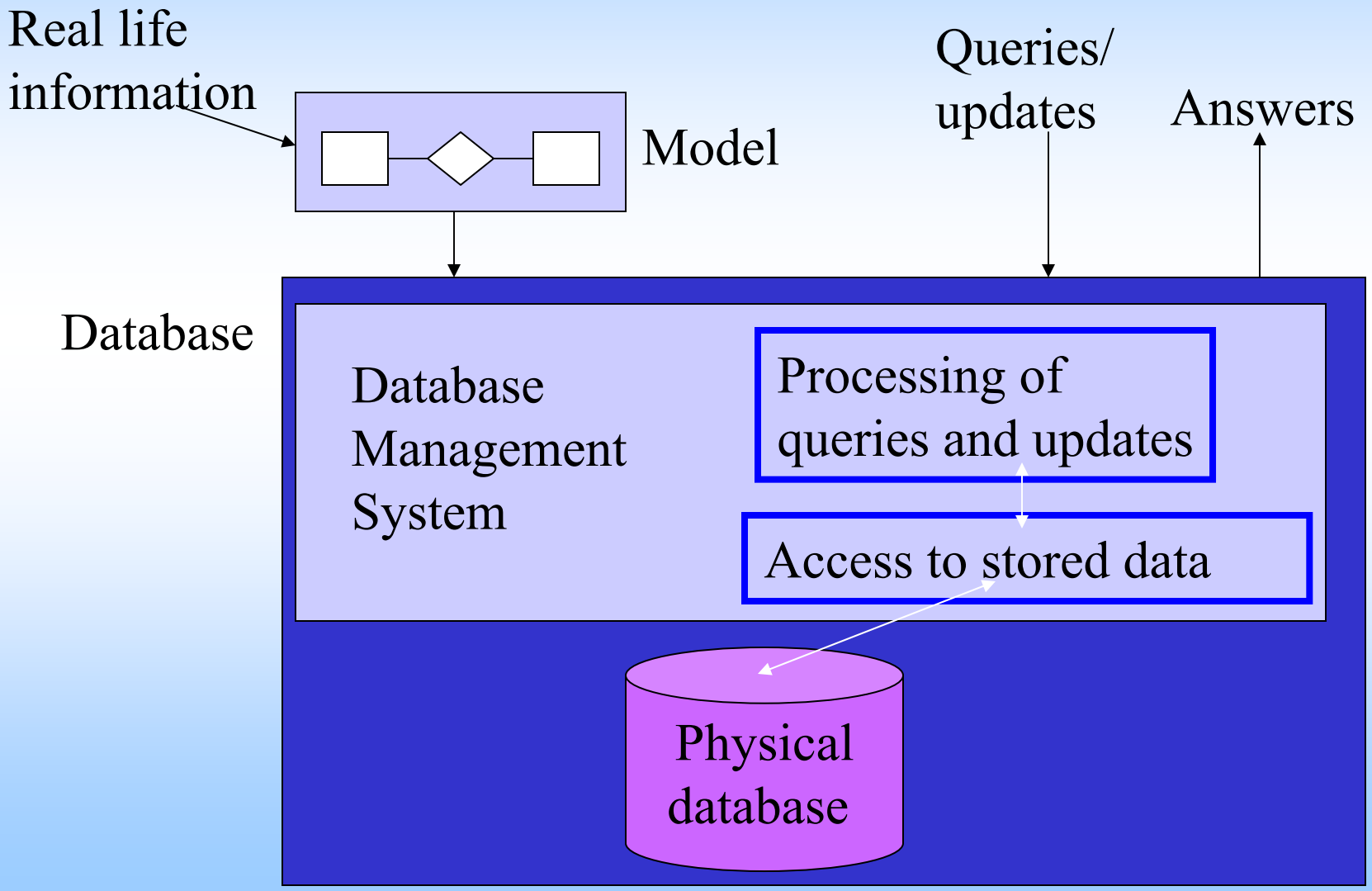
Object-oriented databases

Databases


Real life information
Model
Queries/ updates
Answers
Database
Database Management System
Processing of queries and updates
Access to stored data
Physical database

Database

- DataBase Management System (DBMS): a collection of programs that allows a user to create and maintain a databank

- database system = physical database + DBMS

Some Issues

- What information is stored?
- How is the information stored? (high level)
- How is the information accessed? (user level)

Some Issues

- How can different types of users be authorized to access different pieces of information?

DEFINITION	Homo sapiens adrenergic, beta-1-, receptor
ACCESSION	NM_000684
SOURCE ORGANISM	human
REFERENCE	1
AUTHORS	Frielle, Collins, Daniel, Caron, Lefkowitz, Kobilka
TITLE	Cloning of the cDNA for the human beta 1-adrenergic receptor
REFERENCE	2
AUTHORS	Frielle, Kobilka, Lefkowitz, Caron
TITLE	Human beta 1- and beta 2-adrenergic receptors: structurally and functionally related receptors derived from distinct genes

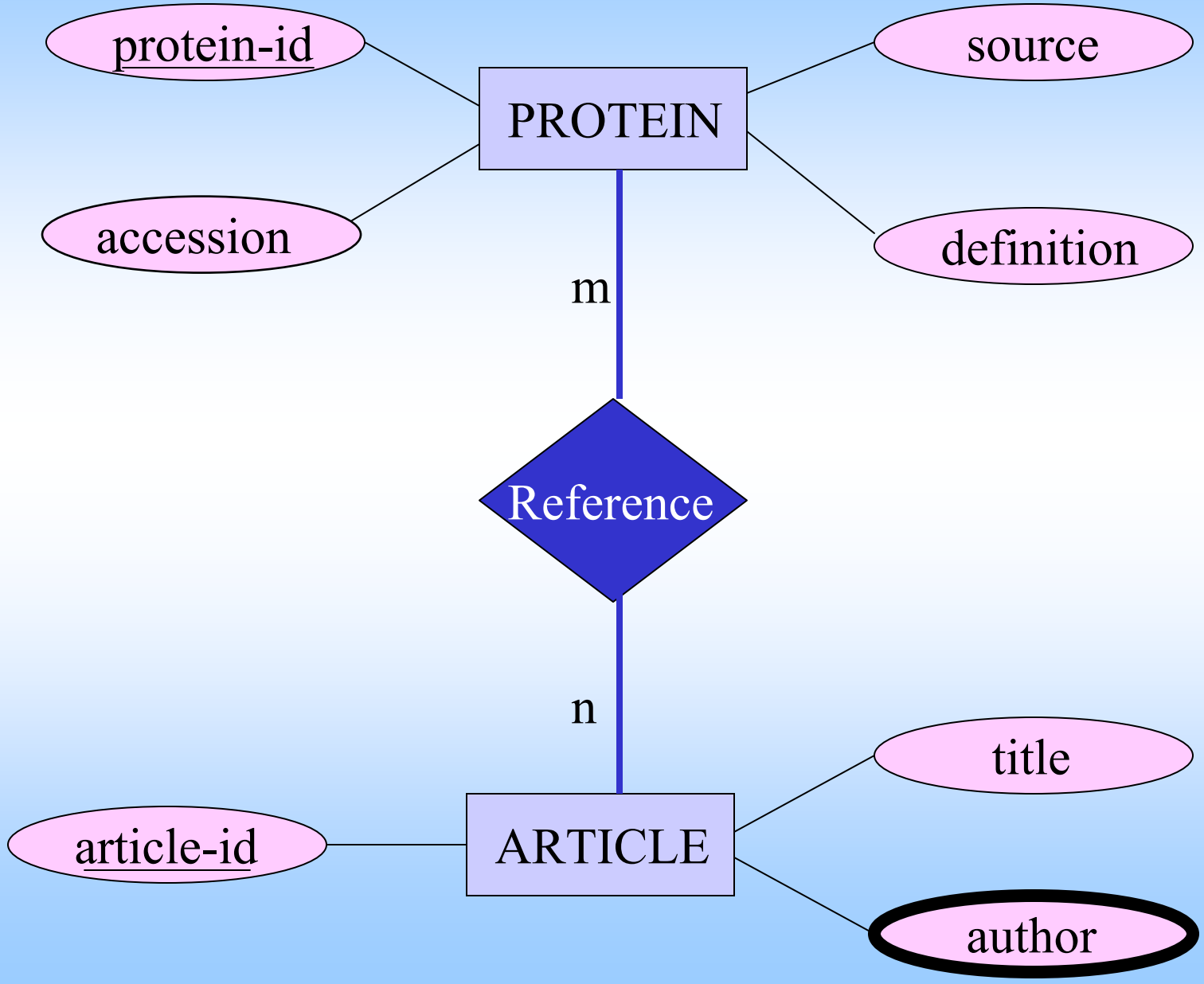
What information is stored?

