P/FDM

Object-oriented database implemented mainly in Prolog

Based on the Functional Data Model (FDM)

- entities (classes; hierarchies)
- functions

  attributes and relationships

  stored and derived values

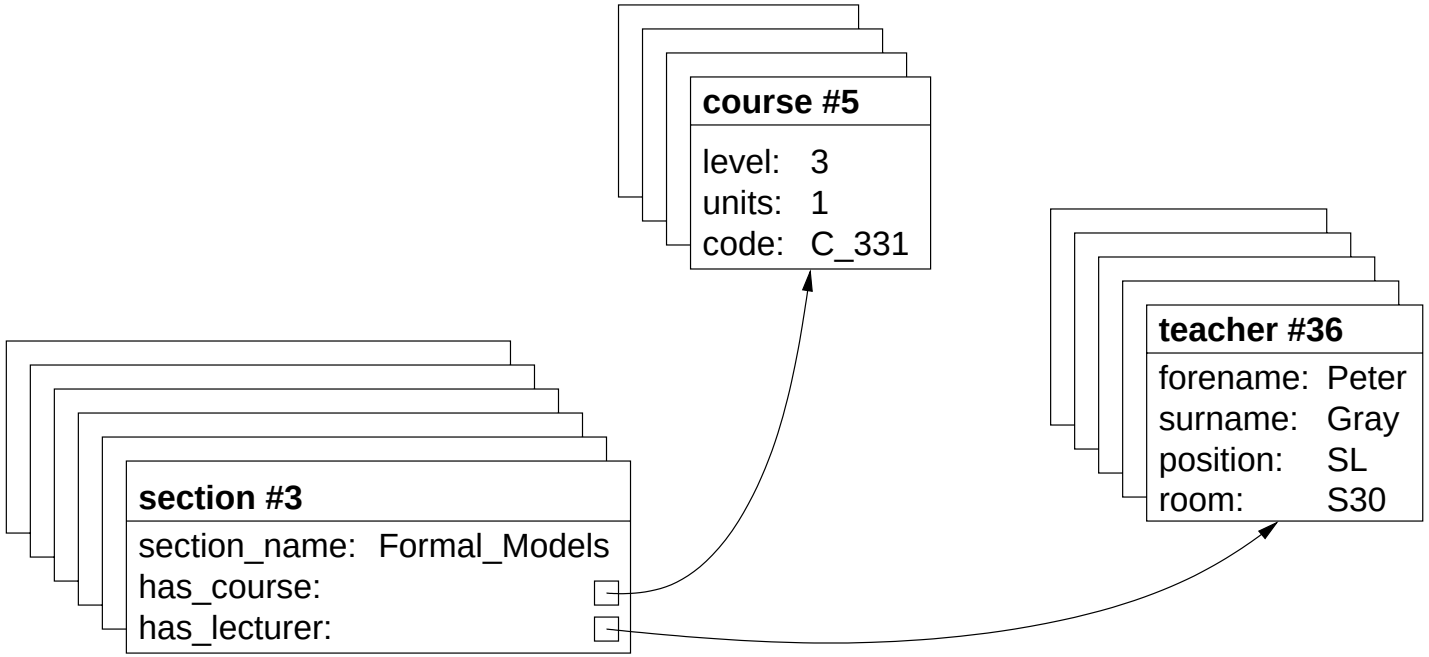
Daplex

- data definition language (DDL)
- data manipulation language (DML)

