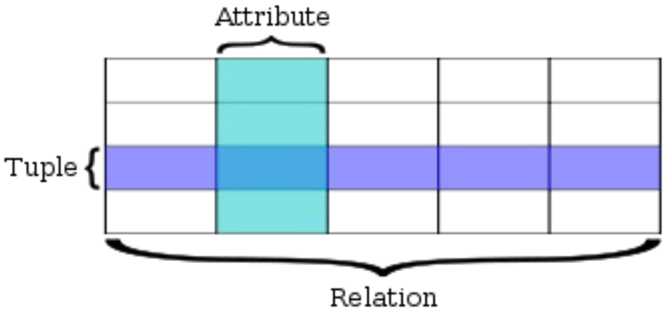
Lore: A Database Management System for Semistructured Data

Authors from Stanford University

Introduction

- Traditional DBMS are relational
- Force data to adhere to rigid schema

- Data can be irregular (null values?)
- Difficult to decide in advance of single correct schema
- Where should we store unstructured data?!



Rigid schema - RDBMS

In Comes Lore

- Takes advantage of structure where it exists
- Handle irregular/unstructured data gracefully
- Uses the OEM (Object Exchange Model) as the data model
- Uses the Lorel query language
- Uses DataGuides in place of a standard schema

Stanford University

This Presentation

- OEM (Object Exchange Model)
- The Lorel Query Language (and OQL)
- High Level System Architecture
- Query Plans and Data Flow
- Query Operators
- Query Plan Construction
- Query Optimization and Indexing
- Index and Update Query Plans
- Physical Storage
- Data Guides

The Object Exchange Model

- Labeled directed graph
- Atomic objects- leaf/edge vertices
- Complex objects - vertices with outgoing edges
- *Names* serve as entry points

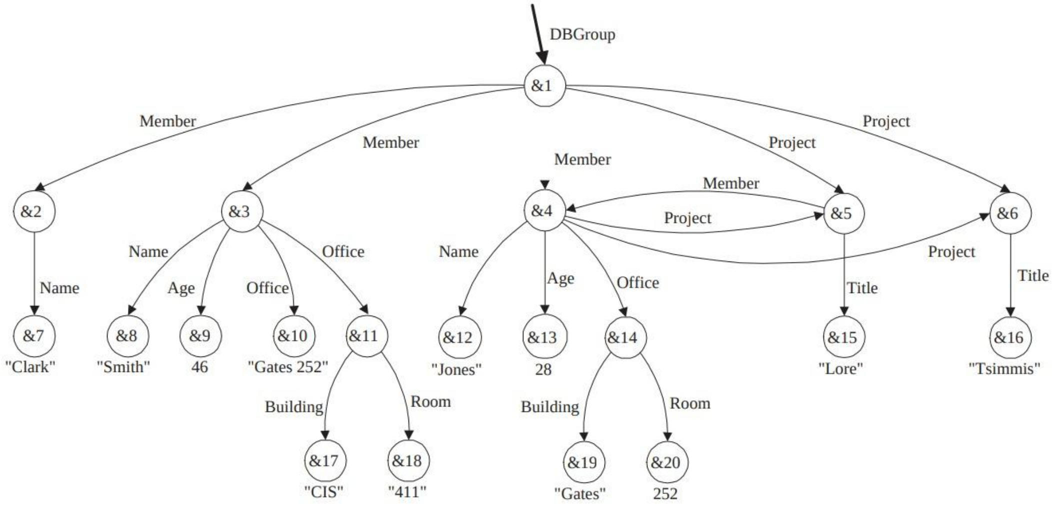


Figure 1: An OEM database

The Lorel Query Language

- Simple Path Expressions
  - DBGroup.Member.Office

```
QUERY
   select DBGroup.Member.Office
   where DBGroup.Member.Age > 30
```

- Rewritten into OQL style

```
select O
from DBGroup.Member M, M.Office O
where exists A in M.Age : A > 30
```

The Lorel Query Language

```
QUERY
   select DBGroup.Member.Office
   where DBGroup.Member.Age > 30
```