- Model of reality

  - Extended Entity-Relationship diagrams (EER)

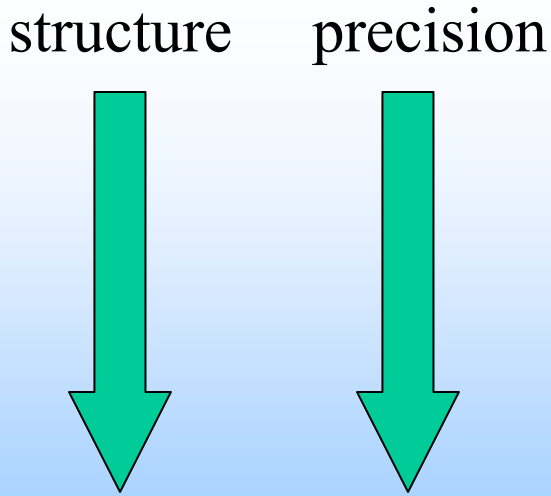
  - Unified Modeling Language (UML)

Entity-relationship



How is the information stored? (high level)
How is the information accessed? (user level)

- Text (IR)
- Semi-structured data
- Data models (DB)
- Rules + Facts (KB)



Databases

- Relational databases:
  - model: tables + relational algebra
  - query language (SQL)
- Object-oriented databases:
  - model: persistent objects, messages, encapsulation, inheritance
  - query language (e.g. OQL)

Relational databases

PROTEIN

PROTEIN-ID	ACCESSION	DEFINITION	SOURCE
1	NM_000684	Homo sapiens adrenergic, beta-1-, receptor	human

REFERENCE

PROTEIN-ID	ARTICLE-ID
1	1
1	2

ARTICLE

ARTICLE-ID	AUTHOR	TITLE
1	Frielle	Cloning of the cDNA for the human ….
1	Collins	Cloning of the cDNA for the human ….
1	Daniel	Cloning of the cDNA for the human ….
1	Caron	Cloning of the cDNA for the human ….
1	Lefkowitz	Cloning of the cDNA for the human ….
1	Kobilka	Cloning of the cDNA for the human ….
2	Frielle	Human beta 1- and beta 2-adrenergic receptors
2	Kobilka	Human beta 1- and beta 2-adrenergic receptors
2	Lefkowitz	Human beta 1- and beta 2-adrenergic receptors
2	Caron	Human beta 1- and beta 2-adrenergic receptors

Relational databases

PROTEIN

PROTEIN-ID	ACCESSION	DEFINITION	SOURCE
1	NM_000684	Homo sapiens adrenergic, beta-1-, receptor	human

REFERENCE

PROTEIN-ID	ARTICLE-ID
1	1
1	2

ARTICLE-AUTHOR

ARTICLE-ID	AUTHOR
1	Frielle
1	Collins
1	Daniel
1	Caron
1	Lefkowitz
1	Kobilka
2	Frielle
2	Kobilka
2	Lefkowitz
2	Caron

ARTICLE-TITLE

ARTICLE-ID	TITLE
1	Cloning of the cDNA for the human beta 1-adrenergic receptor
2	Human beta 1- and beta 2-adrenergic receptors: structurally and functionally related receptors derived from distinct genes

SQL

select source
from protein
where accession = NM_000684;

PROTEIN

PROTEIN-ID	ACCESSION	DEFINITION	SOURCE
1	NM_000684	Homo sapiens adrenergic, beta-1-, receptor	human

SQL

```
select title
from protein, article-title, reference
where protein.accession = NM_000684
  and protein.protein-id
                = reference.protein-id
  and reference.article-id
                = article-title.article-id;
```

REFERENCE

PROTEIN-ID	ARTICLE-ID
1	1
1	2

PROTEIN

PROTEIN-ID	ACCESSION	DEFINITION	SOURCE
1	NM_000684	Homo sapiens adrenergic, beta-1-, receptor	human

ARTICLE-TITLE

ARTICLE-ID	TITLE
1	Cloning of the …
2	Human beta 1- …

From relational to object model

- CASE
- CAD
- office automation
- multimedia applications

Object-Oriented Databases (OODB)

- World is modeled using objects.
- An object has a state (value) and a behavior (operations).
- Persistent objects - permanent storage (sometimes transient objects are allowed)

Object

- An object has an object identifier (OID) that is not visible to the user.
- OID cannot be changed.
- object versus value

  (a value has no OID)
- object structure can be arbitrarily complex (atom, tuple, set, bag, list, array)

Example - object state

- o1(id1, tuple,
  <accession: NM_000684,
  source : human,
  definition: 'Homo sapiens adrenergic …',
  reference: o2>)
- o2(id2, set, {o3,o4})

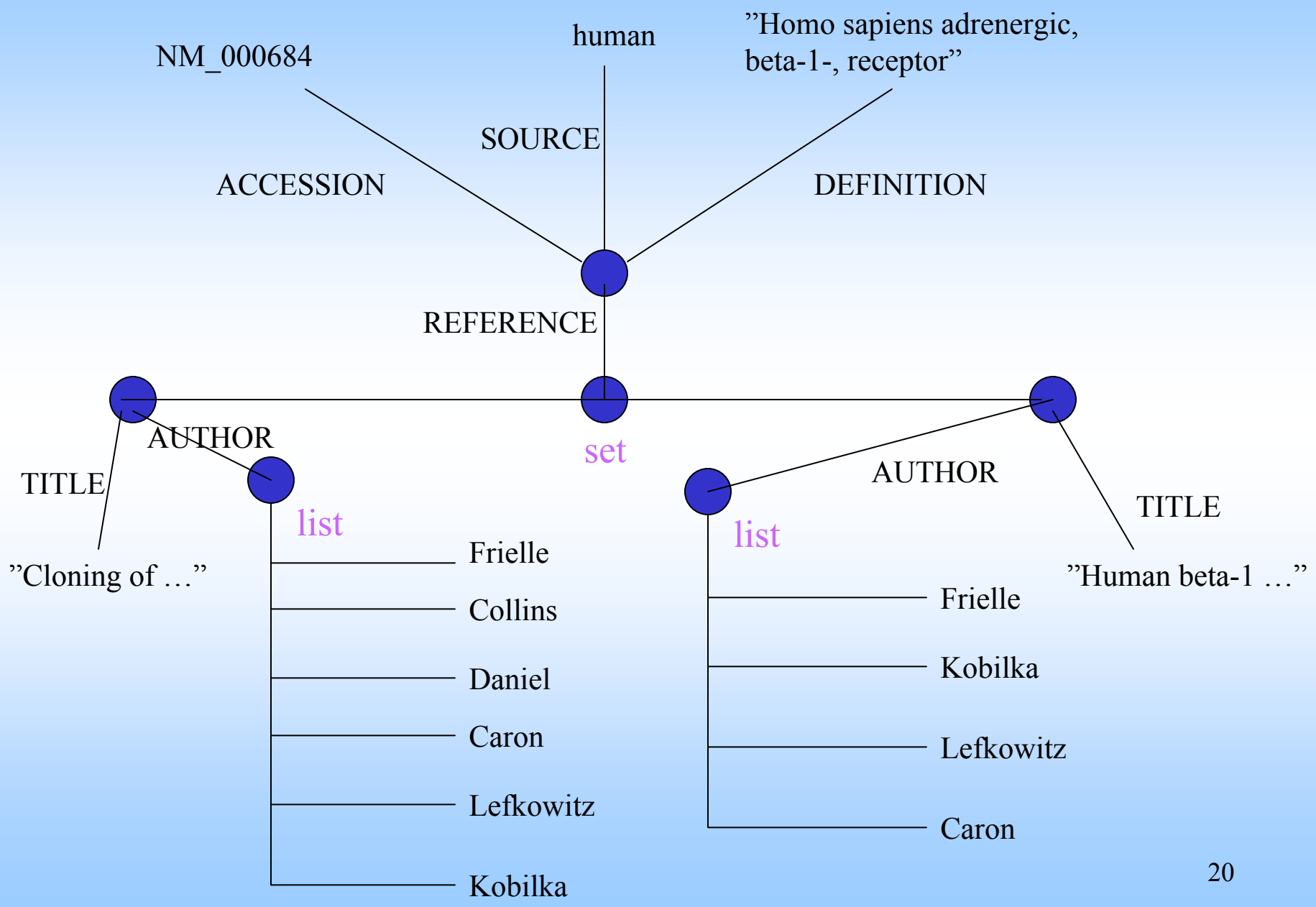
Remark: These examples do not use a standard syntax

