Programs Offered

Post Graduate Programmes (PG)

- Master of Business Administration
- Master of Computer Applications
- Master of Commerce (Financial Management / Financial Technology)
- Master of Arts (Journalism and Mass Communication)
- Master of Arts (Economics)
- Master of Arts (Public Policy and Governance)
- Master of Social Work
- Master of Arts (English)
- Master of Science (Information Technology) (ODL)
- Master of Science (Environmental Science) (ODL)

Diploma Programmes

- Post Graduate Diploma (Management)
- Post Graduate Diploma (Logistics)
- Post Graduate Diploma (Machine Learning and Artificial Intelligence)
- Post Graduate Diploma (Data Science)

Undergraduate Programmes (UG)

- Bachelor of Business Administration
- Bachelor of Computer Applications
- Bachelor of Commerce
- Bachelor of Arts (Journalism and Mass Communication)
- Bachelor of Arts (General / Political Science / Economics / English / Sociology)
- Bachelor of Social Work
- Bachelor of Science (Information Technology) (ODL)

AMITY UNIVERSITY | DIRECTORATE OF DISTANCE & ONLINE EDUCATION
Amity Helpline: 1800-102-3434 (Toll-free), 0120-4614200
For Distance Learning Programmes: dladmissions@amity.edu | www.amity.edu/addoe
For Online Learning programmes: elearning@amity.edu | www.amityonline.com

Product code
81030112000594

Advanced Database Management Systems

AMITY

Advanced Database Management Systems

AMITY UNIVERSITY | DIRECTORATE OF DISTANCE & ONLINE EDUCATION

Advanced Database Management Systems

AMITY UNIVERSITY | DIRECTORATE OF DISTANCE & ONLINE EDUCATION

SLM & Learning Resources Committee

Published by Amity University Press for exclusive use of Amity Directorate of Distance and Online Education, Amity University, Noida-201313

Contents

Module - I: Query Processing

Notes

Key Learning Objectives :

At the end of this module, you will lean:

- Basic Concept about Query Processing.
- How to Convert SQL Queries into Relational Algebra.
- Basic Algo Used for Query Processing

Notes

Unit - 1.1: Basic concepts of Query Processing

Main aim of query processing is to transform the query written in high –level language (Structured Query Language) into low –level language (implementing relational algebra) with efficient , correct and sequential manner for retrieving the data from the database as per the requirement. One of the most important aspect of query processing is query optimisation (select an efficient execution strategy for processing a query). Query processing mainly divided into four phase.

- Decomposition
- Optimisation
- Code generation
- Execution.

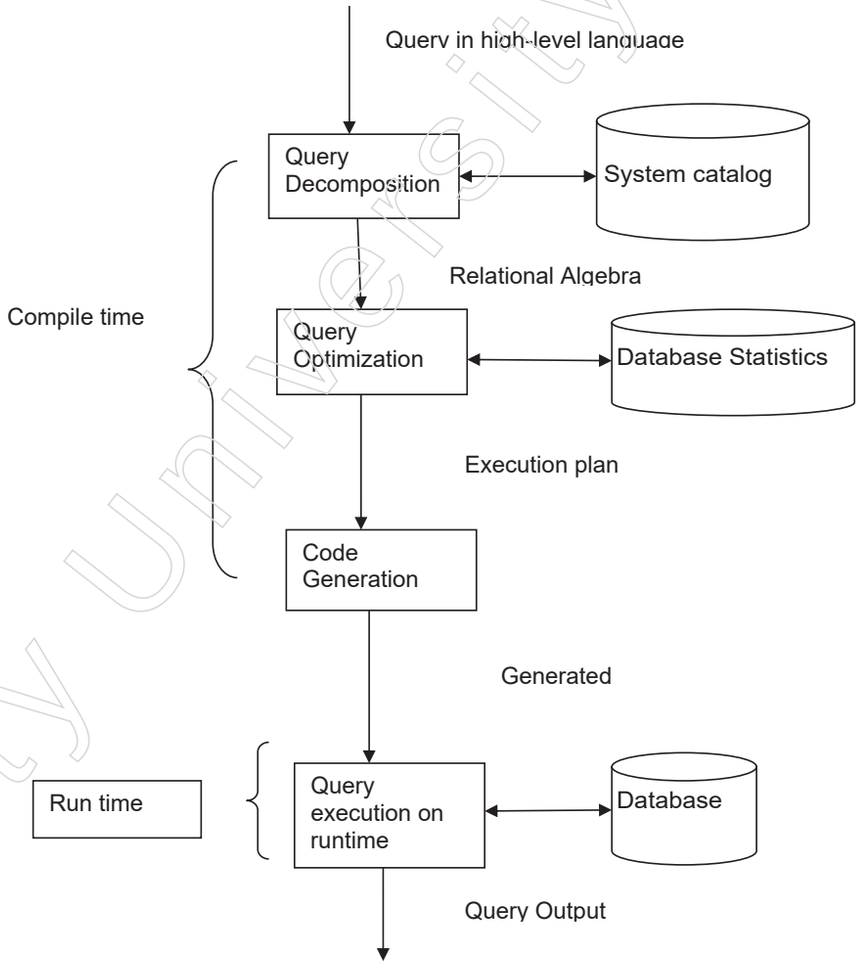


Fig 1.1.1 Query Processing Procedure

Decomposition: That is the first phase of query processing, here the high level language into a relational algebra query and check whether the query syntactically and semantically correct or not. The main phase of the decomposition are

- Analysis (Programming Compiler analysed the lexical and syntactical correctness of query)

Notes

- Normalisation (Convert query into normalised form for easy manipulation)

  Two Types

  - Conjunctive Normal Form (connected with AND operator)
  - Disjunctive Normal form (Connected with OR operator)
- Semantic analysis (Reject incorrectly normalised query)

  Mainly by using two graphical techniques

  - Relation connected graph (showing the incorrect query).
  - Normalised attribute connection graph (showing query is contradictory)
- Simplification
  - Detect redundant qualifications
  - Eliminate common sub expressions
  - Transfer the query into semantically equivalent
  - Access restriction
  - View Definitions
  - Integrity Constraints
- Query Restructuring (Query is restructured for providing efficient implemantaion).

Notes

Unit - 1.2: Converting SQL Queries into Relational Algebra

Basic SQL Relational Algebra Operations

Relational Algebra devided in various groups

- Unary Relational Operations
  - SELECT (symbol: σ)
  - PROJECT (symbol: π)
  - RENAME (symbol: ρ)
- Relational Algebra Operations From Set Theory
  - UNION (υ)
  - INTERSECTION (∩),
  - DIFFERENCE (-)
  - CARTESIAN PRODUCT (x)
- Binary Relational Operations
  - JOIN
  - DIVISION

Let's study them in detail with solutions:

SELECT (σ)

The SELECT operation is used for selecting a subset of the tuples according to a given selection condition. Sigma(σ)Symbol denotes it. It is used as an expression to choose tuples which meet the selection condition. Select operator selects tuples that satisfy a given predicate.

$\sigma_p(r)$

σ is the predicate.

r stands for relation which is the name of the table.

p is prepositional logic.

Basic Relational Algebra Operations (Unary): Selection, Projection.

Picks tuples from the relation

Selection: s <condition(s)> (<relation>)

Picks columns from the relation

Projection: π <attribute-list> (<relation>)

Relational Algebra Operations (Set): Union, Set Difference

Union: (<relation>) U (<relation>)

Notes

New relation contains all tuples from both relations, duplicate tuples eliminated.

Set Difference: R – S

Produces a relation with tuples that are in R but NOT in S.

Relational Algebra Operations (Set): Cartesian Product, Intersect

Cartesian Product: R x S

Produces a relation that is concatenation of every tuple of R with every tuple of S

The Above operations are the 5 fundamental operations of relational algebra.

Intersection: R ∩ S

All tuples that are in both R and S

Relational Algebra Operations (Join): Theta Join, Natural Join

Theta Join: R ⋈ F S = s F (R x S)

Select all tuples from the Cartesian product of the two relations, matching condition F

When F contains only equality "=", it is called Equijoin

Natural Join: R ⋈ S

Equijoin with common attributes eliminated

Relational Algebra Operations:

Division

Standard language for querying and manipulating data

Structured Query Language

Relational algebra queries to SQL queries

- FROM clause produces Cartesian product (x) of listed tables
- WHERE clause assigns rows to C in sequence and produces table containing only rows satisfying condition (sort of like s)
- SELECT clause retains listed columns (P)

Example Relational Algebra Translation to SQL

Though it is tough to convert a high level language (SQL) into a low level relational algebra but I try here to define with logic and example of few. To overcome from ambiguities, for relation symbols in a SQL statement assigned a distinct name through the alias (SQL aliases are used to give a table, or a column in a table, a temporary name. Aliases are often used to make column names more readable. An alias only exists for the duration of the query.) mechanism of SQL. In other.

The SQL statements in where a relation symbol occurs multiple times, for example,

Notes

SELECT * FROM R, R is rewritten into a SQL statement of the form

SELECT * FROM R, R R1

Here every occurrence is given a distinct (alias) name.

1. Select-from-where statements without sub queries

Consider a general SELECT-FROM-WHERE statement of the form

SELECT Select-list

FROM R1, . . . , R2 T1, . . .

WHERE (Where-condition)

Science here query does not use sub queries in where-condition then it can be translate, into the relational algebra as follows:

π Select-list σWhere-condition(R1 X ……..X ρT1(R2) _ _ _ _)

Note: Here an alias R2 T1 in the FROM-clause corresponds to a renaming ρT1(R2).

If there is no WHERE clause then there no need to include the selection σ in the expression.

For omitting the projection (π) we obtain the translation of the following

special case:

SELECT *

FROM R1, . . . , R2 T1, . . .

WHERE Where-condition

E.g.: SQL SELECT-FROM-WHERE statement is

SELECT Cinema_Name

FROM Actors_In, Cinema_Actor

WHERE Actor_Name = name AND date of birth = 1980

Translating relational algebra like

πCinema_Name**σ**Actor_Name=name (Actors_In X Cinema_Actor):

^ date of birth =1980

Example with Sub queries : The SQL query is

SELECT LastName, FirstName

FROM EMPLOYEE

Notes

WHERE Salary >(SELECTMAX (Salary)

FROM EMPLOYEE

WHERE IdNo = 2);

It can be split into two following Sub queries like

- **SELECT** LastName, FirstName

  FROM EMPLOYEE

  WHERE Salary > 10000

  Respective R.A expression like that

  πLastName, FirstName(**σ** Salary>10000(EMPLOYEE))

- **SELECT** MAX (Salary)

  FROM EMPLOYEE

  WHERE IdNo = 2

  Respective R.A expression like that

  ℱMAX Salary (**σ**IdNo = 2 (EMPLOYEE))

  Translating Joins

  (SELECT * FROM R R1) JOIN (SELECT * FROM R R1) ON R1.A = R2.B

  Ans: ρR1(R) ⋈ R1.A= R2.B R2(R) .

  Group and Having

SELECT Select-list

FROM From-list

WHERE Where-condition

GROUP BY Group-list

HAVINGHaving-condition

Ans: **π**A1;:::;An;Select-list**σ** Having-condition **ϒ** A1;:::;An;Group-list;Agg-list(E):

E.g.: SELECT name, SUM(length)

SQL

FROM Cinema Exce, Cinema

WHERE cert = SeniorProducer

GROUP BY name

Notes

HAVING MIN(year) <1960

R.A

Ans: **π**name;SUM(length) **σ**MIN(year)<1960**ϒ**name;MIN(year);SUM(length)

σ cert=SeniorProducer(CinemaExec**X** Cinema).

Nested loop:

R(A;B) and S(C):

SELECT R_1.A, R_1.B

FROM R R_1, S

WHERE EXISTS

(**SELECT** R_2.A, R_2.B

FROM R R_2

WHERE R_2.A = R_1.B AND EXISTS

(**SELECT** R_3.A, R_3.B

FROM R R_3

WHERE R_3.A = R_2.B AND R_3.B = S.C))

Total statement divided into three part where full query is query 1, middle sub query is query2 and inner sub query denoted by query3.

Ans: Now convert for the middle query that means query2 like that

SELECT *

FROM R R_2

WHERE R_2.A = R_1.B

Translation like:

σ R_2:A=R_1:B(ρR_2(R) X ρR_1(R))

Translate query 3 as follows:

σ R_3:A=R_2:B(ρR_3(R) X ρR_2(R) XS)

^R_3:B=S:C

Notes

Query3 already specifies for which tuples in R_2 and R_3 correcttuples in S exist . Let A_1……;A_pbe the list of all parameters of context relations of query. We

can translate EXISTS(query) by joining E with the \space of parameters" for

E query, namely _A_1……..A_p (Equery):

E := E ⋈ πA_1……..A_p (Equery):

Let us continue the translation of query 2:

E = ρR_2(R) X ρR_1(R)

Equery3 = σ R_3:A=R_2:B(ρR_3(R) X ρR_2(R) XS)

^R_3:B=S:C

Joining E and Equery3 on the parameters of query3 it

ensure that here link with" the correct tuples from E with the correct tuples of

E query3. The tuples in R_1 for which tuples in R_2, R_3,

and S exist that satisfy the requirements of query2 .

Simplify this expression

(ρR_1(R) X ρR_2(R)) ⋈ πR_2:A;R_2:B;S:C σR_3:A=R_2:B (ρR_3(R) X ρR_2(R) X S)

^R_3:B=S:C

and

ρR1(R) ⋈ πR_2:A;R_2:B;S:C σR_3:A=R_2:B (ρR_3(R) X ρR_2(R) X S)

^R_3:B=S:C

Union, Intersect, Except

consider the relations R(A;B)

and S(C), as well as the following query.

10

```
SELECT S_1.C, S_2.C
FROM S S_1, S S_2
WHERE EXISTS (
(SELECT R_1.A, R_1.B FROM R R_1
```

Notes

WHERE A = S_1.C AND B = S_2.C)

CROSS JOIN

(SELECT R_2.A, R_2.B FROM R R_2

WHERE B = S_1.C)

)

πS_1:C;S_2:C;R_1:A→A;R_1:B→B(E_1)Uπ[_S1:C;S2:C;R_2:A→A;R_2:B→B(E_2 X ρS_2(S))

E_1 = πS_1:C;S_2:C;R_1:A;R_1:B σ A=S_1:C(ρR_1(R) _X ρS_1(S) X ρS2(S))

^B=S_2:C

E_2 = πS_1:C;R_2:A;R_2:B σB=S_1:C (ρR_2(R) XρS_1(S))

E.g.:

(SELECT name, address from CinemaStar)

EXCEPT

(SELECT name, address from CinemaExec)

Ans:

(πname;address(CinemaStar) - πname;address(CinemaExec))

Count bug:

query over the relations R(A;B) and T(A;C):

SELECT T.A FROM T

WHERE T.C = (SELECT COUNT(R.B) FROM R

WHERER.A=T.A)

If T contains the (10; 0) and R have no ple with A = 10, then the output of the query certainly contains the value 10. Let us translate this query into the relational algebra. First, we rewrite the sub query into the

EXISTS form:

SELECT T.A FROM T

WHERE EXISTS (SELECT COUNT(R.B) FROM R

WHERE R.A=T.A

HAVING COUNT(R.B) = T.C)

The relational algebra translation is then:

πT.AσCOUNT(R.B)=T.C **ϒ** T.A,T.C,COUNT(R.B) σ R.A=T.A(R X T)

If R contains no tuple with A = 10, then the section σR.A=T. A will eliminate the T-tupel (10; 0). Hence, in that case, 5 will not occur in the output, which is incorrect.This problem is well-known and only poses itself with these kinds of subqueries. The solution here consists in replacing the cartesian product with the context relations by an outer join.

Translation is πT.AσCOUNT(R.B)=T.C ϒ T.A,T.C,COUNT(R.B) $(R \overset{\circ}{\bowtie}_{R.A=T.A} T)$

Notes

Notes

Unit - 1.3: Basic Algorithms for Executing Query Operations:

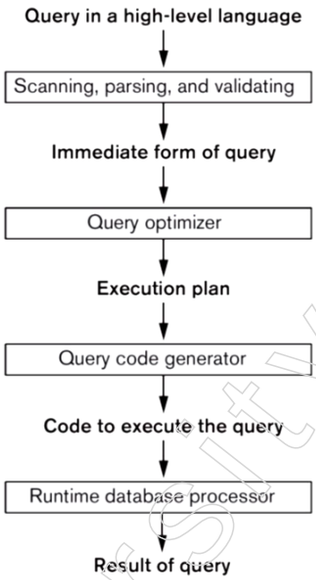


Fig 1.3.1 show the flow of query operations.

- Firstly high-level query language(SQL) must be scanned, parsed, and validated.
- Scanner: Scan and identify the token of the language.
- Parser: Syntax of the query checked
- Validate: Checked all the attributes and relations either valid or not.
- After scanning, parsing, and validating the query internally a query tree or a query graph is created.
- Then DBMS import an execution strategy for retrieving the data from the database files.
- Choose a efficient strategy for query processing.
- There are two main techniques for implementing query processing.
    - Heuristic rules for re-ordering the operations.
    - Systematically estimating the cost and choosing the lowest cost strategy.

Algorithms for External Sorting

- External sorting algorithms are applicable for large records files which is not stored in main memory.
- It used sort and merge strategy, first sorting small sub files and then merges them to create a large file called turn.

The two phases of algorithm are

- Sorting phase
- Merging phase.

Notes

Sorting phase:

- Runs of the files which can adjust in the current available buffer area to read into main memory, by using an internal sorting algorithm temporarily sorted sub files called runs there have some functions, these are

  Number of initial runs =nR,

  Number of blocks presents in files=b,

  Available buffer space=nb

so nR = b/nB

Merging phase:

- During one or more passes the sorted sub files are merged.
- In each pass how many number of sub files are merged together is called the degree of merging (d_M)
- In each pass there needs one buffer block to hold one block for each merge, and need another one block for storing the merge result.
- Consider d_M= MIN(nB– 1,nR,) and the number of passes is $\log d_M(nR)$.

Algorithms for SELECT and JOIN Operations:

Following five examples for comparison with the algo.

- (EX1): σ SSN=123456789 (EMPLOYEE)
- (EX2): σDNUMBER>6(DEPARTMENT)
- (EX3): σ DNO=6(EMPLOYEE)
- (EX4): σDNO=6 AND SALARY>30000 AND SEX=F(EMPLOYEE)
- (EX5): σESSN=123456789 AND PNO=12(WORKS_ON)

For selection, use the following search methods

S1 .Brute force Search: Fetch each and every record and check whether it is reach the selection condition or not.

S2 Binary search: If a key attribute involves an equality comparison on the selection condition then used more efficient search binary search rather than linear(compare EX1).

S3:Using hash key: nTo retrieve the record, If a key attribute involves an equality comparison on the selection condition(compare EX1).

SP4: To fetch multiple records used a primary index: If the comparison condition are like>;_; <;_ on a key field with a primary index and satisfy the equality condition then fetch the data and stored in a file.(compare EX2)

S5. Retrieving multiple records by clustering index : If a non-key attribute with a clustering index involved with equality comparison on the selection condition like

Notes

DNO = 6 in EX3 , then by index retrieve all the records.

S6: Used B+ tree on an equality comparison: If the indexing field is a key then retrieve one record, or multiple records if it is nota key. It is also used to compare >;_;<;_.

Search methods for complex (conjunctive) selection:

S7. Conjunctive selection: If any attribute involved with any simple single condition in the conjunctive condition with an access path which permits to use of one of the methods S2 to S6 to retrieve the data and to check whether each retrieved record satisfies the remaining conjunctive condition or not.

S8. Conjunctive selection trough a composite index: If more than one attributes are involved in equality conditions. And a composite index exists on the combined fields, the use the index directly. Like, in EX5 if, there have a composite index (ESSN; PNO) for WORKS ON _le.

S9. Conjunctive selection by record pointer's intersection: If more than one fields in the conjunctive of condition are available on secondary indexes if the indexes include record pointers rather than block pointers. The intersection of these sets of record pointers would satisfy the conjunctive condition, which are then used to retrieve those records directly. If only some of the simple conditions have secondary indexes, each retrieved record is further tested for the remaining conditions.

Algorithms for JOIN :

Join (EQUIJOIN, NATURAL JOIN)

- Two–way join: Apply join on two files like R A=B S.
- Multi-way joins: Applying join on more than two files like R A=B S C=D T.

Examples

(EX6): EMPLOYEE DNO=DNUMBER DEPARTMENT

(EX7): DEPARTMENT MGRSSN=SSN EMPLOYEE

The algorithm of join operation like that

R ⋈ S

A=B

JN1. Nested-loop join (Brute force algo): Tested whether the two records one from outer loop and one from inner loop satisfy the join condition. Just like from the above example for each record t in R (outer loop), and every record from S (inner loop) satisfy t[A] = s[B].

JN2. Single-loop join (using aaccessible structure for retrieving the matching records): If a hash key used for one of the two attributes , B of S retrieve each record t in R, at a time, and then by the access structure retrieve directly all matching records s from S and satisfy

Notes

s[B] = t[A].

JN3. Sort-merge joins: First using external sorting sort R and S based on the corresponding attributes A and B. Then merge sorted R and S with the help of fetched matching records t and s which satisfy the join condition t[A] = s[B].

J4. Hash-join:

- Partitioning phase: a single pass through the file with few records(R) hashes its records with the hash key A to the hash file.
- Probing phase: a single pass through the other file (S) then hashes each of its records to store in to the appropriate bucket.

Algorithms for PROJECT and SET Operations:

Algorithm for PROJECT operations

p < list of attribute>(R)

- If < list of attribute > has a key in relation R and fetch all tuples from R for the values attributes in < list of attribute >.
- If < list of attribute>have NOT include with a key of relation R and duplicated tuples must be delete from the results.

Methods used to remove duplicate are

- Sorting
- Hashing

Algorithm for SET operations

Set operations:

- UNION,
- INTERSECTION,
- SET DIFFERENCE
- CARTESIAN PRODUCT

CARTESIAN PRODUCT for relations R and S with allpossible combinations of records from R and S where The result's attributes included all attributes of R and S.

Cost analysis of CARTESIAN PRODUCT

If relation R with n records and j attributes and relation S with m records and

k attributes, the result like n*m records and j+k attributes.

UNION

- Sort the two relations depending on the same attributes.
- Sorted files are concurrently Scan and merge both.
- If the same tuple exists in both relations then only one is placed in the merged results.

Notes

Intersection

- Sort the two relations depending on the same attributes.
- Sorted files are concurrently Scan and merge both.
- If the same tuple exists in both relations then only both is placed in the merged results

SET DIFFERENCE R-S

Tuples which are appear inrelation R but not in relation S Kept in the merged results.

Implementing Aggregate Operationsand Outer Joins

Implementing Aggregate Operations:

Aggregate operators: MIN, MAX, SUM, COUNT and AVG

Options to implement aggregate operators:

Table Scan

Index

Example

SELECT MAX (SALARY)

FROM EMPLOYEE;

If there have an ascending order index of SALARY for the employee relation, then optimizer can traversing the index for the largest value from the root to a leaf with right most pointer in each index node.

Implementing Aggregate Operations (contd.):

SUM, COUNT and AVG

For a dense index (every record has has single index entry):

In the index Apply the associated computation to the values

For a non-dense index.

Total number of records associated with each index entry must be accounted .

With GROUP BY: Aggregate operator must be worked individually on each group of tuples.Useing sorting or hashing on the group attributespartitioned the file

into the appropriate groups and computes the aggregate function for the tuples for each group.

