EXPRESS-X Tutorial

PDES, Inc. Offsite

March 11, 1998

John D. Valois
valois@steptools.com

STEP Tools, Inc.
Rensselaer Technology Park
Troy, New York 12180

(518) 276-2848 (518) 276-8471 fax
info@steptools.com http://www.steptools.com

Overview

STEP Tools, Inc.

- Introduction and background
  - What is a mapping language?
  - What is it good for?
  - When can I get one?

- First version of EXPRESS-X
  - Fundamental principles and capabilities
  - Shortcomings

- Next version of EXPRESS-X
  - Fundamental principles and capabilities

What is it?

STEP Tools, Inc.

A mapping language:

- Allows formal specification of how elements of two data models are related
  - Entities, attribute values, relationships are included
  - Constraints are not
- It is implementable
  - Has an execution model
- It is not:
  - A programming language
  - A replacement for the SDAI

Out of scope

STEP Tools, Inc.

- Mapping of data defined using means other than EXPRESS
- Mapping of data defined using the second edition of EXPRESS
- Identification of the version of an EXPRESS schema
- Graphical representation

Applications

STEP Tools, Inc.

A mapping language can be used to specify:

- Translation between different models
- Migration between versions of a model
- Interoperability between models
- High-level (ARM, business object, etc.) views
- Interfaces to legacy data ("pull" mapping)

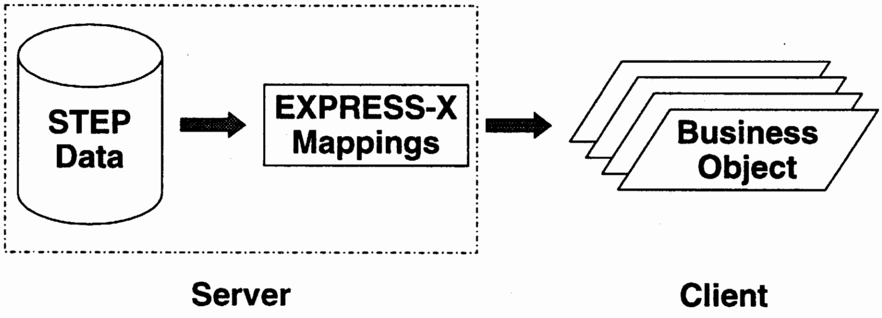
Early Users

STEP Tools, Inc.

NIIP Project

(National Industrial Information Infrastructure Protocols)

- Defining high-level views for implementing business objects based on STEP.

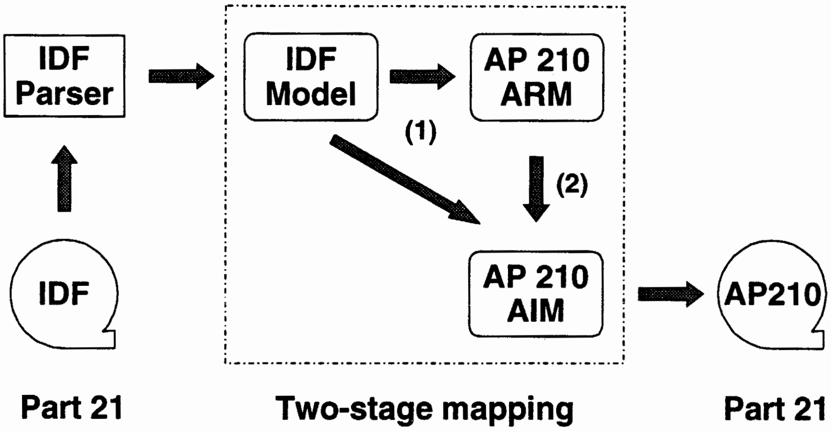


Early Users

STEP Tools, Inc.

PDES, Inc. Electromechanical Pilot

- Mapping IDF into AP210



IDF Parser

IDF Model

AP 210 ARM

(1)

(2)

IDF

AP 210 AIM

AP210

Part 21

Two-stage mapping

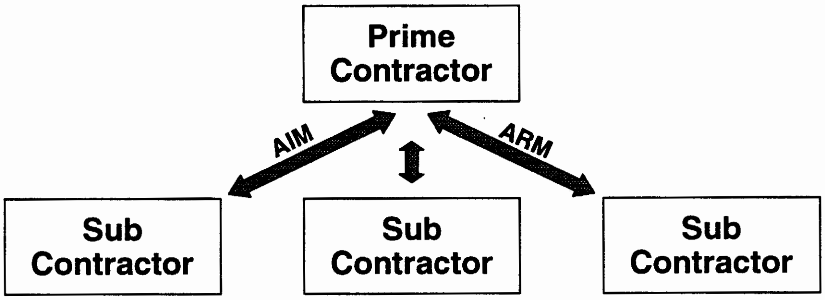
Part 21

Early Users

STEP Tools, Inc.

PCALS AP227 Project

- Exchange using either AIM or ARM



Prime Contractor

AIM

ARM

Sub Contractor

Sub Contractor

Sub Contractor

Early Users

STEP Tools, Inc.

- Others:
  - VÅV project (EuroSTEP, EPM)
  - JSTEP HLDAI project
  - POSC PML

Current Status

STEP Tools, Inc.

When will it be ready?

- How did we get where we are
- What is happening now
- What will happen in near future

Current Status

STEP Tools, Inc.

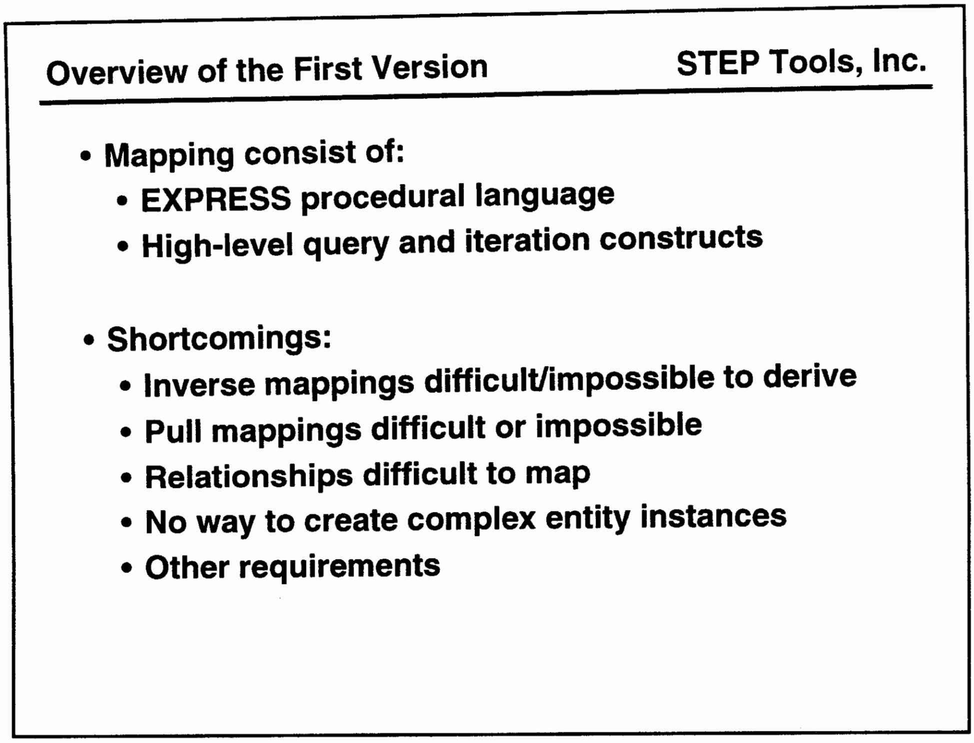
Pre-history

- EXPRESS-V
    - Developed by RPI and STEP Tools, Inc.
    - Goal: High-level “views” of STEP data.
- EXPRESS-M
    - Developed by Ian Bailey and CIMIO.
    - Goal: Translation of legacy data.
- BRITTY
    - Developed by Günter Sauter and Daimler-Benz.
    - Goal: “Pull” mapping of legacy data.

