THE ODMG OQL OPTIMISATION

Object vs. Relational Query Optimization

- What is a Join in the Object Context?
  - Joints between Multi-sorted Collections
  - Dependent Joins: navigating through multivalued attributes
  - Implicit Joins: navigating through complex-value attributes
  - Navigation vs Joins: indexing and clustering of objects

- The issue of Object Methods
  - Methods may have side effects
  - Methods’ cost is unknown
  - Late binding of methods’ body
Joins between Multi-sorted Collections

- The collections in the from clause are not always collections of n-arity relations
  - collections of objects or literals, collections of collections, etc.
- Solutions
  a) The joins add a level of nesting between the n-arity relations ([Shaw & Zdonic DOOD89])

\[
C_1 \bowtie (C_2 \bowtie C_3) = \{[T_1, [T_2, T_3]]\}
\]

- The join loses its associative nature

b) Dereferencing objects before a join ([Venderberg & DeWitt SIGMOD91])

- Not always collections of objects
- Loss of object identity (methods' calls ?)

Dependent Joins

- The collections in the from clause are not always distinct

\[
\text{select } v \quad \text{from} \quad \text{COMPANIES } c, \ c->\text{employees } e, \ e->\text{cars } v
\]

- The Cartesian product does not always express the dependence of the variables domains
- Solution: Introduction of a Dependent Join (Djoin) operator where the evaluation of the second argument (e.g., \(e\)) depends on the evaluation of the first (e.g., \(c\))
Implicit Joins

- **Path Expressions** are used to navigate through complex values instead of associative access (joins)
  
  ```
  select u->president->name
  from UNIVERSITIES u
  where u->president->address->city = "Paris"
  ```

- Solution: Deconstruction and regrouping of path expressions using joins ([Blakeley McKenna Graefe SIGMOD93])
  
  1. \(\sigma_{\text{president.address.city}} = \text{"Paris"} \) UNIVERSITIES
  
  2. UNIVERSITIES \( \bowtie \) PRESIDENTS \( \bowtie \) \(\sigma_{\text{city}} = \text{"Paris"} \) ADDRESSES
  
  3. UNIVERSITIES \( \bowtie \) \(\sigma_{\text{address.city}} = \text{"Paris"} \) PRESIDENTS

Object Navigation and Indexing

- Paths may involve multiple collections qualified by predicates

- Different types of indices for paths of length > 1 ([Bertino Kim TKDE89])
  - Used for selections but sometimes partial
  - Used to navigate at lower cost
  - Used for joins as in the relational world
Object Navigation and Clustering

- Different strategies for object clustering ([Bertino Saad Ismail 95]
  - Clustering by composition: it is better in order to avoid joins
  - Clustering by class: less I/O for joins
  - Dynamic Clustering: navigation ?
  - Random Clustering : ?

Object Methods

- The methods can have side effects
  - The re-ordering of operations may cause false results
- Solution: classification of methods in two groups: with and without side effects
  - Pb: the modification of one method implies the re-compilation of all the methods to maintain the dependence graph
- The cost of methods is unknown
  - The standard optimizations (of the relational model) are not valid
- Solutions:
  a) The cost of methods is provided by the programmer and used by the optimizer ([Hellerstein Stronebraker SIGMOD93])
  b) Pre-evaluation of the methods results ([Kemper Moerkotte SIGMOD91])
Object Methods

- The late resolution of methods
  - The implementation of the method is not known until the execution of the queries
- Solution: Algebraic form of the method’s code ([Graefe Maier TR Oregon 89])
  - Sometimes impossible: the code is very complex
  - Sometimes partially possible: detection of the part of the code that is common to different implementations of the method

I) ALGEBRAISATION OF OQL QUERIES
Preprocessing of Queries

- Find the presidents of companies who reside in Paris and the employees of the same company driving a car manufactured by “Peugeot”

  ```sql
  select struct(pres:c->president, empl:e->name) 
  from COMPANIES c, c->employees e, e->cars v 
  where c->president->address->city = "Paris" and 
  v->manufacturer->name = "Peugeot"
  ```

- Factorization of constant expressions and their common sub-expressions

- Evaluation of dependencies
Translation Of The From Clause

- **COMPANIES c → COMPAGNIES[c]**
  - The result is {{c:c1}, [c:c2] ...}
  - This transformation allows standard joins (relational) and preserves the identity of objects
  - The transformation is logical and at the physical level may correspond to a collection scan

- **COMPAGNIES c, c->employees e, e->cars v →**
  - COMPAGNIES[c] <c->employees[e] > <e->cars[v]>
  - The result is {{c:c1, e:e1, v:v1}, [c:c1, e:e1, v:v2], ...}
  - The transformation of dependent joins to standard joins is not always possible (class extensions)