Outer Join Operators:

LEFT OUTER JOIN

RIGHT OUTER JOIN

Notes

FULL OUTER JOIN.

The full outer join's result is equivalent to result of the union left and right outer joins.

Example:

SELECT F_NAME, D_NAME

FROM (EMPLOYEE LEFT OUTER JOIN DEPARTMENT

ON D_NO = D_NUMBER);

Note: The result of this query is a table of employee names and their associated departments. It is similar to a regular join result, only exception is though any employee doesn't have an associated department then the employee's name still appear in the result but department name would be null.

Modifying Join Algorithms:

Nested Loop or Sort-Merge joins can be modified toimplement outer join. In left outer join, use the left relation (outer relation) and construct result and if it match then concatenated tuple is saved in to the result.

And if not then the tuple isstill included in the result with a null value.

Executing a combination of relational algebra operators.

Implement the previous left outer join example

(Compute the JOIN of the EMPLOYEE and DEPARTMENTtables)

TEMP1←πF_NAME,D_NAME(EMPLOYEED_NO=D_NUMBER DEPARTMENT)

(Find the EMPLOYEEs that do not appear in the JOIN)

TEMP2 ←πF _NAME (EMPLOYEE) - πF_NAME (Temp1)

{Pad each tuple in TEMP2 with a null D_NAME field}

TEMP2 ←TEMP2 x 'null'

{UNION the temporary tables to produce the LEFT OUTER JOIN}

RESULT ← TEMP1 ∪ TEMP2

The cost of the outer join, as computed above, would include the cost of the associated steps like join, projections and union.

Notes

Unit - 1.4: Query Tree and Querygraph:

Query tree:

Query tree is a relational algebra expression based tree data structure where the input relations of the query as leaf nodes and the relational algebra operations as internal nodes.

Execution of the query tree consists of executing an internal node operation whenever its operands are available and then replacing that internal node by the relation that results from executing the operation.

Query graph:

A query graph is a relational calculus expression based data structure. Science here a single graph dedicated for each query so here no need to maintain any operation order.

- Relation nodes: Represent by single circles.
- Constant nodes: Represent by double circles.
- Graph edges: Represent the selection and join conditions.
- Note: The retrieved attributes are displayed by square brackets above each relation.

Heuristic Optimisation of Query Trees:

The query parser generate a standard query tree depend on the perticular SQL query without any optimization.

The canonical query tree is generated by the following sequence.

Firstly the CARTESIAN PRODUCT of particular relations defined by the FROM clause.

The selection and join conditions of the WHERE clause are applied.

The projection are applied on the attributes of SELECT clause.

The initial inefficient query tree convert in to a final executable efficient query tree by the heuristic query optimizer.

Example:

For every project located in 'Stafford', retrieve the project number, the controlling department number and the department manager's last name, address and birthdate.

Relation algebra:

pPNUMBER, DNUM, LNAME, ADDRESS, BDATE (((sPLOCATION='STAFFORD'(PROJECT))

DNUM=DNUMBER (DEPARTMENT)) MGRSSN=SSN (EMPLOYEE))

SQL query:

Q2: SELECT P.NUMBER,P.DNUM,E.LNAME, E.ADDRESS, E.BDATE

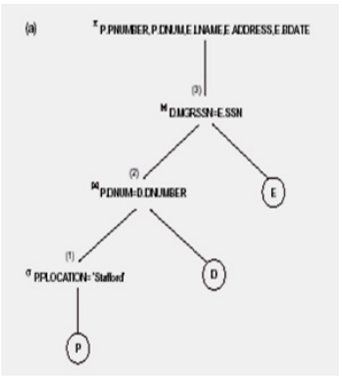
FROM PROJECT AS P,DEPARTMENT AS D, EMPLOYEE AS E

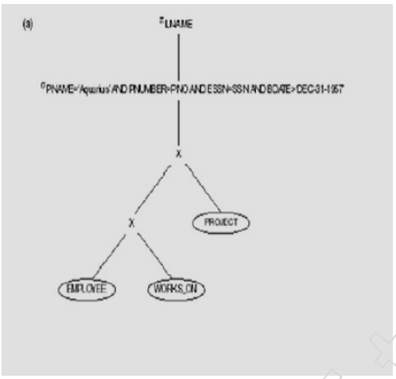
Notes

WHERE P.DNUM=D.DNUMBER AND D.MGRSSN=E.SSN AND

P.PLOCATION='STAFFORD';

The same query could correspond to many different relational algebra expressions — and hence many different query trees.





Ex:Consider the SQL query below.

SELECT L_NAME

FROM EMPLOYEE, WORKS ON, PROJECT

WHERE P_NAME='Aquarius' and P_NUMBER=P-NO and ESSN=SSN

and B_DATE > '1984-12-31';

General transformation rules for relational Algebra operations are

- Cascadeσ: σ c1 and c2 and …and cn(R) =σc1(σc2(….. (σcn(R)) ….))
- Commutativity ofσ: σc1(σc2(R)) =σc2(σc1(R))
- Cascade of π: πList1(_List2(…. (πListn(R))….)) =πList1(R)
- Commuting σ with π: If the selection condition c involves only those attributes A1;A2; : : : ;An in the projection list.

  πA1;A2…..An(σc(R)) =σc(πA1;A2…..An(R))
- Commutativity of ⋈ (and X):

  R ⋈c S = S .c ⋈ R

  R XS = S X R

Notes

Unit -1.5: Using Heuristics in Query Optimisation

Process for heuristics optimization

- The parser of a high-level query generates an initial internal representation;
- Apply heuristics rules to optimise the internal representation.
- A query execution plan is generated to execute groups of operations based on the access paths available on the files involved in the query.

The main heuristic is to apply first the operations that

reduce the size of intermediate results.

E.g., Apply SELECT and PROJECT operations before

applying the JOIN or other binary operations.

Heuristic Optimisation of Query Trees:

The same query could correspond to many different relational algebra expressions — and hence many different query trees.

The task of heuristic Optimisation of query trees is to find a final query tree that is efficient to execute.

Example:

Q: SELECT LNAME

FROM EMPLOYEE, WORKS_ON, PROJECT

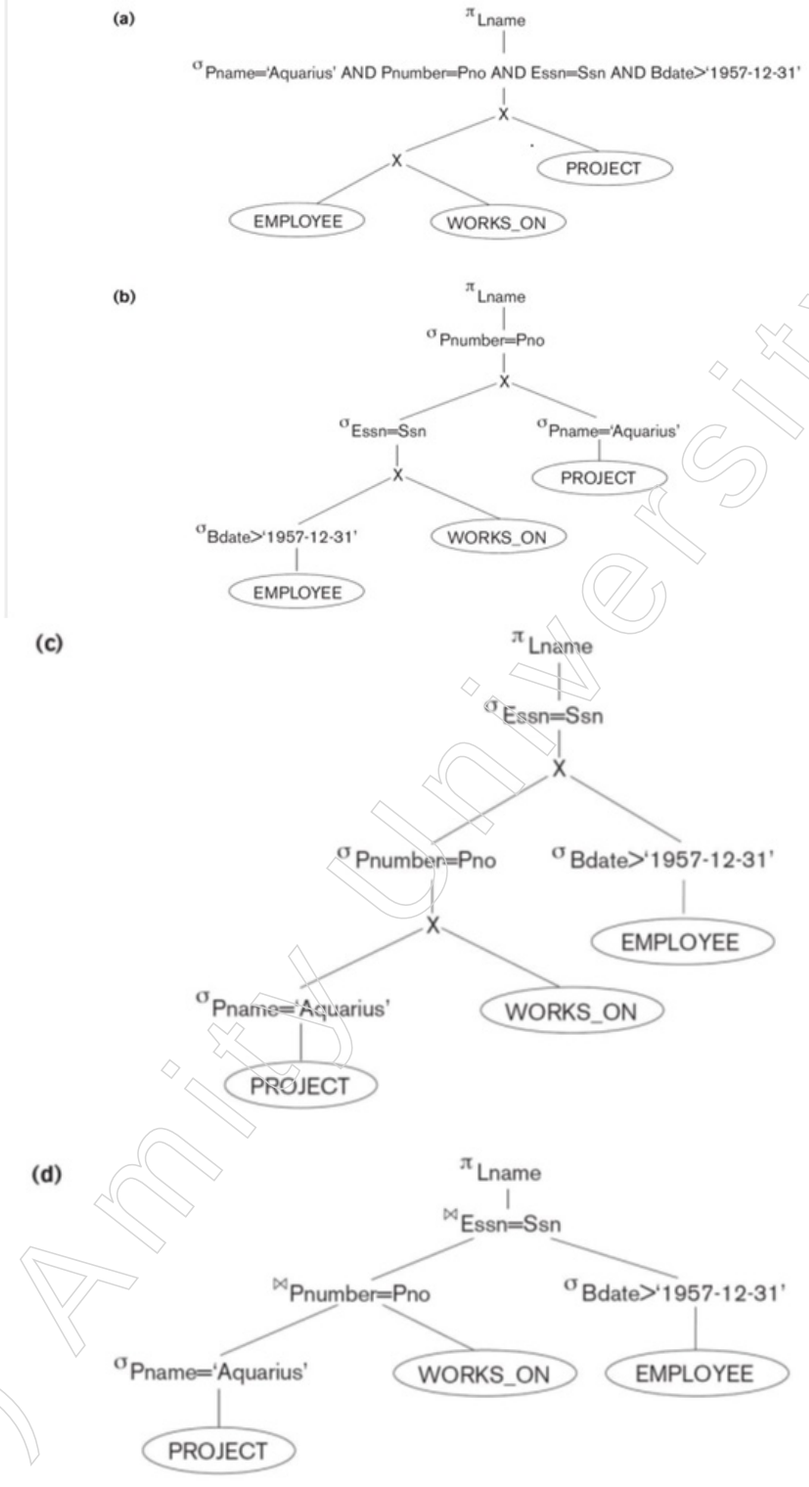
WHERE PNAME = 'AQUARIUS' AND

PNMUBER=PNO AND ESSN=SSN

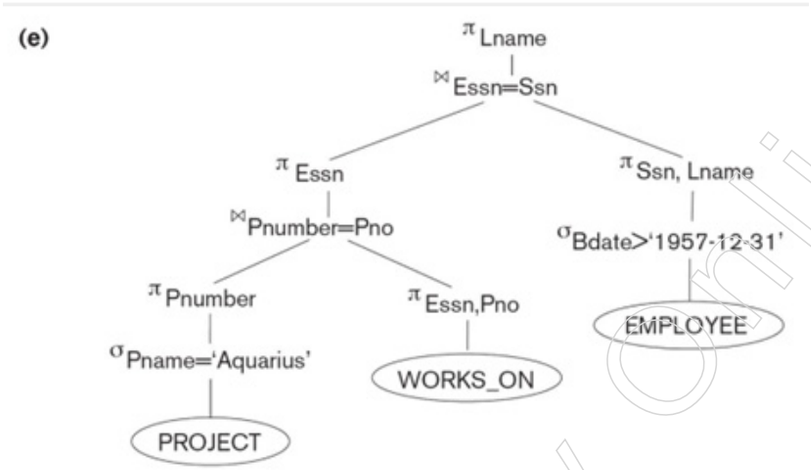
AND BDATE > '1957-12-31';

Notes

Steps in converting a query tree during heuristic optimization.
(a) Initial (canonical) query tree for SQL query Q.
(b) Moving SELECT operations down the query tree.
(c) Applying the more restrictive SELECT operation first.
(d) Replacing CARTESIAN PRODUCT and SELECT with JOIN operations.
(e) Moving PROJECT operations down the query tree.



Notes



All the above figure shows the above examples

(Ref : fundamental of database system Elmsri&Navathe)

Unit - 1.6 : Functional Dependency

Notes

- Functional dependencies (FDs) are used to specify formal measures of the "goodness" of relational designs .
- FDs and keys are used to define normal forms for relations .
- FDs are constraints that are derived from the meaning and interrelationships of the data attributes .
- A set of attributes X functionally determines a set of attributes Y if the value of X determines a unique value for Y .
- X → Y holds if whenever two tuples have the same value for X, they must have the same value for Y
- If t1[X]=t2[X], then t1[Y]=t2[Y] in any relation instance r(R).
- X → Y in R specifies a constraint on all relation instances r(R).
- FDs are derived from the real-world constraints on the attributes.
- Social Security Number determines employee name

  SSN → ENAME.
- Project Number determines project name and location

  PNUMBER → {PNAME, PLOCATION}.
- Employee SSN and project number determines the hours per week that the employee works on the project

  {SSN, PNUMBER} → HOURS.
- An FD is a property of the attributes in the schema R.
- The constraint must hold on every relation instance r(R).
- If K is a key of R, then K functionally determines all attributes in R (since we never have two distinct tuples with t1[K]=t2[K]).

Inference Rules for FDs

- Given a set of FDs F, we can infer additional FDs that hold whenever the FDs in F hold
- Armstrong's inference rules
  - A1. (Reflexive) If Y subset-of X, then X à Y
  - A2. (Augmentation) If X à Y, then XZ à YZ

    (Notation: XZ stands for X U Z)
  - A3. (Transitive) If X à Y and Y à Z, then X à Z

A1, A2, A3 form a sound and complete set of inference rules

Additional Useful Inference Rules

Decomposition

If X → YZ, then X → Y and X → Z

Notes

Union

If X → Y and X → Z, then X → YZ

Psuedotransitivity

If X → Y and WY → Z, then WX v Z

Closure of a set F of FDs is the set F+ of all FDs that can be inferred from F

If we take an example like that

Student(sid,sname,address)

Course(cid,cname)

Enrolled(sid,cid,grade)

Then the several functional dependency work like that

We can say that sid determines address – We'll write this

sid → address

- This is called a functional dependency (FD)
- (Note: An FD is just another integrity constraint)

We'd expect the following functional dependencies to hold in our Student database

– sid → sname,address

– cid → cname – sid,

cid → grade

- A functional dependency X → Y is simply a pair of sets (of field names) – Note: the sloppy notation

  A,B → C,D rather than {A,B} → {C,D}

  Formalities

  Given a relation

  R=R(A1 :τ1 , …, An :τ n),

  and X, Y (⊆{A1 , …, An }), an instance r of R satisfies X→Y,

  if – For any two tuples t1 , t2 in R, if t1 .X=t2 .X then t1 .Y=t2 .Y

  Note: This is a semantic assertion. We can not look at an instance to determine which FDs hold (although we can tell if the instance does not satisfy an FD!)
- Assume that X → Y and Y → Z are known to hold in R. It's clear that X → Z holds too.
- We shall say that an FD set F logically implies X → Y, and write F [X → Y – e.g. {X → Y, Y → Z} [X → Z
- • The closure of F is the set of all FDs logically implied by F, i.e. F + @ {X→Y | F [

Notes

X→Y}

- The set F+ can be big, even if F is small.

Candidate keys and FDs

- If R=R(A1 :т1 , …, An :т n) with FDs F and X⊆ {A1 , …, An }, then X is a candidate key for R if – X → A1 , …,An ∈ F+ – For no proper subset Y⊆X is Y → A1 , …,An ∈ F+.

Armstrong's axioms

Reflexivity: If Y⊆X then F \ X→Y – (This is called a trivial dependency) – Example: sname,address→address

- Augmentation: If F \ X→Y then F \ X,W→Y,W – Example: As cid→cname then cid,sid→cname,sid
- Transitivity: If F \ X→Y and F \ Y→Z then F \ X→Z – Example: As sid,cid→cid and cid→cname, then sid,cid→cname

Consequences of Armstrong's axioms:

Union: If F \ X→Y and F \ X→Z then F \ X→Y,Z • Pseudo-transitivity: If F \ X→Y and F \ W,Y→Z then F \ X,W→Z • Decomposition: If F \ X→Y and Z⊆Y then F \ X→Z

Equivalence:

Two sets of FDs, F and G, are said to be equivalent if F+ =G+

For example:

{(A,B→C), (A→B)} and

{(A→C), (A→B)} are equivalent

F + can be huge – we'd prefer to look for small equivalent FD sets.

Minimal cover:

An FD set, F, is said to be minimal if

- Every FD in F is of the form X→A, where A is a single attribute
- For no X→A in F is F-{X→A} equivalent to F
- For no X→A in F and Z⊂X is (F-{X→A})∪{Z→A} equivalent to F

For example, {(A→C), (A→B)} is a minimal cover for {(A,B→C), (A→B)}

FACT:

If F is an FD set, and X→Y∉F + then there exists an attribute A∈Y such that X→A∉F +

Why Armstrong's axioms?

Soundness – If F \ X→Y is deduced using the rules, then X→Y is true in any relation in which the dependencies of F are true

Notes

Completeness – If X→Y is is true in any relation in which the dependencies of F are true, then F \ X→Y can be deduced using the rules.

Soundness • Consider the Augmentation rule: – We have X→Y, i.e. if t1 .X=t2 .X then t1 .Y=t2 .Y – If in addition t1 .W=t2 .W then it is clear that t1 .(Y,W)=t2 .(Y,W)

Consider the Transitivity rule: – We have X→Y, i.e. if t1 .X=t2 .X then t1 .Y=t2 .Y (*) – We have Y→Z, i.e. if t1 .Y=t2 .Y then t1 .Z=t2 .Z (**) – Take two tuples s1 and s2 such that s1 .X=s2 .X then from (*) s1 .Y=s2 .Y and then from (**) s1 .Z=s2 .Z

Unit - 1.7: Normalisation

Notes

- Normalisation: Process of decomposing unsatisfactory “bad” relations by breaking up their attributes into smaller relations
- Normal form: Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

2NF, 3NF, BCNF based on keys and FDs of a relation schema

4NF based on keys, multi-valued dependencies

First Normal Form:

- Disallows composite attributes, multivalued attributes, and nested relations; attributes whose values for an individual tuple are non-atomic.

Considered to be part of the definition of relation

(a)

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS

(b)

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

(a)

EMP_PROJ

SSN	ENAME	PROJS	
		PNUMBER	HOURS

(b)

EMP_PROJ

SSN	ENAME	PNUMBER	HOURS
123456789	Smith,John B.	1	32.5
		2	7.5
666884444	Narayan,Ramesh K.	3	40.0
453453453	English,Joyce A.	1	20.0
		2	20.0
333445555	Wong,Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya,Alicia J.	30	30.0
		10	10.0
987987987	Jabbar,Ahmad V.	10	35.0
		30	5.0
987654321	Wallace,Jennifer S.	30	20.0
		20	15.0
888665555	Borg,James E.	20	null

(c)

EMP_PROJ1

SSN	ENAME

EMP_PROJ2

SSN	PNUMBER	HOURS

- Second Normal Form: Uses the concepts of FDs, primary key
- **Definitions:**

Prime attribute - attribute that is member of the primary key K

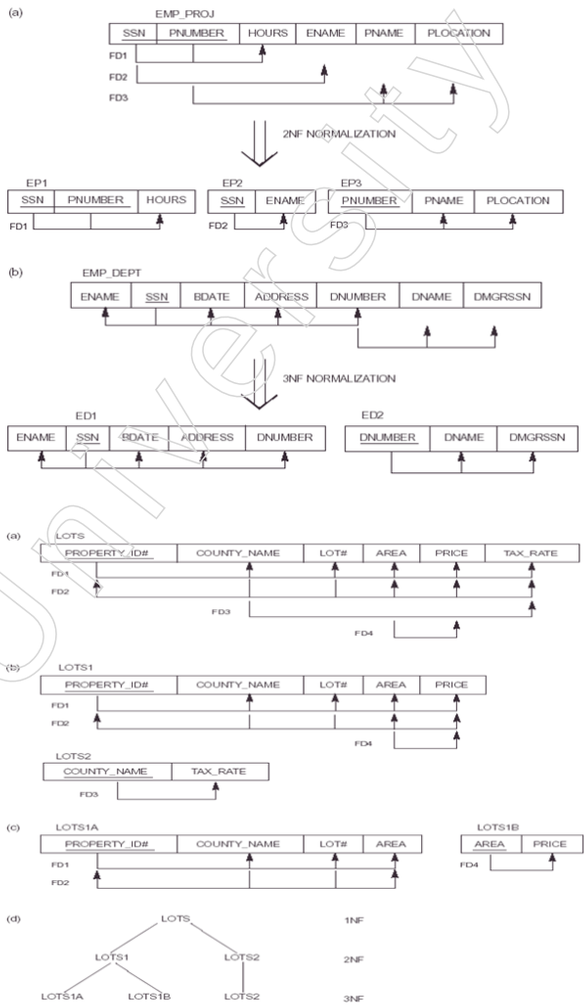
Full functional dependency - a FD $Y \rightarrow Z$ where removal of any attribute from Y means the FD does not hold any more .

Notes

Examples

Second Normal Form:

- {SSN, PNUMBER} → HOURS is a full FD since neither SSN → HOURS nor PNUMBER → HOURS hold
- {SSN, PNUMBER} → ENAME is not a full FD (it is called a partial dependency) since SSN → ENAME also holds
- A relation schema R is in second normal form (2NF) if every non-prime attribute A in R is fully functionally dependent on the primary key
- R can be decomposed into 2NF relations via the process of 2NF Normalisation



Third Normal Form:

Definition

Transitive functional dependency – if there a set of attributes Z that are neither a primary or candidate key and both X à Z and Y à Z holds.

Examples:

SSN → DMGRSSN is a transitive FD since

Notes

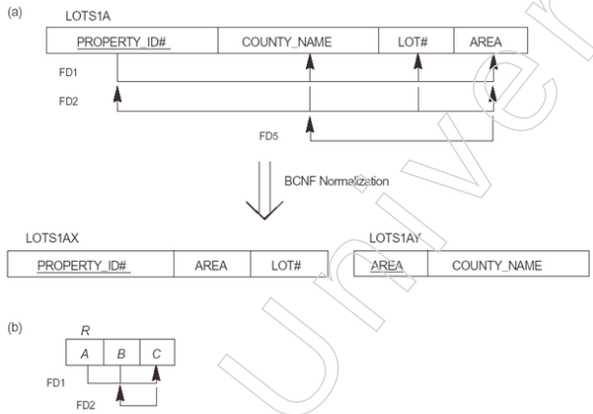
SSN v DNUMBER and DNUMBER → DMGRSSN hold

SSN → ENAME is non-transitive since there is no set of attributes X where SSN → X and X → ENAME

A relation schema R is in third normal form (3NF) if it is in 2NF and no non-prime attribute A in R is transitively dependent on the primary key .

BCNF (Boyce-Codd Normal Form):

- A relation schema R is in Boyce-Codd Normal Form (BCNF) if whenever an FD X à A holds in R, then X is a superkey of R
  - Each normal form is strictly stronger than the previous one:
    - ♦ Every 2NF relation is in 1NF
    - ♦ Every 3NF relation is in 2NF
    - ♦ Every BCNF relation is in 3NF
  - There exist relations that are in 3NF but not in BCNF
  - The goal is to have each relation in BCNF (or 3NF)



TEACH

STUDENT	COURSE	INSTRUCTOR
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe

- {Student,course} à Instructor
- Instructor à Course
- Decomposing into 2 schemas

Notes

- {Student,Instructor} {Student,Course}
- {Course,Instructor} {Student,Course}

{Course,Instructor} {Instructor,Student

Example:

- Given the relation

Book(Book_title, Authorname, Book_type, Listprice, Author_affil, Publisher)

The FDs are

Book_title → Publisher, Book_type

Book_type → Listprice

Authorname → Author_affil

Fourth normal form (4NF):

Fourth normal form (4NF) is a level of database Normalisation where there are no non-trivial multivalued dependencies other than a candidate key. It builds on the first three normal forms (1NF, 2NF and 3NF) and the Boyce-Codd Normal Form (BCNF). It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency.