Current Status

STEP Tools, Inc.

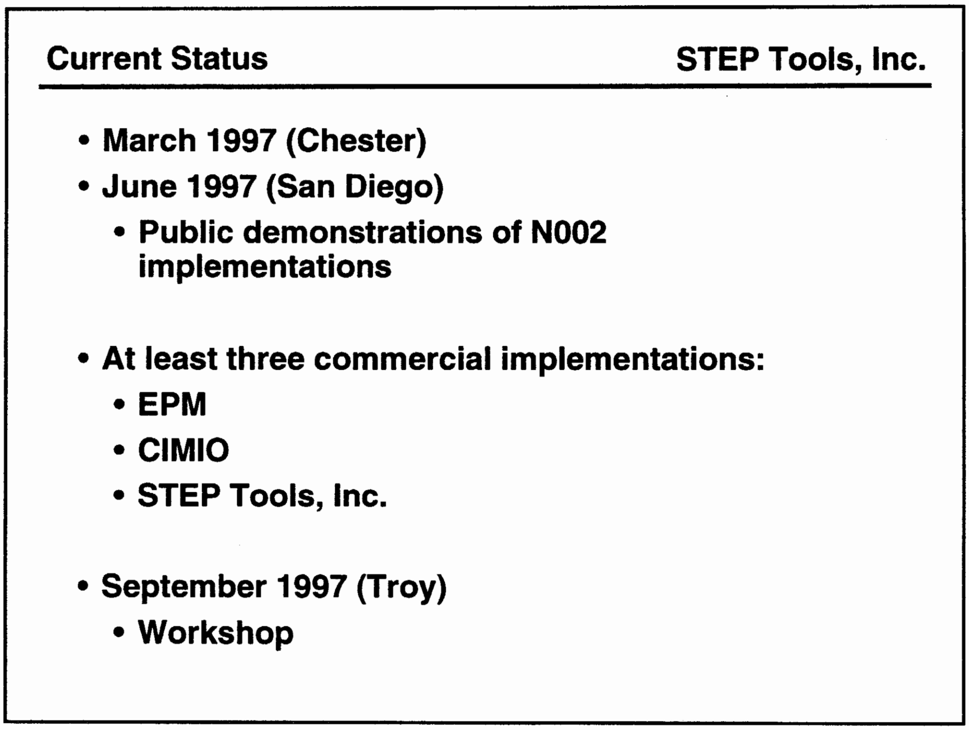
- June 1996 (Kobe)
    - PWI resolution passed
- August 1996
    - First version (WG11 N002), combining:
        - EXPRESS-V
        - EXPRESS-M
        - Expressions and statements from EXPRESS
- October 1996 (Toronto)
    - Requirements gathering, issue log initiated



Overview of the First Version

STEP Tools, Inc.

- Mapping consist of:
  - EXPRESS procedural language
  - High-level query and iteration constructs
- Shortcomings:
  - Inverse mappings difficult/impossible to derive
  - Pull mappings difficult or impossible
  - Relationships difficult to map
  - No way to create complex entity instances
  - Other requirements



Current Status

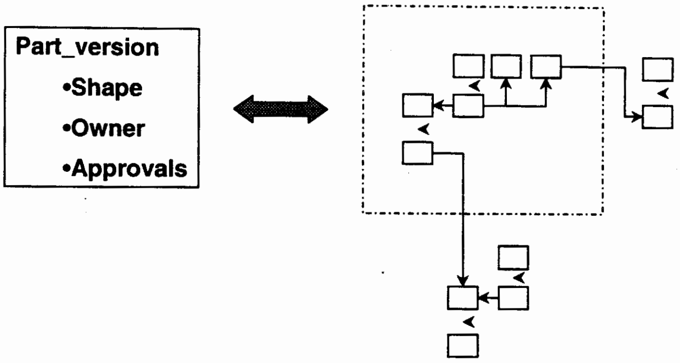
STEP Tools, Inc.

- March 1997 (Chester)
- June 1997 (San Diego)
  - Public demonstrations of N002 implementations
- At least three commercial implementations:
  - EPM
  - CIMIO
  - STEP Tools, Inc.
- September 1997 (Troy)
  - Workshop

View Mapping

STEP Tools, Inc.

- Data is reorganized
- Named, reusable query
- References to underlying data

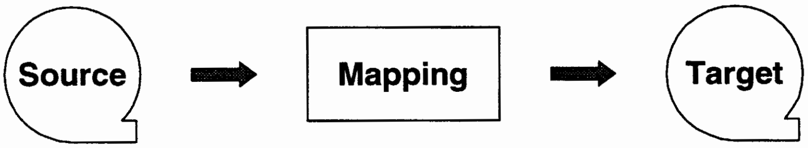


Part_version
•Shape
•Owner
•Approvals

Push Mapping

STEP Tools, Inc.

- Batch translation; all or nothing
- May be highly procedural
- Not necessarily reversible



Source

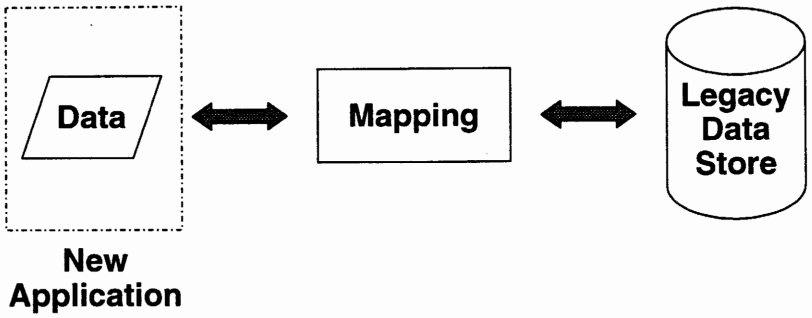
Mapping

Target

Pull Mapping

STEP Tools, Inc.

- Translation is on demand
- Important when source data size is large
- Mappings can be combined



Current Status

STEP Tools, Inc.

- February 1998 (Orlando)
  - 1st draft of next version
- June 1998 (Bad Aibling)
  - Wider distribution of document
  - Enable vendor implementation
  - Begin qualification process
- TBA
  - NWI ballot with CD

First Version

STEP Tools, Inc.

What is a “mapping”?

- Define one or more *schema instances.*
  - A data set, like a Part 21 file.
  - May have more than one based on same schema.
  - Data may or may not already exist.
  - No explicit notion of source or target.

Fundamental Principles

STEP Tools, Inc.

- Mappings create new instances.
  - Created in a *target* schema instance.
  - Based on one or more *source* instances.
  - May also create *manually instantiated* instances “out of thin air”.
  - Source instances may have a selection criteria based on attribute values (logical expression).
  - New instance is populated by a sequence of EXPRESS statements.

Fundamental Principles

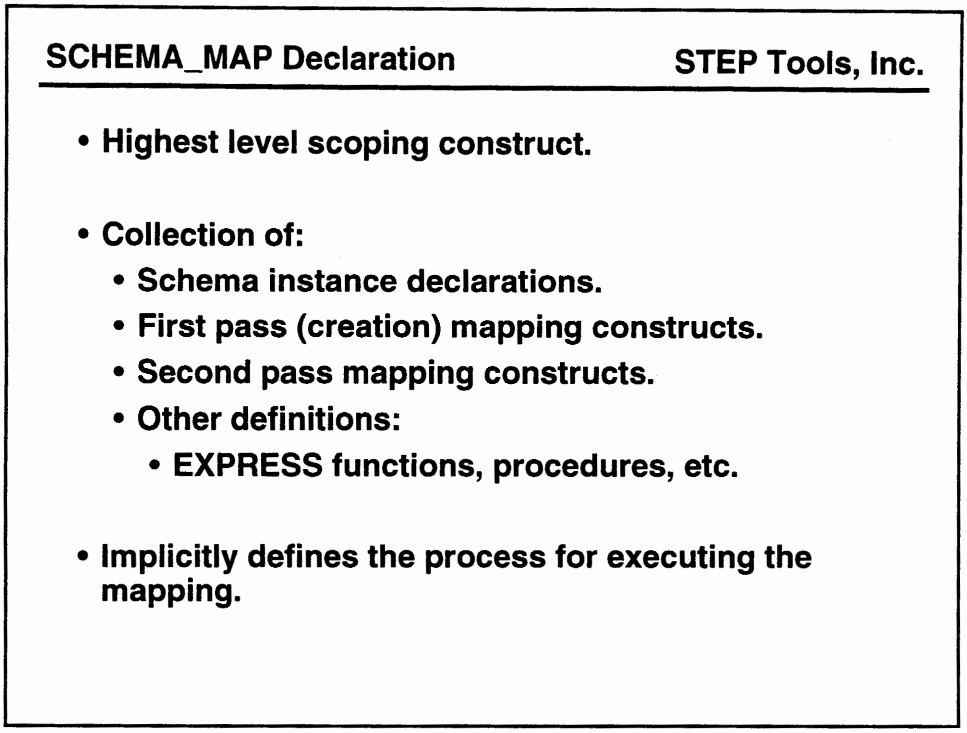
STEP Tools, Inc.