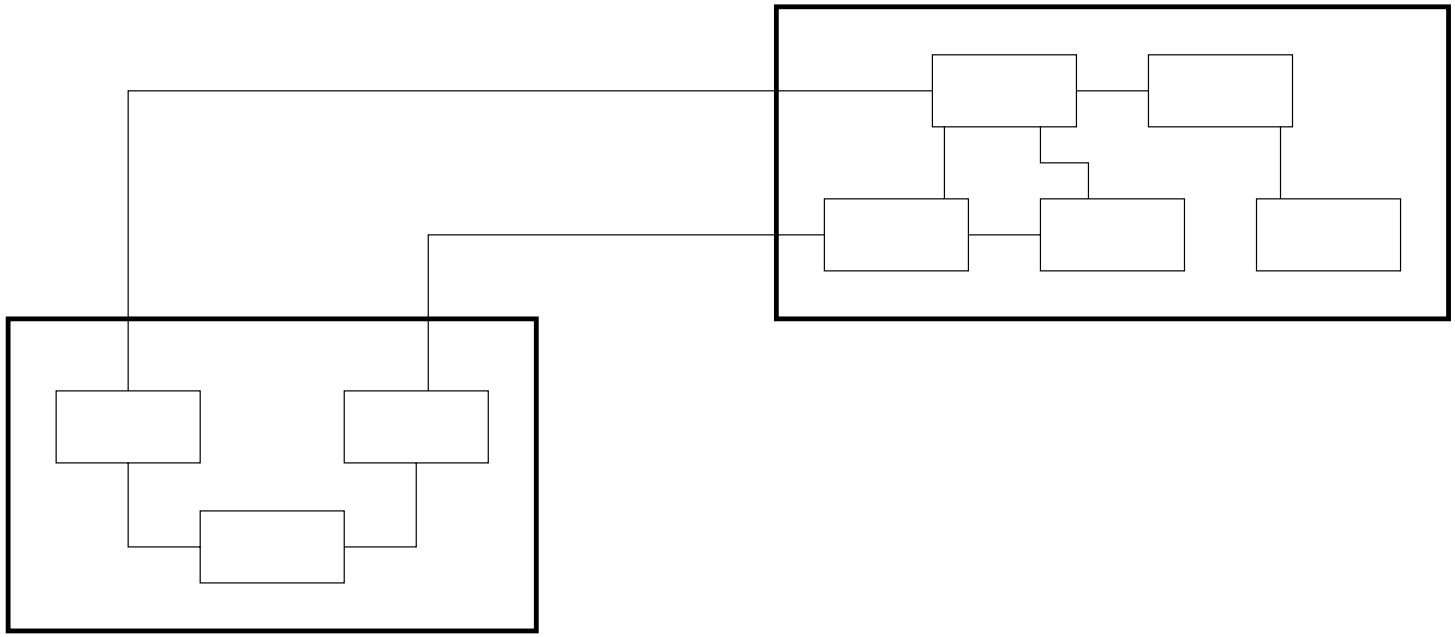
List comprehensions — Intermediate code (ICode)

Keys

Object instances



Modules in P/FDM



- Partition data
- Serve as the locking unit (one writer; many readers)
- Private extensions to the database
- Can have different physical representations

Accessing data in P/FDM

getentity(+Class, ?InstId)
getentity(+Class, +Key, ?InstId)

returns an instance of an entity class, e.g.

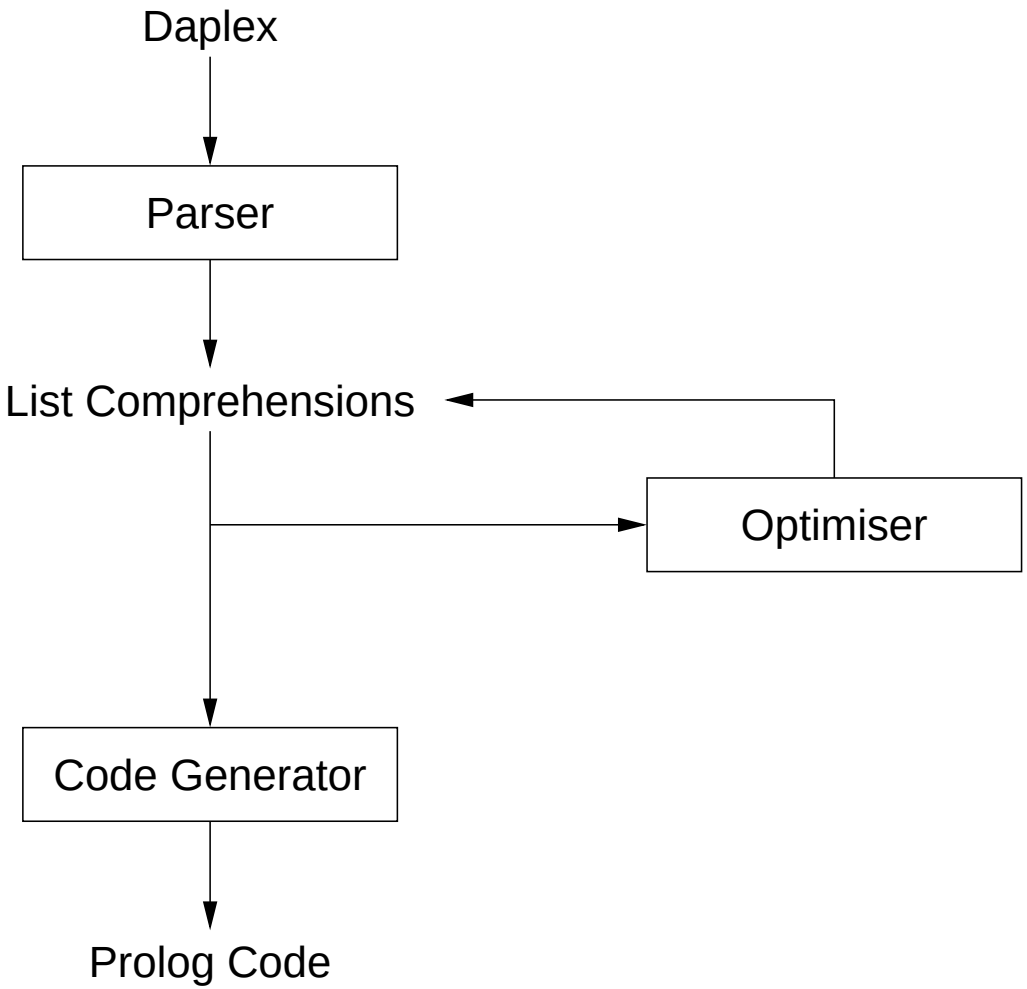
getentity(person, P)
getentity(person, [’Gray’,’Peter’], P)

getfnval(+Function, +Arguments, ?Result)

returns the value(s) of a function, e.g.