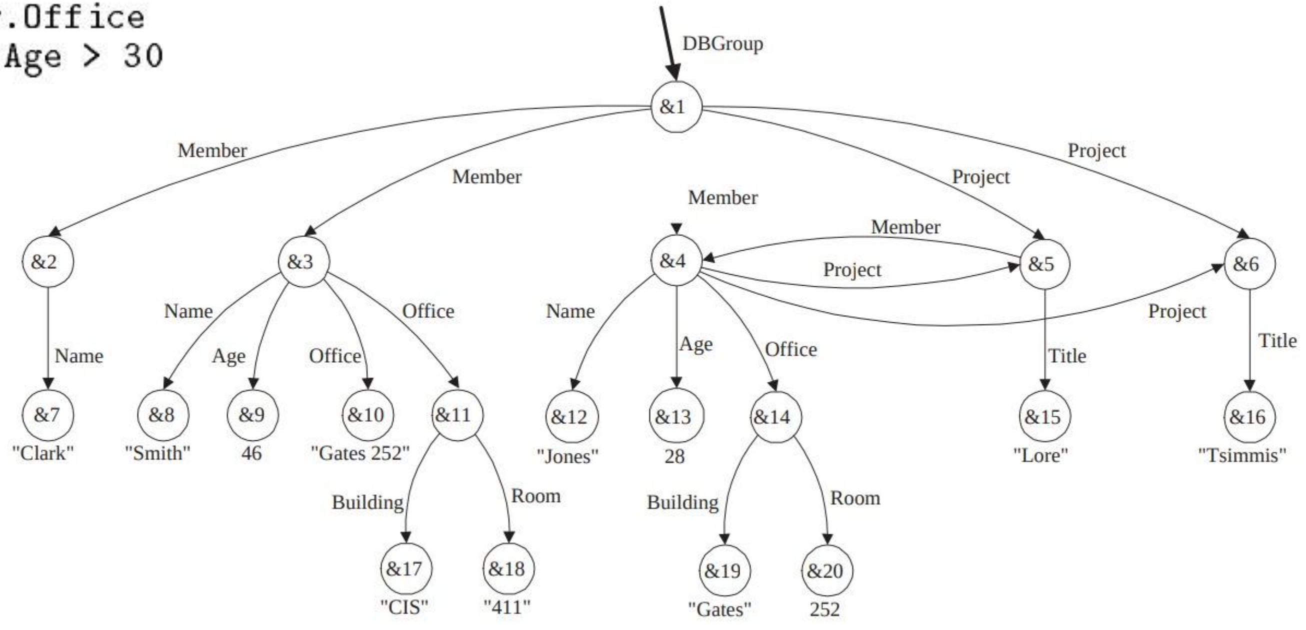


Figure 1: An OEM database

```
RESULT
   Office "Gates 252"
   Office
      Building "CIS"
      Room "411"
```

The Lorel Query Language

- Complex Path Expressions

```
QUERY
 select DBGroup.Member.Name
 where DBGroup.Member.Office(.Room%|.Cubicle)?
          like "%252"

RESULT
   Name "Jones"
   Name "Smith"
```

The Lorel Query Language

- Subqueries

```
QUERY
  select M.Name,
         ( select M.Project.Title
           where M.Project.Title != "Lore" )
  from DBGroup.Member M
  where M.Project.Title = "Lore"

RESULT
  Member
      Name "Jones"
      Title "Tsimmis"
```

The Lorel Query Language

- Updates
  - insertion/removal of edges
  - creation of vertices
  - modifications of atomic values
  - modifications of name assignments
  - **no object deletion** (handled by garbage collector)

```
update P.Member +=
    ( select DBGroup.Member
      where DBGroup.Member.Name = "Clark" )
from DBGroup.Project P
where P.Title = "Lore" or
      P.Title = "Tsimmis"
```