- Mappings may also make a second pass over the data.
- Like first pass, but no instance is created.
- Intent: enable population of target relationships.
- Can be used more creatively as well.

Fundamental Principles

STEP Tools, Inc.

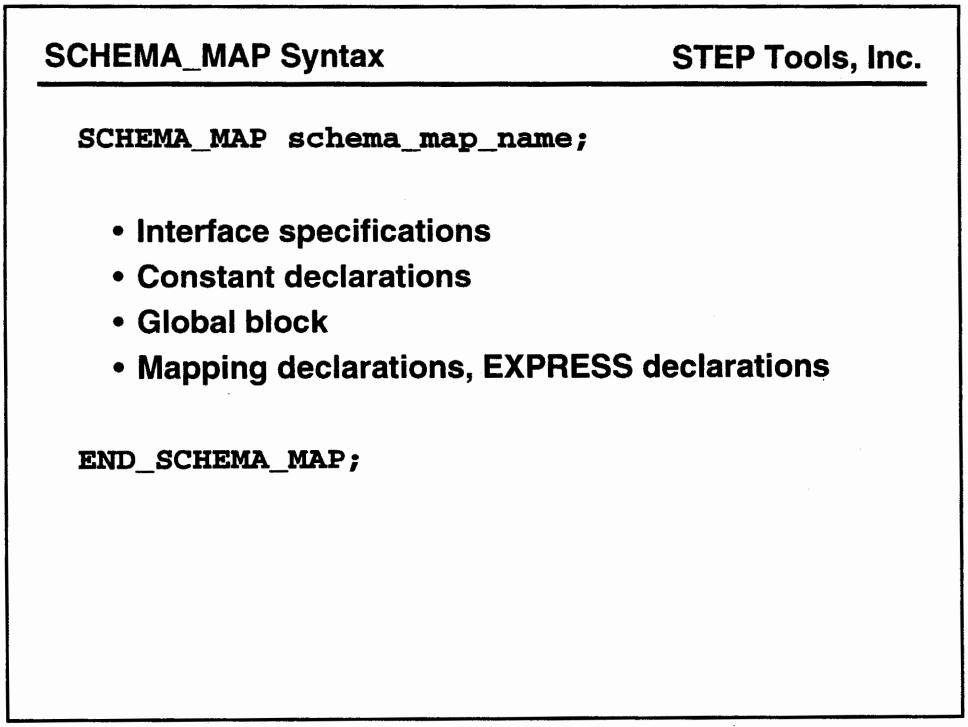
- Other odds and ends:
  - Ability to define "groups" of instances.
  - Handful of extensions to EXPRESS statements and expressions.
  - Ability to create and destroy instances at the statement level.
  - Ability to iterate over source data at statement level.



SCHEMA_MAP Declaration

STEP Tools, Inc.

- Highest level scoping construct.
- Collection of:
  - Schema instance declarations.
  - First pass (creation) mapping constructs.
  - Second pass mapping constructs.
  - Other definitions:
    - EXPRESS functions, procedures, etc.
- Implicitly defines the process for executing the mapping.



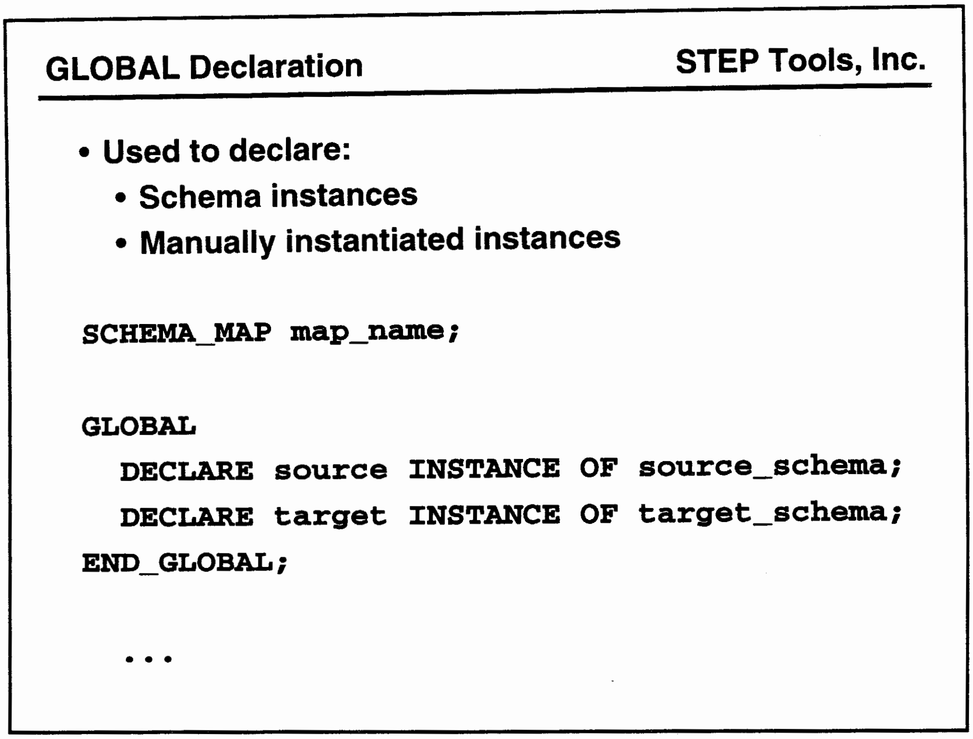
SCHEMA_MAP Syntax

STEP Tools, Inc.

```
SCHEMA_MAP schema_map_name;
```

- Interface specifications
- Constant declarations
- Global block
- Mapping declarations, EXPRESS declarations

```
END_SCHEMA_MAP;
```