Properties – A relation R is in 4NF if and only if the following conditions are satisfied:

- It should be in the Boyce-Codd Normal Form (BCNF).
- the table should not have any Multi-valued Dependency.

A table with a multivalued dependency violates the Normalisation standard of Fourth Normal Form (4NK) because it creates unnecessary redundancies and can contribute to inconsistent data. To bring this up to 4NF, it is necessary to break this information into two tables.

Example – Consider the database table of a class whaich has two relations R1 contains student ID(SID) and student name (SNAME) and R2 contains course id(CID) and course name (CNAME).

SID	SNAME
S1	A
S2	B

CID	CNAME
C1	C
C2	D

When there cross product is done it resulted in multivalued dependencies:

Notes

SID	SNAME	CID	CNAME
S1	A	C1	C
S1	A	C2	D
S2	B	C1	C
S2	B	C2	D

Multivalued dependencies (MVD) are:

SID->->CID; SID->->CNAME; SNAME->->CNAME

Joint dependency – Join decomposition is a further generalisation of Multivalued dependencies. If the join of R1 and R2 over C is equal to relation R. then we can say that a join dependency (JD) exists, where R1 and R2 are the decomposition R1(A, B, C) and R2(C, D) of a given relations R (A, B, C, D). Alternatively, R1 and R2 are a lossless decomposition of R. A JD ⋈ {R1, R2, …, Rn} is said to hold over a relation R if R1, R2, ….., Rn is a lossless-join decomposition. The *(A, B, C, D), (C, D) will be a JD of R if the join of join's attribute is equal to the relation R. Here, *(R1, R2, R3) is used to indicate that relation R1, R2, R3 and so on are a JD of R.

R1

COMPANY	PRODUCT
C1	CD
C1	DVD
C2	HD
C2	HD

R2

AGENT	COMPANY
ARKA	C1
ARKA	C2
RAJA	C1

R3

AGENT	PRODUCT
ARKA	CD
ARKA	DVD
ARKA	HD
RAJA	HD

Table – R1 ⋈ R2 ⋈ R3

Notes

COMPANY	PRODUCT	AGENT
C1	CD	ARKA
C1	DVD	ARKA
C2	HD	HD
C1	HD	ARKA

Fifth Normal Form / Projected Normal Form (5NF):

A relation R is in 5NF if and only if every join dependency in R is implied by the candidate keys of R. A relation decomposed into two relations must have loss-less join Property, which ensures that no spurious or extra tuples are generated, when relations are reunited through a natural join.

Properties – A relation R is in 5NF if and only if it satisfies following conditions:

- R should be already in 4NF.
- It cannot be further non loss decomposed (join dependency)

Example – Consider the above schema, with a case as "if a company makes a product and an agent is an agent for that company, then he always sells that product for the company". Under these circumstances, the ACP table is shown as:

AGENT	COMPANY	PRODUCT
A1	PQR	Nut
A1	PQR	Bolt
A1	XYZ	Nut
A1	XYZ	Bolt
A2	PQR	Nut

The relation ACP is again decompose into 3 relations. Now, the natural Join of all the three relations will be shown as:

R1

AGENT	COMPANY
A1	PQR
A1	XYZ
A2	PQR

R2

AGENT	PRODUCT
A1	Nut
A1	Bolt
A2	Nut

Notes

R3

COMPANY	PRODUCT
PQR	Nut
PQR	Bolt
XYZ	Nut
XYZ	Bolt

Result of Natural Join of R1 and R3 over 'Company' and then Natural Join of R13 and R2 over 'Agent' and 'Product' will be table ACP.

Hence, in this example, all the redundancies are eliminated, and the decomposition of ACP is a lossless join decomposition. Therefore, the relation is in 5NF as it does not violate the property of lossless join.

Module Out comes:

From the above, we learn

- Query processing technique
- Heuristic operation of Query processing
- Several Algorithm Used in Query processing

Self Assessment

- Try to draw the quer graph of the given 5 questions above.
- What are the main algorithm used in query processing.

Exercise

Fill in the blanks through the right answers.

1. Query tree is a relational algebra expression based ---------- data structure.
   a) Tree
   b) Graph
   c) Heuristic
   d) Hierarchical
2. Nested Loop or Sort-Merge joins can be modified to implement …….. join.
   a) Outer
   b) Inner
   c) Theta
   d) Gama
3. ……..operator must be worked individually on each group of tuples.

Notes

a) Aggregate

b) Join

c) Union

d) Add

4. External sorting algorithms are applicable for …….records files

a) large

b) small

c) text

d) medium

5. External sorting algorithms is not stored in……..

a) main memory

b) external memory

c) data structure

d) flash disk

Identify the following statement are true or false

1. High –level language is structured query Language.
2. Query processing mainly divided into 3 phase
3. Conjunctive Normal Form connected with OR operator
4. SQL statement assigned a distinct name through the alias
5. DBMS import an execution strategy for retrieving the data from the database files.

Answer Keys

Question	Answer	Question	Answer	Question	Answer
1	a	2	a	3	a
4	a	5	a		

Question	Answer	Question	Answer	Question	Answer
1	T	2	F	3	F
4	T	5	T		

Case study of Query Processing:

Let us discuss the whole process with an example. Let us consider the following two relations as the example tables for our discussion;

Employee(Eno, Ename, Phone)

Proj_Assigned(Eno, Proj_No, Role, DOP)

Notes

where,

Eno is Employee number,

Ename is Employee name,

Proj_No is Project Number in which an employee is assigned,

Role is the role of an employee in a project,

DOP is duration of the project in months.

With this information, let us write a query to find the list of all employees who are working in a project which is more than 10 months old.

SELECT Ename

FROM Employee, Proj_Assigned

WHERE Employee.Eno = Proj_Assigned.Eno AND DOP >10;

Input:

A query written in SQL is given as input to the query processor. For our case, let us consider the SQL query written above.

Step 1: Parsing

In this step, the parser of the query processor module checks the syntax of the query, the user's privileges to execute the query, the table names and attribute names, etc. The correct table names, attribute names and the privilege of the users can be taken from the system catalog (data dictionary).

Step 2: Translation

If we have written a valid query, then it is converted from high level language SQL to low level instruction in Relational Algebra.

For example, our SQL query can be converted into a Relational Algebra equivalent as follows;

πEname(σDOP>10 Λ Employee.Eno=Proj_Assigned.Eno(Employee X Prof_ Assigned))

Step 3: Optimiser

Optimiser uses the statistical data stored as part of data dictionary. The statistical data are information about the size of the table, the length of records, the indexes created on the table, etc. Optimiser also checks for the conditions and conditional attributes which are parts of the query.

Step 4: Execution Plan

A query can be expressed in many ways. The query processor module, at this stage, using the information collected in step 3 to find different relational algebra expressions that are equivalent and return the result of the one which we have written already.

Notes

For our example, the query written in Relational algebra can also be written as the one given below;

π_{Ename}(Employee $\bowtie_{Eno}$ ($\sigma_{DOP>10}$ (Prof_Assigned)))

So far, we have got two execution plans. Only condition is that both plans should give the same result.

Step 5: Evaluation

Though we got many execution plans constructed through statistical data, though they return same result (obvious), they differ in terms of Time consumption to execute the query, or the Space required executing the query. Hence, it is mandatory to choose one plan which obviously consumes less cost.

At this stage, we choose one execution plan of the several we have developed. This Execution plan accesses data from the database to give the final result.

In our example, the second plan may be good. In the first plan, we join two relations (costly operation) then apply the condition (conditions are considered as filters) on the joined relation. This consumes more time as well as space.

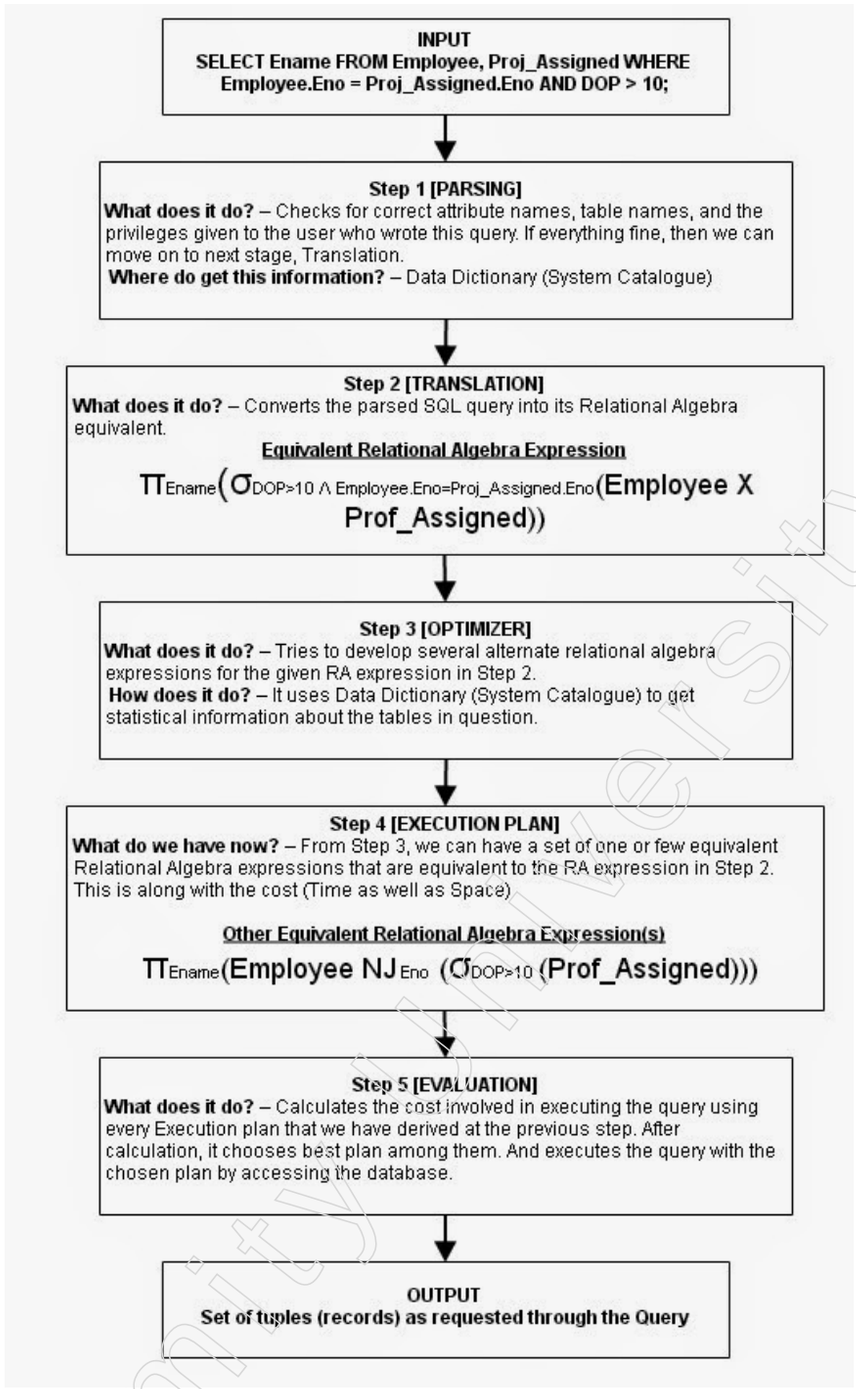
In the second plan, we filter one of the tables (Proj_Assigned) and the result is joined with the Employee table. This join may need to compare less number of records. Hence, the second plan is the best (with the information known, not always).

Output:

The final result is shown to the user.

The overall information discussed above are depicted in Figure 2 below;

Notes



Notes

Module - II - Object Oriented and Extended Relational Database Technologies

Key Learning Objectives:

At the end of this module, you will learn:

- Architecture of OODBMS
- Architecture and modelling about ORDBMS
- What are Specialization and Generalisation
- What are Aggregation and Associations
- What are the ObjectQueryLanguage

Basic Concept of oops:

Object oriented programming is a type of programming which uses objects and classes its functioning. The object oriented programming is based on real world entities like inheritance, polymorphism, data hiding, etc. It aims at binding together data and function work on these data sets into a single entity to restrict their usage.

Some basic concepts of object oriented programming are −

- CLASS
- OBJECTS
- ENCAPSULATION
- POLYMORPHISM
- INHERITANCE
- ABSTRACTION

Class : A class is a data-type that has its own members i.e. data members and member functions. It is the blueprint for an object in object oriented programming language. It is the basic building block of object oriented programming in c++. The members of a class are accessed in programming language by creating an instance of the class.

Some important properties of class are −

- Class is a user-defined data-type.
- A class contains members like data members and member functions.
- Data members are variables of the class.
- Member functions are the methods that are used to manipulate data members.
- Data members define the properties of the class whereas the member functions define the behaviour of the class.

A class can have multiple objects which have properties and behaviour that in common for all of them.

Notes

Object : An object is an instance of a class. It is an entity with characteristics and behaviour that are used in the object oriented programming. An object is the entity that is created to allocate memory. A class when defined does not have memory chunk itself which will be allocated as soon as objects are created.

Syntax

class_name object_name;

Encapsulation: In object oriented programming, encapsulation is the concept of wrapping together of data and information in a single unit. A formal definition of encapsulation would be: encapsulation is binding togather the data and related function that can manipulate the data.

Let's understand the topic with an easy real life example,

In our colleges, we have departments for each course like computer science, information tech. , electronics, etc. each of these departments have their own students and subjects that are kept track of and being taught. Let's think of each department as a class that encapsulates the data about students of that department and the subjects that are to be taught. Also a department has some fixed rules and guidelines that are to be followed by the students that course like timings, methods used while learning, etc. this is encapsulation in real life, there are data and there are ways to manipulate data.

Due to the concept of encapsulation in object oriented programming another very important concept is possible, it is data abstraction or Data Hiding. It is possible as encapsulating hides the data at show only the information that is required to be displayed.

Polymorphism: The name defines polymorphism is multiple forms. which means polymorphism is the ability of object oriented programming to do some work using multiple forms. The behaviour of the method is dependent on the type or the situation in which the method is called.

Let's take a real life example, A person can have more than one behaviour depending upon the situation. like a woman a mother, manager and a daughter. And, this defines her behaviour. This is from where the concept of polymorphism came from.

In c++ programming language, polymorphism is achieved using two ways. They are operator overloading and function overloading.

Operator overloading: In operator overloading and operator can have multiple behaviour in different instances of usage.

Function overloading: Functions with the same name that can do multiple types based on some condition.

Inheritance: It is the capability of a class to inherit or derive properties or characteristics other class. It is very important and object oriented program as it allows reusability i.e. using a method defined in another class by using inheritance. The class that derives properties from other class is known as child class or subclass and the class from which the properties are inherited is base class or parent class.

Several types of inheritances are

Notes

- Single inheritance
- Multiple inheritance
- Multi level inheritance
- Hierarchical inheritance
- Hybrid inheritance

Abstraction: Data abstraction or Data Hiding is the concept of hiding data and showing only relevant data to the final user. It is also an important part object oriented programing.

Let's take real life example to understand concept better, when we ride a bike we only know that pressing the brake will stop the bike and rotating the throttle will accelerate but you don't know how it works and it is also not think we should know that's why this is done from the same as a concept data abstraction.

In C plus plus programming language, write two ways using which we can accomplish data abstraction −

- using class
- using header file

Object-Oriented DBMS:

Object-Oriented Database Management Systems (OODBMS) are an extension of OO programming language techniques into the field of persistent data management. For many applications, the performance, flexibility, and development cost of OODBMSs are significantly better than RDBMSs or ORDBMSs. The chief advantage of OODBMSs lies in the way they can achieve a tighter integration between OO languages and the DBMS. Indeed, the main standards body for OODBMSs, the Object Database Management Group (ODMG) defines an OODBMS as a system that inte- grates database capabilities with object-oriented programming language capabilities.

The idea behind this is that so far as an application developer is concerned, it would be useful to ignore not only questions of how an object is implemented behind its interface, but also how it is stored and retrieved. All developers have to do is implement their applica- tion using their favourite OO programming language, such as C++, Smalltalk, or Java, and the OODBMS takes care of data caching, con- currency control, and disk storage.

In addition to this seamless integration, OODBMSs possess a num- ber of interesting and useful features derived mainly from the object model. In order to solve the finite type system problem that constrains SQL-92, most OODBMSs feature an extensible type system. Using this technique, an OODBMS can take the complex objects that are part of the application and store them directly. An OODBMS can be used to invoke methods on these objects, either through a direct call to the object or through a query interface. And finally, many of the structural deficiencies in SQL-92 are overcome by the use of OO ideas such as inheritance and allowing sets and arrays.

OODBMS products saw a surge of academic and commercial interest in the early 1990s, and today the annual global market for OODBMS products runs at about $50 million per year. In many application domains, most notably computer-aided design or

Notes

manufacturing (CAD/CAM), the expense of building a monolithic system to manage both database and application is balanced by the kinds of perfor- mance such systems deliever.

Object-Relational DBMS

ORDBMSs synthesise the features of RDBMSs with the best ideas of OODBMSs.

Although ORDBMSs reuse the relational model as SQL interprets it, the OR data model is opened up in novel ways. New data types and functions can be implemented using general-purpose languages such as C and Java. In other words, ORDBMSs allow developers to embed new classes of data objects into the relational data model abstraction.

ORDBMS schema have additional features not present in RDBMS schema. Several OO structural features such as inheritance and polymorphism are part of the OR data model.

ORDBMSs adopt the RDBMS query-centric approach to data manage-ment. All data access in an ORDBMS is handled with declarative SQL statements. There is no procedural, or object-at-a-time, navigational interface. ORDBMSs persist with the idea of a data language that is fun- damentally declarative, and therefore, mismatched with procedural OO host languages. This significantly affects the internal design of the ORDBMS engine, and it has profound implications for the interfaces developers use to access the database.

From a systems architecture point of view, ORDBMSs are implemented as a central server program rather than as a distributed data architec-tures typical in OODBMS products. However, ORDBMSs extend the functionality of DBMS products significantly, and an information sys- tem using an ORDBMS can be deployed over multiple machines.

Notes

Unit - 2.1: Over View Object-oriented Database

Object database management systems (ODBMSs) are the Database where the data represents in the form of objects and class based on the logic of object-oriented programming. We know in OOP's object is a real world entity and objects are stored in class. Where objects with the members like as fields, properties, and methods.

Object databases are used in applications likes high performance, calculations, faster results, real-time systems, architectural and engineering for 3D modelling, telecommunications, and scientific products, molecular science, and astronomy, etc.

In the ODBMS data model , data is stored in form of objects, which are instances of classes and these classes and objects together makes an object oriented data model.

Unit - 2.2: OO Concepts

Notes

Components of Object Oriented Data Model:

There have three major components in ODBMS these are

- Object structure.
- Object classes.
- Object identity.
    - **Object Structure:** The object structure is the properties of object by which an object is stand up. These properties are the attributes of an object (entity). Science an object is a real world entity then with the help of certain attributes that can makes an object structure. Also an object encapsulates the data code into a single unit, which can provide data abstraction by hiding the implementation details from the user.

The object structure is stands in three types of components these are

- Messages
- Methods
- Variables

Messages :

A message be an interface or a communication medium for building up a communication between an object and the outside world user. A message can be of two types:

- **Read-only message:** If the entreat method doesn't change the value of a variable, then that message called read-only message.
- **Update message:** If the entreat method changes the value of a variable, then the invoking message is said to be an update message.

Methods :

After passing the a message the body of code is executed which is known as a method. In every execution of method, it returns a value called output. Method can be of two types:

- **Read-only method:** When the value of a variable is not affected through a method, then it called read-only method.
- **Update-method:** When the value of a variable changes through a method, then it called update method.

Variables –

It stores distinguishable data for each and every object by which we can identify each variable separately.

- **Object Classes:** Science an object is an instance of a class. Then first, we need to define a class and then the objects are made which differ in the values they store but share the same class definition. The objects, in turn, corresponds to various messages and variables stored in it.

Notes

Example –

```
class STUDENT


{ //variables
    char name;
    string address;
    int id;
    int phone number;


   //messages
    char get_name();
    string get_address();
    int get_phonenumber();
 };
```

In above example, we can see, STUDENT is a class that holds the object variables and messages.

OODBMS supports inheritance in an extensive manner like in a database there have many classes formed with similar methods, variables and messages and the concept of class hierarchy is maintained to represent the similarities among various classes.

The concept of encapsulation that is the data or information hiding is also supported by object oriented data model. And data model also support the facility of abstract data types apart from the built-in data types like char, int, float. ADT's (abstract data type) are the user defined data types that hold the values within it and can also have methods attached to it.

Thus, OODBMS provides huge numbers of facilities for it's users in the form of both built-in and user defined. It incorporates the properties of an object oriented data model with a database management system, and supports the concept of programming paradigms like classes and objects along with the support for other concepts like encapsulation, inheritance and the user defined ADT.

2.3.1: Architecture of OODBMS:

There have several types of OODBMS architecture, Here. we discus about the Six layer architecture of OODBMS.

Notes

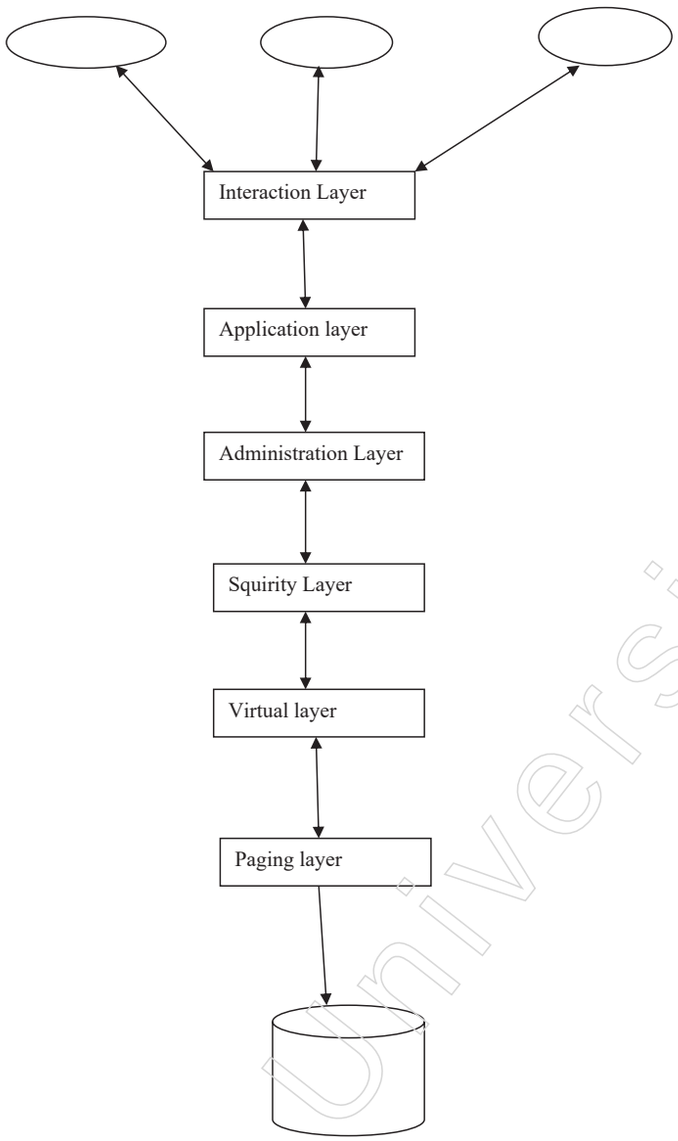


Fig. Describe the six layer architecture of OODBMS

Here we discus about the layers architecture of object oriented databases. Here data is transfered from data server to client through six layers. Each six layers with different responsibilities depending on the requirement.