High Level System Architecture

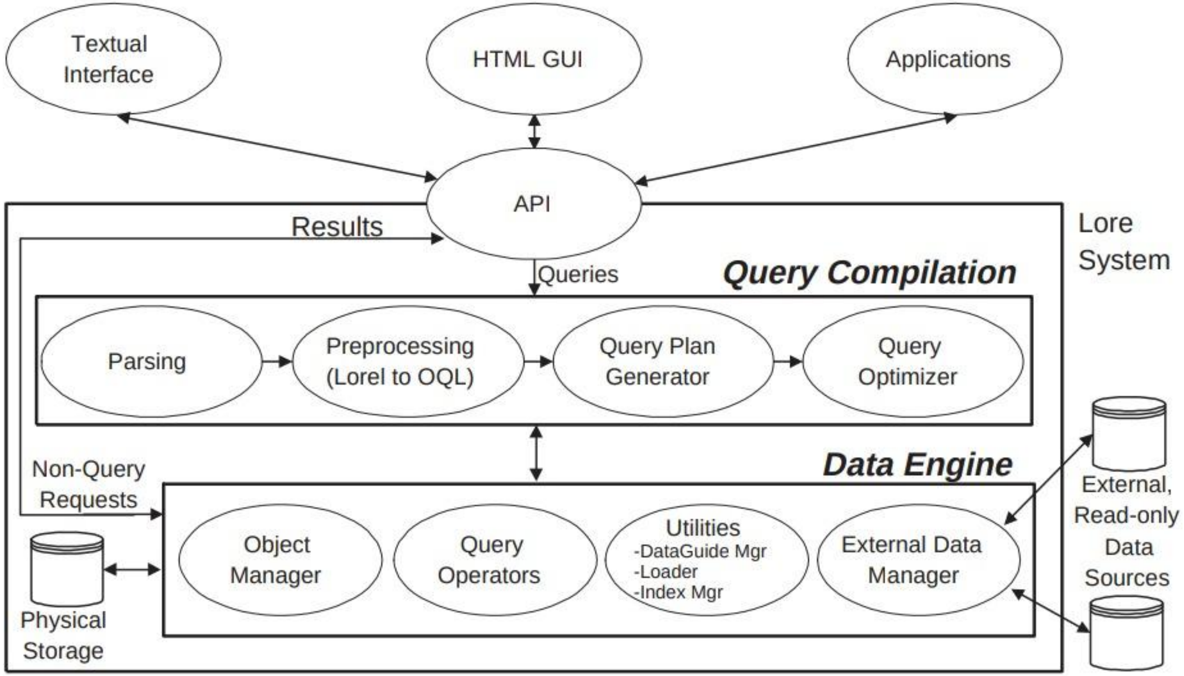


Figure 2: Lore architecture

Query Plans and Data Flow

```
QUERY
   select DBGroup.Member.Office
   where DBGroup.Member.Age > 30
```

- Execution begins at the top
- Iterator approach avoid creation of temporary relations
- Each OA slot holds the oid of a vertex

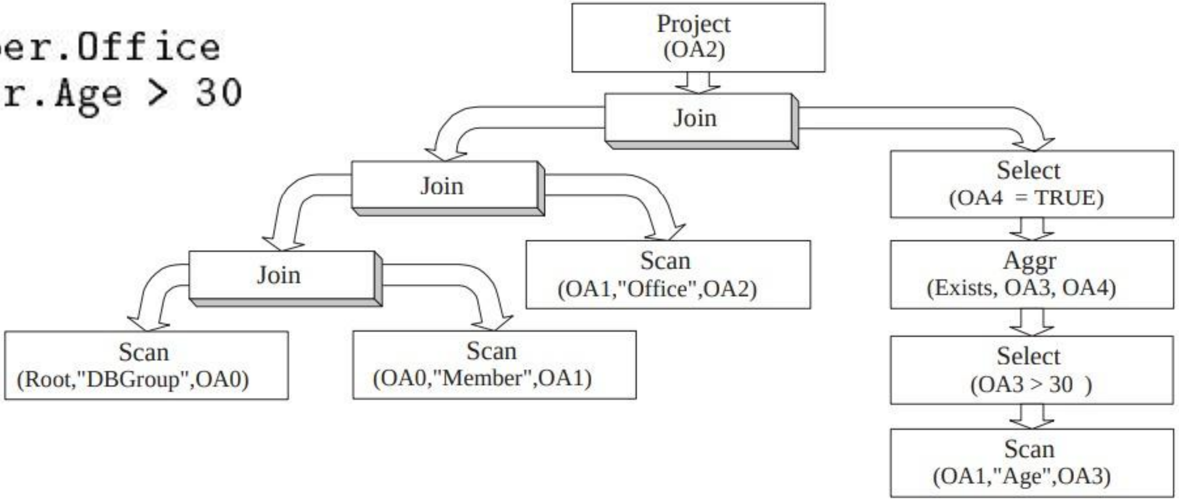


Figure 3: Example Lore query plan

OA0	OA1	OA2	OA3	OA4
(DBGroup)	(OA0.Member)	(OA1.Office)	(OA1. Age)	(true/false)

Figure 4: Example object assignment

Query Operators

```
QUERY
   select DBGroup.Member.Office
   where DBGroup.Member.Age > 30
```