GLOBAL Declaration

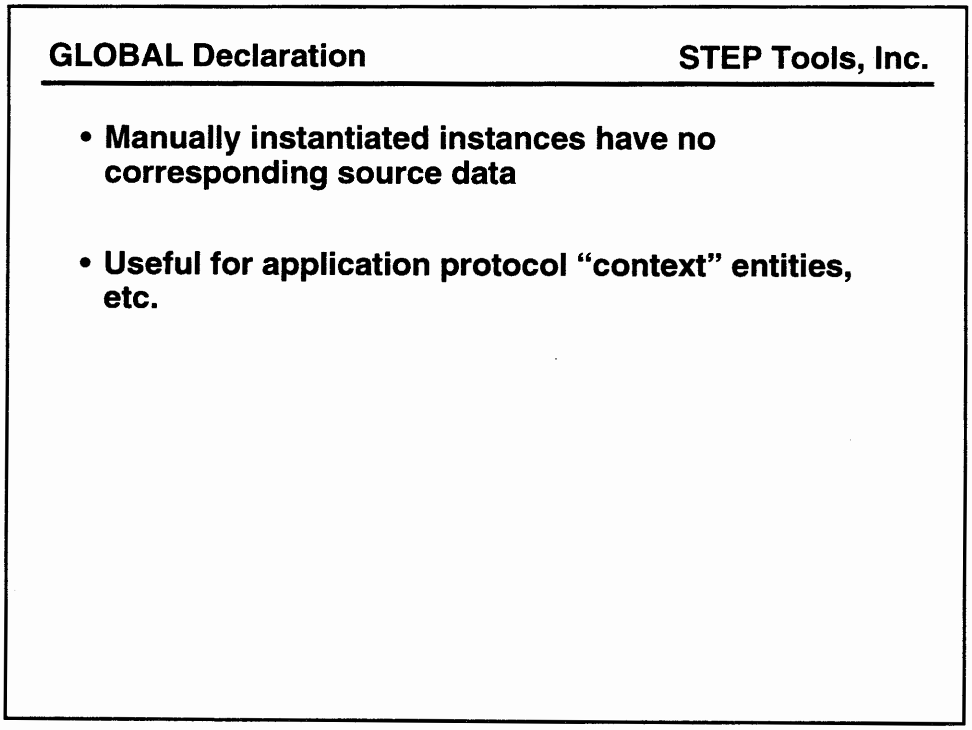
STEP Tools, Inc.

- Used to declare:
  - Schema instances
  - Manually instantiated instances

```
SCHEMA_MAP map_name;

GLOBAL
  DECLARE source INSTANCE OF source_schema;
  DECLARE target INSTANCE OF target_schema;
END_GLOBAL;

  ...
```



GLOBAL Declaration

STEP Tools, Inc.

- Manually instantiated instances have no corresponding source data

- Useful for application protocol “context” entities, etc.

Extended Identifiers

STEP Tools, Inc.

- Used to specify:
  - The scope containing the declaration of an identifier's type.
  - The schema instance in which to find/create the data.
- Example:

```
schema_name::identifier
```

- We will see this in other contexts as well.

Example

STEP Tools, Inc.

```
SCHEMA_MAP map;

GLOBAL
  DECLARE source INSTANCE OF arm_schema;
  DECLARE target INSTANCE OF aim_schema;
  #target::app_context_instance =
    application_context(...);
END_GLOBAL;

  ...

END_SCHEMA_MAP;
```

VIEW Declaration

STEP Tools, Inc.

- First pass mapping construct; creates instances.
- Header declares:
  - Type of instance to create.
  - Target schema instance.
  - Source instances (types and schema instances).
  - Source instance selection criteria.
- Body: sequence of EXPRESS statements.
  - Typically assignment statements, but others useful as well.

VIEW Declaration

STEP Tools, Inc.

Parts of a VIEW declaration:

```
VIEW
```

- target entity specification
- source data specification (“from” clause)
- source constraints (“when” clause)

```
BEGIN_VIEW
```

- EXPRESS statements to populate target entity

```
END_VIEW;
```

Example

STEP Tools, Inc.

```
VIEW da : target::dated_approval;
FROM (
  apo : source::approval_person_organization,
  d : source::cc_design_date_and_time_assignment)
WHEN
   apo IN d.items;
   d.role = 'sign_off_date';
BEGIN_VIEW
year_approved := d.assigned_date_and_time
                   .date_component
                   .year_component;
   ...
END_VIEW;
```

Execution Model

STEP Tools, Inc.

- When are target instances created?
  - Iterate over all combinations of source instances (Cartesian product).
  - Ignore those that do not satisfy all domain rules in the WHEN clause.
  - For each that does, instantiate a target instance, and
  - Execute the body.

Source Instance Types

STEP Tools, Inc.

- Note on source instances:
  - Includes both instances of explicit type and subtypes, complex instances.

```
DECLARE s INSTANCE OF some_schema;
...
FROM (x : s::t) ...
```

- Will iterate over:
  - All instances x in schema instance s;
  - Satisfying:
    `'SOME_SCHEMA.T' IN TYPEOF(x)`

Example

STEP Tools, Inc.

```
VIEW ...
FROM (
  apo : source::approval_person_organization,
  d : source::cc_design_date_and_time_assignment)
WHEN
   apo IN d.items;
   d.role = 'sign_off_date';
...
```

- Examine all pairs of instances of the two entities (and subtypes)
- Eliminate:
  - Those not related by items attribute
  - Those where role is not correct

VIEW Body

STEP Tools, Inc.

- Essentially an EXPRESS procedure
  (may define local variables, etc.)

- May use any EXPRESS statement or expression.

- Useful extensions:
  - Enhanced assignment operators.
  - Explicit instantiation.
  - Iteration over a population.

Enhanced Assignment

STEP Tools, Inc.

- Two additional operators.

- Aggregate addition: '+='
  - Useful to build up aggregate attributes, usually within some kind of loop.

- Aggregate removal: '-='
  - Not well motivated or used in practice.

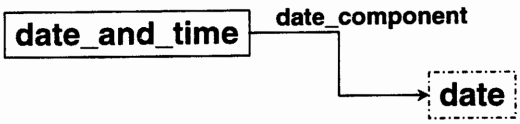
Explicit Instantiation

STEP Tools, Inc.

- The NEW statement creates a new instance.
- Must specify an l-value expression.

- Example

```
VIEW d : target::date_and_time;
FROM ...
BEGIN_VIEW
  NEW d.date_component;
END_VIEW;
```



Iteration Over a Population

STEP Tools, Inc.

- Iterates over combinations of instances that match a selection criteria.

- Similar to the FROM/WHEN header, but at the statement level.

FROM

- source specification

WHEN

- constraints

BEGIN

- statements

END;

Example

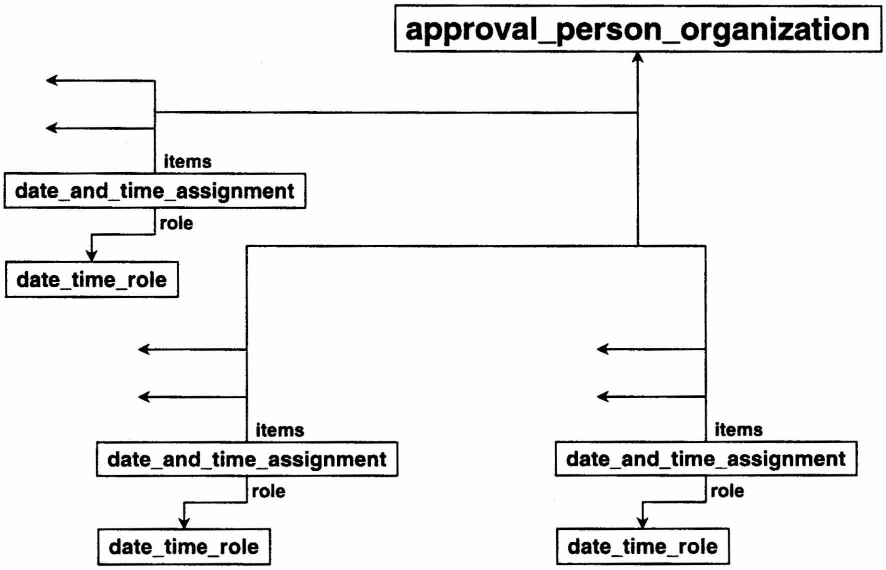
STEP Tools, Inc.

- What are the roles of the dates assigned?

```
VIEW da : target::dated_approval;
FROM (
  apo : source::approval_person_organization)
WHEN TRUE;
BEGIN_VIEW
  FROM (d : source::date_and_time_assignment)
  WHEN apo IN d.items;
  BEGIN
    da.date_roles += d.role;
  END;
END_VIEW;
```

Example

STEP Tools, Inc.



approval_person_organization

items

date_and_time_assignment

role

date_time_role

items

date_and_time_assignment

role

date_time_role

items

date_and_time_assignment

role

date_time_role

Other Extensions

STEP Tools, Inc.

- WHEN statement:
  - Like EXPRESS IF...THEN...

```
WHEN (logical_expression)
  BEGIN
    statements ...
  END;
```

- DELETE statement:
  - Destroys instances.
  - Not well motivated or used in practice.

