Record Types

```
type EmployeeRecd = {age:int, name:string, salary:int};

- val r = {name="Jones", age=25, salary=45000};
val r = {age=25, name="Jones", salary=45000}
  : {age:int, name:string, salary:int}

- (#age r);
val it = 25 : int

- val {name=name_r, age=age_r, ...} = r;
val age_r = 25 : int
val name_r = "Jones" : string

- val {name, age, ...} = r;
val age = 25 : int
val name = "Jones" : string
```

Record Types

```
- fun YearsToRetirement(butler) =
      67 - (#age butler);
```

```
stdIn:3.1-4.29 Error: unresolved flex record
   (can't tell what fields there are besides #age)
```

Partial record specifications. A field selection that omits some of the fields does not completely specify the record type; a function may only be defined over a complete record type. For instance, a function cannot be defined for all records that have fields *born* and *died*, without specifying the full set of field names (typically using a type constraint). This restriction makes ML records efficient but inflexible. It applies equally to record patterns and field selections of the form #*label*. Ohori (1995) has defined and implemented flexible records for a variant of ML.

(L.C. Paulson, *ML for the Working Programmer (2nd ed.)*, sec 2.9, page 35)

Record Types

```
- fun YearsToRetirement(butler : EmployeeRecd) =
      67 - (#age butler);
```

```
val YearsToRetirement = fn : EmployeeRecd -> int
```

Record Types

```
- fun YearsToRetirement(butler : EmployeeRecd) =
      67 - (#age butler);
val YearsToRetirement = fn : EmployeeRecd -> int

- fun YearsToRetirement({age,...} : EmployeeRecd) =
      67 - age;

- fun YearsToRetirement({age,name,salary}) =
      67 - age;
```

Record Types

```
- val r = {name="Jones", age=25, salary=45000}

- YearsToRetirement r;
val it = 42 : int

- map (#salary) [r];
val it = [45000] : int list

- map (#salary);
stdIn:17.1-17.14 Error: unresolved flex record
   (can't tell what fields there are besides #salary)
```