- Scan
- Join
- Select
- Aggregate
- Project

- SetOp
- ArithOp
- CreateSet
- GroupBy

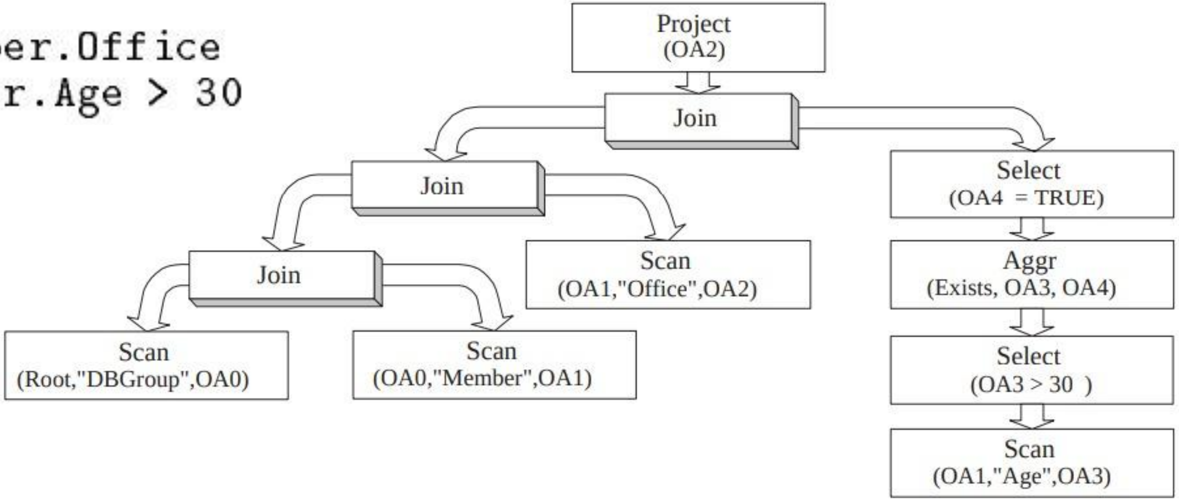


Figure 3: Example Lore query plan

OA0	OA1	OA2	OA3	OA4
(DBGroup)	(OA0.Member)	(OA1.Office)	(OA1.Age)	(true/false)

Figure 4: Example object assignment

Query Plan Construction

```
select M.Name, count(M.Publication)
from DBGroup.Member M
where M.Dept = "CS"
```