Interaction layer: It is the first layer of Six Layers Architecture Model for Object Oriented Databases. Here, user can interact with the databases like user send data to the databases and retrieved data from database .

Application layer: Robert Greene told that, if by chance an early adopter choose an ODB who's architecture was ill suited for their applications needs, the reasoning lead immediately to the conclusion that no ODB was suited to solve their needs.

Administration Layer: This layer is responsible for management of administrative information. This layer can change responsibility as per requirement. The success in preserving complex structures like databases depends crucially on the amount of information that is lost in the process and this is inversely related to the amount of

Notes

metadata included in the preservation package. Administration layer control the flow of data as well as provide the permission to access the data.

Security Layer: The security layer plays an important role in this model. The security layer is responsible to provide the full security to data and also provide to the security of application used to manage the data also. This layer can provide the authentication to the uses as well as the authentication to databases administrators.

Virtual Layer: In this model, the virtual layer manage the data virtually to manage the large volume of data. Virtually stored means stored the data not in main memory outside of the main memory and depending on the requirement data will be converted in real memory. In this way, to handle a large data becomes very easy'.

Paging Layer: It is involved in distributing a large rule base and efficient query handling. Here, all the data is the data in the form of pages for easy handle. The data are divided into same size of pages frame. The page frame is equal number of partitions of memory.

2.3.2: Architecture of ORDBMS:

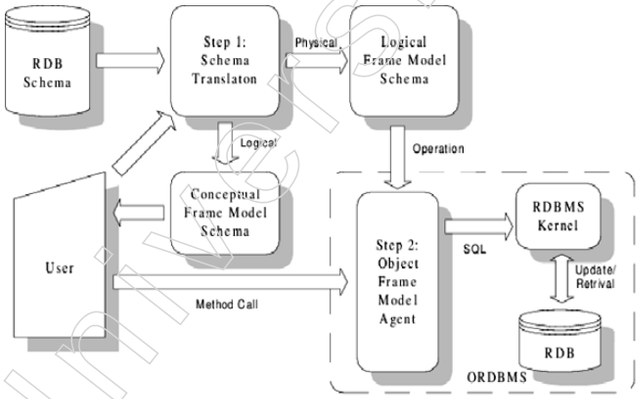


Fig. define the frame model of ORDBMS

Unit - 2.4 OODBMS Modeling:

Notes

Object-oriented databases established a data model with the support of object-oriented features discussed above and abstract data types. OO databases provide unique object identifiers by which the objects can be easily identified. This unique identifier is similar to a primary key in the relational model. It can utilise the power of object-oriented programming languages for providing excellent database programming capability. In object-oriented database management systems, the data is managed by two sets of relations.

- One is the interrelations of data items
- Another one is abstract relationships
- These systems established both relation types for coupling data items with procedural methods (encapsulation) to establish a direct relationship between the application data model and the database data model. For this strong connection between application and database there, minimum number of code.
- Create more natural data structures
- Better and reusability of code.

The lack of a defining standard was a drawback for OODBMS but Object Data Management Group proposed a standard known as ODMG-93 or ODMG 1.0 standard. That standard consists of

- The object model, the object defining language (ODL),
- The object query language (OQL),
- The bindings of OO programming languages.

The ODMG data model consists of data types, type constructors, etc. similar to the SQL report describing the standard model for relational databases. Here ODL is designed on semantic constructs object model. A key difference between relational databases and OO databases is the procedure in which relationships are handled. In OO databases, the relationships are represented explicitly with OIDs, which improves the data access performance.

Notes

Unit - 2.5: ORDBMS Modeling

An Object-Relational database added features of objectoriented systems .

It is a user-extensible type system supports

- Encapsulation
- Inheritance
- Polymorphism
- Dynamic binding of methods
- Complex objects including first normal form objects
- Object Identity

Disadvantages of ORDBMS

Complexity

Increased costs

Provision of a language(s) which will front end to SQL and will provide a migration path for existing SQL users.

ORDBMS SQL is a language used for

- Data definition,
- manipulation

ORDBMS design has to achieve the benefits of both the relational and the object models. It has a data model that which can incorporate OO features.

When a tabular entries may contain bigger data structure then its called abstract data types. It supports SQL3. The ORDBMS has occupied the relational model because to store the data in the form of tables with rows and columns. Hence, the characteristics of an ORDBMSs are:

- Base datatype extension,
- Support complex objects,
- Inheritance, and
- Rule Systems.

It also allow users to define datatypes, functions and operators.

An example schema of a student relation which ORDBMS supports is :

STUDENT(fname,lname,ID,sex,major,address,dname,location,picture)

In the above example, extra attributes “location” and “picture” which are not present in the traditional EMPLOYEE relation of RDBMS. And, the datatype of “location” is “geographic point” and that of “picture” is “image”.

Notes

2.5.1: Generalisation:

Generalisation is a bottom-up approach where two lower level entities combine together to form a higher level entity and the higher level entity again combine with other lower level entities to make further higher level entity and so on. Generalisation merge all the entities those who contain some common properties to form a new entity. It reducing the size of schema. It is always applied to the group of entities to form a single entity.

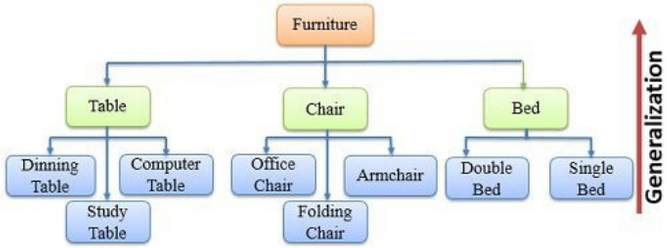


Fig. give an example of Generalisation

2.5.2: Specialisation:

It is a top-down approach where one higher level entity can be broken down into two lower level entities and so on . On the other hand, specialisation spilt an entity into multiple new entities those inherit some properties from the root node. Here, a higher entity may not have any lower entity present. Specialisation is just opposite it increases the number of entities. Thereby, increasing the size of a schema.

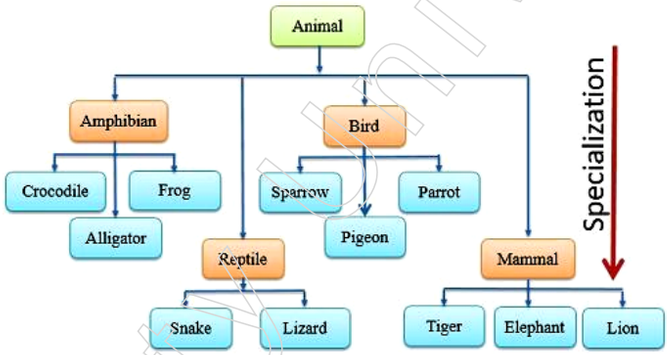


Fig. example of Spelization

2.5.3 Aggregation:

Aggregation is a process when the relation of two entities is treated as a single entity. It is a part of relationship where two distinguished objects are involved, one of them is a part of other to relatess instances.

Notes

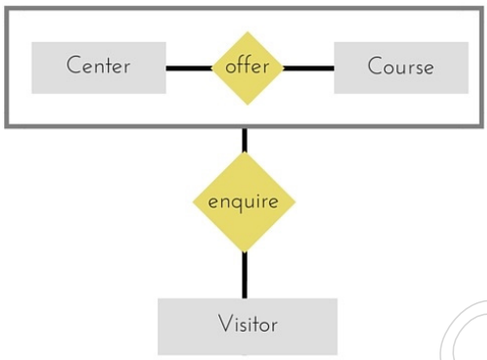


Fig. describe the aggregation

From the above diagram, the relationship between Center and Course together acting as single Entity in a relationship with another entity Visitor just like if a Visitor or a Student visits a Coaching Center, he/she will never enquire about the center only or just about the course, rather he/she will ask enquire about both.

2.5.4: Association:

- It is a description of a group of links (A physical or conceptual connection between objects) with common structure and semantics.
- It has a name (roles for problem description)
- A link is an instance of an association
- Qualified association
  - The objects in many roles are partially or fully disambiguated by an attribute called the qualifier.
  - A small box at the end of the association lies near the source class
- Multiplicity
  - The number of instances of one class that may relate to a single instance of an associated class one, many, zero or one.

Unit - 2.6: OODB Query Language

Notes

There are various types of query languages just like

- ONTOS (from Oncologic)
- O2 (from O2 Technology)
- IRIS (from HP)

All support object-orientedextension of SQL.

In IRIS, each entity type specifies one or more properties these are the functionapply to the instances . Entity types have extensions and can inherit from one another.

Example: The type PERSON can be created by:

Create Type Person (

Name char(20),

Age integer

Address address)

In IRIS, Name, Age and Address are called functions apply to instances of type Person.

Object SQL Query: retrieve the name and district of all people who are older than 25:

Select Name(p), Distrct(Address(p)

for each Person p

where Age(p) > 25

The user can compose the function application:

P5(P4(...P1(P))...)

in the select statement.

EXAMPLE:

Employee is a subtype of Person

SalesPerson is a subtype of Employee

Addresses have street names

Salespeople have sales managers

sales managers have secretaries

secretaries are employees

Out comes of this module:

- Learn about OODBMS

Notes

- Modelig of ORDBMS
- OO concept used in database model

Exercise:

1. Differentiate between Specialization and Generalisation
2. Discus about the Object Oriented concept.
3. Define class ane method with proper example.

Fill in the blanks with appropriate answer.

1. OODBMS supports inheritance in an …….manner.
   a) extensive
   b) exclusive
   c) extraneous
   d) Non of the above
2. The concept of encapsulation is that the data or information is ……..
   a) hiding
   b) wrapping
   c) binding
   d) all .
3. ……..layer is responsible for management of administrative
   a) administrative
   b) application
   c) insertion
   d) interaction
4. …… layer is involved in distributing a large rule base .
   a) virtual
   b) sequrity
   c) paging
   d) application
5. …….is a top-down approach
   a) Specialization
   b) Generalisation
   c) aggregation
   d) Association

Identify the following statement are true or false

Notes

1. links is a physical or conceptual connection between objects.
2. When the value of a variable changes through a method, then it called update method
3. An object is a real world entity
4. Object is an instance of a class
5. ORDBMS support inheritance.

Answer Keys

Question	Answer	Question	Answer	Question	Answer
1	a	2	d	3	a
4	c	5	a		

Question	Answer	Question	Answer	Question	Answer
1	T	2	T	3	T
4	T	5	T		

Notes

Module - III: Parallel and Distributed Database

Key Learning Objectives:

At the end of this module, you will learn:

- Introduction of parallel databases
- Design of Parallel databases
- Parallel QueryEvaluation
- Distributed databasesprinciples
- Architectures, design andimplementation of distributed databases

Unit - 3.1: Introduction

Notes

Parallel database architectures mainly developed by the use of external hardware in a software parallel dataflow architecture .These new designs processing relational database queries in an impressive speedup and scale up manner. Mainly relational queries are suitable for parallel execution because they applied uniform operations on streams uniform of data. Here, every new relation produced by a particular operator can be composed into highly parallel dataflow graphs that means the output of one operator be a input of another operator just worked like a series with pipelined parallelism. The input data By partitioned among multiple processors and memories, and operator can be split into many independent operators, each of them working on a particular part of the data. This partitioned data and execution mentioned partitioned parallelism.

These dataflow approach to database system depends on message-based client-server operating system for executing the relational operators which are interconnect with parallel processes. Parallel database systems are two types

- Data placement and query processing.
- Database clusters

It is mainly used for maintaining.

Large volume of data by using external disk and large main memory.

Memory access bottleneck features that means I/O bottleneck, it is

Microprocessor's speed>> Speed of RAM >> Disk's speed .

I/O bandwidthincrease .

Partitioning Data

Accessing data in parallel manner.

The Architecture of data flow in parallelism technique is like that

Inter-operation(partitioned data allows partitioned parallelism)

The same query paralalyaccess by different operations of on the different data source .

Notes

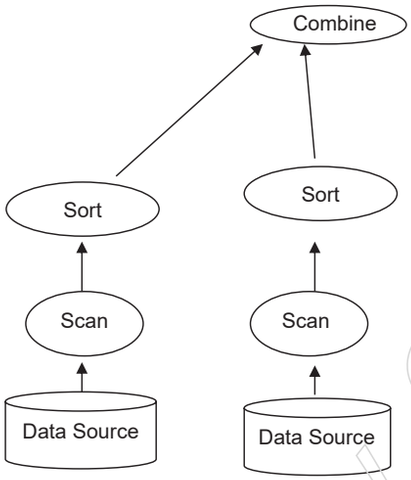


Fig 3.1.1 describe the flow in paralielism technique

3.2.1: Design of Parallel databases:

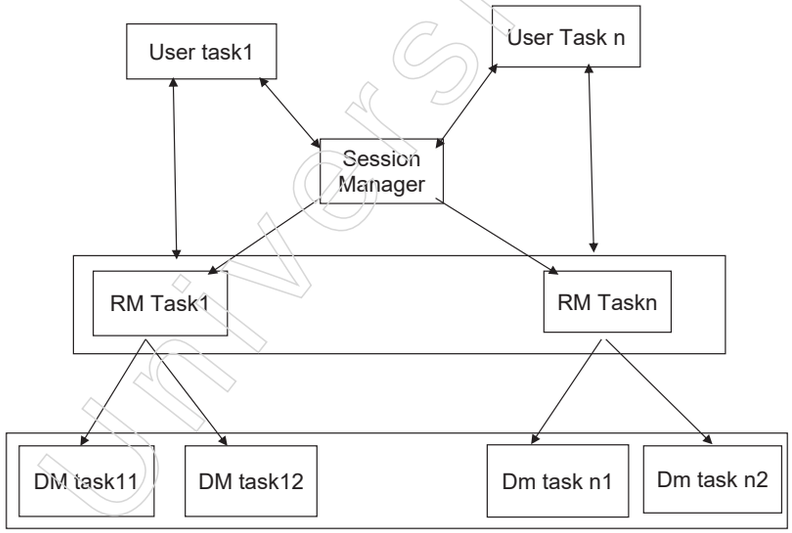


Fig 3.2.1.1 represent the Functional architecture of parallel DBMS

Session manager

- Host interface
- Transaction monitoring for OLTP

Request manager

- Compilation and optimisation
- Data directory management
- Semantic data control
- Execution control

Data manager

- Execution of DB operations
- Transaction management support

Notes

- Data management

System Architecture:

- Multiprocessor architecture alternatives
- Shared memory (SM)
- Shared disk (SD)
- Shared nothing (SN)
- Hybrid architectures
- Non-Uniform Memory Architecture (NUMA)
- Cluster

3.2.2: Parallel Query Evaluation:

In parallel database systems, we have to speed up the evaluation of relational queries. As per distributed database systems, knowledge data may be stored across different machines in potentially different databases. Science parallelism is cheaper and more scalable to use in more machines rather than faster processors. There have two main notions by which measure performance in the presence of parallelism.

- Speed-up: Using more processors, how much faster does the query run when the problem size is fixed?
- Scale-up: Using more processors, does performance remain the same as we increase the problem size?

In parallel database systems, there are three different systems architectures:

- **Shared memory:** Machine share RAM and disk. That kind of architecture is easy to program, but expensive to scale.
- **Shared disk:** The nodes access the same disk it is also hard to scale.
- **Shared nothing:**
  - Machines have their own RAM and disk, they share no resources.
  - The nodes are connected with a fast network.
  - This is the cheapest solution and scale up for thousands of machines.
  - It is very hard to program!

Types of Parallelism

Inter-query parallelism: Here each query runs in one processor, but for every different queries, assign different nodes. Use a common platform for the transaction processing where each transaction can be executed in a different node.

Inter-operator parallelism: Every query can runs in multiple processors. In the sense of parallelism the different operators of a query running in different processors.

Intra-operator parallelism: A single operator is distributed among multiple processors. This is also commonly referred to as data parallelism.

We will focus mostly on intra-operator parallelism. It provides the largest speed-up.

Notes

Data Partitioning

Before we discuss some simple algorithms by which we can evaluate the parallel relational queries, we will keep a look in to the different ways we can partition the data across the nodes.

Example: Consider the relation R(K, A, B, C), where K is the key with n tuples p numer of node.

Block partitioning: Arbitrarily partition the data that same amount of data n/p is placed at each node.

Hash partitioning: Partitioned the data by applying a hash function on some attribute of the Relation to send each tuple to the corresponding node.

Range partitioning: Partition the data according to a specific range of an attribute.

Parallel Operators

Selection sA=5(R): If the data is hash partitioned with the selection condition on a attribute that means execute the selection condition only in the 5machine .

Join R(A, B) ./ S(B, C): The standard parallel join algorithm is called parallel hash join. Where use a hash function h to partition both R, S on their join attribute B. Each tuple of R(a, b) will go to machine h(b), and each tuple S(b, c) will go to machine h(b).

Cartesian Product. In the cartesian product q(x, y) : -R(x), S(y) suppose whereIRI= nR and ISI = nS. The key idea is to organize the p machines into a pRX pS rectangle, such that p = pR.pS. By which we can identify each machine with a pair of coordinates. Then pick two hash functions,

hR :dom ! f1, . . . , pR

hS :dom ! f1, . . . , pS.

Then, we send each tuple R(a) to all themachines with coordinates (hR(a), _*) and send S(b) to all the machines with coordinates (_*, hS(b)). After data partitioned, each machine again locally computes the Cartesian product.

The algorithm is true,since each tuple (a, b) can be discovered in the machine with coordinates (hR(a), hS(b)).

Multiway Joins. Consider a queryof triangle q(x, y, z) = R(x, y), S(y, z), T(z, x). There use two step to compute this query in parallel manner.

Firstly, perform a parallel has join between R, S to obtain an intermediate relation RS(x, y, z).

Secondly, step consider another parallel hash join between RS and T.

Here, we organise the p machines into a 3-dimensionalhypercube: p = px _ py _ pzits give the identity of each machine.

The algorithm uses individual hash function for each variable.

Now, each tuple R(a, b) will be sent to all machines with coordinates (hx(a), hy(b), *). Similarly,

Notes

each tuple S(b, c) will be sent to coordinates (*_, hy(b), hz(c)) and each tuple T(c, a) to (hx(a), *, hz(c)).

Assume for the moment that we choose px = py = pz = p1/3. Then, we can see that each tuplewill be replicated to p1/3 machines.

Notes

Unit - 3.3: Distributed Databases Principles:

- A distributed database is a logically interrelated collection of data or a description of this data and physically distributed on a computer network.
- Extended concurrency control to maintain consistency of replicated data
- Extended recovery services to take into account of individual sites and communication links
- Extended communication services to provide access to remote sites and allow transfer of queries and data among sites using a network
- Distributed query processing, including query Optimisation and remote data access
- Extended system catalog to store data distribution details

DDBMS can be classified into homogeneous or heterogeneous. In a heterogeneous system, sites may run different DBMS products which needs not be based on the same underlying data model and so system may be composed of relational, network, hierarchical or OODBMS's. Heterogeneous systems usually results when individual sites have implemented their own databases and integration is considered much later. Translations are required to allow communication between different DBMSs.

In homogeneous systems, all sites use the same DBMS products. They are easier to design and manage as new sites can be easily added.

Unit - 3.4: Architectures DDBMS

Notes

Client Server Architecture of DDBMS

- This client server architecture is a two-level architecture which make it easier to control the complexity of modern DBMS and the complexity of distribution.
- Here the server make most of the data management works like
  - Query processing and optimisation
  - Transaction management
  - Storage management etc.
- And the client portion is the application as well as the user interface. It can
  - Manage the data that is cached to the client.
  - Manage the transaction locks.

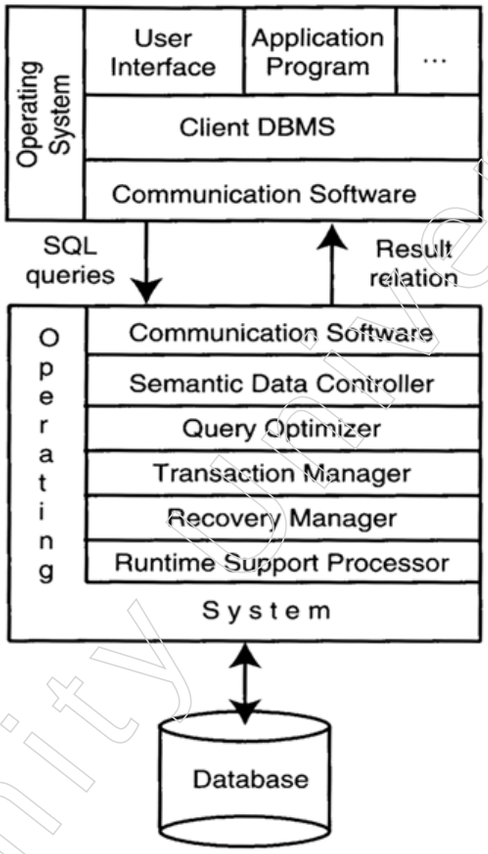


Fig 3.4.1.1 Define the client server architecture ofDDBMS

Multiple clients - single server

This is like centralised databases because the database is stored on only one machine called server. It also hosts the software to manage. Though there have some differences from centralised systems in the form of transactions are executed and caches are managed.

Notes

Multiple clients - multiple servers

In this strategy, there are two alternative possibilities, either each client manages its own connection to the appropriate server or each client knows of only its "home server" which then communicates with other servers as required.

Peer to peer architecture.

- The physical structure of data organisation may different for each machine.
- Local internal scheme (LIS) - It is an individual internal schema definition for each and every site.
- Global conceptual schema (GCS)–Used for describing the enterprise view of the data.
- Local conceptual schema (LCS) –Used to describe the logical schema of data for each site.
- External schemas (ESs)–Supporting the user applications and user access to the database

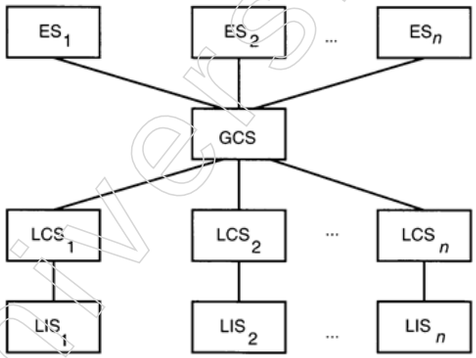


Fig 3.4.1.2 define the peer to peer architecture of DDBMS

3.4.2&3: Design and Implementation

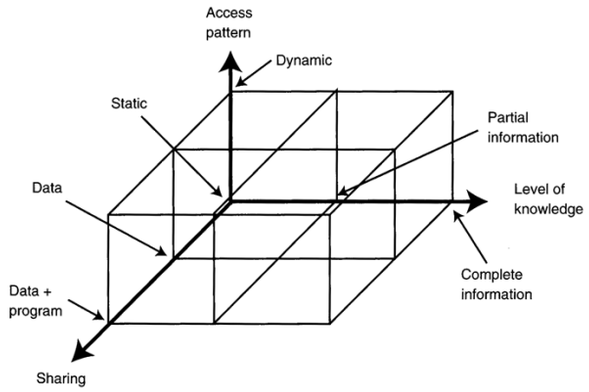


Fig 3.4.2&3.1 describe the design principle of DDBMS

Distribution framework

The distributed systems mainly established along three orthogonal dimensions:

Notes

- Level of sharing
- Behaviour of access patterns
- Level of knowledge

Level of sharing: Sharing mainly in three kind

- No sharing–each and every application and its corresponding data must be executes at one site.
- Data sharing - all the required program files are replicated on all of the sites but data files are not replicated.
- Data and program sharing –If required, then both the data and programs file may be shared.

