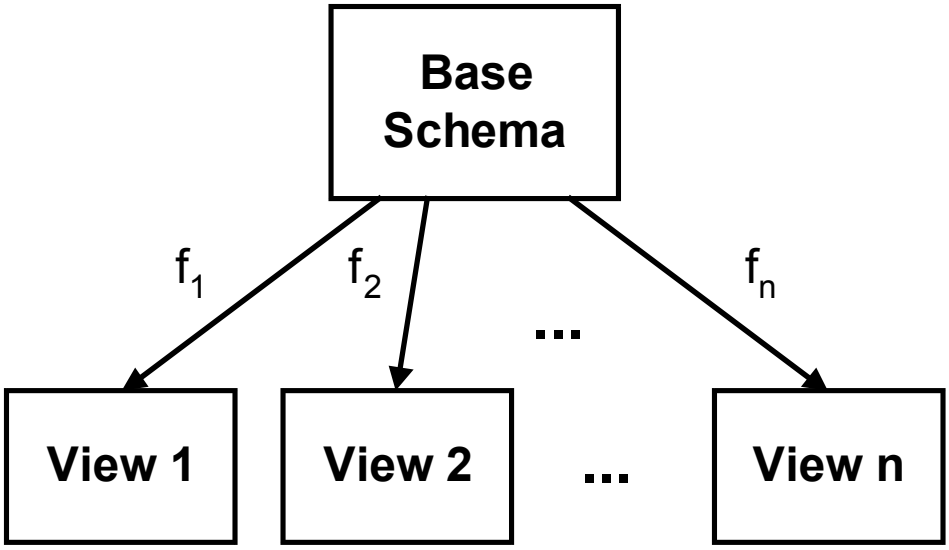
Views

What is a view?



- A view is just a functional image of a database schema.

  How does a view differ from a query?

- Structurally, they are identical.
- Functionally:
  - A view persists in time much longer than a query response.
  - A view tracks the changes made in the state of the base schema.
  - A view may itself be queried.
  - A view may even be updated under certain circumstances.

Three reasons why view are important:

- Security
- Simplicity
- Summary

Some examples from the running text example to illustrate:

Security: Allow a user to see the names of employees, but not their salaries.

```
Create View  Emp_Names
as   Select    LName, FName, MInit, SSN
     From      Employee
```

Simplicity: Show just the project names and their locations.

```
Create View Simp_Proj
as   Select    PName, PLocation
     From      Project
```

Summary: Provide a user with the minimum, maximum, and average salary, by department.

```
Create View Salary_Summary
Select    Min(Salary) as Min_Salary,
          Max(Salary) as Max_Salary,
          Avg(Salary) as Avg_Salary,
          DNo
From      Employee, Department
Where     Employee.DNo =
                    Department.DNumber
Group by       DNo
```

Note: While Microsoft Access does not support the SQL Create View operator, PostgreSQL does support it quite nicely. ☺

An ideal property of a view:

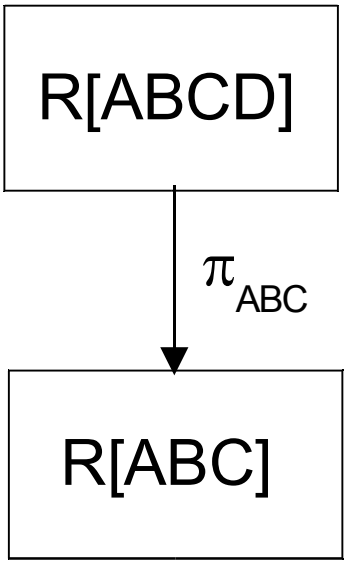
- A view should be *functionally indistinguishable* from a base relation.
  - Queries
  - Updates

With respect to queries, this is not too difficult to achieve in practice.

With respect to updates, it is usually unachievable, even for simple relational views defined by projection, selection, and join (SPJ queries).

- The view mapping f_l need not be injective, so there need not be a unique way to reflect the update back into the base schema.
- The *implied constraints* on the base schema may be extremely complex, even if the constraints on the base schema are extremely simple.

Some simple examples of views on relations which have only key constraints:



A few $(\forall)(\forall)$
2-easy
constraints

Infinitely many
$(\forall)...(\forall)$
constraints;
for each n, at least
one of which is not
n-easy

Since foreign keys are always a problem if the associated keys for both relations are not available, they will not be mentioned explicitly.

Selection:

- Deletion is usually not a problem.
- Update is not a problem if the primary key is not altered
- Insertion is usually impossible.

```
Create View      Female_Employees
   Select    *
   From           Employee
   Where          Sex = 'F'

Create View     Female_NonManagers
   Select       *
      From           Employee
      Where    Sex = 'F'
                  and not Exists
                        (Select *
                        From Department
                        Where MgrSSN = SSN)
```

Projection:

- Deletion is usually not a problem.
- Update is not a problem, as long as a candidate keys are included in the projection.
- Insertion is possible only if the attributes which are not projected may be null. Nulls are then inserted. (This implies that all key attributes are projected.)

```
Create View    Names_and_SSN
   Select      LName, FName, MInit, SSN
   From        Employee
```

Join:

- Characterizing updatability through a join is very difficult, and there no nice, clean, general rules.
- Specific conditions will not be provided here.