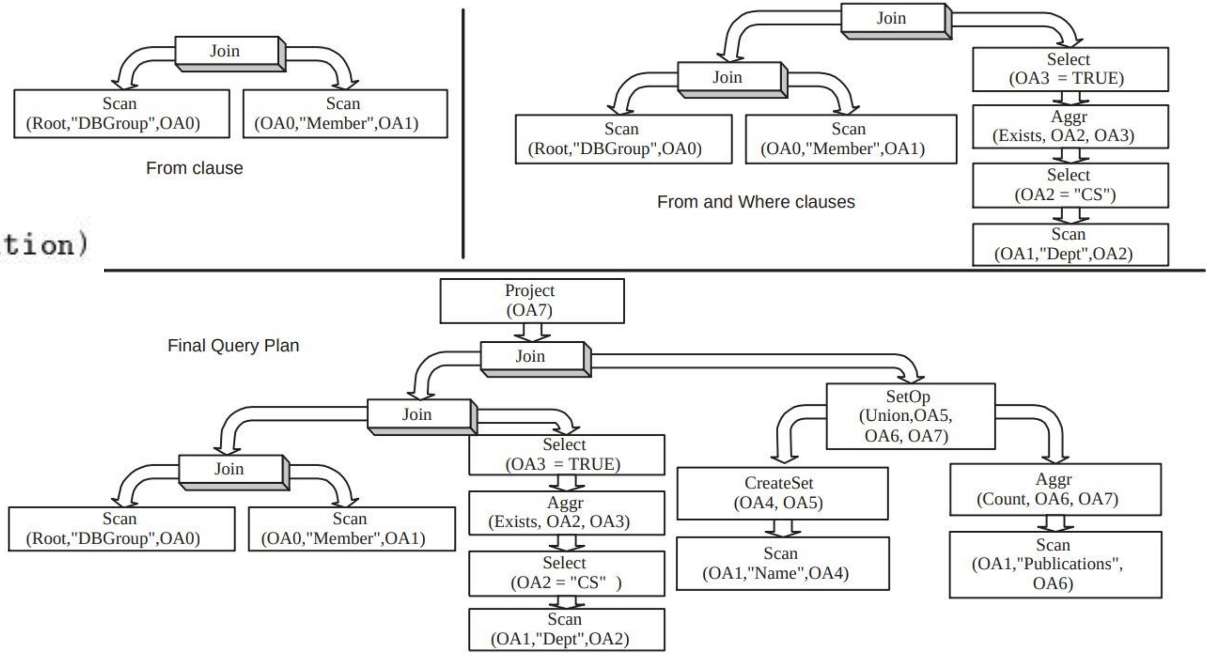


Figure 5: Steps in constructing a query plan

Query Optimization and Indexing

- Lacks sophisticated query planning
- Selections are pushed down
- Two types of indexes:
  - Lindex (Parent Link Index)
  - Vindex (Value Index)
- Lindexes implemented using linear hashing
- Vindexes implemented using B+-Trees

arg2 / *arg1*	*string*	*real*	*int*
string	–	*string* → *real*	*both* → *real*
real	*string* → *real*	–	*int* → *real*
int	*both* → *real*	*int* → *real*	–

Table 1: Coercion for basic comparison operators

Index Query Plans

```
QUERY
   select DBGroup.Member.Office
   where DBGroup.Member.Age > 30
```

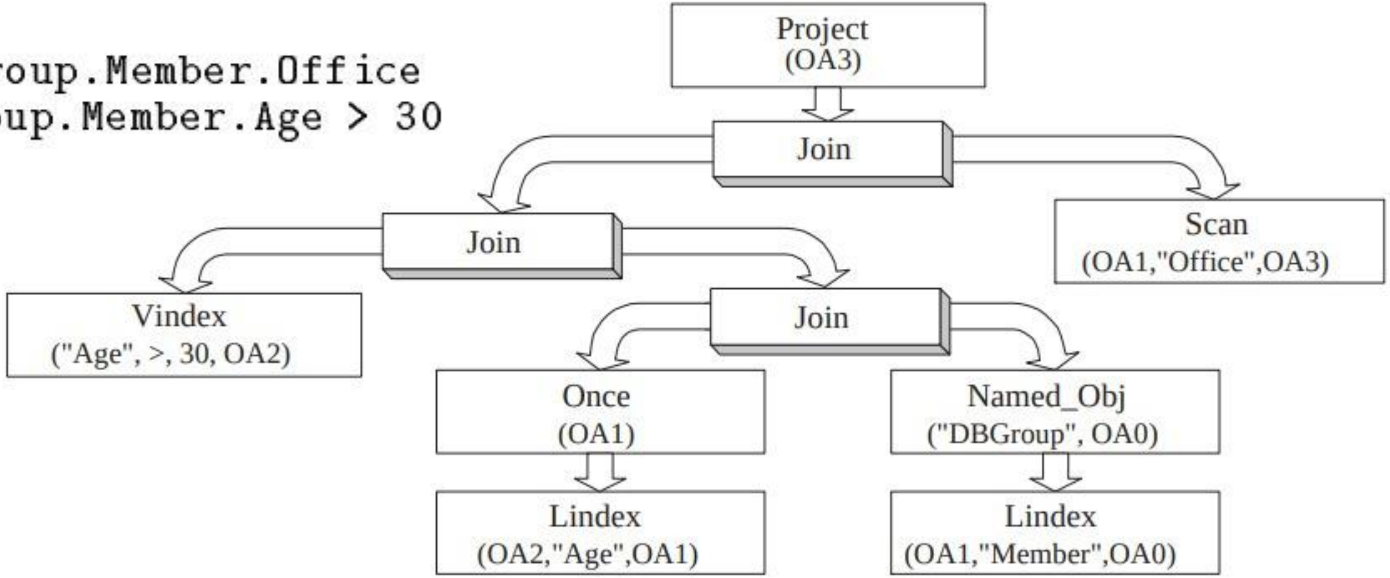


Figure 6: A query plan using indexes

Update Query Plans

```
update P.Member +=
   ( select DBGroup.Member
     where DBGroup.Member.Name = "Clark" )
from DBGroup.Project P
where P.Title = "Lore" or
      P.Title = "Tsimmis"
```