Type of access patterns: It basically two types.

- Static - user requests access patterns do not change at the time interval.
- Dynamic - user requests access patterns change at the time interval.

Level of knowledge on access pattern behaviour

- Complete information – Though the access patterns can depend on the prediction, it does not deviate significantly from the predictions.
- Partial information–Information derived from the predictions.

There are used two main techniques to implement the design of the distributed database. These are

- Top-down approach
- Bottom-up approach

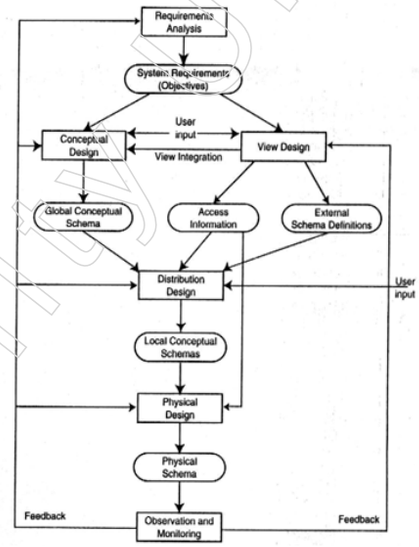


Fig 3.4.2&3. 2 describe the implementation procedure

From the above design we can describe all the above elaborately like.

View design - describing the interfaces for end users.

Notes

Conceptual design– It is the process used to examine and determine entity types as well as the relationships among these entities. It mainly divided in the following activity groups likes:

- Entity analysis –is responsible for determining the entities, the attributes of this entity, and the relationships among them.
- Functional analysis– is responsible to determine the fundamental functions are involved with the modeled enterprise.
- Distributions design - by distributing the entities over the sites of the distributed system design the local conceptual schemas. That distribution design activity are likely be divided in two parts.
    - fragmentation
    - allocation
- Physical design–That is the process to maps the local conceptual schemas on to the physical storage devices.
- Observation and monitoring - The result is mainly desired from the feedback and depending on that result we can think about the next step of the design.
- This top-down design is much suitable approach of database designed from scratch.

But if there are already a number of database and we have to integrate them to form a new database, then the bottom-up approach is suitable for this type of environment. Bottom-up design is starting from the individual local conceptual schemas. The process integrates all the local schemas into a global conceptual schema.

- The important issue in implementing is that the appropriate unit of distribution. For many reasons, it is only natural to consider subsets of relations as distribution units.
- If applications defined on a given relation reside at different sites.
- For that reason, the set of program as well as the data are replicated in different sites. These activities help the parallel execution of the data .
- There have another facility called fragmentation by which execution of a single query by dividing it into a set of sub queries which can resolve the concurrency control problem. That means it can increase the level of concurrency which is mainly helps the parallel distributed database.

Unit - 3.5: Fragmentation

Notes

To speed up the parallel database, fragmentation is more useful by which execution of a single query by dividing it into a set of sub queries which can resolve the concurrency control problem. Fragmentation is mainly two types , these are

- Horizontal fragmentation
- Vertical fragmentation
- The fragmentation by nature be nested but if the nesting's are build in different ways, then it is called hybrid fragmentation.
- The extent by which the database should be fragmented that affects the performance of query execution.

The degree of fragmentation goes to extreme, that means, there are no fragments at all, to the other extreme. Degree of fragment to the level of individual tuples consider in the case of horizontal fragmentation and the level of individual attributes consider in the case of vertical fragmentation.

Completeness

If a relation instance R is decomposed into fragments R1,R2, ..., Rn, that means a data item if present in R, then it must present in one or more of Ri's. This is an important property of fragmentation when their happens a lossless mapped between the global data and fragments.

Reconstruction

If a relation R is decomposed into fragments R1,R2, ..., Rn, it should be possible to define a relational operator Ñ such that:

$R = \nabla Ri, \ldots . \forall Ri \in FR$

The reconstructability of the relation from its fragments ensures that constraints defined on the data in the form of dependencies are preserved.

Disjointness

If a relation R is horizontally decomposed into fragments R_1,R_2, ..., R_n and data item di is in R_j, but it not in any other fragment R_k ($k \neq j$) that criteria ensures that the horizontal fragments are disjoint.

And if a relation R is vertically decomposed, then its primary key attributes are typically repeated in all its fragments. That means in vertical partitioning, disjointness is defined only on the nonprimary key attributes.

Notes

Unit - 3.6: Transparency

Though DDBS is distributed, but it should appear as a single system to the user . If DDBMS established distribution transparency, then user does not need to know about

- Replication of data
- Location of data
- Partitioning

Types of transparency

Distributed database system is providing the appearance of centralised system to end user.

There have eight transparencies to incorporate the desired functions of a DDBMS.

These are

- Access Transparency
- Location Transparency
- Concurrency Transparency
- Replication Transparency
- Failure Transparency
- Migration Transparency
- Performance Transparency
- Scaling Transparency

Access Transparency: There have no distinct differences between local and remote access methods that means explicit communication may be hidden. The access method to a remote object sould be identical to access a local object of the same class. It have two parts

- Between distributed and non-distributed access have a syntactical or mechanical consistency.
- Keep the same semantics science the semantics of remote access are failure prone.

Location Transparency: Every time clients search for a uniform filename space where files or set of files may be reallocated without changing their path. Which is important to support the movement of the resources and the availability of services. These locationand access transparencies together are called as Network transparency.

Concurrency Transparency:

Users and applications should able to access shared data without interfere each other. The shared objects are accessed simultaneously, like ATM.

Replication Transparency: This transparency mainly include for the distributed file systems, it replicates the data at two or more sites to make more reliable.

Notes

Failure Transparency: It enables to hiding of faults, user and application programs completing their tasks in spite of the failure of hardware or software components. Fault tolerance is provided by the mechanisms that relates to access transparency.

Migration Transparency: This transparency make the user to be unaware of the movement of information or processes within a system without affecting the running operations of the users and the applications that are. which allows for the load balancing of any particular client.

Performance Transparency: Provide the ability to reconfigured for improving the performance as.

Scaling Transparency: Allow a system to grow without affecting application algorithms. A system must be able to off scaled down to small environments where required, and be space and/or time efficient as per requirement.

Execution Transparency: The user must not be able to see the process Migration.

Configuration Transparency: The configuration must not affect the programming and use of the system.

Network Transparency: The programmer and the user have no idea about the network where system may looks like a stand-alone computer.

Name Transparency: Names will not change when the system is reconfigured.

Levels of Transparency

The three levels transparency used to hide certain complexities from the user for managing the database in effective manner.

Fragmentation transparency: Maximum level of transparency divides the original database into fragments for dispersing them throughout the DDBMS. For why user does not need to specify fragment names or locations for accessing .

Location transparency: It ensures that the user can query on any table or fragment of a table.The fact that the table or its fragments are stored at remote site in the distributed database system, should be completely unwire to the end user.

Local mapping transparency: The lowest level of transparency requires the user to know the name and location of fragments.

Notes

Unit - 3.7: Transaction Control in Distributed Database

A set of actions that involved to make consistent transformations of system states to preserve the system consistency is called transaction. The out comes of a transaction process are

- Concurrency transparency
- Failure transparency

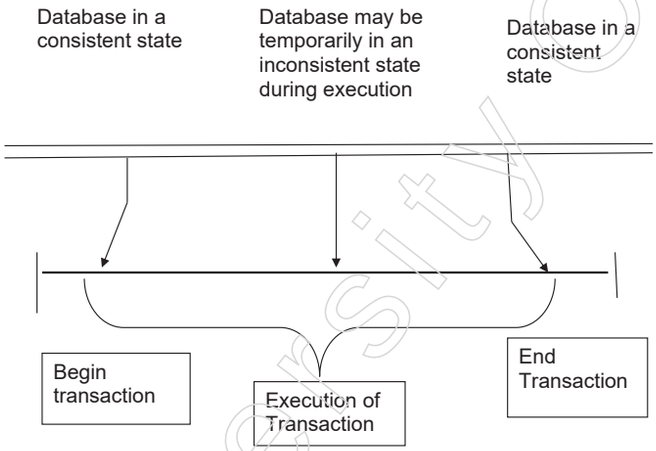


Fig 3.7.1 is the process structure of distributed transaction.

Characterisation of transaction :

Read set(RS): Transaction read a set of data items .

Write set(WS): Transaction update the value of data set.

Base set(BS): RS U WS(union of read and write set)

Principles of Transactions: The whole transaction procedure mainly depends on the following principle, which is called ACID property.

- **ATOMICITY:**
  - Either all transaction happened or nothing.
  - Either all or none of the transaction's operations are performed.
  - If any transaction is interrupted, then it should be incomplete.
  - Preserving the transaction's atomicity in presence of transaction aborts due to input errors, system overloads, or deadlocks is called transaction recovery.
  - The activity to established atomicity in the presence of system crashesis called crash recovery.
- **CONSISTENCY:**
  - Must satisfy the integrity constraints never violate the constrains.
  - Transaction that executes uniquely against a consistent database leaves it in a consistent state.
  - Transactions are correct programs.

Notes

- **ISOLATION**
  - Serialisability: It supports the concurrency control that means there concurrently execute several transactions and produced the same results .
  - Incomplete results: An incomplete transaction cannot disclose its results to other transactions until it's completed. Necessary to avoid cascading.
  - Dirty read:
    - T_1 modifies x which is then read by T_2 before T_1 terminates; T_1 aborts.
    - T_2 has read value which never exists in the database.
  - Non-repeatable (fuzzy) read: T_1 reads x; T_2 then modifies or deletes x and commits. T_1 tries to read x again but reads a different value or can't find it.
  - Phantom: T_1 searches the database according to a predicate while T_2 inserts new tuples that satisfiues the predicate.
  - Read Uncommitted: For transactions operating at this level, all three phenomena are possible.
  - Read Committed: Fuzzy reads and phantoms are possible, but dirty reads are not.
  - Repeatable Read: Only phantoms possible.
  - Anomaly Serialisable: None of the phenomena is possible.
- **DURABILITY**
  - Once a transaction commits, the system must guarantee that the results of its operations will never be lost, in spite of subsequent failures.
  - Database recovery

Characterization of Transactions Based on:

- Application areas
  - Distributed vs Non-distributed
  - Transactions Compensating
  - Heterogeneous transactions
- Timing
  - Short-life (On-line) vs lLong-life (batch)
- Structure
  - Simple transactions
  - Nested transactions
  - Workflows

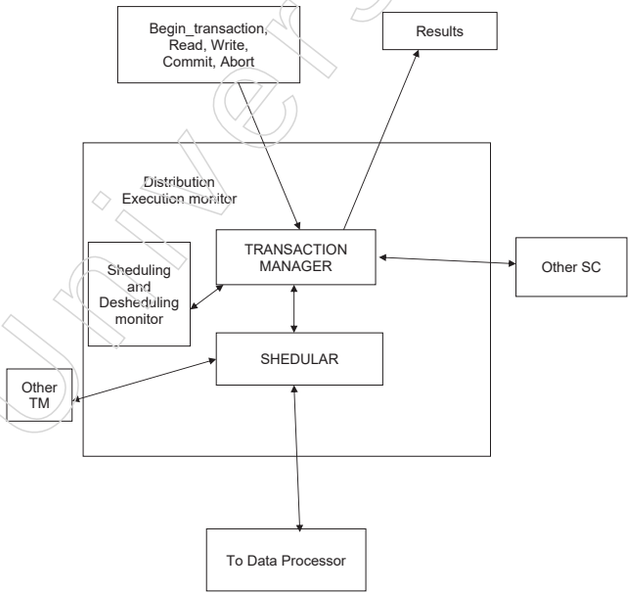
Transactions Provide:

- Atomic and reliable execution with failures
- Perfect execution with the multiple user accesses
- Correct management of replicas

Transaction Processing Issues:

Notes

- Transaction structure
  - Flat, nested
- Internal database consistency
  - Semantic data control algorithms
- Reliability protocols
  - Atomicity & Durability
  - Local recovery protocols
  - Global commit protocols
- Concurrency control algorithms
  - How to synchronize concurrent transaction executions (correctness criterion)
  - Intra-transaction consistency, Isolation
- Replica control protocols
  - How to control the mutual consistency of replicated data
  - One copy equivalence and ROWA



Unit - 3.8: Distributed Query Processing Mainly Works With

Notes

- Query Processing Methodology
- Distributed Query Optimization

Distributed Query Processing Methodology:

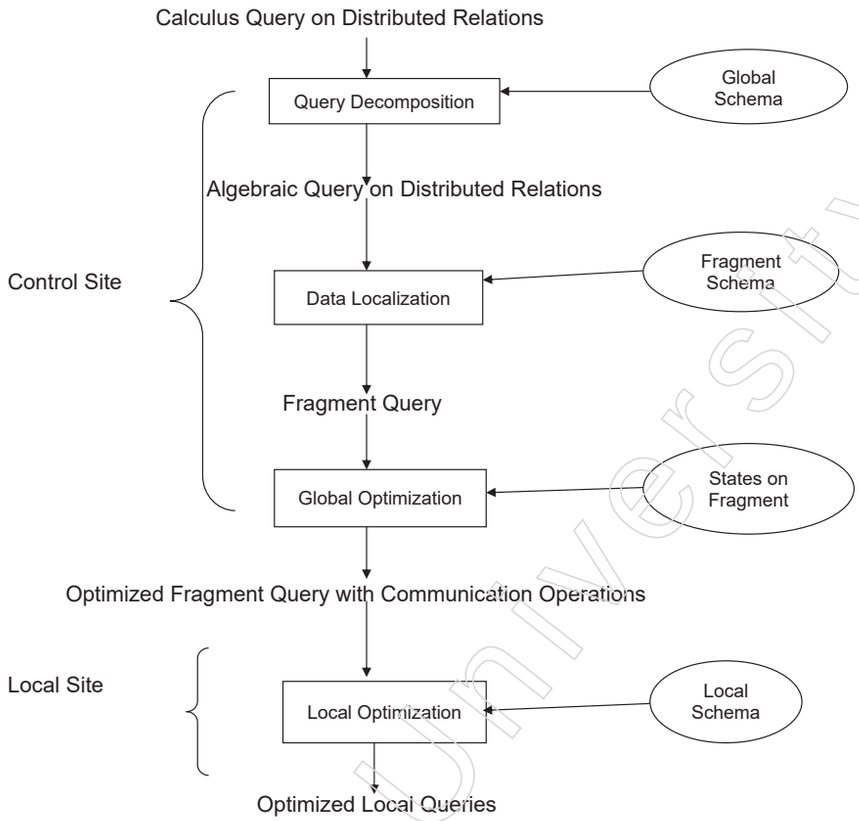


Fig 3.8.1 represent the query processing methodology

Query Decomposition:

- Input: Run calculus query on global relations .
- Normalisation - Quantifiers and qualification of queries are manipulated
- Analysis – Identify the wrong one and reject.
- Queries – Happened only for a relational calculus subset.
- Simplification - Redundant predicate's are eliminated.
- Restructuring:
  - calculus query -algebraic query .
  - more than one translation is possible .
  - use transformation rules

Notes

Normalisation

- Lexical and syntactic analysis used for –
- Validity checking like as compiler.
- Checking the attributes and relationship.
- Qualification type checking .
- Normal form-
- Construct the conjunctive normal form like

  (p11$\vee$ p12$\vee$ … $\vee$ p1n) $\vee$ … $\vee$ (pm1 $\vee$ pm2 $\vee$ … $\vee$pmn) .
- Disjunctive normal form like

  (p11 $\wedge$ p12 $\wedge$ … $\wedge$ p1n) $\wedge$ … $\wedge$ (pm1 $\wedge$ pm2 $\wedge$ … $\wedge$pmn)
- Logical OR's mapped by union.
- Logical AND's mapped into either join or selection.

Analysis

Reject incorrect queries

Incorrect Occur:

- Ifdefining of its attribute or relation names are not happened in the global schema.
- If operations are applied in wrong manner to the attributes.
- Semantically incorrect
- Components do not contribute in any way to the generation of the result
- only a subset of relational calculus queries can be tested for correctness
- Those that do not contain disjunction and negation
- To detect
  - connection graph (query graph) (It is a graph database e that uses graph structures for semantic queries with nodes, edges, and properties to represent and store data. ... Graph databases hold the relationships between data as a priority. Querying relationships is fast because they are perpetually stored in the database.)
  - join graph (a distributed join graph (DJG) is an acyclic directed graph where nodes represent sub-queries and edges represent cooperation relations between the nodes.)

Example

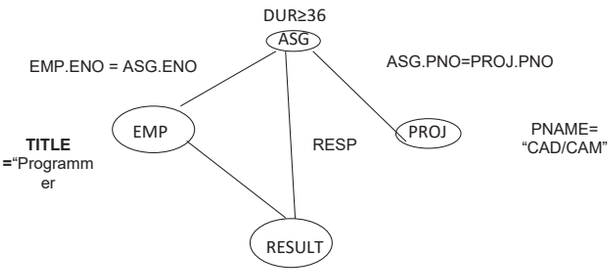
```
SELECT    ENAME,RESP
FROM      EMP, ASG, PROJ
WHERE     EMP.ENO = ASG.ENO
AND       ASG.PNO = PROJ.PNO
AND       PNAME = "CAD/CAM"
```

Notes

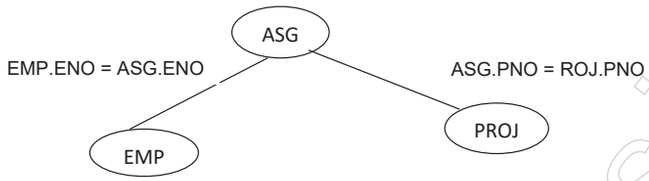
AND DUR ≥ 36

AND TITLE = “Programmer”

Query graph :



Join graph:



Simplification:

- Used to eliminate the redundancies
- Using following rules

  $p1 \wedge \neg(p1) \Leftrightarrow$ false

  $p1 \wedge (p1 \vee p2) \Leftrightarrow p1$

  $p1 \wedge$ false $\Leftrightarrow p1$ …
- Using transitivity rules
- Using integrity rules

Example:

```
SELECT      TITLE
FROM        EMP
WHERE       EMP.ENAME = “J. Doe”
OR          (NOT(EMP.TITLE = “Programmer”)
AND         (EMP.TITLE = “Programmer”
OR          EMP.TITLE = “Elect. Eng.”)
AND         NOT(EMP.TITLE = “Elect. Eng.”))
```

After Simplification we get

```
SELECT      TITLE
FROM        EMP
WHERE       EMP.ENAME = “J. Doe”
```

Notes

Restructuring:

- obtained relational algebra from relational calculus.

  using of query trees

- Example Find the names of employees other than J. Doe who worked on the CAD/CAM project for either 1 or 2 years. SELECT ENAME

  FROM EMP, ASG, PROJ

  WHERE EMP.ENO = ASG.ENO

  AND ASG.PNO = PROJ.PNO

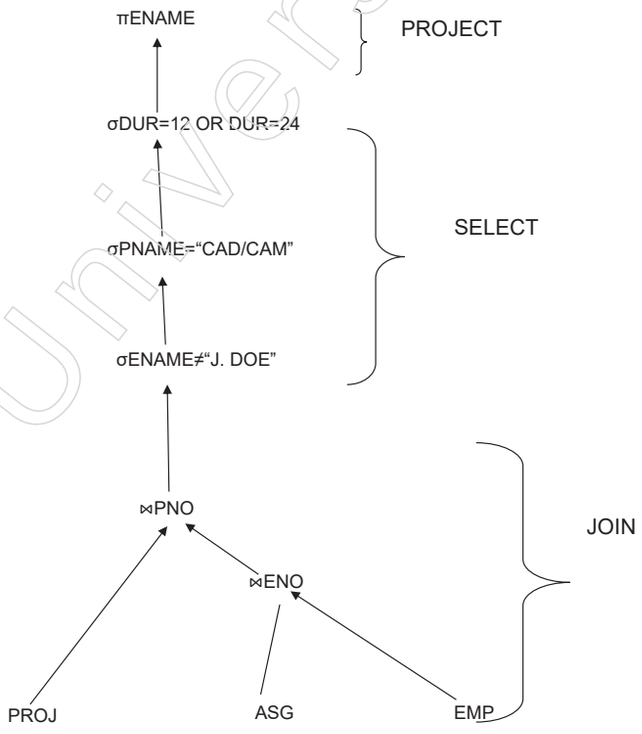
  AND ENAME≠ “J. Doe”

  AND PNAME = “CAD/CAM”

  AND (DUR = 12

  OR DUR = 24)

  Ans:



Data Localisation

Input: Applied algebraic query on distributed relations

- Determine the involved fragments .
- Localisation program

Notes

Substitute each global query by its materialisation program .

Optimise

Global Query Optimisation

Input:

Fragment query

- Finding the best way.

global schedule

Used to reduced a cost function

Distribute join processing

Either Bushy or by linear trees

Which relation ship to where?

Either Ship-whole or ship-as per requirement.

Using of semi joins

Semi join reduced expenses of communication and local processing.

Join methods

It may nested loop .

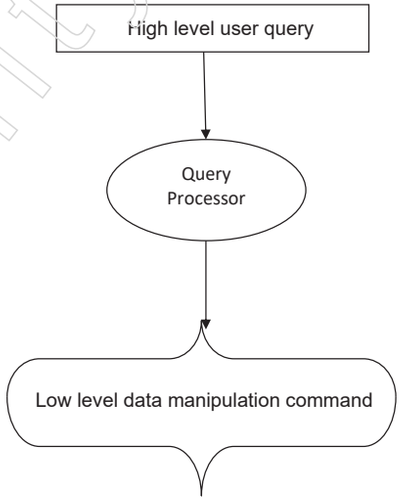
It may ordered joins like merge join or hash join.

Search Space

Search space categorised depending on by alternative execution

- Focused on join trees
- For N relations, used equivalent join trees obtained from commutatively and associatively rules.

Query Processing:



Components of Query Processing :

Notes

- Query language : That is use SQL relating to data define.
- Query execution methodology: Through this step the high-level declarative queries are executing.
- Query Optimisation :To find and determine the best execution plan?.

Selecting Alternatives

SELECT E_NAME

FROM EMP,ASG

WHERE EMP.E_NO = ASG.ENO

AND DUR > 37

Strategy i

π E_NAME(σ DUR >37∧ EMP.ENO=ASG.E_NO=(EMP ASG))

Strategy ii

π E_NAME(EMP E_NO (σ DUR >37 (ASG)))

there have no Cartesian product to be used thats why it is good.