Example - object state

- o3(id3, tuple,

  <title: `Cloning of …’, author: o5 >)
- o4(id4, tuple,

  <title: `Human beta-1 …’, author: o6 >)
- o5(id5, list, [Frielle, Collins, Daniel, Caron, Lefkowitz, Kobilka])
- o6(id6, list, [Frielle, Kobilka , Lefkowitz, Caron])


NM_000684
human
”Homo sapiens adrenergic, beta-1-, receptor”
ACCESSION
SOURCE
DEFINITION
REFERENCE
set
AUTHOR
TITLE
list
”Cloning of …”
Frielle
Collins
Daniel
Caron
Lefkowitz
Kobilka
AUTHOR
TITLE
list
”Human beta-1 …”
Frielle
Kobilka
Lefkowitz
Caron

Classes

define class protein
type tuple (
    accession: string;
    source : string;
    definition: string;
    reference: set(article););
operations
   create-protein(string,string,string,set(article)): protein;
   get-accession: string;
   get-source: string;
   get-definition: string;
   get-references: set(article);
   add-reference(article): void;
end protein;

Classes

```
define class article
type tuple (
    title: string;
    author: list(string); );
operations
   create-article(string, list(string)): article;
   get-title string;
   get-authors: list(string);
   print-article-info string;
end article;
```

Example program

program

variables: article1, article2, protein1;

begin

article1 := create-article('Cloning….', list(Frielle, Collins, Daniel, Caron, Lefkowitz, Kobilka));

protein1 := create-protein(NM_000684, human,'Homo sapiens adrenergic …', set(article1));

article2 := create-article(' Human beta-1….', list(Frielle, Kobilka , Lefkowitz, Caron]);

protein1.add-reference(article2);

end;

Operations

- encapsulation: operation = interface + body
  - interface: how is the operation called?

    What is the result of the operation?

    > visible to user, used in programs
  - body: how is the operation implemented?

    > invisible for user
- program is based on message passing

Inheritance

- journal-article **subtype-of** article:
  journal-name journal-volume page-numbers

journal-article inherits all attributes and operations from article and has in addition also journal-name, journal-volume and page-numbers as attributes

- human-protein **subtype-of** protein (source = ’human’)

Composite objects

- Composite objects are complex objects that are conceptualized as a hierarchy of objects such that the hierarchical links represent the part-of relation.
- Dependent parts: existence of part depends on existence of the whole

  →special semantics for delete operation
- Exclusive and shared parts: exclusive part can belong to only one whole at the time; a shared part can belong to different wholes at the same time.

Operator overloading

- The same operator name can be used for different implementations
- example:

  print-article-info for article prints information on title and author.

  print-article-info for journal-article prints information on title, author and also on the journal's name, volume and page number..

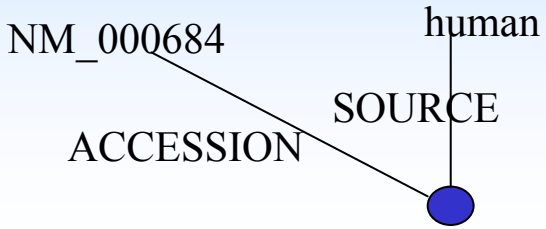
Query language OQL

- select … from … where

  select distinct … from … where
- iterator variables
- path expressions
- struct

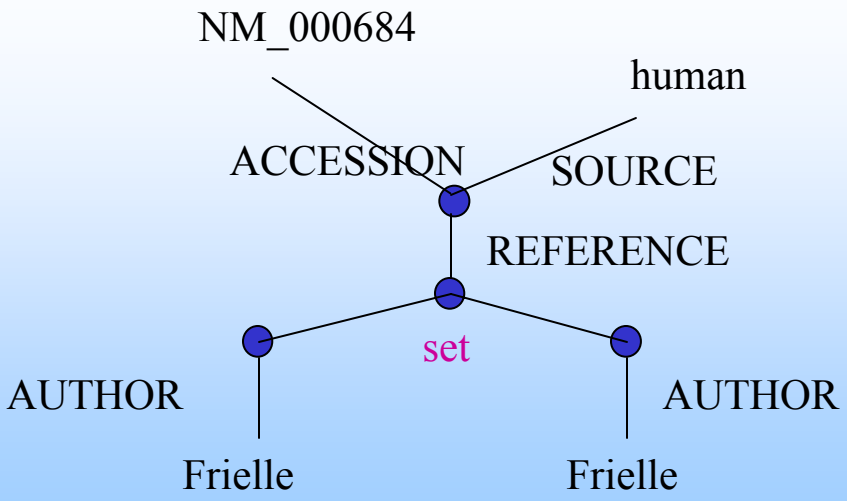
Queries

select o.source
from o **in** protein
where o.accession = NM_000684;



Queries

select struct (accession: o.accession, source: o.source**)**
from o **in** protein
where Frielle **in**
(select a.author
from a **in** o.reference**);**



Query language OQL

OQL also allows:

- views
- aggregation
- special operations for list and array
  (first, last, nth)
- order-by
- group-by

Third-Generation DB Manifesto

- Objects and Rules

- rich type system
- inheritance
- methods and encapsulation
- unique identifiers
- rules (triggers, constraints)

Third-Generation DB Manifesto

- DBMS functionality

- access through non-procedural high-level language

- specify collections intensionally and extensionally

- updatable views

- no performance indicators in the model

Third-Generation DB Manifesto

- Open systems

- accessible via several high-level languages
- persistency
- SQL-like language
- queries and answers are the lowest level of communication between client and server

OODBS Manifesto

Thou shalt ...

- complex objects
- object identity
- encapsulation
- types and classes
- inheritance
- overriding, overloading, late binding

OODBS Manifesto

- computational completeness
- extensibility
- persistence
- secondary storage management
- concurrency
- recovery
- query facility

OODBS Manifesto

Optional

- multiple inheritance
- distribution
- long and nested transactions
- versions

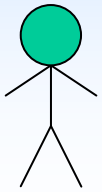
Thou shalt question the golden rules.