IS Operator

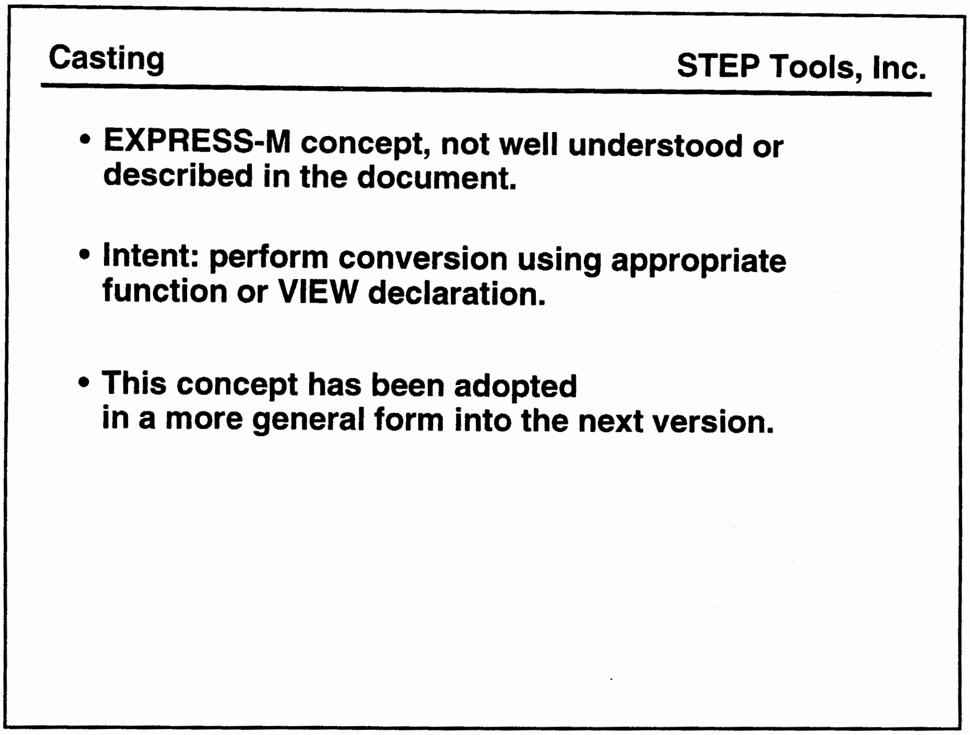
STEP Tools, Inc.

- Shortcut logical operator.

```
DECLARE s INSTANCE OF some_schema;
  ...
x IS s::t
```

is equivalent to

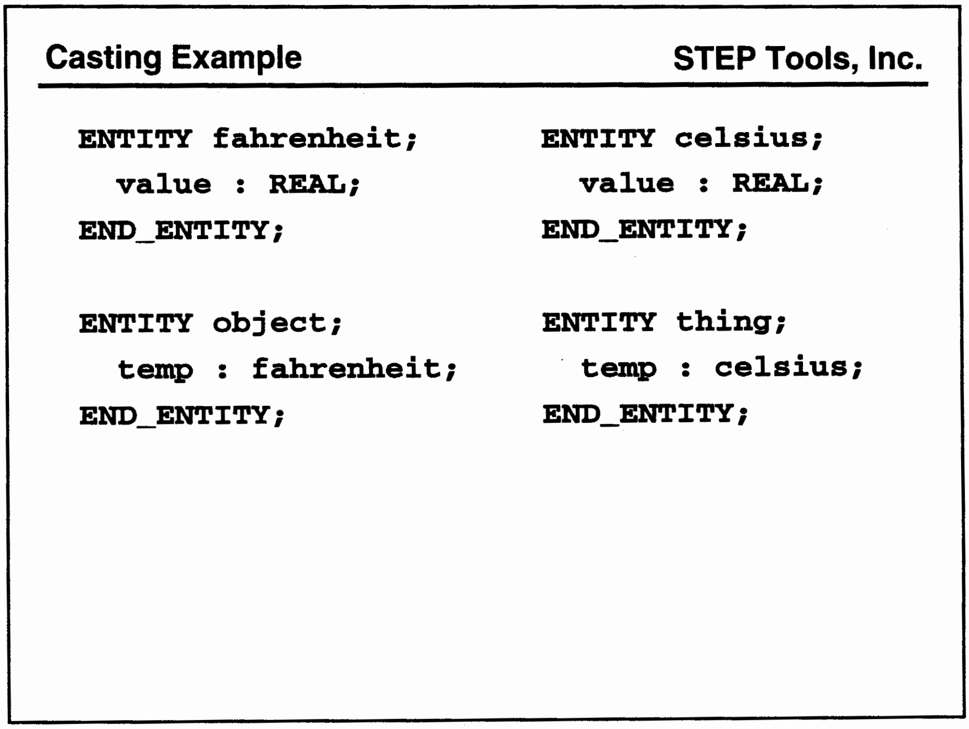
```
'SOME_SCHEMA.T' IN TYPEOF(x)
```



Casting

STEP Tools, Inc.

- EXPRESS-M concept, not well understood or described in the document.
- Intent: perform conversion using appropriate function or VIEW declaration.
- This concept has been adopted in a more general form into the next version.



Casting Example

STEP Tools, Inc.

```
ENTITY fahrenheit;           ENTITY celsius;
  value : REAL;                value : REAL;
END_ENTITY;                  END_ENTITY;

ENTITY object;               ENTITY thing;
  temp : fahrenheit;           temp : celsius;
END_ENTITY;                  END_ENTITY;
```

Casting Example

STEP Tools, Inc.

```
VIEW f : target::fahrenheit;
FROM (c : source::celsius) WHEN TRUE;
BEGIN_VIEW
  f.value := c.value * 9/5 + 32;
END_VIEW;

VIEW obj : target::object;
FROM (t : source::thing) WHEN TRUE;
BEGIN_VIEW
  obj.temp := {fahrenheit} t.temp;
END_VIEW;
```

cast

Coercion

STEP Tools, Inc.

- Another EXPRESS-M concept, not well suited to the EXPRESS-X paradigm.
- EXPRESS-M supported *implicit instantiation*; EXPRESS-X does not.
- Used with assignment statements to control instantiation of subtypes and SELECT base types.

Coercion Example

STEP Tools, Inc.

```
ENTITY polygon;
  edges : LIST OF edge;
END_ENTITY;

ENTITY edge SUPERTYPE OF
            ONEOF(solid, dashed);
END_ENTITY;

ENTITY solid;                ENTITY dashed;
  thick : INTEGER;             size : INTEGER;
END_ENTITY;                  END_ENTITY;
```

Coercion Example

STEP Tools, Inc.

```
VIEW p : source::polygon;
FROM ...
BEGIN_VIEW
  {(dashed)} edges[1].size := 1;
END_VIEW;
```

- Intent is to implicitly instantiate a dashed edge.

- Specification is unclear and does not generalize (for example, to nested selects).

COMPOSE Declaration

STEP Tools, Inc.

- Second pass mapping construct; populate attributes in previously instantiated entities.
- Header and body identical to VIEW (FROM clause is optional).