What is the Problem? If we can process the above queries in to subparts, then we can get the following queries like

1. ASG1= σ E_NO ≤"E3"(ASG)

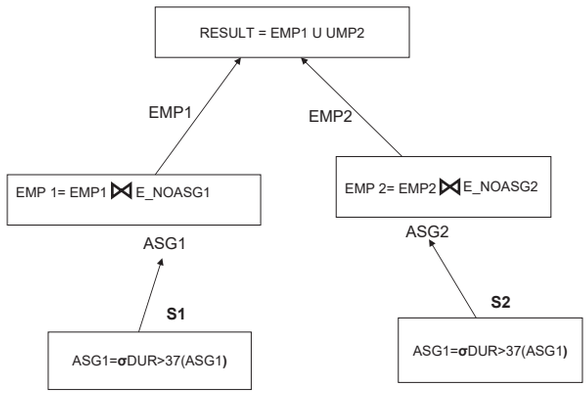
2. ASG2= σ E_NO >"E3"(ASG)

3. EMP1= σ E_NO≤"E3"(ASG)

4. EMP2= σ E_NO>"E3"(ASG)

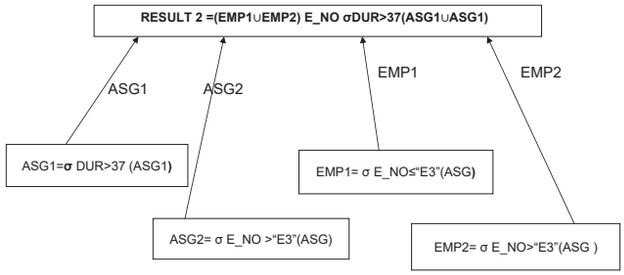
Now the results become like that

Result 1:



Result 2:

Notes



Cost of Alternatives

Assume:

Size (EMP) = 600, size (ASG) = 1200

Tuple access cost = 2 unit; Tuple transfer cost = 15 units

NOW Calculating cost for strategy one.

- Produce ASG : (15+15) x tuple access cost(2) 60
- Transfer ASG' to the sites of EMP: (15+15)∗ tuple transfer cost(15) 450
- Produce EMP': (10+10) ∗tuple access cost∗ 240
- 4. Transfer EMP' to result site: (10+10) ∗tuple transfer cost 200

Total cost 460

Strategy 2

- Transfer EMP to site 5:400∗tuple transfer cost 4,000
- Transfer ASG to site 5 :1000∗tuple transfer cost 10,000
- produce ASG':1000∗tuple access cost 1,000
- Join EMP and ASG':400∗20∗tuple access cost 8,000

Total cost 23,000

Query Optimisation Objectives

Minimize a cost function

I/O cost + CPU cost + communication cost these might have different weights in different distributed environments.

Outcomes of this Module

- Know about the query Optimisation technique
- About the transaction process mechanism
- Know about DD
- What is fragmentation

Notes

Exercise:

1. Describe the various transparency used in DDBS
2. Is there required fragmentation in DDBS and why?
3. What are the utilisation of transaction process?

Fill in the blanks with appropriate answer.

1. …….are access pattern used to design DDBS
   a) Extensive
   b) Static
   c) Dynamic
   d) Both a & b
2. …….. are two main technique to implement the design of the distributed database hiding
   a) Top down
   b) Bottom up
   c) Serializibility
   d) Both a&b
   e) Botha&c
3. fragmentation is used to……..of parallel database
   a) Speed up
   b) Speed down
   c) Control
   d) Interaction free
4. Every time clients search for a uniform file name space that is called……..
   a) Location transparency
   b) Access transparency
   c) Concurrency transparency
   d) None of the above
5. …….. satisfy the integrity constraints never violet the constrains.
   a) Isolation
   b) Atomicity
   c) Concurrency
   d) Durability

Identify the following statement are true or false

Notes

1. Write set (WS) used toTransaction update the value of data set.
2. Durability used to data base recovery
3. Reliability protocols are Atomicity &Durability,Local recovery protocols And Global commit protocols
4. In query decomposition Input used to run calculus query on global relations .
5. Search space used to execute the join tree.

Answer Keys

Question	Answer	Question	Answer	Question	Answer
1	a	2	d	3	a
4	a	5	c		

Question	Answer	Question	Answer	Question	Answer
1	T	2	T	3	T
4	T	5	T		

Notes

Module - IV: Databases on the Web and Semi Structured Data

Key Learning Objectives:

At the end of this module, you will learn:

- What is the web interface?
- What is XML and how to define it ?
- Storage of XML
- Application of XML

Unit - 4.1: Web Interfaces

Notes

As per the concept of websites the web interface is a page by which users can interact with the site when it is fully accessible through a web browser. Science any website is a collection of code (programs), but this code is not suitable enough for user interaction. In order to assurance of the visitors, can use a site there must need a web interface for easy communicate or easy interact by the users.

In several manner, most of the web design is about to build a web interface which allow visitors to command a site as acting in a useful manner. For this reason, it's much more essential to build a web interface which is both attractive and intuitive. Users have to be able to find the information they're searching. This niche is often referred as User Interface (UI) design. That's why every designer design their website as per their customer who can basically access that .

Web Interface Components

At the time of designing a website or blog, it is essential to keep the following website interface component in mind certain components in mind to build a effective websites. The components are

Immediate Information: When a visitor arrives access a web site, they should be aware immediately that they are at the right place. That means the user interface design needs to clearly indicate the actual identity of the website.

At the same time visitors should also be able to find quickly the links of website. As modern attention spans continue to grow feeble, so we don't want to take risk that web users pushing the back row because they unable to find actually what they want.

Easy Access: It is mostly related to the above component, easy access is a very critical component of any good interface design. We need to make sure that our visitors of the site have an intuitive click-through stream to find what they are searching for.

Layered intent: Make sure our site is layered as per the user intent in mind. Try to make it as easy as possible to access the most valuable things because you know where everything is but visitors does not so make it useful for the visitors.

Attractive Design: At the end of the day it is so simple no one can trust a site which looks unprofessional. Making an attractive, professional design can help visitors to trust of your website and brand.

Avoid Overdoing it: If your design is about to like old fashion in manner then it may impact on page performance. Where page speed is a vital component of Search Engine Optimisation .If we want to build our rank high in Google's search results then it's essential that our site operates efficiently. Users also prefer to interact with sites that load quickly, so slowing your page speed will increase your bounce rate.

Keep it simple: Keep as simple as possible the web application sphere and all things should be straightforward. because all fancy ideas may create a big issues and make it unobstructed interaction for the interface.

Compatibility issues. Eliminate all the compatibility issues to make interface should look, work, and feel as per the the screen size, operating system, browser, etc.

Notes

Make navigation flawless: Keep things as simple as possible mainly the navigation because users should navigate between products and functionality without a hesitation.

High performance: Website should not keep your users waiting for that every components should run fast.

Delivering messages: Web application design should clearly explain the purpose of the product. Structure everything using such time-proven components as tables, grids, carousels, etc.

Usability of interface

According to the world leaders in research-based user experience, usability is based on several important components:

- learning ability (is it easy for newcomers to accomplish the task),
- efficiency (how quickly can users perform their tasks),
- memo ability (how easily returned users can reestablish proficiency in working with your interface),
- errors (how easily can users recover from the mistakes made while interacting with the interface),
- satisfaction (is your website pleasant to work with),
- Utility (does your website provide services that your users need).

These essential quality components help to make your site useful for your prospects.

Unit - 4.2: Overview of XML

Notes

XML, namely known as extensible Mark-up Language. It is a mark-up language likely to be HTML but there have several differences between them like HTML have a collection of predefined tags using these in right way we can editing our HTML files, like <h1> for heading, <font> for font etc.

Where as XML tags are not predefined. Using XML user must have to define their own tags for every specific application as per their requirement. For example, if we want to describe a note firstly we have to define <note>, <heading>, <body> ,<to> ,<from> and then described them in following nested fashion.

<note>
<to> raj </to>
<from>rohit</from>
<heading> remainder</heading>
<body> not to forget about our get together </body>
</note>

In XML, firstly define the required tags, then assembled them to compose a user-defined markup language and then use the language for describing data. Mainly XML uses Document Type Definition (DTD) with an XML Schema to define tags. For that reason, it looks like a meta-language, since, it can be used to describe a markup language instead of concrete data directly, that's why it's called extensible. XML was designed for describing the data HTML for displaying data. Just like HTML tags control the way data is presented to browser users, color, font size, spacing, etc but XML deals with the logic meaning of data, or semantics. XML used to store data where data can be exchanged between incompatible systems. Though In real world computer and database hold data in incompatible formats. Converting the plain data in to XML minimize the complexity and create data in easy readable mode. Science the XML data is stored in plain text format as per the software- and hardware. So it's also called a cross-platform, software and hardware independent tool used for transmitting information.

4.2.1: Structure of XML data:

- XML document is an ordered, labeled tree
- Character data leaf nodes contain the actual data (text strings) „ Element nodes, are each labeled with „ a name (often called the element type), and „
- A set of attributes, each consisting of a name and a value, „ can have child node

Illustrate with an example:

</book>
<chapter id='abd'>
<chapheading> story1</chapheading>
<para> the chapter manage the following command
<fn>fcab</fm> internet application
</para>
</chapter>
</book>

Notes

Unit - 4.3: XML Schema:

XML Schema is an XML application also called syntax definition of XML. It can

- Provides simple data types likes
  - String
  - Integer
  - Date Time
  - Duration
  - Language, etc
- Defining possible values for the elements
- Describing types by deriving from existing types
- Allows complex types
- Can posing constraints depending on the occurrence of elements
- Maintaining the uniqueness and works with foreign keys
- Solving complex must cover an introductory talk
- Using formal language to expressing XML schemas just like Document Type Definition „
- Make job is much easier by using schema.