Translation Of The Define Clause

- **cp=c->president, cpac=cp->address->city, en=e->name... →**
  - ... Map en : e->name (Map cpac:cp->address->city (Map cp:c->president (COMPAGNIES[c] <c->employees[e] > <e->cars[v]>))))
  - The result is {{en:'toto', cpac:'San Diego', cp:p1, c:c1, e:e1, v:v1} ...}
  - The Map operations correspond to operations non interpretable (e.g. for the factorization) by the algebra
  - The optimizer may decide on the deconstruction and the regrouping of nested path expressions in the Map operations using the typing information (class extensions)
Translation Of The Where Clause

- \( ... c \rightarrow \text{president} \rightarrow \text{address} \rightarrow \text{city} = \text{“Paris”} \)
  and \( v \rightarrow \text{manufacturer} \rightarrow \text{name} = \text{“Peugeot”} \) \(
\sigma\ c_{\text{president}} = \text{“Paris”} \) and \( v_{\text{manufacturer}} = \text{“Peugeot”} \)

\[ \left( \text{Map}_{c_{\text{president}} : c \rightarrow \text{address} \rightarrow \text{city}} \left( \text{Map}_{c : c \rightarrow \text{president}} \left( \text{COMPAGNIES}[c] <c \rightarrow \text{employees}[e] > <e \rightarrow \text{cars}[v] > \right) \right) \right) \]

- Standard selections (relational)

Translation Of The Select Clause

- \( ... \text{struct}(\text{pres} : c \rightarrow \text{president}, \text{empl} : e \rightarrow \text{name}) \) \(
\text{Map}_{\text{pres} : c \rightarrow \text{president}, \text{empl} : e \rightarrow \text{name}} \left( \sigma\ c_{\text{president}} = \text{“Paris”} \) and \( v_{\text{manufacturer}} = \text{“Peugeot”} \) \( \text{COMPAGNIES}[c] <c \rightarrow \text{employees}[e] > <e \rightarrow \text{cars}[v] > \right) \)

- The Map operation may be transformed to a standard projection with the correct renaming of the variables (attributes)
Translation Of The **Group Clause**

- ... **group by** city: u->address->city ➔
  
  \[ \Gamma_{\text{partition}; \text{city}}; (\text{Map } u \rightarrow \text{address} \rightarrow \text{city} \ldots) \]

- The result is \{[city:'San Diego',partition:{{name:'UCSD',address:a1, ...] ...}}] ...\}

Translation Of The **Order Clause**

- ... **order by** u->address->city, u->name ➔
  
  \[ S_{u \rightarrow \text{address} \rightarrow \text{city}, u \rightarrow \text{name}} (\text{UNIVERSITIES}[u]) \]

- The result is <u1, u2,...>
**Formal Definitions**

select $f$
from $f_1^{1} x_1^{1} f_2^{1} x_2^{1} \ldots f_n^{1} x_n^{1}$
where $p(f_1^{2}, f_3^{2}, \ldots, f_n^{2})$
group by $f_1^{3} x_1^{2} f_2^{3} x_2^{2} \ldots f_n^{3} x_n^{2}$
having $p'(f_1^{4}, f_2^{4}, \ldots, f_n^{4})$
order by $f_1^{5}, f_2^{5}, \ldots, f_n^{5}$

$F = f_1^{1}[x_1^{1}] <f_2^{1}[x_2^{1}]> \ldots <f_n^{1}[x_n^{1}]>$

$W = \sigma \ p(w_1, \ldots, w_n2) \ (Map \ w_1:f_1^{2}, \ldots, w_n:f_n^{2}(F))$

$G = \Gamma \ \text{partition;} x_1^{2}, \ldots, x_n^{2} \ \{Map \ x_1^{2}:f_1, \ldots, x_n^{2}:f_n^{2}(W)\}$

$H = \sigma \ p(h_1, \ldots, h_n) \ (Map \ h_1:f_1^{3}, \ldots, h_n:f_n^{3}(G))$

$O = S \ (o_1, \ldots, o_n) \ (Map \ o_1:f_1^{4}, \ldots, o_n:f_n^{4}(H))$

$S = Map \ f \ (O)$

---

**II) ALGEBRAIC EQUIVALENCES**
D-joins and Joins

1. COMPAGNIES[c] <c->employees[e]>

2. COMPAGNIES[c] <σ_e in c->employees EMPLOYEES[e]>

3. σ_e in c->employees (COMPAGNIES[c] <EMPLOYEES[e]>)

4. COMPAGNIES[c] e in c->employees EMPLOYEES[e]

For this transformation the typing information is used.