Authorization

Real world example

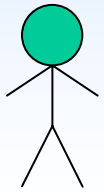
Roles: Manager, researcher, PR person, employee

Manager



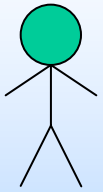
Can read/write any document

Researcher



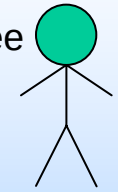
Can read public PR-material and read/write research material

PR



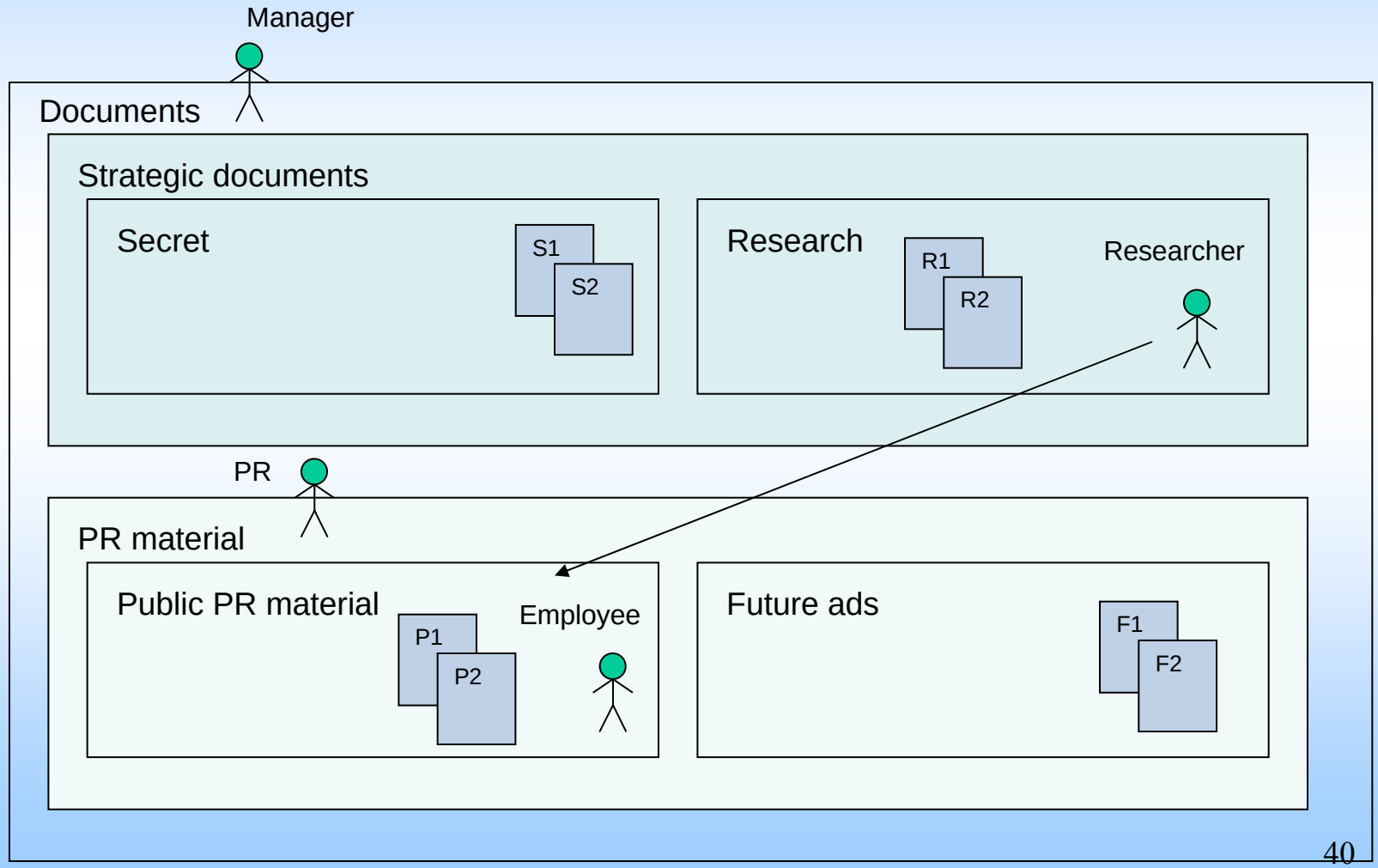
Can read/write public PR material and non-public PR material (work in progress)

Employee



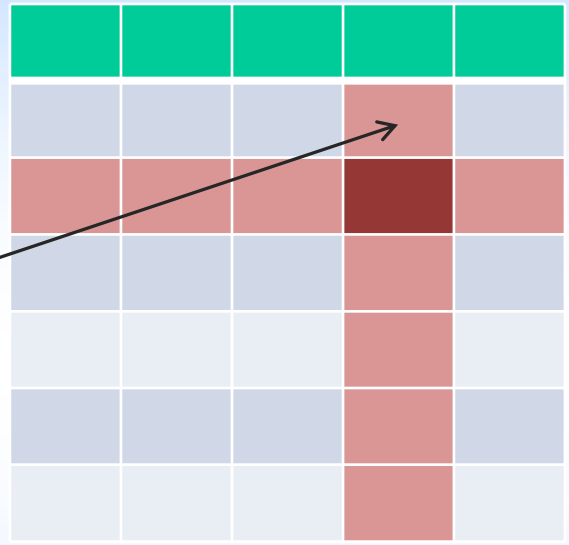
Can only read public PR material

Real world example



Authorization model in relational DBs

- Coarse-grained
- Units of authorization:
  - Relation (record)
  - Attribute (field)



Authorization

- Extensions for the OO model

  (class, inheritance, composite objects, (versions))
- Authorization mechanisms

  (implicit/explicit, strong/weak, positive/negative)