Example of Schema:

```
<xs:schema>
<xs:elementname=“story“>
<xs:complexType>
<xs:sequence>
<xs:elementname=“writer“type=“xs:string“/>
<xs:element name=“title“type=“xs:string“/>
<xs:element name=“text“>
<xs:complexType>
<xs:sequence>
<xs:elementname=“abstract“type=“xs:string“/>
<xs:elementname=“section“type=“xs:string“
minOccurs=“0“maxOccurs=“unbounded“/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Unit - 4.4: QueryingXML :

Notes

Querying XML means SQL for XML

It uses scenarios

- Human-readable documents
- Data-oriented documents
- Mixed documents

It mainly depends on

- XPath
- XML Schema datatypes
- XQuery

Main things of XQuery expressions are:

- Path expressions
- Construct element
- Flower (FLWR) expressions
- List expressions
- Expressions of condition.
- Quantified expressions
- Datatype expressions
- Context Evaluation.

But, here mainly discus on Querying XML through XPath andXQuery .

XPath andXQuery are query languages for XMLdata.

Their mainly searching through using

- logical conditions over the attribute and element.

Like first-order predicate logic (Xpath)

- regular expressions for matching pattern of element names, paths and the sub trees.

Like + joins, grouping, aggregation, transformation, etc.(XQuery)

XPath

- XPath is a simple language for identifying the parts of XML document its necessary for future processing.
- It workson the tree representation of the document
- XPath's result expressed by asetof elements or attributes.

Notes

It is basically alocation path of location steps, separated by using / like /article/text/abstract.

- Aleading /always mentioned the root element.
- Every location step identified by a node of the tree these are called context node.
- Possible location steps likes

  Childelement x: select all child elements with name x

  Attribute @x: select all attributes withname x

  Wildcards(*)any child,@*(any attribute)

  Multiple matches, separated by| likex|y|z.
- Standard:/(context node is the result of the preceding location step)

  article/text/abstract (allthe abstract nodes of articles)

- Select any descendant,not onlychildren: //article//index(anyindex elementin articles)
- Select the parent element: ..•Select the content node:.

  Predicate Added with [].
- Used to restricts elements .

  a[b]elements athat have a subelement b

  a[@d]elementsathathave anattribute d Plus conditions on content/value:

  o a[b=„c“]

  o A[@d>10]

  o <,<=,>=,!=,…

  Example of Xpath

/literature/book/author	retrieves allbook authors: startingwiththe root, traverses the tree,matcheselement names literature, book, author,and returns elements

<author>Samaresh, Uttar Dakhshin</author>,

<author>Nimai ,RagAshabari</author>

<first name>Haribansha</first name>

<last name>Rai</last name></author>

/literature/(book|article)/author	authors ofbooksorarticles
/literature/*/author	authors ofbooks,articles, story, etc
/literature//author	authorsthat are descendantsof literature

Notes

/literature//@year	value of the year attribute of descendantsof literature
/literature//author[first name]	authorsthathave a sub elementfirst name
/literature/book[price>„100“]	highpricedbooks
/literature/book[author//country =„India“]	books ofindianauthor

XQuery :

It is powerful querylanguage for XMLdata.

A query as the formof FLWRexpression:

FOR$var1INexpr1,$var2INexpr2,...

LET$var3:=expr3,$var4:=expr4,...

WHEREcondition

RETURNresult-doc-construction

Here the FORclause evaluates expressions and merge the resultingelements to variables.For merging eachvariable denotes exactly one element.

The LETclause merge entire sequences ofelements to variables.

The WHEREclauseevaluates thelogical condition with eachpossiblevariable merging andselects them only if satisfy thecondition.

And RETURNclauseconstructs XMLresult tree fromeach ofthe variable merging. It may also involve in grouping andaggregationand evencomplete subqueries.

Example of XQuery:

//findWeb-relatedarticlesby Ramanfromtheyear of 2000

<results>{

FOR$aIN document(“literature.xml“)//articleFORnINa//author,tINa/title

WHERE$a/@year=“2000“

ANDcontains($n,“Raman“)ANDcontains($t,“Web“)RETURN<result>nt</result>}</results>

//findarticlesco-authoredand authorswhohavejointlywrittenabookafter 1997

<results>{

FOR$aIN document(“literature.xml“)//articleFOR$a1IN$a//author,$a2IN$a//author

WHERESOME$bINdocument(“literature.xml“)//bookSATISFIES$b//author=$a1AND$b//author=$a2AND$b/@year>“1997“

RETURN<result>$a1$a2<wrote>$a</wrote></result>}</results>

Notes

Unit - 4.5: Storage of XML data:

In XML, use Data structures for storing XML data. For developing a XML data structure, we need to follow some conditions, rules, protocols, steps which are given bellow.

Data Retrieval-

- Which kind of primitives we need.
- Inverted index: assign all elements matching text query.
- Treat each and every element as a document.
- Assign all elements (immediately) below any instance of the Book element.
- Combination of the above

Links Between parent and Child -

- Numbering each element
- Maintain a proper list of parent-child relationships
- Like Module:3 ← Book:2
- Assign immediate parent of consequent child.

Propositional indexes -

- Consider XML document as a text document
- Create a positional index for each element
- Mark the starting and ending for each element like,

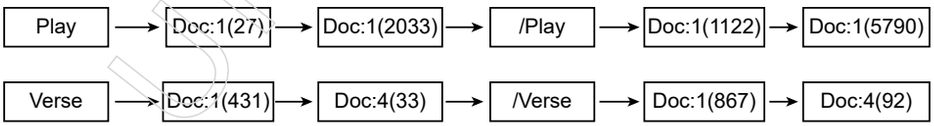


Fig 4.5.1 represent the links of paren child

Summary:

- Path containment can be solved by positional inverted indexes
- Retrieval consists of “merging” postings
- Complications arise from insertion/deletion of elements, text within elements

4.5.1: XMLapplications

There are several XML application here we discus few More in Below .

Wireless Markup Language (WML): It is XML based markup language used for the devices which implements the Wireless Application Protocol (WAP) specifications like mobile phones.

- It provides navigational support on data input, hyperlinks, hyper media, text and image presentation, etc.

Notes

- It increases the processing power of mobile devices.

XML News: It is a set of specifications for converting news objects like stories, images, and audio clips in a standard format depending on the different applications and different OS.

It uses XML and developed as per industry standards by the International Press Telecommunications Council and the Newspaper Association of America. It has two parts

- **XMLNews-Story -** It is an XML document type for text-based news and information. It assign the format of a news content and it support News Industry Text Format (NITF), the XML document type definition (DTD) for designing mark up and deliver news content in a variety of ways like print, wireless devices, and theWeb.
- **XMLNews-Meta-** It defines the format of any metadata associated with news and it based on the World Wide Web Consortium's Resource Description Framework (RDF).

Channel Definition Format (CDF) :

It is a file format developed by Microsoft to create a file which defines a Web "channel," .

- For using this channel, user needs the Microsoft Internet Explorer 4 or later browser.
- The CDF file identifies the Web page and subpages which will be appear to the user after selecting a channel on the browser. The channel developer puts the CDF file on the Web server.
- The CDF is an application of XML which Microsoft is proposed as a standard way to describe a Web site channel.

Open Software distribution (OSD):

- It is a vocabulary used to describe software packages and their dependencies for heterogeneous clients. It is much more useful in automated software distribution environments.

Open Financial Exchange (OFX):

It is a data-stream format for exchanging financial information that extract from Microsoft's Open Financial Connectivity (OFC) and Intuit's Open Exchange file formats.

- OFX is provide the solution to the financial services industry's in a simplified way to exchange electronic financial data with consumers and small businesses partners.
- It is open and unified specification for the exchange of financial data over the Internet.
- Open Financial Exchange helped to accelerate the adoption of online financial services and enabled financial institutions to offer their customers safe, secure banking, bill pay, investments and other services over the Internet.

Notes

RDF/XML :

- It is a syntax, defined by the W3C used to express an RDF graph as an XML document.
- It is sometimes misleading called simply RDF.
- The RDF/XML format is still in use because of it's more human-friendly.
- The RDF model is made up of 3times efficiently implemented and stored.
- The RDF model is essential for the canonicalisation of a directed graph.
- It can be processed even in absence of detailed information on the semantics because it allows basic inferences to take place which it can treat as logical fact basis.

Mathematical Markup Language (MathML):

- It is used to describe mathematical notations and capturing both its structure and content. It can integrate mathematical formulae into World Wide Web pages and other documents.

Platform for Privacy Preferences Project(P3P) :

- P3P is a protocol give permission to the websites for declaring their intended useing information they collect about web browser users.
- Designed for giving users more control on their personal information when browsing.
- P3P was developed by the World Wide Web Consortium(W3C) .
- Only Microsoft Internet Explorer is support P3P.

Human Resource Management Markup Language(HRMML):

It is an XML-based markup language for job postings, job descriptions, and preparing resumes.

It currently is described as two draft DTD one for resumes and one for job postings.

Voice Extensible Markup Language (VXML):

VXML is a technology developed by Motorola for creating a voice dialogue with a Web site in which a user can call a Web site by phone and interact with it through speech recognition.

- It allow developers to create a script of the conversation.
- The user calling in is connected to a client program called a voice browser it passes requests on to the Web server.
- The markup defined in VXML to define the strategic data definition language for the Internet

Vector Markup Language (VML) :

VML is used for vector graphics on the Internet. It was proposed to the World Wide Web Consortium as a standard for vector graphics rendering by Microsoft, Autodesk.

- Description of images in vector format.

Notes

- Resolution Independent
- Smaller in Size
- Faster Speed

Scalable Vector Graphics (SVG): It is an XML-based vector image format used for two-dimensional graphics with support for interactivity and animation.

- It is Resolution Independent
- It can reducing HTTP Request
- Styling and Scripting
- Can be animated and Edited
- Smaller in File Size

Synchronised Multimedia Integration Language (SMIL) :

It is a World Wide Web Consortium recommended Extensible Markup Language to describe multimedia presentations. It describes markup for timing, layout, animations, visual transitions, and media embedding, etc. It allows presenting media items, such as text, images, video, audio, links to other SMIL presentations, and files from multiple web servers.

Three dimensional mark up language(3DML):

3DML is a format to creating three-dimensional websites by combining similar sized building blocks.

3DML files can be delivered from standard web servers and shown within a browser plugin and independent 3DML browser called Flatland Rover.

It is designed for non-programmers.

It allows cross-toolkit development.

It is the integration of VRML, 3DS, OBJ, etc.

3dml allows reuse of elements.

Notes

Unit - 4.6:The Semi Structured Datamodel:

Entity Relationship model, Relational model , ODL models all are based on schema. These are all called structured data model where

- Structure of data is rigid and known in advance
- Efficiently implement in various storage and optimize in processing

Where as Semi structured data is schema less, that means in semi structure data model data can be represent without schema may likely be graphical representation. That model

- is much more flexible in representing data
- have different structure and properties for different objects.
- give permission to data is describing itself.
- It is harder to optimize but efficient in implementation.

In that model basically used describing the following kind of data storage and implement. This data is collected from that means the origin of these data are collect

By Integrating all heterogeneous sources

From data sources with non-rigid structure like

- Biological data
- Web data etc.

These data can be described through Various models like

OEM (Object Exchange model)

Semi_Structured Schema Graph

Semi Structure Schema Graph

Here is the Relational model for the data base Movie

Collection of records (touples)

title	*year*	*studioName*
Star Wars	1977	Fox
Gone With the Wind	1939	MGM
Wayne's World	1992	Paramount

Movie

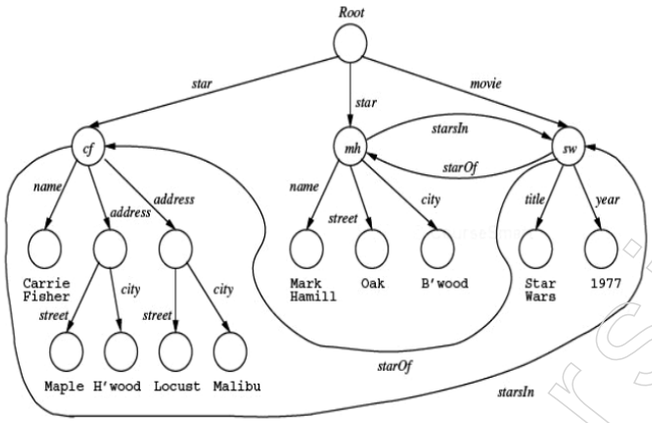
name	*address*
Carrie Fisher	123 Maple St., Hollywood
Mark Hamill	456 Oak Rd., Brentwood
Harrison Ford	789 Palm Dr., Beverly Hills

Star

Notes

title	*year*	*starName*
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone With the Wind	1939	Vivien Leigh
Wayne's World	1992	Dana Carvey
Wayne's World	1992	Mike Meyers

Star in relationship



Corresponding Semi structure Data representation

(ref:cs561-spring 2012 wpi, mohamed e itabakh)

Here

- Leaf nodes represent the data
- Either objects or attributes represent by Internal nodes .
- Each link is either an attribute link or relationship link

Characteristics of semi-structured data are

- Missing or additional attributes
- Multiple attributes
- Different types in different objects
- Heterogeneous collections

Notes

Unit - 4.7: Issues on Implementation

A database in a RDBMS be a special form of semi structured data involves few anomalies likes

Node correspondence: Every node v in data graph D is associated with one and only one node V of role R in graph G .We call V in the schema the definition node of the data node v.

Edge correspondence: A component edge in data graph D is represented by a solid arrow line. If v1 is connected to v2 through a component edge e with tag T, then their definition nodes in G must be likewise connected through a component edge E with tag T .

We call the edge E in G the definition edge of the edge e in D. Likewise for the referencing edge which is represented by a dashed arrow line.

Root correspondence: For each node v , there must be another node w whose definition node is a root node in G, that means either v is w itself or v is connected to w through component edge.

Data type correspondence: Each node v whose definition node is a leaf node V with data type in G must be associated with a data of the matching data type.

Cardinal correspondence: For each component edge e with tag T connecting u to v if the tag associated with e's definition edge is not suffixed with "*"then u cannot connect to another node w with the same tag T.

Unit - 4.8: Indexes for Textdata:

Notes

Full text index: Text Index an index associated with the XML document to search full-text from all of the content in the XML document. The XML tags in the document are transparent to the text for allowing quick and efficient text searches on the document. In other words, we can say the text index is created in the database on the content to removed all XML markup.

By default the text index is created on the values all element/attribute. The XML document store the values of all elements and/or attributes in an SDE_XML_IDX table which supports full doc text queries.

For example: In the following XML document, a text index will search only for values of elements such as “New India”, “15”, and attribute values such as “YES” for the LISTED attribute in MARK tag.

<gn>

<coverage>New India</coverage>

<suptheme>15</suptheme>

<MARK NUMBER=”1” LISTED=”yes” TYPE=”natural”/>

</gn>

BTREE index: Science Full-text index is bigger and slower than BTREE indexes so updating from outside of the transaction is inefficient when used to index feature property sets. And also, full text indexes are not available on all databases. To overcome the limitation of full text indexes, the tag-indexing model has been extended to include Numbers, Strings, and Text indexed using a BTREE, while text uses the full-text indexer. This implementation supports only simple strings or single words using indexing technology for every database.

To implement BTREE indexes, a new column, STRING_TAG, has been added to the existing SDE_XML_IDX table. Since the value of this column assign to be small size set as 250 so at the time of inserting an XML document, the DOUBLE_TAG, STRING_ TAG, and TEXT_TAG columns will be populated with appropriate values based on tag type.

Thus, we set a new index type, SE_XML_INDEX_VARCHAR_TYPE.

In BTREE, By default, only string will be indexed, and a full-text index will be created on text tag values.

The following XML document, consider the “metadata/idCitation/resTitle” node for full-text index and the “metadata/idCitation/citRespParty/rpOrgName” node for BTREE index.

<?xml version = \”1.0\” ?>

<metadata>

<Esri MetaID=\”1000\”>

Notes

<idCitation>

<resTitle>

Migratory Birds and Spread of West Nile Virus in the Asia.

</resTitle>

<citRespParty>

<rpOrgName>ACASIAN</rpOrgName>

</citRespParty>

</idCitation>

</Esri>\

</metadata>

Example 1

Find all documents with a “metadata/idCitation/resTitle” node equal to “West Nile”

Xpath: //metadata/idCitation[resTitle = “West Nile”]

SQL:

SELECT sde_xml_id

FROM sde_xml_idx<xmlcolumn id> x INNER JOIN sde.sde_xml_index_tags t

ON x.tag_id = t.tag_id

WHERE t.tag_name = ‘//metadata/idCitation/resTitle’ AND

CONTAINS (x.text_tag, ‘West Nile’) > 0;

Example 2

Find all documents with a metadata/idCitation/citRespParty/rpOrgName node equal to “ACASIAN”.

Xpath: //metadata/idCitation/citRespParty[rpOrgName = “ACASIAN”]

SQL:

SELECT sde_xml_id

FROM sde_xml_idx<xmlcolumn id> x INNER JOIN sde.sde_xml_index_tags t

ON x.tag_id = t.tag_id

WHERE t.tag_name =

‘//metadata/idCitation/citRespParty/rpOrgName’ AND

x.string_tag = ‘ACASIAN’;

Different predicates are used for searching text and string. For text, use the CONTAINS function and for string, use usual text operator “=” or “!=”.

Notes

For indexing, a text is accomplished using the database's full text indexing capabilities. Which words will be used in the database's text index it depend on the indexing rules which are used. Because every RDBMS may use slightly different rules to index the same text. Mainly keep it in mind that the words which have meaning, like as river, pollution, and population, will be considered as text and include in to the text index, but articles such as and, the, and in will not be included. Different rules are used to index text are written in different languages.

Exercise

1. What is the difference between HTML and XML
2. Which are the area of application of XML language?
3. Describe the components of web interface?

Fill in the blanks with proper alternatives .

1. Eliminate all the compatibility issues to make ……..should look good.
   a) Interface
   b) B0 web page
   c) Web site
   d) None
2. XML document is an ordered……,
   a) Tree
   b) Graph
   c) Path
   d) none
3. XML Schema is an XML application also called ……definition of XML
   a) Syntax
   b) Graphical
   c) Architectural
   d) All the above
4. Querying XML means ……for XML
   a) SQL
   b) HTML
   c) DQL
   d) All
5. Main things of XQuery expressions are…….
   a) Path expressions

Notes

b) Construct element

c) Flower (FLWR) expressions

d) All

Define from the following statement which are either true or false.

1. First-order predicatelogic called Xpath.
2. Xquery ispowerful querylanguage for XMLdata.
3. In XML using Data structures for storing XML data
4. WML styands for Wireless Markup Language
5. World Wide Web Consortium's Resource Description Framework is called RDF.

Answer Keys

Question	Answer	Question	Answer	Question	Answer
1	a	2	a	3	a
4	a	5	d		

Question	Answer	Question	Answer	Question	Answer
1	T	2	T	3	T
4	T	5	T		

Module - V: Advance Transactions and Emerging Trends

Notes

Key Learning Objectives:

At the end of this module, you will learn:

- Several advance transaction trends.
- Know about saga
- Data ware house and data mining
- Activate database
- Spatial database

Notes

Unit - 5.1: Multi Level Transaction

The main features of these transaction models are that

- The semantic properties of operations may be used to relax the isolation of concurrent transactions.
- Consequence, atomicity is achieved by compensation rather than state-based process.
- Sub transactions can be developed persistent independency on their commit state by which the global visibility is updates.

That advanced transaction models and new correctness criteria for transaction introduced for the following reasons

- To provide better support for long-lived activities.
- Keep relax the classical ACID paradigm just like provide more edibility in concurrent transactions.
- Support cooperation between the members of a group of designers.
- Testing more smoothly with object-oriented data models.
- Capable for more semantics about the operations of advanced DBMS applications.
- To enhance inter-transaction parallelism by exploring the semantics.
- Support model intra-transaction parallelism.
- It allow user-defined and system-defined intra-transaction save points one kind of rollbacks.
- It support conversational transactions.

Established multiple autonomous subsystems in a federated DBMS environment.

When a transaction can meet the items 5, 6, 7, 8, 10 then it called multilevel transactions and open nested transactions and if it meet the demand of items 1, 2, and 4. Moreover, open nested transactions can serve as a basic building block in more sophisticated models for cooperative transactions and other long-lived activities. Unlike many other advanced transaction models, multilevel transactionsis much more efficient and effective.

Unit - 5.2: Long-lived Transactions(Saga)

Notes

The Long-Lived Transaction is a transaction with a whole time larger than the formal database transaction. That means LLT keep hold on the database resources for a long time periods and make a delay termination shorter and common transactions .For reducing these problems introduce saga. The Long-Lived Transactions become a saga if it can be written in a sequence of transactions which must be interconnect with the other transactions. It is guarantees that either all the transaction of a saga are successfully executed or partially executed. Both the concept of saga are simple to implement and have potentiality to improve the performance.

For better understanding consider an example like

Course

Course name: String

Course Objectives: string

Credits: Integer

Bibliography: Publication

Department

It's a simplified design of the domain model of an application for higher education model. In this easiest domain model, a course belongs to a department along with a name, its objectives, the credits score, and the recommended bibliography. The domain consider as a consistent only if all the attributes of course are with a finite value and each department managed its courses. The introducing a new course must be executed transitionally, because other users involved with the system is not able to see the course in an inconsistent state.

- Saga mainly applied to all business transactions like
  - SCM(Supply Chain Management)
  - ERP(Enterprise Relationship Pattern)
  - HCM(Human Capital Management)
- It take a potential step forward for completing
  - Buy
  - Sell
  - Reserve
  - Deposit transaction
- Appling scalar data types(auto-compensating data types)
- User-defined compensation may supported other data types.
- It impels a higher increment of concurrency for multiple local transactions by a Saga
- For loosely coupled services it used ACID property.
- The notion of saga is supports a simple nested transaction like

Notes

- It only supports two levels of nesting where the top level is saga with simple transactions.
- And the outer level does not provide full atomicity that means sagas may seen the partial results of other sagas.

The corresponding features supporteed by are saga described below

User Facilities: From the point of view of an application programmer, it is required to inform about the beginning and end of a saga, it happens through a command begin-saga and followed by the series of begin transaction and end-transaction commands to complete the total transaction like accessing the database, manipulating local variables etc. Sometimes, it can start with a user-defined abort by issuing an abort transaction command which can terminates the current transaction, but if put the command abort-saga then it first abort the running transaction and then entire saga. Finally use end-saga command to complete the saga. In some casesprogrammer initiate the save-point command for saga check point. This command should be used between transactions forced the system to save the running state of the application program with a save-point identifier for future use. Just like if there are execution series like T1, T2, C2, T2, T3, T4, T5, C5, C4, T4, T5, T6 .

And if, after executing T2 for the first time, the system crashed any how then the save-point taken after T1, but at time of restart the system first undo T2 by running C2. Then, the saga can be restarted and T2 re-executed and similarly for the second time failure occurred after the execution of T5. That means if there have no partially recovery sequence but the saga save point automatically restart the transaction. Here C for check-point and T for transaction.

Code Reliability: After a crash in traditional transaction processing system there no needed to restore application code to make the database consistent. If such a failure destroys the code of currently executing transaction, the system logs holds enough information about this which is more than enough to restored the effects of the transaction. But in saga there need to complete a running saga after a crash it is necessary to either complete the destroying transactions or run compensating transactions to abort the saga. In either case transaction systems used abstract data type to handled a similar problem. In saga there have a procedure that when saga starts running, It must assumed that all transactions and recovery transactions are all ready predefined. Its only working If the DBMS can manage code stored reliable code for sagas. For example if 1st transaction T1 of the saga enters into the database all other transactions which may used in future after T1 commits, the remaining saga is ready to start.

Recovery: At the time of interruption of saga by some failure then there have twochoices to recover the executed transactions is

- Backward recovery and
- Forward recovery

Backward Recovery: In backward recovery the system needs compensating transactions. The SEC saga execution component manage sagas as well as for controlling the execution of the individual transactions it call another component transaction execution component (TEC). Both TEC and SEC execute transactions as

Notes

a unit value while. Science the transactions controlled by SEC must be interleaved with other transactions that's why there no need of concurrency control .

All saga commands and database actions are stored through the SEC like all the begin-transaction and end-transaction commands with all database actions, are forwarded to the TEC. For that reason when there occur any abort saga command the SEC starts backward recovery.

Example like assume a saga already executed transactions T1 and T2 and at the halfway of the execution of T3 then abort-saga command is proceed then SEC records the command in to the log instructs the TEC to abort the current transaction T3 which is again rolled back using conventional techniques.

Forward Recovery: In forward recovery, SEC needs a reliable copy of the code for every missing transactions with a save-point for data recovery. The used of save point either specified by the application or by the system totally depending on which saga need to be recall. Like Assume a saga executes transactions respectively T1 , T2, a save-point command, and T3 after that at the time of execution of T4 system crash is occurred . Then for the recovery system first perform through the save-point and try to ensure that the code for running T3, T4, is available or not after confirming the SEC restart the saga.

Notes

Unit - 5.3: Data Warehouse And Data Mining

In modern era, Data Warehouses and Data Mining becomes more necessary and more indivisible for handling huge volume of data, helping business executives to take important decisions analysis data.

Data warehouse

Data Warehouse handled a huge volume of data which make it developers so dynamics as per the fact of response. The main two windows where data warehouse become more precious. These are

1. Economic window.
2. Technological window.

These two windows are in so much close relationship with economic and dynamic markets, just like

Making Trade globalized. By which

- There make a drastic change dramatic sharpening of competition.
- Spectacular changes happened in product's life cycle due to dynamic technologic.
- It can be a collection of extremely high quality requirements.

The economic windows are:

- Increasing the computing power.
- Low Price

We can go through the following conclusions likes

- Existence of informatics systems.
- From any where we can accessed the data.
- The need for information is much more essential.
- Computing power is very high.
- Storage capacity is very large but so cheaper.
- Availability of software tools.

So the above windows are essential for implementing a Data Warehouse. It is a collection of organised data used for the fundamentals of management decision. It contains various types of data to present a coherent image of business conditions of a company at a point of view. It works like a system that monitor the activity of organisations depending on the reports generating through a standard formats for analysing issues and made a coordination decisions. Data Warehouse and Data Mining operations are inter dependent the raw materials of Data Mining supplied by the Data Warehouse.

Basic features of a Data Warehouse are:

1. The decisions taken depends on the analysis of data of operations and transactions in daily basis .

Notes

2. It neither delete any data and nor overwritten.
3. For the integrated system data is collected may from operating systems, may from databases, files etc., and transformed the data in to the representation format.
4. Data integration makes a vital stand to construct a Data Warehouse. The costs required for creating and maintaining a Data Warehouse, are equally divided into following manner for
    - Hardware and data storage systems.
    - Software used to extract, processes, store and analyze data.
    - Professional services.
    - Building a Data Warehouse.

A Data Warehouse is made out of the following levels with their own structure and functionality:

1. **Data sources level:** It refers collect and stored electronic information in Data Warehouse. It mainly uses as data sources are:
    - Mainframe databases (DB2, IBMS, VSAM, ISAM, etc).
    - Client-Server databases (Oracle, Informix).
    - Small databases, from PC's (MS Access).
    - Spreadsheets and other types of files.
2. **Data processing level:** In that level the collected data are converted in to a standard form (Internal format of the Data Warehouse). For that some special applications are used like ETL (Extract, Transform and Load).

Data transformation may require operations of "cleaning" data (accuracy, consistency, totals, subtotals etc).

3. **Data Warehouse level:** By using relational databases stored the data for certain period of time in Data Warehouse level .
4. **Reporting level:** This level generates and analysis reports for monitoring organizational activity. This procedure can be done by using some specific tools like:
    - Business intelligence tools;
    - Executive information systems;
    - Enterprise Information Systems;
    - OLAP;
    - Data Mining and KDD

(Knowledge Discovery in Databases), that uses statistic analysis techniques, "form" recognition to discover correlations, rules, knowledge, etc.

5. **Metadata level:** Here metadata is represent including administration information of the Data Warehouse like last date of update, number of users connected etc.
6. **Operations level:** This level mainly used for loading, manipulating and extraction of data in the Data Warehouse. And also represented by the user management, security, capacity and other administration functions also.

Notes

There are 3 types of a Data Warehouse implementation for

- Analytical,
- For standardization of reports and
- For homogenization and consolidation of data.

Analytical. It needs a lot of attention from design to implementation. and It can addressed who can interpret the data.

For standardisation of reports. These are the most common types known on the market (80%) because they are easy to interpret based on developing standard Reports.

For homogenisation and consolidation of data: Here combine multiple sources of information in order to data and Data Warehouses double their size within the first 12 to 18 months.

Data Mining

Along with the Data Warehouse the data mining is assist organisations to take decisions. Data Mining is a “deeper search” in the both data from the Data

Warehouse and also from other data is called source data. It also known as “knowledge discovery in large Databases” by using powerful IT instrument to extract useful information.

Data Mining is a process to extract knowledge from Data Warehouse.

Using unknown knowledge extracting new unanticipated knowledge and these knowledge must be translated as per real requirement.

In Data Mining techniques extracting information can be predictive or descriptive , used for describe an event.

The area where Data Mining are used these are :

- Analysis sales to established customer behaviour.
- Medical research.
- Fraud detection and maintenance, electronic fraud and (cyber) terrorism.
- Risk analysis.

Artificial Intelligence technologies used in Data Mining tools to process and extract data. Artificial Intelligence solutions are more and more present in the data analysis software for business activities. Business Intelligence solution is more appropriate for both the management departments and other departments where each and every department can use for specific work.

The advantages are:

- In financial department it can be used to analyze data and generate reports, financial statistics and help to take important financial decisions.
- Operations department estimate stocks depend on requests to make the production process more efficient.

Notes

- Analyzing the response marketing and sales department can promote and marketing campaigns, and estimate sales by analysis the market behaviour.
- Relaxing IT departments from tasks.

Implementing the system

To designing and implementing a Data Warehouse and Data Mining must go through some necessary stages like

Stage 1. Implementation decision:

When a person or a n organization want to implement a Data Warehouse then he try to realize my asking him/her self “Do I really need this?”

The answer may be multiple.

Ten years ago answer may be probably “No” but now days it must be emphatically “Yes”.

Today’s companioning market companies are forced to store a huge volume of data in a very short time to take a decision instantly which helping by data warehouse and data mining.

Stage 2. The analysis of the existing economic system:

At this stage, a strategic analysis is referring to the company’s assets and the ways to achieve these objectives and analyzing the existing information. From the existing economical structure, retrieve the consequent data and management factors take a quick and correct decision.

Stage 3. Analysis of the existing IT system

At this stage, 2 things must be established:

If the existing IT infrastructure can be used for implementing Data

Warehouse and Data Mining.

Whether the organisation has a well defined and formalized security policy.

Stage 4. Data Warehouse architecture design

The existing architecture can be used in the future architecture because we try to use re used technique to implement a new technique. Here define physical and logical configurations, the data, the necessary applications and financial and architectural support are designed.

Stage 5. Selection of the technological solution

In this stage identify the implementation tools of data and applications, and the technical and architectural support need for the tools. Selecting these tools depending on there structure and the complexity. On the basis of their functions, these tools are classified into some categories likes:

- Transforming and extracting time,
- Data cleaning,
- Data loading and refreshing,
- Data access,

Notes

- Security providing,
- Version control and configuration management,
- Database management,
- Data backup and recovery,
- Disaster recovery,
- Performance monitoring,
- Data modelling,
- Metadata management.

Stage 6. Development

At this stage the front-end design of the applications are define in detail level by level for every operation and also designed the data stores mechanism including data quality and metadata as per the purpose of data extraction.

Stage 7. Testing and implementation:

In this stage total design and all elements are merging to build a system and after that the system will tested properly.

Stage 8. Operation and maintenance:

In this stage the complete application are to be installed, evaluate it's performance and security level and maintained as per the requirements.

Journey of implementation

At first define a strategic plan for initial and further implementation as per the financial condition, time and have specialists of the company. In the case of financial resources

Then implement the lower level Data Warehouse and Data Mining .

Creating and developing mainly depends on the following three stages:

- Implementation from scratch.
- Improving the existing system.
- Converting an older system.

Implementation from scratch: In this process there have an advantage that it not need to audit just modify the scratch which implies a higher cost benefit.

Improving the existing system: By this procedure there no need to extra investment in rebuilding the application but it would may not satisfy the new age of software conditions, science it's a reformation of old one but it should be applicable as per the money consideration.

Converting an older system: It is much more complex than the other procedure because here operating system may make an issues, the existing data base may configure and also it's a time taken procedure.

Unit - 3.4: Active Database

Notes

We know about the conventional database model which is also known as the passive data base, but here we discuss about the active database. In active database recognise the predefined situations in database and then trigger (it's a procedure that runs automatically when a certain event occurs in the DBMS) start actions like database updates, stop calls an external programs etc.

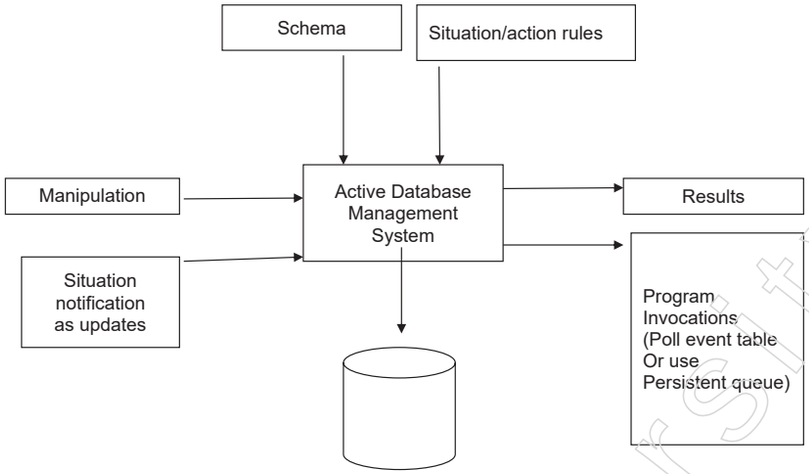


Fig 5.4.1 process diagram of activate database

ADBMS :

- Provides all primitives of regular DBMS along with
- Definition of application for defining situations.
- Triggering of application for defining reactions.
- Used for computation of derived data.
- View manifestation of derived data like

e.g. Enumerated incremental issues of view of sum of salaries for each department.

Salsum (d_id, total),

Enumerated from employee(ssn,d_id,salary)

or invalidation of materialised view when relevant update occurs

  - Rematerialize view if materialised view invalid
- In Oracle materialised views can defined all the result query.
- Science it's not standard SQL so triggers provide an alternative Semantics of ECA rules.

ECA(Event-Condition-Action):

Events at databases-

- Update of a single database record like row addition, deletion .
- Parameterized by pseudo tables in a single row (added, updated, or deleted) by

Notes

REFERENCING clause.

Conditions are checked- The SQL conditional query on database state

Like a database query

empty marks => condition is FALSE

non-empty marks => condition is TRUE

Actions are happened if conditions are satisfied- Database update statements like

Stored procedure call

Unconditional (Event - Action) rules like:

ON ... DO

Condition-Action rules

Not used in databases

- Difficult to identify situation when rule triggered both for user and DBMS
  - SQL + procedures
  - All data actions performed within the same transaction by the trigger .

Triggers :

- It's a procedure that automatically runs If a certain event occurs in the DBMS.
- Implement the Event-Condition-Action rules
- Procedures perform some action like
  - Certain value checking.
  - Fill by some values.
  - Other records may
    - Insert
    - Delete
    - Update
- Checked business constraints are satisfied or not.
- Either cancel or roll back the forbidden actions as per requirements.

E.g: Event –Action rule with no condition

EMPLOYE1(S_SN, D_NO, SALARY)

SALSUM(D_NO, TOTAL) <= Materialized from EMPLOYE1

CREATE TRIGGER EMPLOYE1_SALARY_MATERIALIZATION

AFTER UPDATE SALARYON EMPLOYE1<--- Event

REFERENCING NEW ROW AS N_ROW, OLD ROW AS O_ROW

Notes

```
FOR EACH ROW <--- every single updated row
BEGIN <--- Action
UPDATE SALSUM S
SET TOTAL = TOTAL – O_ROW.SALARY FROM O_ROW
WHERE S.D_NO = O_ROW.D_NO
UPDATE SALSUM S
SET TOTAL = TOTAL + N_ROW.SALARY FROM N_ROW
WHERE S.D_NO = N_ROW.D_NO
END.
```

Example of ECA to maintain constraints.

Department table named DEP with number and manager's S_SN:

```
DEP(D_NO, MGR_S_SN)
CREATE TRIGGER SALARY_SITUATION1
AFTER UPDATE OF SALARY ON EMPLOYE1<--- Event
REFERENCING NEW ROW AS N_ROW, OLD ROW AS O_ROW
FOR EACH ROW <--- C and A per updated row
IF N_ROW.SALARY >(SELECT M.SALARY <--- Condition
FROM EMPLOYE1 M,DEP D,N_ROW
WHERE
N_ROW.D_NO = D.D_NO AND
D.MGR_S_SN = M.S_SN)
THEN BEGIN <--- Action
UPDATE EMPLOYE1 E
SET SALARY = O_ROW.SALARY*0.8 FROM O_ROW
END
```

Manager's need SALARY_CONSTRAINT

```
CREATE TRIGGER SALARY_SITUATION2
AFTER UPDATE OF SALARY ON EMPLOYE1
REFERENCING NEW ROW AS N_ROW, OLD ROW AS O_ROW
FOR EACH ROW
```

Notes

```
IF N_ROW.SALARY < (SELECT E.SALARY
FROM EMPLOYE1 E,DEP D, N_ROW
WHEREE.D_NO = D.D_NO AND
D.MGR_S_SN = N_ROW.S_SN )
THEN
ROLLBACK
```

For assertions :

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK(NOT EXISTS
(SELECT *
FROM EMPLOYE1 E, EMPLOYE2 M,
DEP D
WHERE E.SALARY> M.SALARY AND
E.D_NO = D.D_NO AND
D.MGR_S_SN = M.S_SN))
```

Above constraint after each update to any of the tables EMPLOYE1 or DEP, does not scale the inefficiency.