Map and Joins

1. Map_{c:p, c->president}(COMPAGNIES[c])

   COMPAGNIES[c] =nil \n_{c->president = p} PRESIDENTS[p]

2. An external join is necessary if there exist companies that do not have a president. In this way, the relevant companies remain in the result and value for their president is nil.

3. In the case where all the companies have one president a simple join is sufficient.
**Relationships Between Map Operations**

- \( \sigma_{\text{vmn} = \text{"Peugeot"}} (\text{Map}_{\text{vmn} : \text{vm} - > \text{name}} (\text{Map}_{\text{vm} : \text{v} - > \text{manufacturer}} (\ldots))) \)

- If the operations before the Map do not (directly) depend on the variables involved in the Map
  \[ \rightarrow \sigma_{\text{vmn} = \text{"Peugeot"}} (\text{Map}_{\text{vmn} : \text{v} - > \text{manufacturer} - > \text{name}} (\ldots)) \]

- In the opposite case, the factorization is necessary
  \[ \rightarrow \text{Map}_{\text{[pres:cp, empl:en]}} (\sigma_{\text{cpac} = \text{"Paris"}} (\text{Map}_{\text{cpac:cp} - > \text{address} - > \text{city}} (\text{Map}_{\text{cp:cp} - > \text{president}} (\ldots)))) \]

- The Map operations may be re-ordered according to the dependency of the variables

**Relationships between Selections**

- \( \sigma_{\text{cpac} = \text{"Paris"} \text{ and } \text{vmn} = \text{"Peugeot"}} (\text{Map} (\ldots)) \)
  \[ \rightarrow \sigma_{\text{cpac} = \text{"Paris"}} (\sigma_{\text{vmn} = \text{"Peugeot"}} (\text{Map} (\ldots))) \]

- Indices on the corresponding collections may be used for the evaluation of selections

- When there exist more than one indices on the same collection the best combination of the indices defined on the same path is used
Selections and Joins

- $\sigma_{\text{cpac} = "Paris"}$ ($\sigma_{\text{vnm} = "Peugeot"}$ ($\text{Map}_{\text{cpac}}$ $\text{Map}_{\text{vnm}}$

- $((\text{COMPANIES}[c]\bowtie_{e \in \text{c->employees}} \text{EMPLOYEES}[e])$

- $\bowtie_{v \in \text{e->cars} \text{VEHICLES}[v])))$)

$\Rightarrow \sigma_{\text{cpac} = "Paris"}$ ($\text{Map}_{\text{cpac}}$

- $\text{COMPANIES}[c]\bowtie_{e \in \text{c->employees}} \text{EMPLOYEES}[e]$

- $\bowtie_{v \in \text{e->cars}} (\sigma_{\text{vnm} = "Peugeot"}$ ($\text{Map}_{\text{vnm}}$ $\text{VEHICLES}[v])))$)

- A condition of the selection may become condition of the join and vice-versa

- Transformation of joins representing semi-joins on the selections (extensive tests)

Relationships Between Joins

- $\cdots (\sigma_{e = u->president \text{ and } u->address->city="Paris"}$ $(\text{COMPANIES}[c])$

- $\bowtie_{c->employees[e]} \bowtie_{c->UNIVERSITIES[u]}$)$

1. $\cdots ((\text{COMPANIES}[c]\bowtie_{e \in \text{c->employees}} \text{EMPLOYEES}[e])\bowtie_{e = u->president} (\sigma_{u->address->city="Paris"} \text{UNIVERSITIES}[u]))$

2. $((\text{COMPANIES}[c]\bowtie_{e \in \text{c->employees}} \exists e \in \text{c->employees} : e = u->president$

$\sigma_{u->address->city="Paris"} \text{UNIVERSITIES}[u]))$

3. $((\sigma_{u->address->city="Paris"} \text{UNIVERSITIES}[u])\bowtie$

$\exists e \in \text{c->employees} : e = u->president \text{COMPANIES}[c])$
Conclusion

- The Query Optimizer is a key component of an ODBMS
- Relational techniques can be generalized
  - Object algebra
  - Cost model
  - Search strategies
- New techniques are required for:
  - Extensibility of data types
  - Optimizing path expressions
  - Optimizing method calls
  - Optimizing bulk data types
  - Path index maintenance and access

VII) REFERENCES

- J. Blakeley and W. McKenna and G. Graefe: "Experience Building the OODB Query Optimizer". In SIGMOD 1993
- S. Vandenberg and D. DeWitt: "Algebraic Support for Complex Objects with Arrays, Identity and, Inheritance". In SIGMOD 1991
- D. Straube and T. Ozsu: "Queries and Query Processing in Object-Oriented Database Systems". In ACM Trans. on Office Information Systems, 8(4), 1990