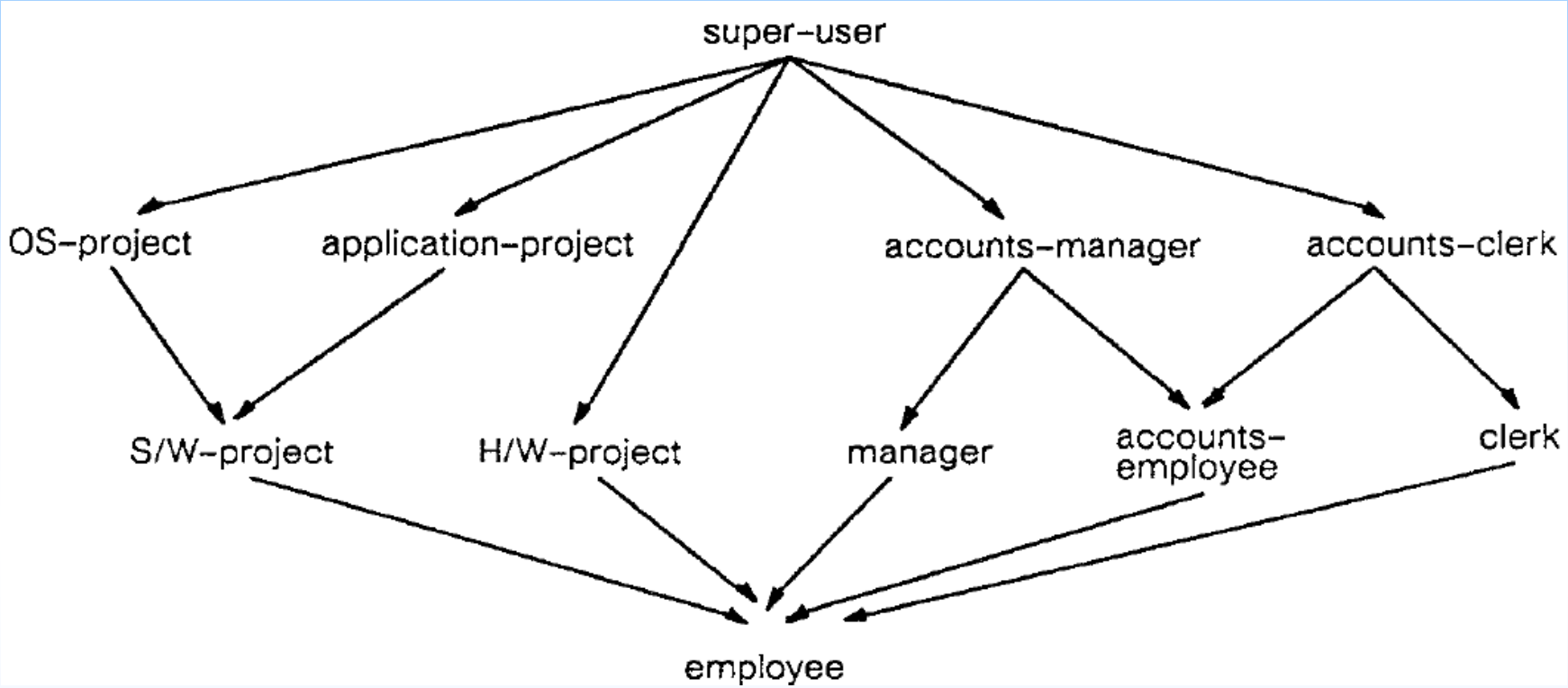
Basic authorization concepts

- (s,o,a) $s \in S$

  $o \in O$

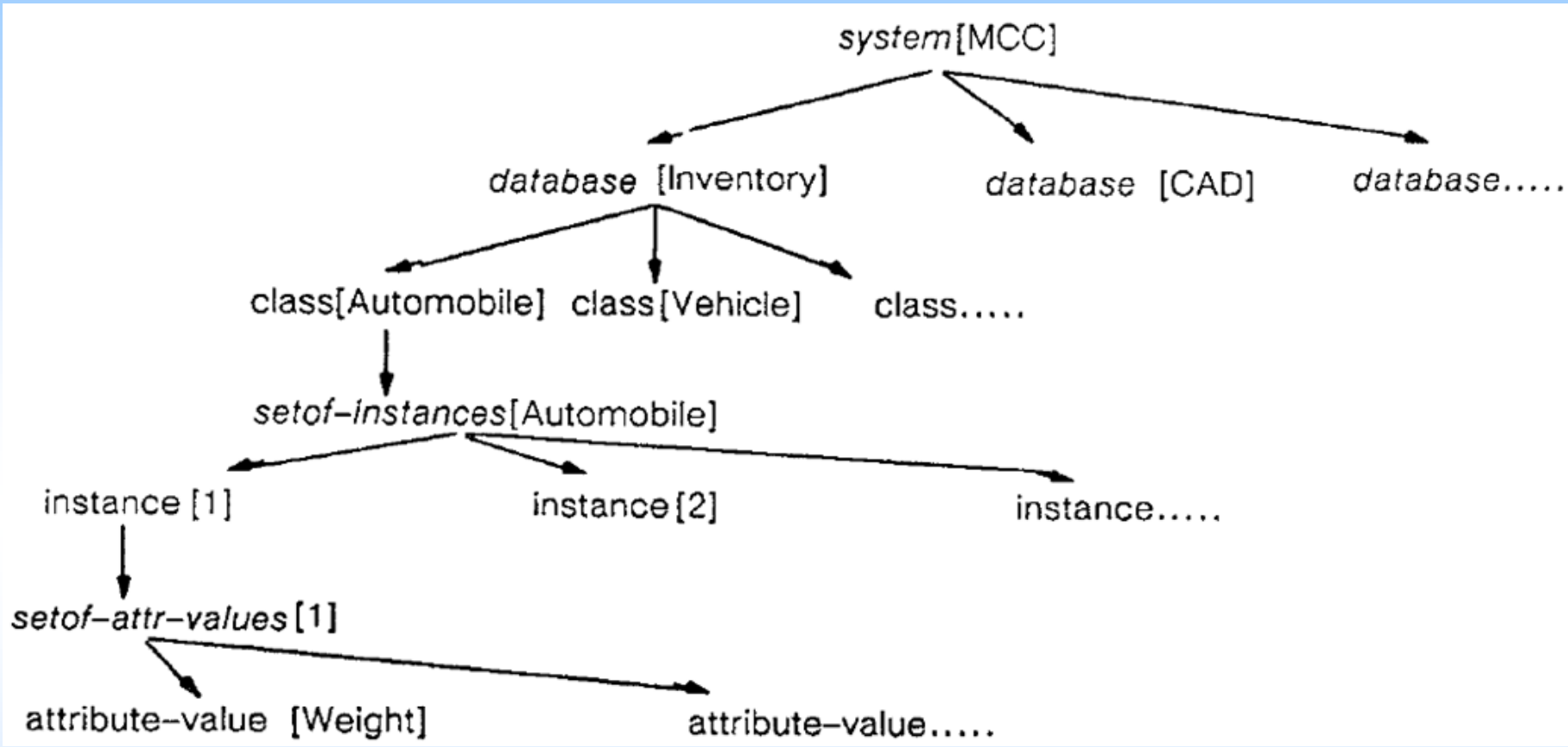
  $a \in A$

  $F: S \times O \times A \rightarrow$(True, false)
- Subject (a user or group of users)
- Authorization object (single object, group of objects, entire database)
- Authorization type (read, update, create, …)



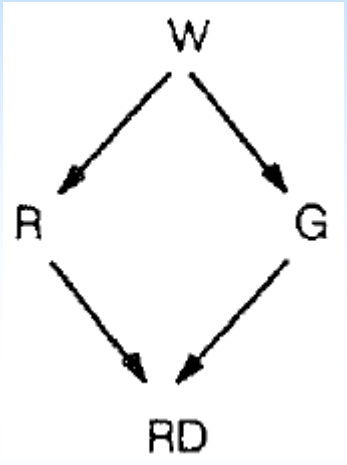
Role lattice

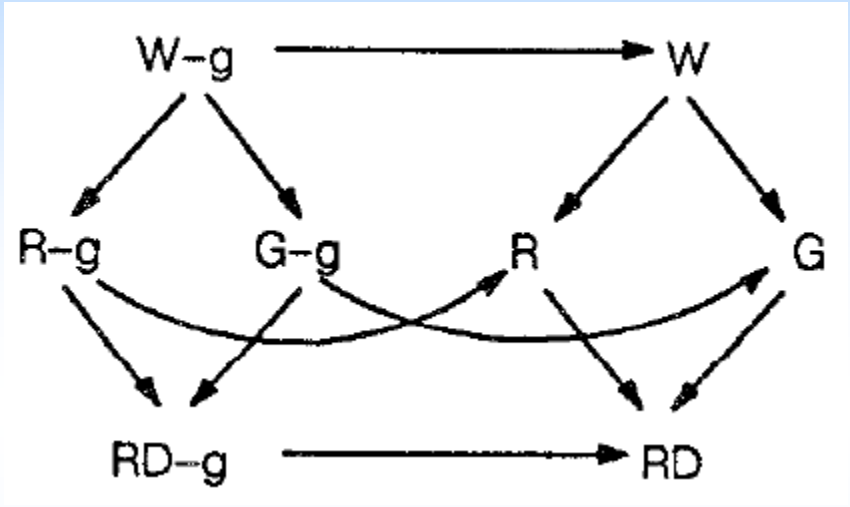
Rabitti, F., Bertino, E., Kim, W., Woelk, D. 1991