`COMPOSE`

- target entity specification
- optional additional source data specification
- source constraints

`BEGIN_COMPOSE`

- EXPRESS statements to populate target entity

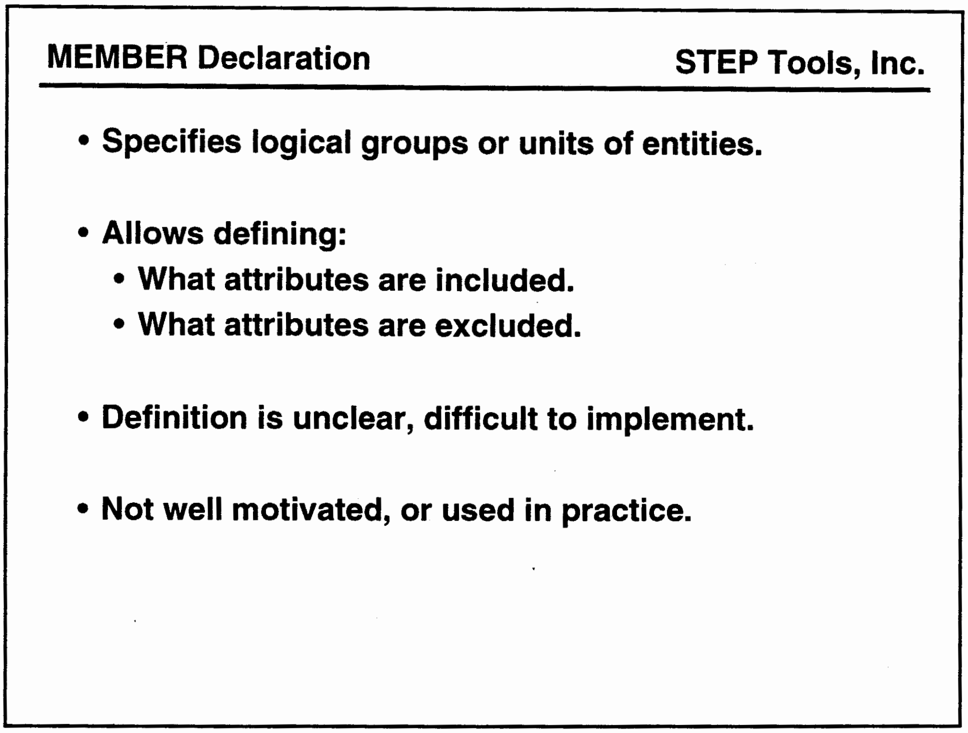
`END_COMPOSE;`

Example

STEP Tools, Inc.

- COMPOSE is useful for populating relationships.

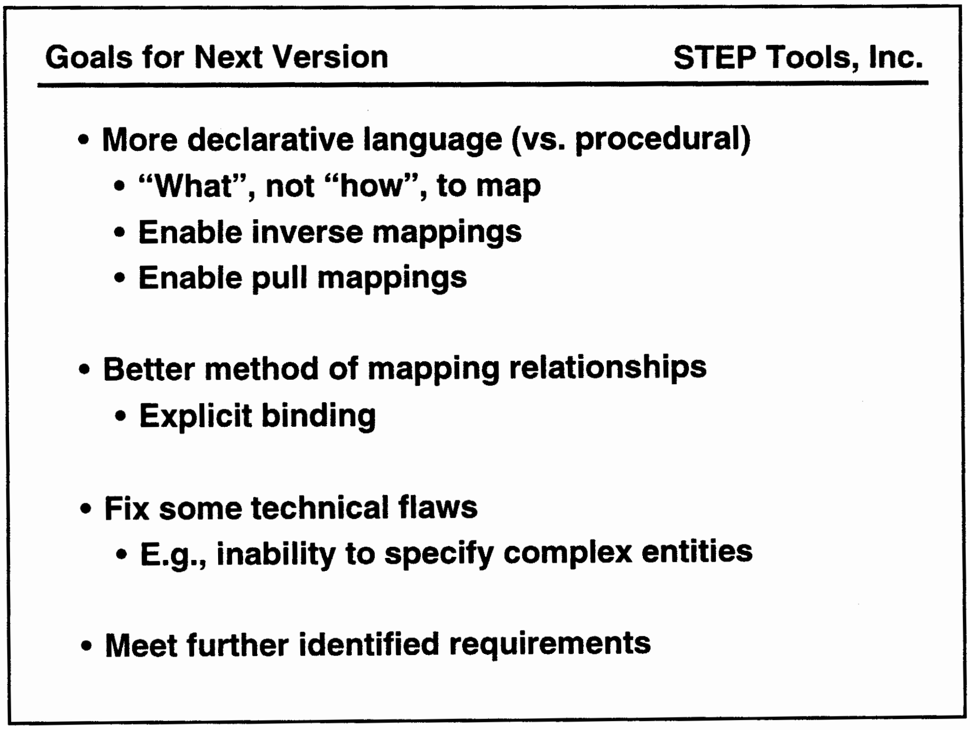
```
COMPOSE h : target::head_of_household;
WHEN TRUE;
BEGIN_COMPOSE
FROM (d : target::dependent)
WHEN (h.family_name = d.family_name);
  BEGIN
    h.dependents += d;
  END;
END_COMPOSE;
```



MEMBER Declaration

STEP Tools, Inc.

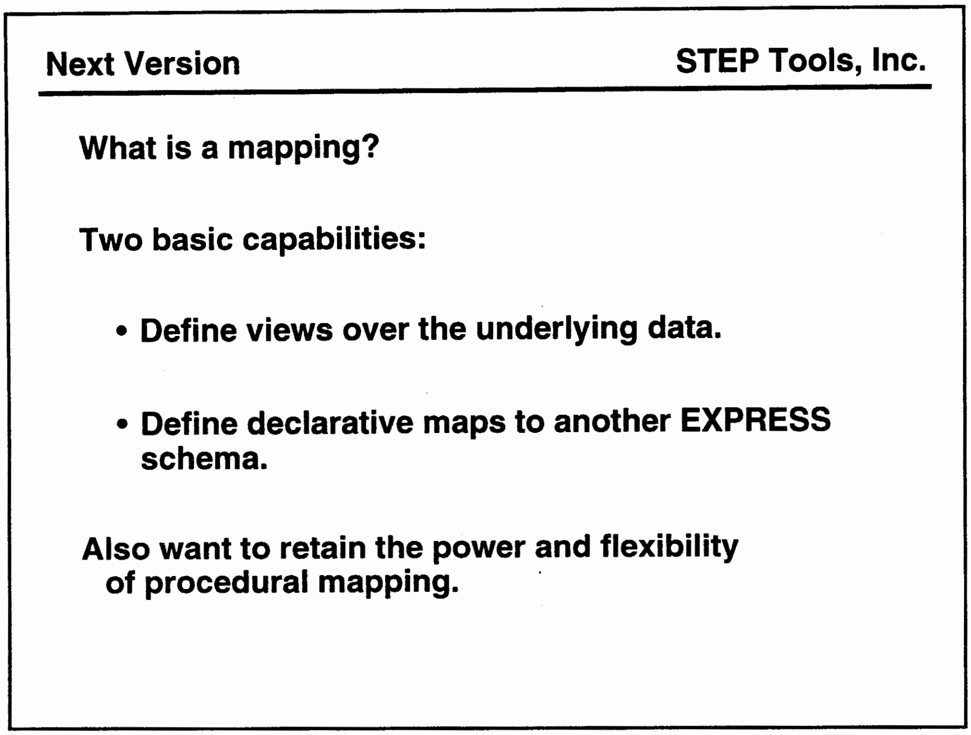
- Specifies logical groups or units of entities.
- Allows defining:
  - What attributes are included.
  - What attributes are excluded.
- Definition is unclear, difficult to implement.
- Not well motivated, or used in practice.



Goals for Next Version

STEP Tools, Inc.