getfnval(surname, [P], Surname)
getfnval(has_lecturer, [S], T)
getfnval(has_lecturer_inv, [T], S)

Processing Daplex queries



A simple Daplex query

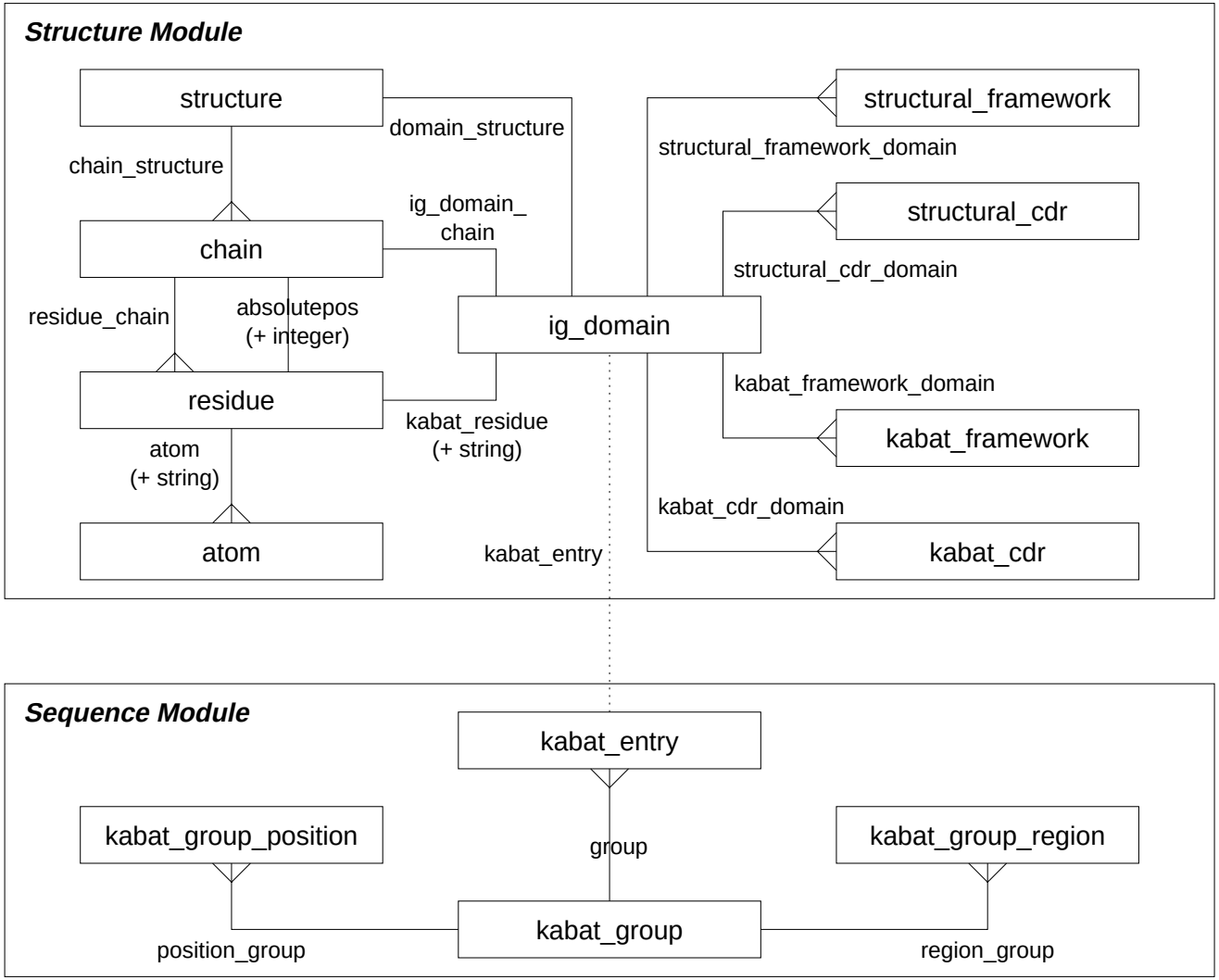
Daplex:

```
|: for each p in person such that forename(p) = 'Peter'
|: print(surname(p));
```

Prolog:

```
run :-
        (   getentity(person, A),
            getfnval(forename, [A], B),
            B='Peter',
            getfnval(surname, [A], C),
            write_list([C]),
            fail
        ;   true
        ).
```

Antibody database schema diagram



Object Instances