Object lattice

Rabitti, F., Bertino, E., Kim, W., Woelk, D. 1991





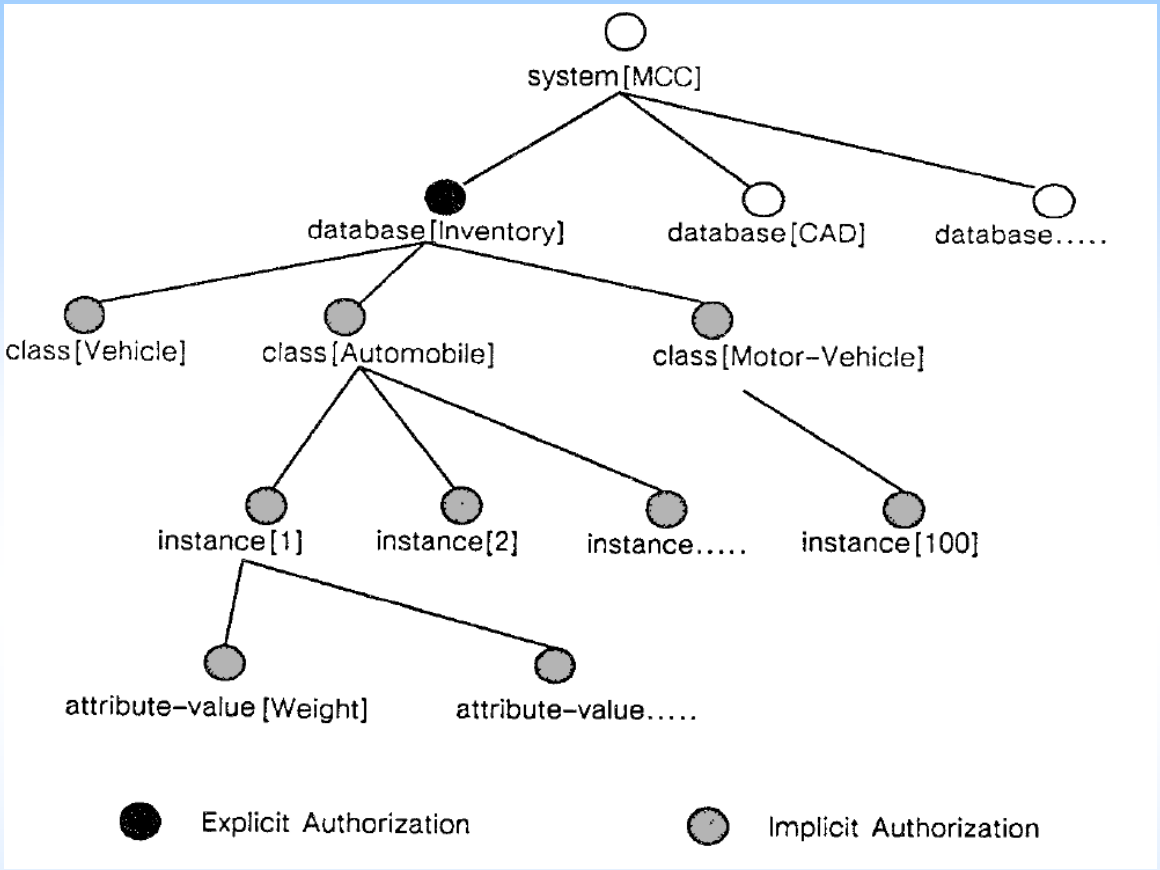
grants

Type lattice

Rabitti, F., Bertino, E., Kim, W., Woelk, D. 1991

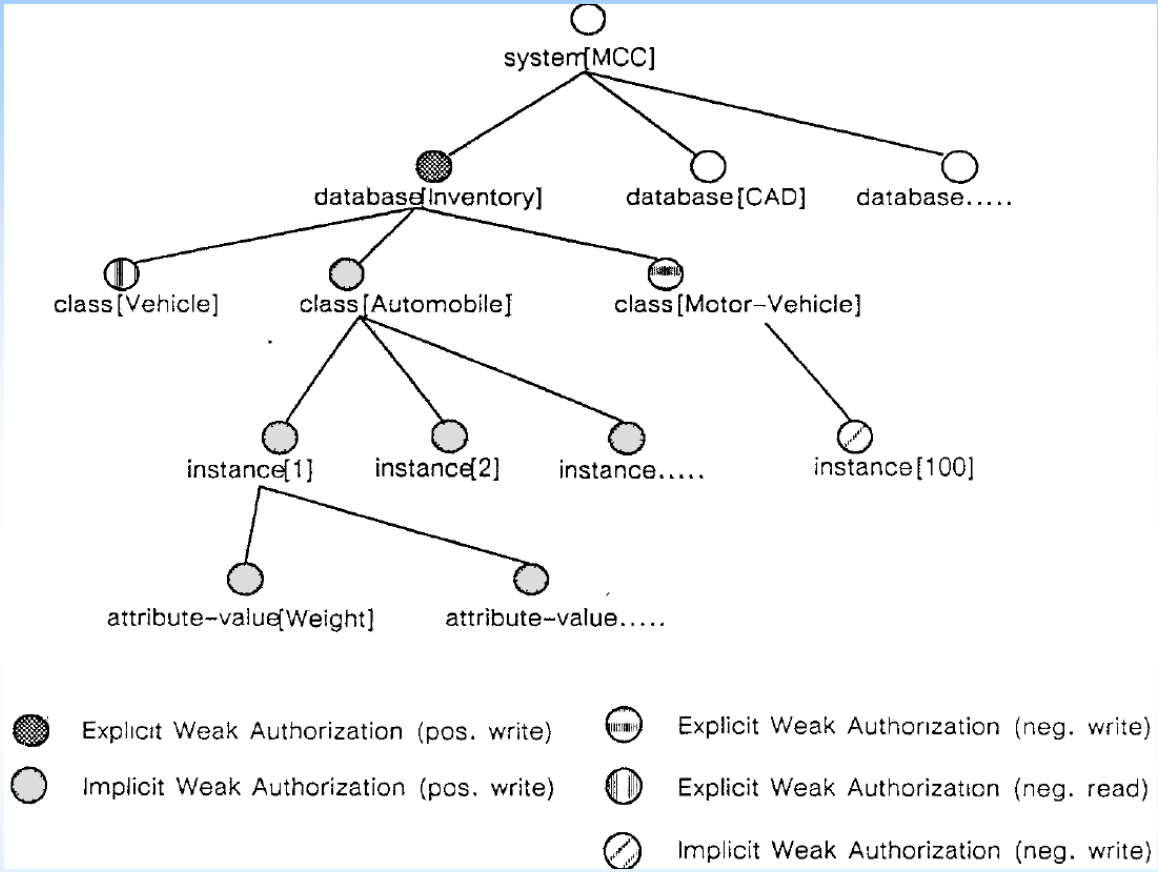
Authorization notions

- Explicit – Implicit
  - Explicit: set <s,o,a> triplets
  - implicit.: derive <s,o,a> triplets
- Positive – Negative
  - Positive: allowed to access
  - Negative: not allowed to access
- Strong - Weak
  - Strong: cannot be overridden
  - Weak: can be overridden