Assertions cannot make different compensating actions as per the situation.

Cautions:

It have powerful mechanism:

Small statement =>greater behavior changes.

Careful design Required

Consequences of rule of specification or changes.

Unnecessaryusing triggers avoid.

Unit - 5.5: Spatial Database:

Notes

Spatial data is a kind of data in a form of spatial or geographic reference by which they can be identified and located with two-or three-dimensional space . The Spatial Database management system is a kind of software module which supports spatial data models, spatial abstract data types (ADTs) and ADTs are supportable query language. SDBMS works with spatial indexing and efficient algorithms by which spatial operations are processed. For query Optimisation it go through domain specific rules.

Example of SDBMS are:

Oracle Spatial data cartridge,

ESRI SDE can work with Oracle

Has spatial data types (e.g. polygon), operations (e.g. overlap) callable from SQL3 query language

Has spatial indices, e.g. R-trees

Beneficiary Are of SDBMS:

Domains

Several important application domains used spatial data and queries. Like

- Army: Identify any enemy troop movement since few times back.
- Insurance: Risk Manager identify the people who can affected by which it's may natural calamities, may be accident etc.
- Medical Doctor: Based on this patient's MRI are they treated somebody with a similar condition or not?
- Molecular Biologist: The geometrical structures of the amino acid biosynthesis gene in the genome is found in any other sequence feature map in the database ?
- Astronomer: To find all blue galaxies within 2 arcmin of quasars.
- Climatologist : Test the various global warming model.
- Corporate Supply manager: Decide on the base of customer in where the new Data warehouse is placed .
- Transport Specialist : Expand the rout map with minimum traffic and minimum time dependency.
- Urban Development Specialist: Decide the new urban development land for a project without violation the highly agricultural area.
- Ski Resort owner: Decide which mountain are easy or safe for beginer's ski run.
- Farmer: How they minimize the use of pesticide in his/her land?
- Golf Entrepreneurs : choose the land for golf with minimum cost, minimum noise, minimum, destruction etc.
- Emergency Services: Like location Sharing for help, location traced for un wanted circumstances .

Notes

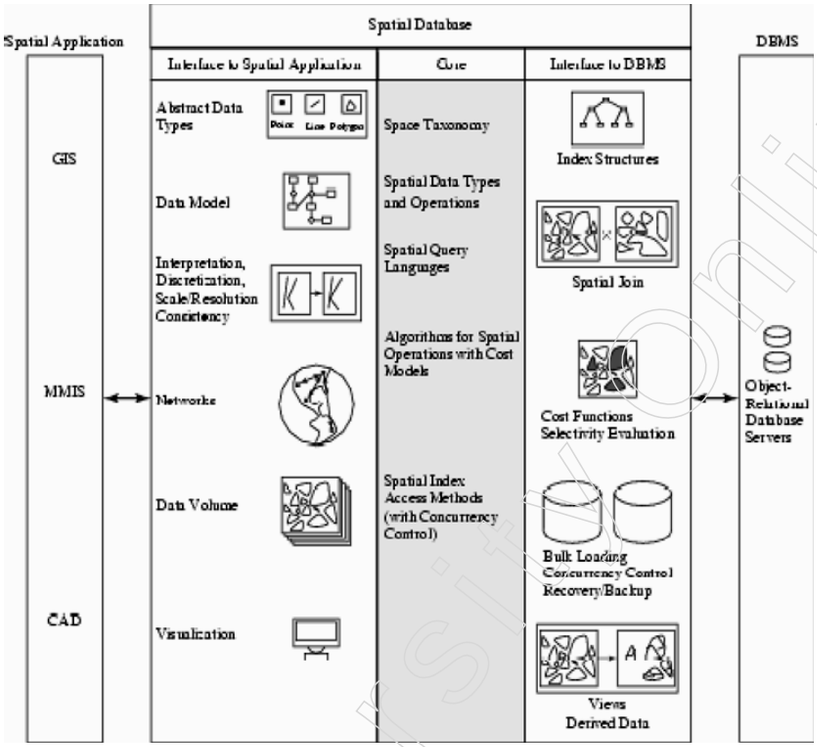
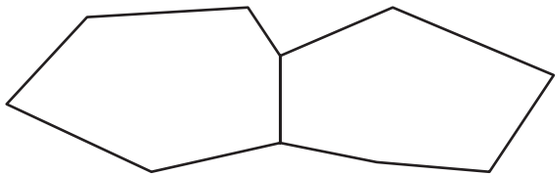


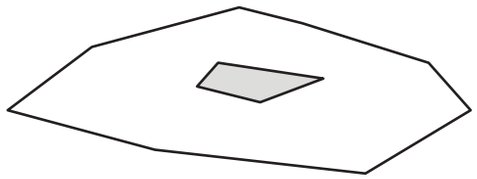
Fig 5.5.1 define the spatial database with three layer

Spatial Taxonomy:

- Gathering of descriptions available to organise space.
- Topology models homeomorphism (deformed into each other by a continuous, invertible mapping) relationships.
- Euclidean(a two- or three-dimensional space In geometry,)space models distance and direction in a plane
- Connectivity link graphs models, Shortest-Path

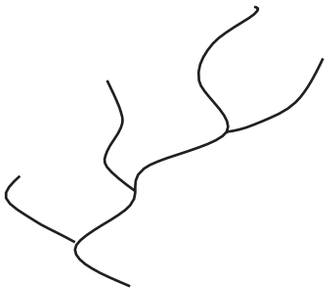


Adjacent polygons



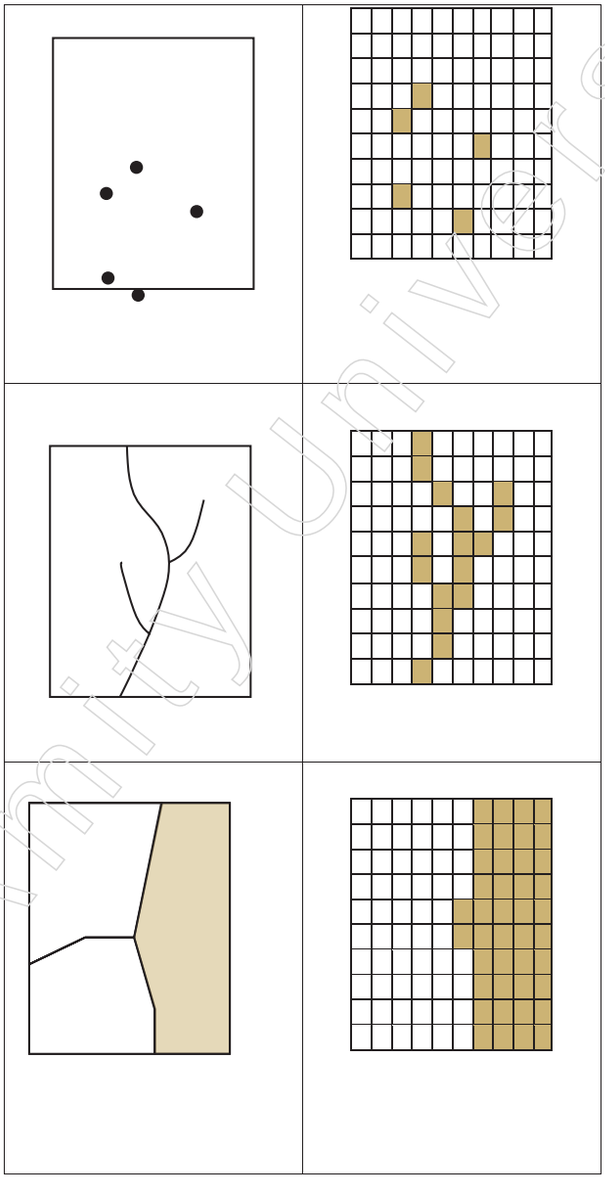
One polygon containedinside another polygon

Notes



Connected stream network

- Rules for identifying the identifiable objects and properties of space.
- Object model used to manage identifiable things like mountains, cities, land-parcels etc.
- Field model help manage continuous and amorphous phenomenon, like wetlands, satellite imagery, snowfall, etc.



Notes

Spatial query language work

- With spatial data types likes point, line string, polygon etc.
- With spatial operations likes overlap, distance, nearest neighbor etc.
- Callable from a query language like

SELECT S.name

FROM Student S

WHERE S. district. Area()> 300

Multi-scan Query Example

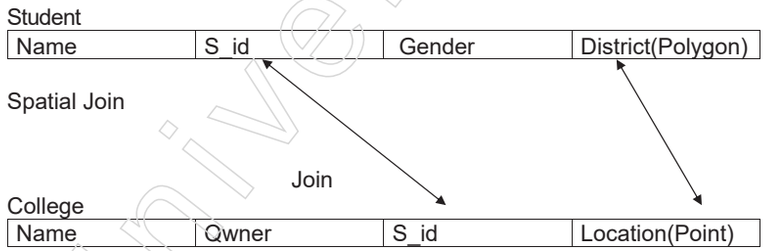
Spatial join example

SELECTS.name FROM Student S, College C

WHERE S.s_id = C.s_id AND AND Within(C. location,S .district)

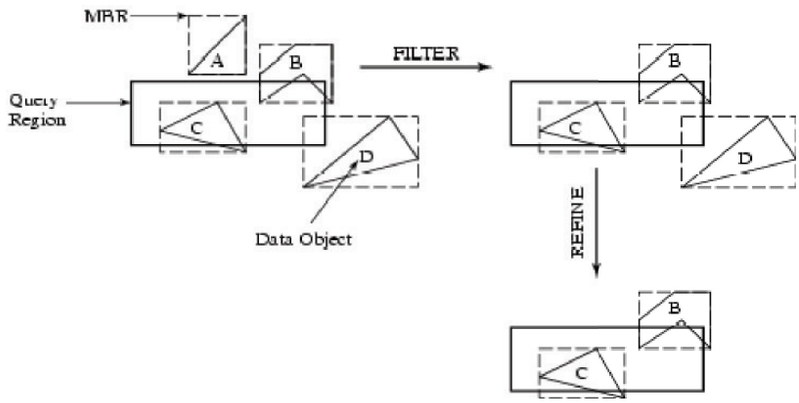
Non-Spatial Join example

SELECTS.name FROM Student S, College C WHERES.s_id=C.s_idAND S.gender=„Male"



Query Processing Using

- Efficient algorithms for answering about spatial queries
- Common Strategy likes filter andrefine where
- Filter -Query Region overlaps with Minimum bounding rectangles with B, C and D B,C andD
- Refine - Query Region overlaps with B andC just like.

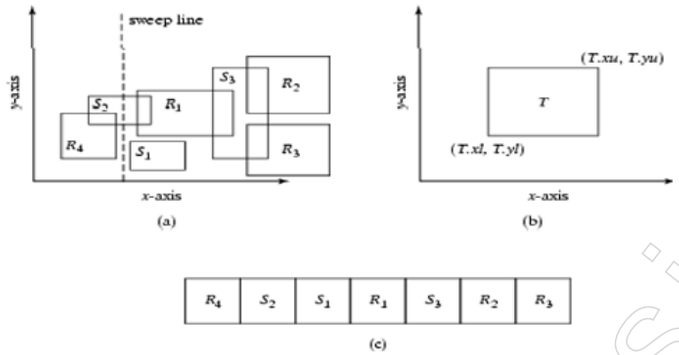


Notes

Query Processing of Join Queries

Example – Determined the pairs of intersecting rectangles

- Two sets R and S of rectangles,
- Rectangle with 2 opposite corners
- Rectangles sorted by smallest X coordinate value



difference between GIS and SDBMSassumptions

- GIS algorithms: dataset is loaded in main memory
- SDBMS: dataset is on secondary storage e.g disk
- SDBMS uses space filling curves and spatialindices to efficiently search disk resident large spatialdataset

Notes

Unit - 5.6: Deductive Database:

Deductive databases is not only used to store explicit information like a relational database but also used to store the rules to enable inferences on the stored data. This is an on growing area of logic programming where mathematical logic is used for direct model computational concepts. These two methods together build a techniques for relational databases in the basis of logic . That means the deductive databases are capable to handled large amounts of information as well as to performed reasoning based on the information.

There have many application areas of deductive database technology.

Areas are ,

Decision support systems:

Any particular organisation's need sufficient information about the current and future status of their resources by which they can plans for the future with the use of effective reasoning.

Another one is

Expert systems:

There have many computing applications by which from a large amounts of information rectified the important facts for simple analysis. Like in a medical analysis and monitoring generate a large amount of data, and obviously error can occur consequently. DD technology can complete the analysis of these data to more efficiently and with a lower chance of error.

Planning systems :

This technology can also support the planning system. Illustrate by an example like

A student is planning to do a course under a university.

Or a passenger planned give a round of the whole world.

For that they need a large amount of information to explore with alternatives and hypotheses. By DD student be able to get advise about pre-requisites and regulations of the subjects and also traveler can gathered knowledge about the financial implications and the rout etc.

Why use DD: It is much more effectible because of it's nature like

- **Declarative:** There have clear understanding about what a database represents .
- **Expressiveness:** If there are some queries which is not only be formulated using relational algebra but also used datalog rules for easy expressed.

Data log Rule

a :- a1, …, an, not b1, …, not bm

where a, a1,..,an,b1,…,bm are atoms and

not is the negation-as-failure operator (n, m can be 0).

Notes

- A program or query and knowledge base is a set of rules.

Rule Means

a :- a1, …, an, not b1, …, not bm

express like

"If a1,…,an are true and b1,…,bm can be assumed to be false then a must be true."

If n=0 and m=0, then a must be true (fact).

If m=0, the rule is a definite rule

Example:

```
nameMovie(M) :- movie(M,X,Y,Z), Z= "name".
durationMovie(M):- movie(M,X,Y,Z), Y>180.
yearMovie(M):- movie(M,X,Y,Z), X<1980.
movieWithSequel(M):- movie(M,X1,Y1,Z1),
                     movie(M,X2,Y2,Z2),
                     X1 <> X2.
```

Do-not-care variables :

```
nameMovie(M) :- movie(M,_,_,Z), Z= "name".
durationMovie(M):- movie(M,_,Y,_), Y>180.
yearMovie(M):- movie(M,X,_,_), X<1980.
movieWithSequel(M):- movie(M,X1,_,_),
                     movie(M,X2,_,_),
                     X1 <> X2.
```

Negation as failure :

```
nonameMovie (M) :- movie(M,_,_,Z), not Z= "name".
notYetProducedMovie (M) :- not movie(M,X,Y,Z).
```

Deductive Database & Relational Algebra:

- Projection

```
nameMovie(M) :- movie(M,_,_,Z), Z= "name".
```

- Selection

```
yearMovie(M,X,Y,Z):- movie(M,X,Y,Z),X<1980.
```

- Join (two relations p(A,B), q(B,C))

Notes

r(X,Y,Z):- p(X,Y), q(Y,Z)

- Union (two relations p(A,B), q(A,B))

  r(X,Y):- p(X,Y).

  r(X,Y):- q(X,Y).

- Set difference (two relations p(A,B), q(A,B))

  r(X,Y):- p(X,Y), not q(X,Y).

- Cartesian product (two relations p(A,B), q(C,D))

  c(X,Y,Z,W):- p(X,Y), q(Z,W).

Implementation Schemes:

The implementation techniques of deductive databases are mainly categorized into three groups:

- Prolog systems loosely coupled to database systems.
- Top-down evaluation with memoing.
- Bottom-up methods.

Prolog Systems Loosely Coupled to Database Systems

Implement deductive databases were to interface a Prolog system with a database system (or a file store) is called prolog database systems. It use Prolog computation and access appropriate database relations on a tuple-at-a-time being. In this approach the systems can be implemented so quickly and easily. But there have a problem in this approach that the result delivery system is extremely inefficient because access of the database in tuple-at-a-time manner which is similar to the nested loop join algorithm performing on several relations simultaneously. Prolog follow the top-down computation method known as backward chaining.

It starts with the query and applies the rules of the program until it arrives at the facts.

The steps of resolutions are as follows:

- Initialize the goal list related to the query.
- Choose a goal (Ai) from the goal list A1,A2, . . . A i . . . An.

Find a rule A :-B1,……, Bm where Aθ = Aiθfor general unifier θ. If there are no such rules then terminatewith failure.

- Update the goal list to (A1, A2,. .. Ai - 1 ,B1,… Bm,, Ai+1,….,an)θ.
- If the goal list is not empty, go back to step 2. Otherwise, terminate with success; an answer to the query is contained in the substitutions.

Step 2 of the top-down algorithm has two forms of non deterministic.

Notes

- The computation rule specifies which literal is to be selected.
- The search rule specifies the order of matching the rules of the program are unified against the selected literal.

These two rules give the shape of the tree developed by the top-down algorithm.

Top-Down With Memoing

To overcome the termination problem of top-down methods on Datalog programs, the technique of memoingis introduced. In backward chaining method the main termination problem is that the refutation procedure does not recognise goals for that there needs so many needless loop.

It ca also build a tree similar to top-down evaluation with few restrictions and differences . like

Answers of subgoals are memoised (tabulised) for future use, when derivation are proceeds from the goal list

A1,...,Ai,...An

to a descendent goal list

(A1,..., Ai-1, Ai+l,... An)θ

the atom Aiθis treat as an answer.

- If the sub goal A is an instance that occurred earlier in a left-to-right pre-order traversal of the tree, then A is not resolved using rules from the program, but is resolved by tabled.
- When a new answer is found, any sub goal that has been resolved using answers must be tested to see is it unifies with the new answer.

Bottom-Up Methods

It also known as forward chaining or fix point computation. It starts with the facts and continuously Appling the rules until it arrives at the query. It mainly used in the semantics of logic programs with many deductive

databases.

This computation method is describing in following steps: Let the

query be q(Q).

- Initialize M the set of known facts. And add the rules of the program:

ans(Q⃗) :- q(Q⃗).

- For each rule A :-A1,…, An, look for substitutions 0 for whichA10,...,

AnO E M. For each such substitution, add AO to M.

- If the set of known facts M has increased, go back to step 2.
- The answer to the query is the set of ans facts in M.

Notes

Unit - 5.7: Multi Media Data Base

Science the multimedia objects require a large amount of memory and disk storage rather than the traditional text or numerical documents . A multimedia database management system should be able to provide the following basic functions:

- Handles image, voice, and other multimedia data types
- Handles a large number of multimedia objects
- Provides a high-performance and cost-effective storage management scheme
- Supports database functions, such as insert, delete, search and update .
- MDBMS needs a sophisticated storage management mechanism, which should also be cost-effective
- MDBMS should take the following issues into consideration:
- Composition and decomposition of multimedia objects
- Operations of multimedia objects with media synchronization
- Persistence object
- Content-based multimedia information retrieval
- Concurrent access and locking mechanisms for distributed computing
- Security
- Consistency, referential integrity, and error recovery
- Long transactions and nested transactions
- Indexing and clustering A multimedia object usually does not exist by itself. A typical multimedia presentation or document may contain a number of objects of various types, such as picture, MIDI music.

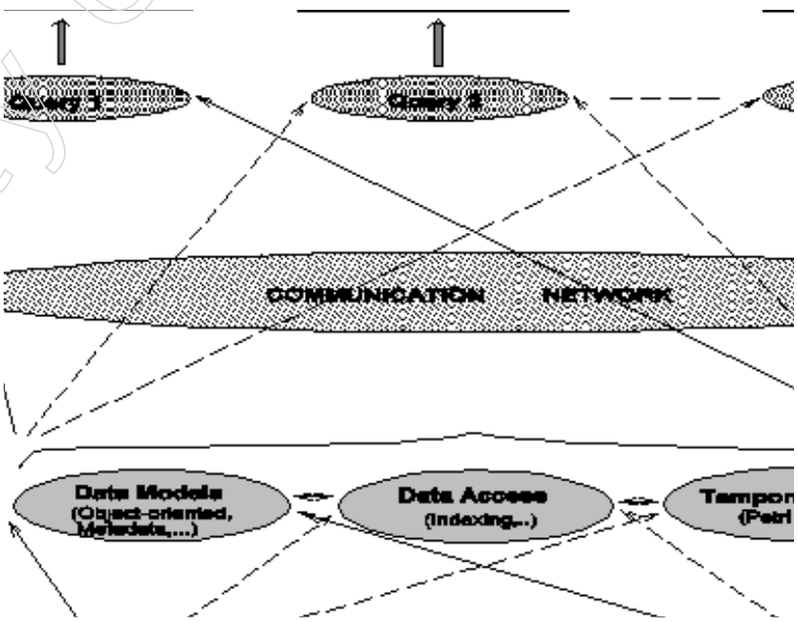


Fig 5.7.1 layer architecture of Multimedia database

Notes

Multimedia Database Applications

- Video-on-Demand (VoD) Servers:
- Store digitized entertainment movies and documentaries.
- Provide services similar to those of a videotape rental store.
- Digitized movies need large storage spaces
- Typically use a number of extremely high capacity storage devices, such as optical disks.
- Users can access a VoD server by searching on stored information such as video's subject title and have a real-time playback of the movie.

Applications.Multimedia Document Management Systems:

- Very general application domain for multimedia databases.
- Involves storage and retrieval of multimedia objects structured into a multimedia document.
- Structuring of objects into a multimedia document involves:
- Temporal relationships among the objects composing the multimedia document
- Spatial relationships that describe how objects are to be presented.
- Applications in CAD/CAM, technical documentation of product maintenance, education, and geographical information systems.
- Interesting aspect of multimedia documents: media objects can be distributed over computer networks.
- Authors can work in a collaborative manner to structure the data into a multimedia document.

Exercise

1. What is the use multilevel transaction on saga?
2. What is multimedia dbms what are the effect of it on MIDI.
3. What is the application of spatial database in present life.

Fill in the blanks with proper alternatives .

1. Saga mainly applied to all …….. transactions
   a) interface
   b) business
   c) web site
   d) money
2. in backward recovery the system needs compensating ……Tree
   a) transactions
   b) forward

Notes

c) backward

d) all the above

3. Data Warehouse handled a huge volume of ……..

a) Syntax

b) data

c) object

d) All the above

4. Data Mining are used …….. field

a) Business

b) Medical

c) Education

d) All the above

5. Spatial data is a kind of data in a form of spatial or …….reference

a) geographic

b) Chemical

c) Regional

d) medical

Define from the following statement which are either true or false

1. Video-on-Demand is named as VoD Servers
2. Bottom up approach is also known as forward chaining
3. That means the deductive databases are not capable to handled large amounts of information as well as to performed reasoning based on the information
4. For query Optimisation spatial database go through domain specific rules.
5. Polygon is example of spatial data .

Answer Keys

Question	Answer	Question	Answer	Question	Answer
1	a	2	a	3	d
4	d	5	a		

Question	Answer	Question	Answer	Question	Answer
1	T	2	T	3	F
4	T	5	T		