- More declarative language (vs. procedural)
  - “What”, not “how”, to map
  - Enable inverse mappings
  - Enable pull mappings
- Better method of mapping relationships
  - Explicit binding
- Fix some technical flaws
  - E.g., inability to specify complex entities
- Meet further identified requirements



Next Version

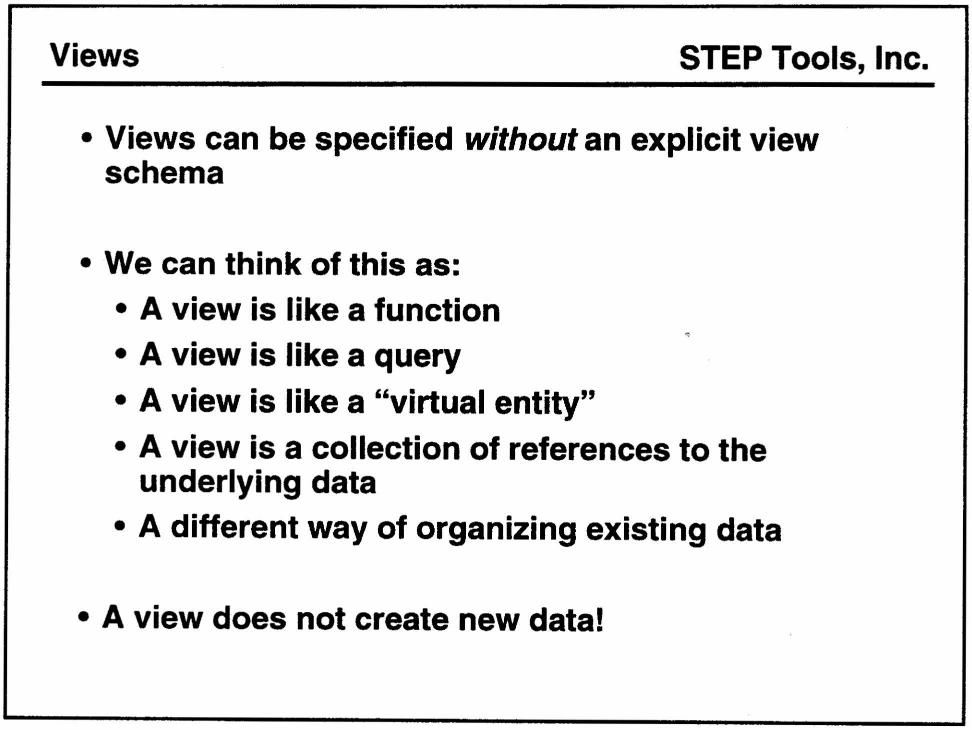
STEP Tools, Inc.

What is a mapping?

Two basic capabilities:

- Define views over the underlying data.
- Define declarative maps to another EXPRESS schema.

Also want to retain the power and flexibility of procedural mapping.



Views

STEP Tools, Inc.

- Views can be specified *without* an explicit view schema
- We can think of this as:
  - A view is like a function
  - A view is like a query
  - A view is like a “virtual entity”
  - A view is a collection of references to the underlying data
  - A different way of organizing existing data
- A view does not create new data!

Fundamental Principles

STEP Tools, Inc.

Extents

- An *extent* is a set of “entity-like” conglomerates of data
- A member of an extent consists of one or more references (“attributes”) to underlying data
- An entity defines an implicit extent
  - Each entity instance is a member of the extent
- A view defines an explicit extent, and the “attributes” of its members

VIEW Declaration

STEP Tools, Inc.

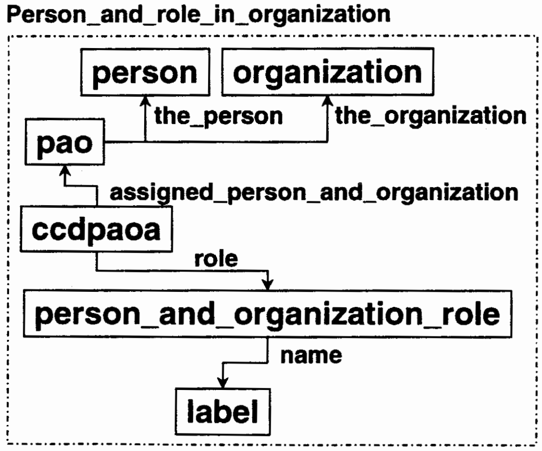
Example VIEW

```
VIEW person_and_role_in_organization
FROM pao : person_and_organization,
   ccdpaoa : cc_design_person_and_organization_assignment
WHERE
   ccdpaoa.assigned_person_and_organization :=: pao;
SELECT
  person := pao.the_person;
  org    := pao.the_organizaion;
  role   := ccdpaoa.role.name;
END_VIEW;
```

VIEW Declaration

STEP Tools, Inc.

Each member of the extent corresponds to a collection of underlying data:



Person_and_role_in_organization

person

organization

the_person

the_organization

pao

assigned_person_and_organization

ccdpaoa

role

person_and_organization_role

name

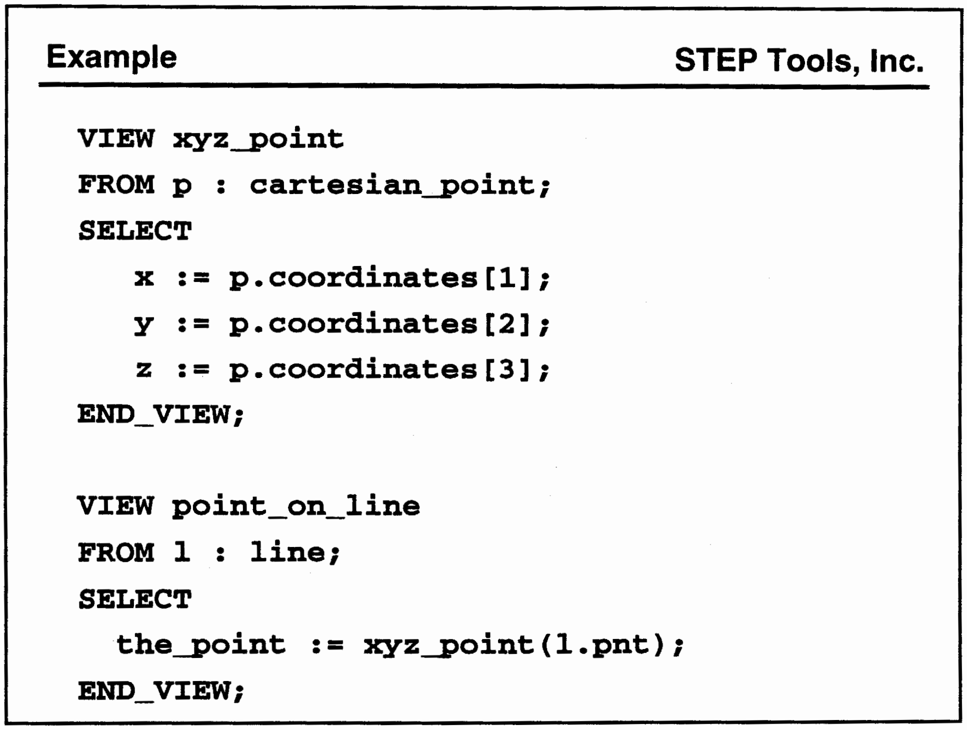
label

Fundamental Principles

STEP Tools, Inc.

Explicit Binding

- Once an extent has been defined, we can refer to individual members
- We reference a particular member by specifying the source instances from which it derives
- This makes it easy to specify relationships