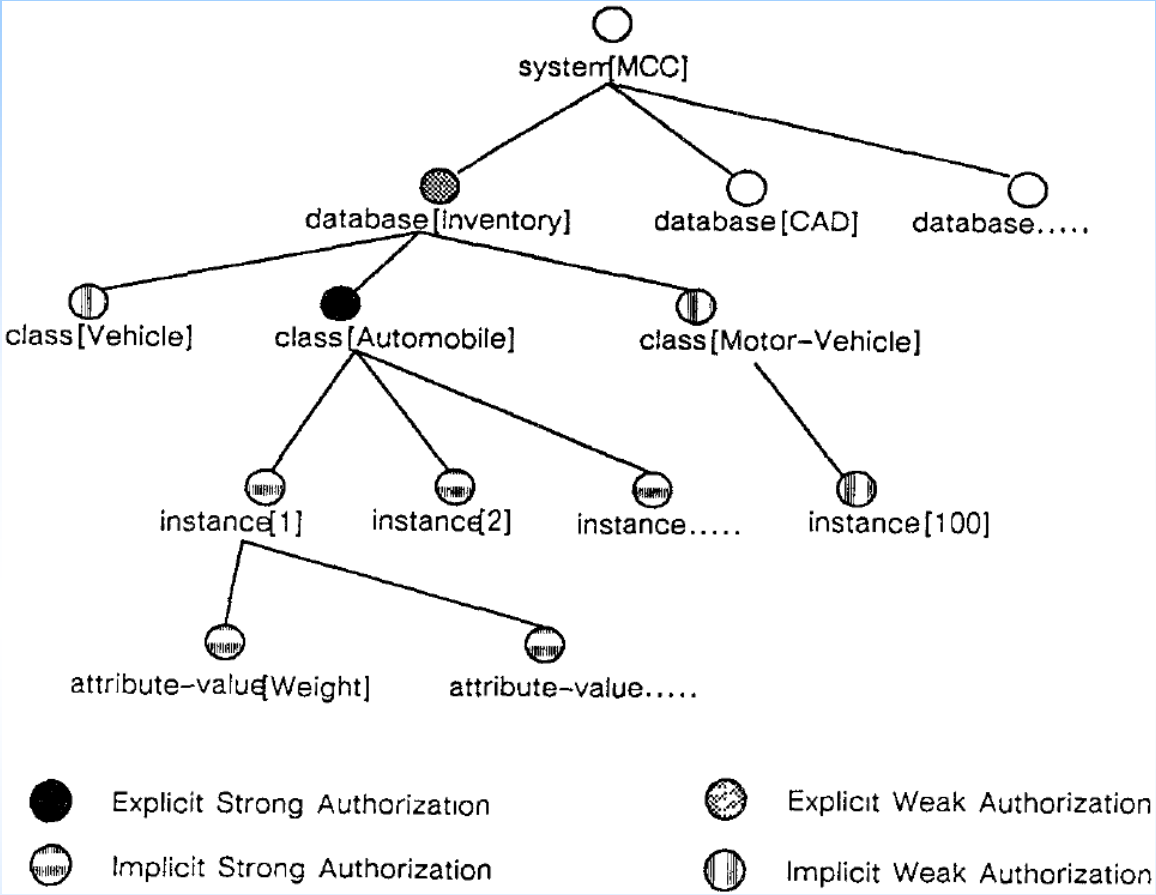
Explcit and implicit authorization

Rabitti, F., Bertino, E., Kim, W., Woelk, D. 1991



Weak authorization

Rabitti, F., Bertino, E., Kim, W., Woelk, D. 1991

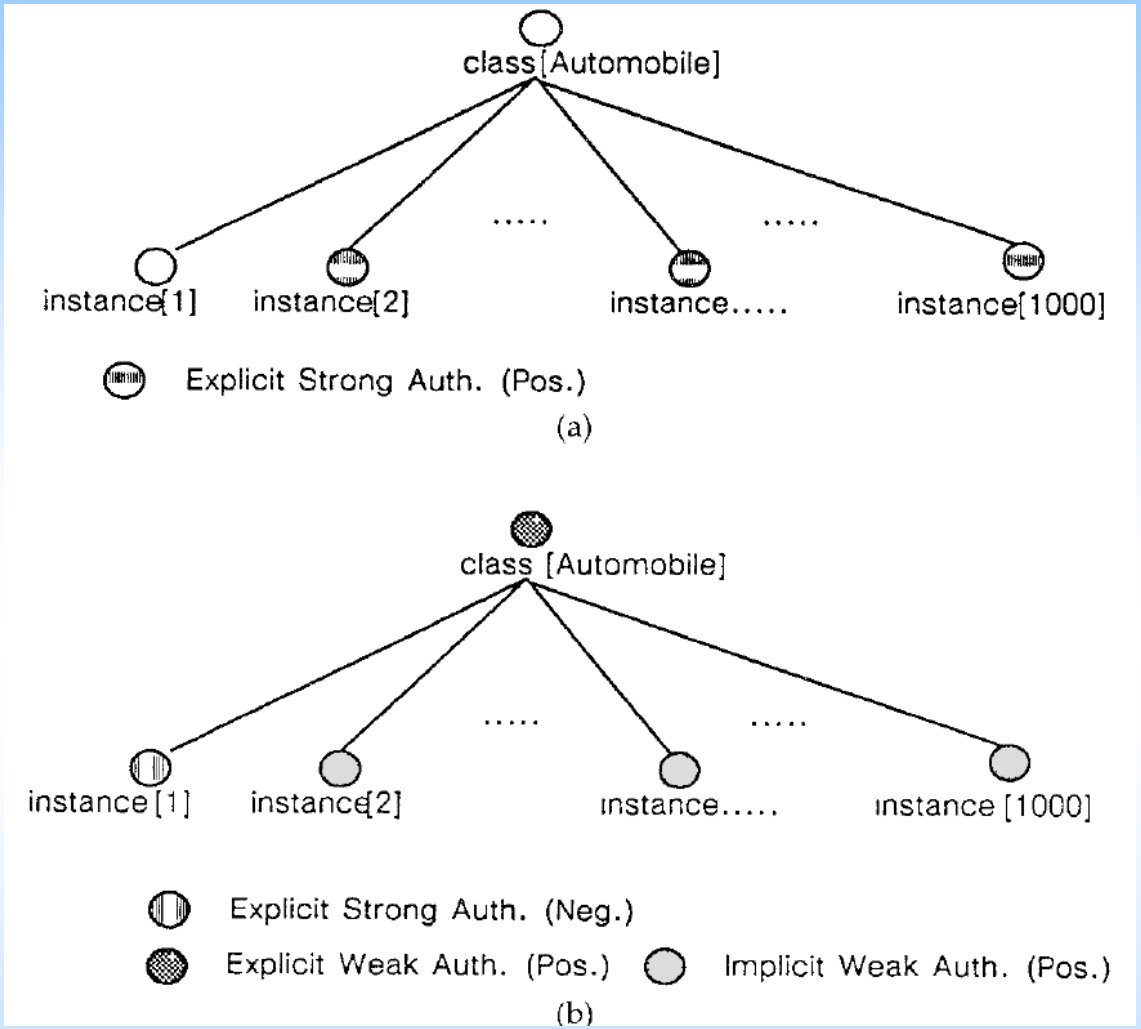


Strong authorization

Rabitti, F., Bertino, E., Kim, W., Woelk, D. 1991

Implicit authorization

- Pros:
  - No need to store all combinations
  - No need to set all combinations
- Cons:
  - Sometimes hard to grasp why a specific authorization is determined as it is
  - Conflicts
  - Computational overhead