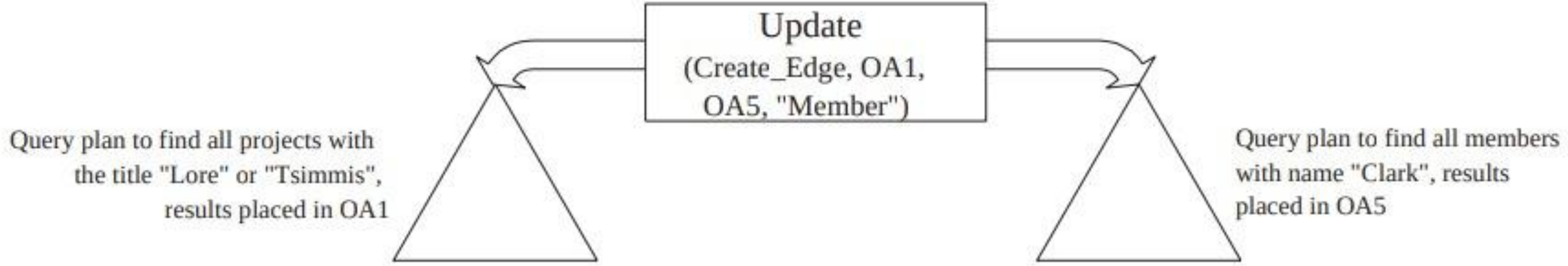


Figure 7: Example update query plan

Physical Storage

- Each page on disk has slots
- One object in each slot
- First-fit algorithm used
- Object forwarding mechanism
- Large objects span many pages
- Object clustering is depth first
- Garbage collector for orphans
- External data also supported

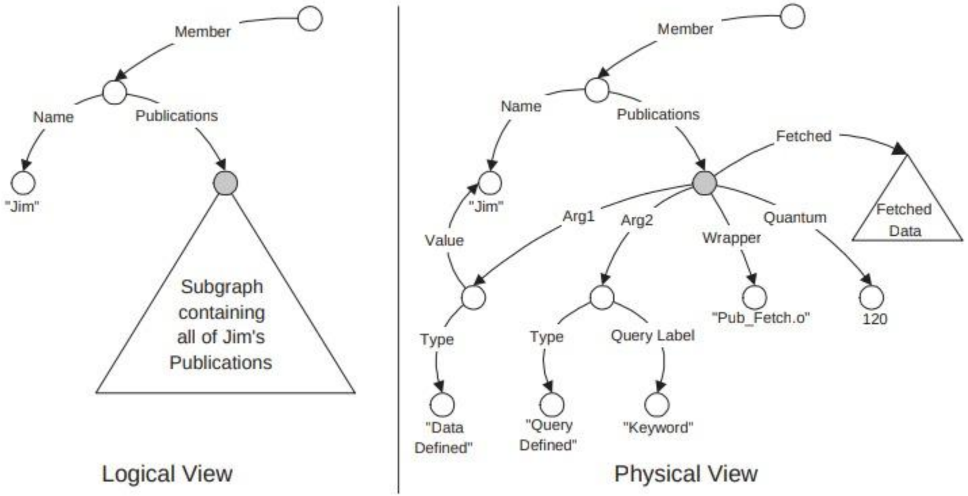


Figure 8: The logical and physical views of the data

Data Guides

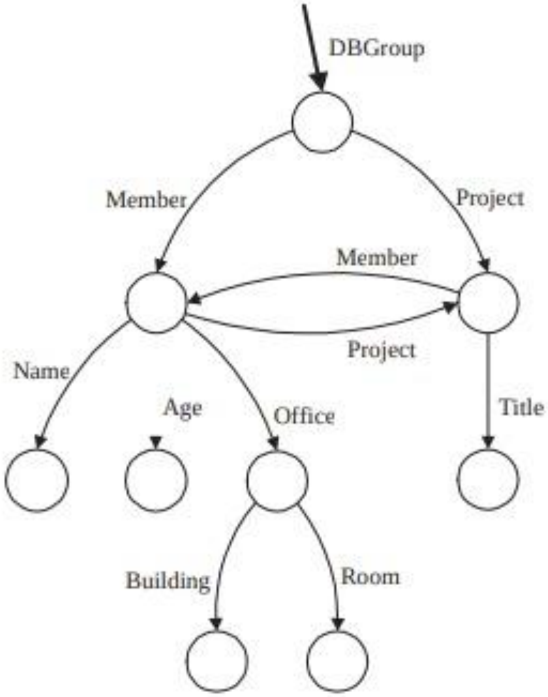


Figure 9: A DataGuide for Figure 1

Thank You! Questions?