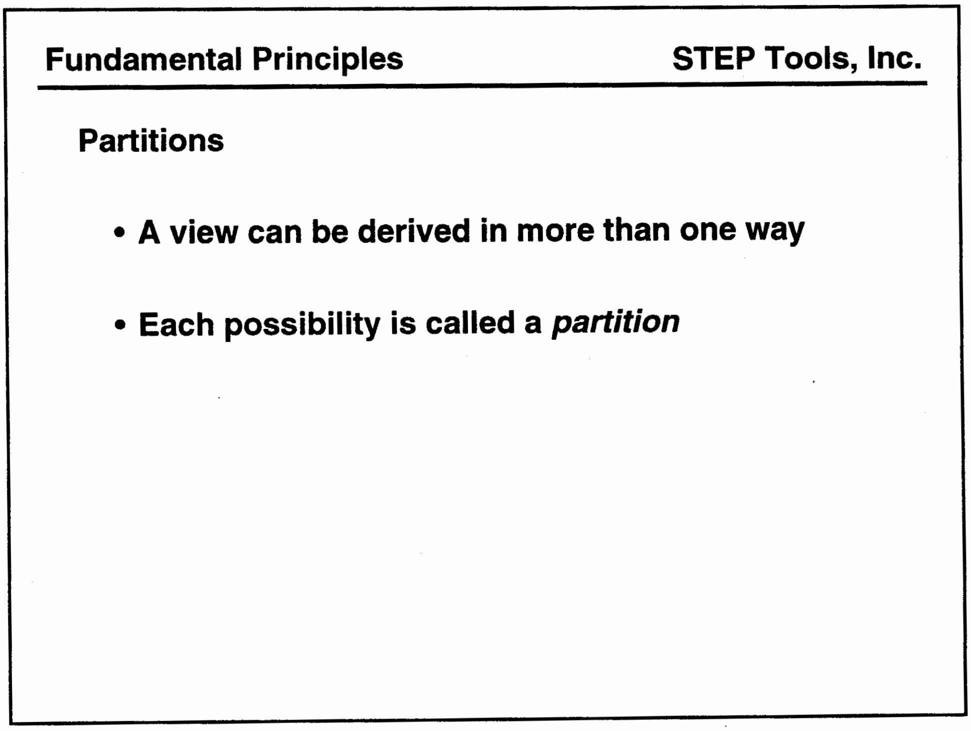


Example

STEP Tools, Inc.

```
VIEW xyz_point
FROM p : cartesian_point;
SELECT
   x := p.coordinates[1];
   y := p.coordinates[2];
   z := p.coordinates[3];
END_VIEW;

VIEW point_on_line
FROM l : line;
SELECT
  the_point := xyz_point(l.pnt);
END_VIEW;
```

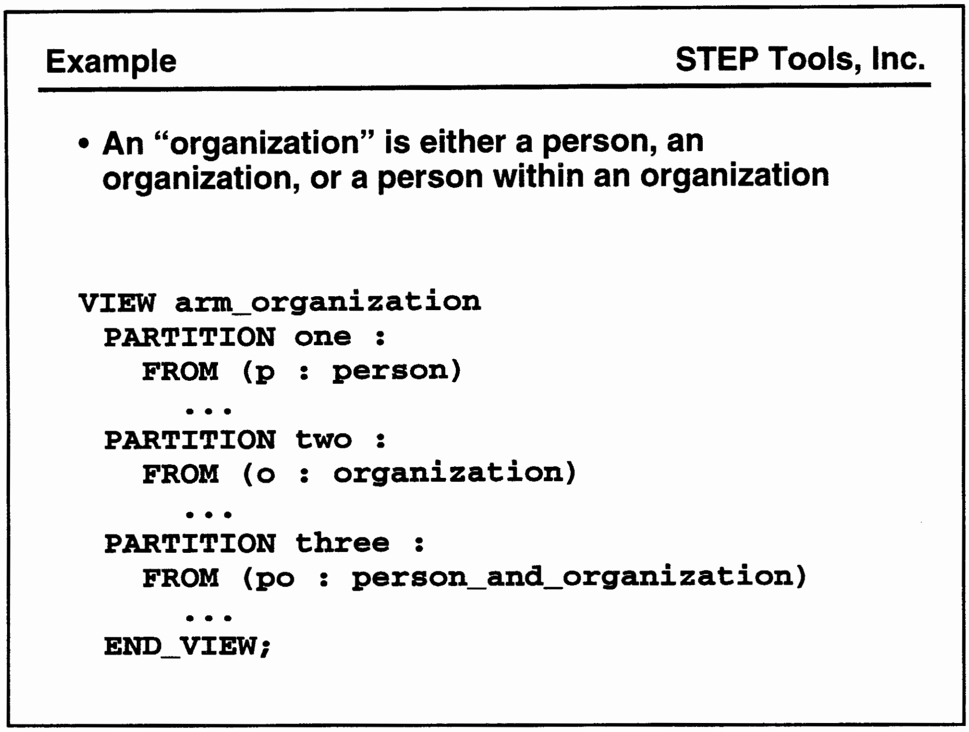


Fundamental Principles

STEP Tools, Inc.

Partitions

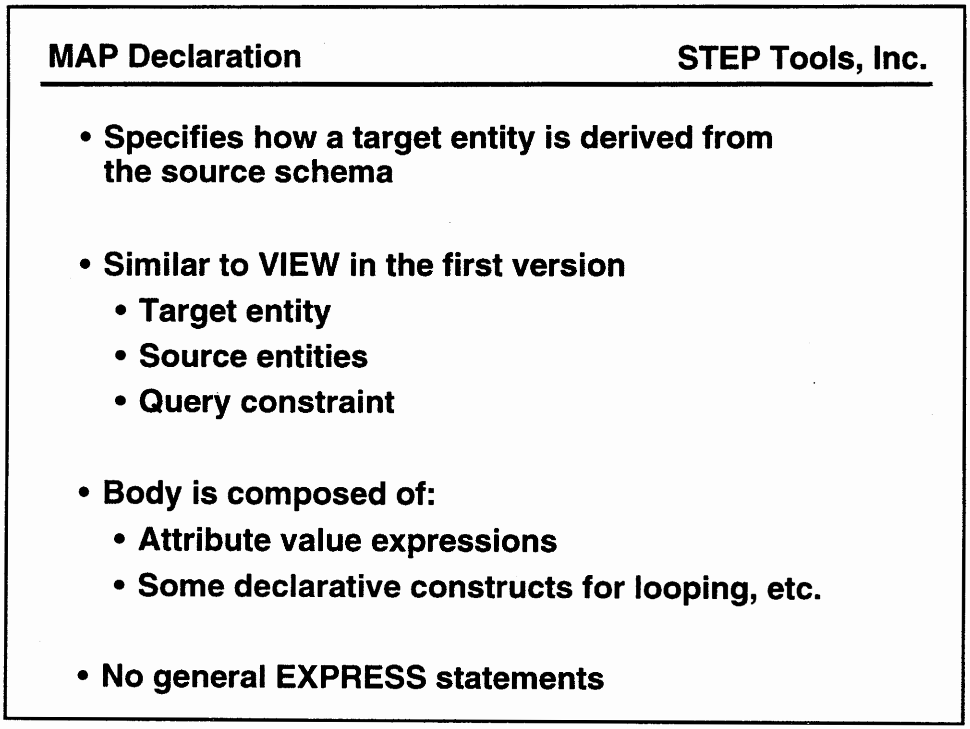
- A view can be derived in more than one way
- Each possibility is called a *partition*



Example STEP Tools, Inc.

- An “organization” is either a person, an organization, or a person within an organization

```
VIEW arm_organization
 PARTITION one :
   FROM (p : person)
     ...
 PARTITION two :
   FROM (o : organization)
     ...
 PARTITION three :
   FROM (po : person_and_organization)
     ...
 END_VIEW;
```



MAP Declaration STEP Tools, Inc.

- Specifies how a target entity is derived from the source schema
- Similar to VIEW in the first version
  - Target entity
  - Source entities
  - Query constraint
- Body is composed of:
  - Attribute value expressions
  - Some declarative constructs for looping, etc.
- No general EXPRESS statements

MAP Declaration

STEP Tools, Inc.

Other aspects of MAPS:

- UNIQUE clause enforces unique instantiation
- GROUP clause allows specifying multiple target entity instances
- Explicit binding and partitioning

Type Maps

STEP Tools, Inc.

- Specifies an implicit two-way conversion between defined types.

```
TYPE dmark = REAL; END_TYPE;

TYPE dollar = REAL; END_TYPE;

TYPE_MAP dmark FROM dollar;
  dmark := 1.5 * dollar;
  dollar := dmark / 1.5;
END_TYPE_MAP;
```

Other Stuff

STEP Tools, Inc.

Other aspects of the next version:

- VIEWs can be built on top of other VIEWs, and can be used in the FROM clause of MAPs
- No COMPOSE; use explicit binding instead
- There will be some kind of "inheritance" allowed between MAP declarations

Conclusion

STEP Tools, Inc.

When will EXPRESS-X solve all my problems?

- One year development cycle:

• Summer 1997	First version
• Summer 1998	Second version (CD?)
• Summer 1999	DIS? Third version?

- Implementations are available now
  - Both pilot projects and commercial users are experimenting with the language
- User feedback has been critical to the project