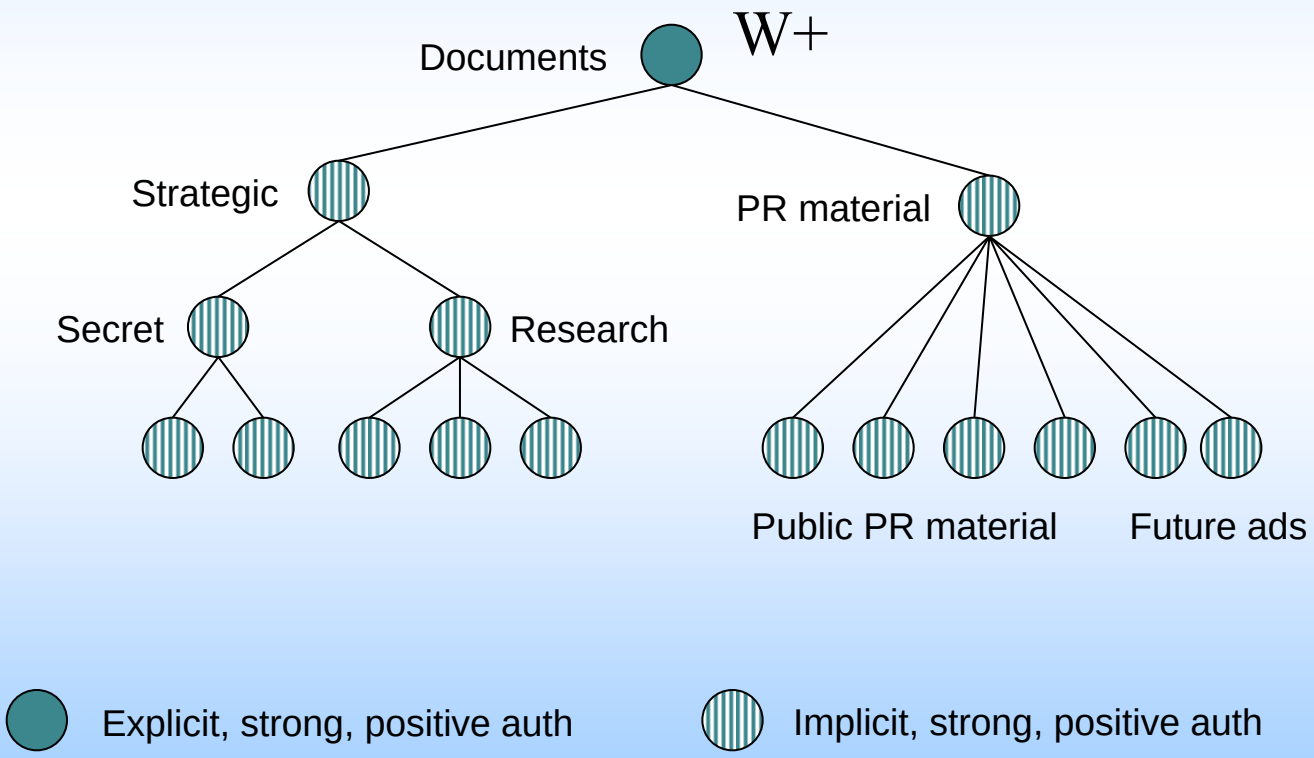


Positive/negative authorization

Rabitti, F., Bertino, E., Kim, W., Woelk, D. 1991

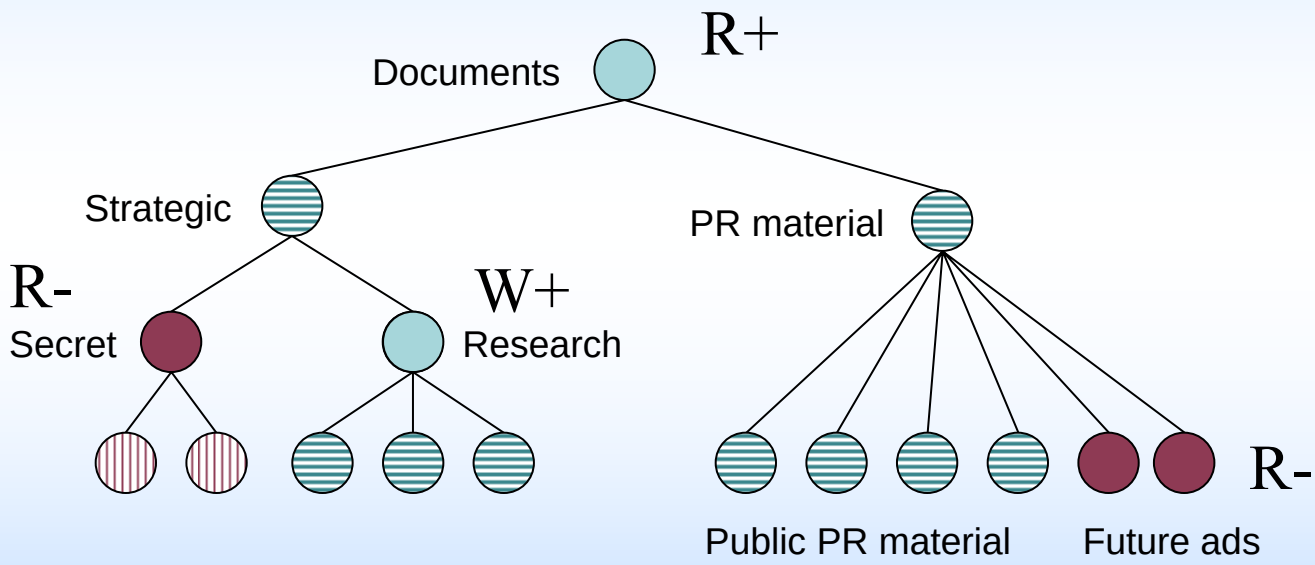
Applied real world example

Manager: Can read/write any document



Applied real world example

Researcher: Can read public PR-material and read/write research material



Explicit, strong, negative auth

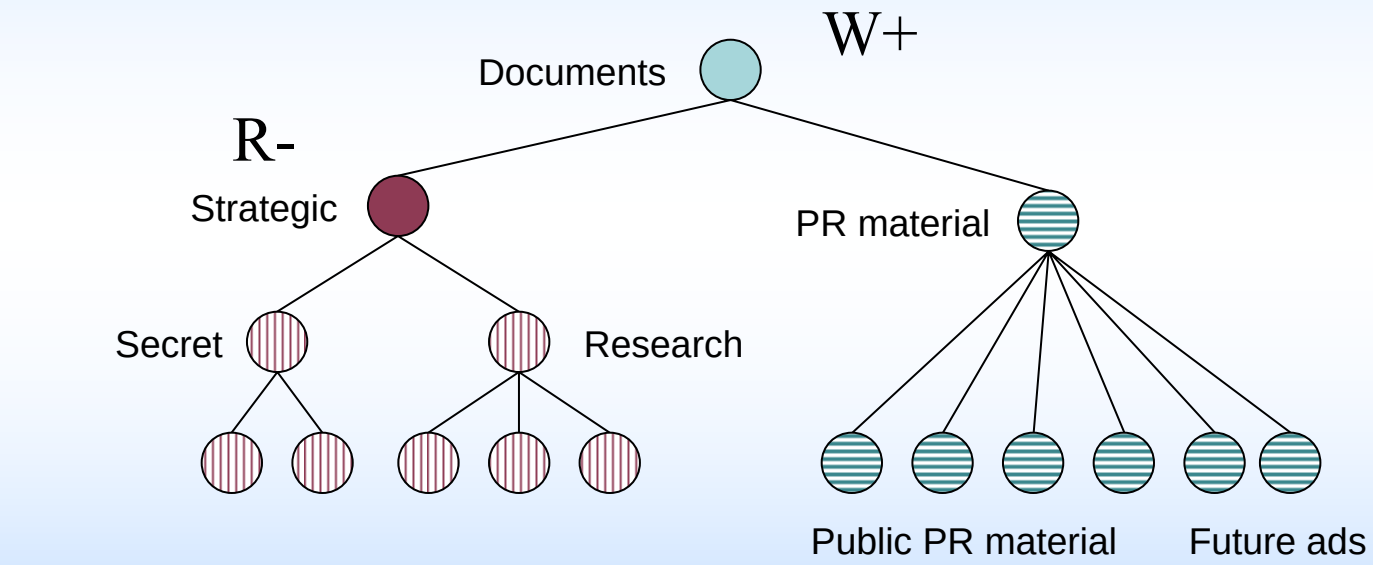
Implicit, strong, negative auth

Explicit, weak, positive auth

Implicit, weak, positive auth

Applied real world example

PR person: Can read/write public PR material and non-public PR material



Explicit, strong, negative auth

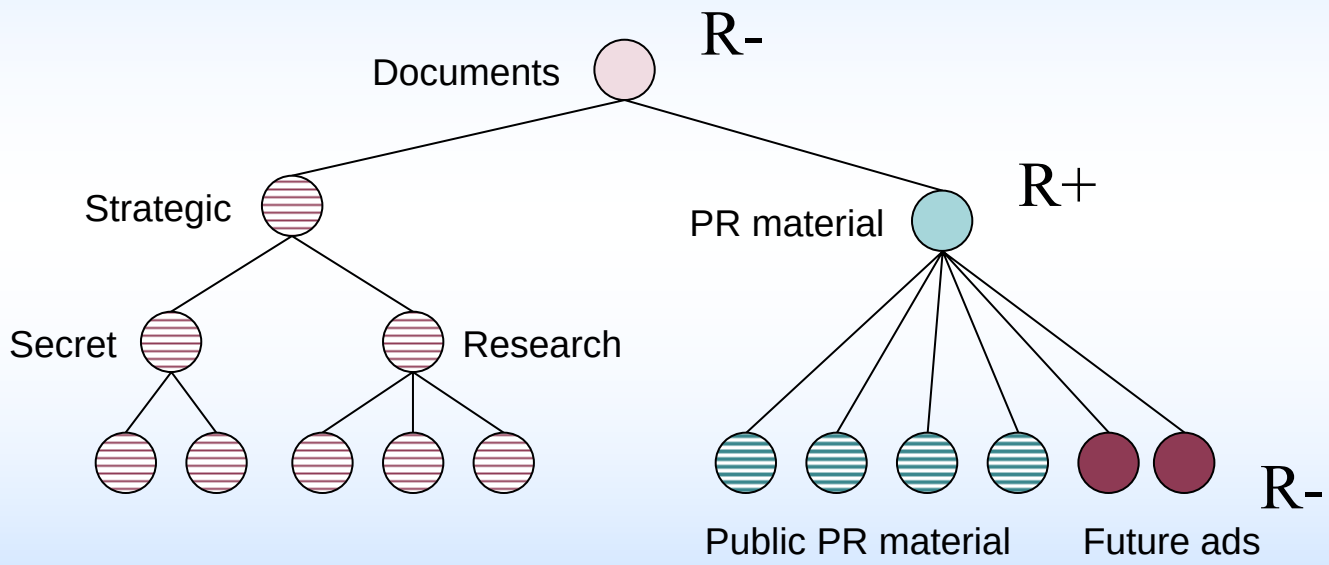
Implicit, strong, negative auth

Explicit, weak, positive auth

Implicit, weak, positive auth

Applied real world example

Employee: Can only read public PR material



Explicit, strong, negative auth

Explicit, weak, negative auth

Explicit, weak, positive auth

Implicit, weak, positive auth

Implicit, weak, negative auth

Possible topics for ’plus’-grade

- Read articles about a specific OO database and summarize.
- Test run a specific OO database and write report.
- Read articles or experiment with OO system regarding a particular issue. Examples for issues are: authorization, versioning, schema evolution, query optimization, duplicate detection in OODBs, storage management and indexing in OODBs.