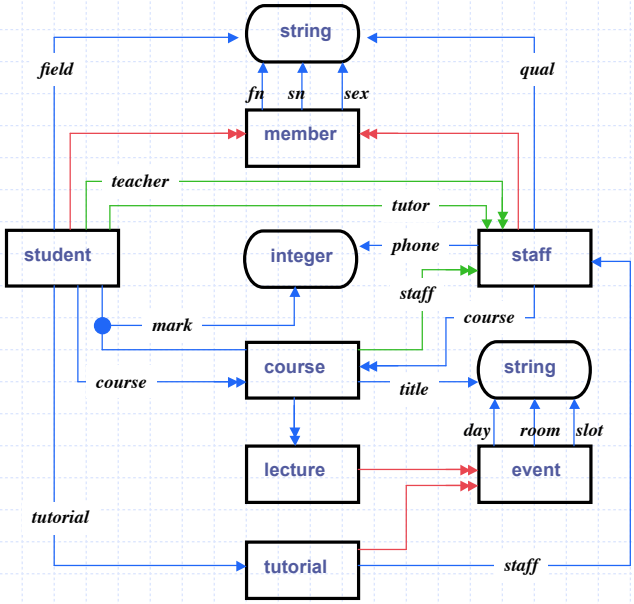
Extended Functional Data Model

- structures
  - function (aruments) → results (single-valued / multi-valued)
  - base function — stored data
  - derived function — algorithm
  - f ( ) — defines new entity type
  - f (a_1, a_2, ...) — defines attributes & relationships
- entity diagram
- operations
- constraints
- user views
- metadata

subtype - supertype

base functions

derived functions



base functions

```
declare
  {   member ()  ⇒  entity
      student ()  ⇒  member
      staff()  ⇒  member
      course()  ⇒  entity
      event ()  ⇒  entity
      tutorial()  ⇒  event
      lecture()  ⇒  event

      fn (member)  →  string
      sn (member)  →  string
      sex (member)  →  string

      course (student)  ⇒  course
      tutorial (student)  →  tutorial
      mark (student, course)  →  integer
      field (student)  →  string

      title (course)  →  string
      lecture (course)  ⇒  lecture

      day (event)  →  string
      slot (event)  →  string
      room (event)  →  string

      course (staff)  ⇒  course
      phone (staff)  →  integer
      qual (staf)  →  string
      staff (tutorial)  →  staff
  }
```


derived functions

```
define
  {   staff(course) ⇒  staff such that
                       some c in course (staff)
                       has c = course          -- inverse of

      teacher (student)   ⇒  staff (course (student))

      tutor (student)     →  staff (tutorial (student))

  }   -- combinations of inverse, composition, recursion,
transitivity
```

derived functions are represented by algorithms accepting arguments to compute results

retrievals

```
-- get the names of all members

for each m in member
get fn(m), sn(m)
```

```
-- get surnames of all female students

for each s in student
such that sex(s) = 'F'
get sn(s)
```


retrievals

```
-- get the names of those students that take a course on FDB

for each s in student
such that
        some c in course (s)
        has title (c) = 'FDB'
get sn(s)
```

retrievals

```
-- get the titles of courses taught by Stefan

for the s in staff
        such that  fn (s) = 'Stefan'
        for each c in course (s) get title(c)

-- error handling procedure is called if more than one Stefan exists
```


updating - insertion

```
a new m in member

-- creates a new member entity,  adds it to the extent of
        member type, associates it with the variable m
```

```
a new s in student

-- creates a new entity, which is included in the extents
        of both student and member entity types
```

updating - new record

```
for a new s in student
  let fn(s) = 'Mary'
  let sn(s) = 'Jones'
  let sex(s) = 'F'
  let field(s) = 'Comp'
```


updating - change values

```
for the s in student such that
fn(s) = 'Mary' and sn(s) = 'Jones'
  let tutorial(s) = the t in tutorial such that
  day(t) = 'Mon' and slot(t) = '09,10' and room(t) = 'm101'
```

updating - adding rules

```
for the s in student such that
fn(s) = 'Mary' and sn(s) = 'Jones'
 include course(s) = {
            the c1 in course such that title(c1) = 'Haskell'
            the c2 in course such that title(c2) = 'Prolog' }

-- similarly exclude
```


constraints

```
constraint unique-id on
            fn(member), sn(member) → unique

constraint must-be-supplied on
            sex(member) → total                -- i.e. not partial

constraint must-differ on
            student, staff → disjoint

constraint non-upd-sex on
            sex(member) → fixed

constraint ris on
            mark (student, course) →
            some c in course(student)
            has c = course
```