Introduction

* **DDB** is a collection of multiple, logically interrelated databases distributed over a computer network.
* **D-DBMS** is the software that manages DDB and provides an access mechanism that makes this distribution transparent to the users.
* **DDBS** (distributed database system) technology is the union of: database system and computer network. (Physically distributed) 🡪 DDBS = DDB + D-DBMS

**Data delivery** in three dimensions: delivery modes (pull-only, push-only, hybrid), frequency (periodic, conditional, ad-hoc or irregular) and communication methods (unicast, one-to-many).

**DDBS promises**: transparency (in independent, network-distributed, fragmented, replicated data), improved reliability (through distributed transactions/failure atomicity), improved performance, scalable + expandable system.

**Distributing/Placing data (design)**:

1. Fragmentation way: partitioned/fragmented/non-replicated
2. Distribution way: replicated (fully/partially) – be aware of mutual consistency (lock, timestamp)

**Distributed DBMS issues**:

1. distributed database design (how to distribute/replicate, and directory management)
2. query processing (optimization issues)
3. concurrency control (consistency, deadlock)
4. reliability (resilient to failures, atomicity, durability)

**Database architecture**: client/server systems, peer-to-peer distributed DBMS, multi-database systems.

1. ANSI/SPARC: users – external views – conceptual view – internal view
2. Generic centralized DBMS: applications – interface – control – compilation – execution – data access – consistency – database
3. Architectural models for distributed DBMSs: **autonomy + distribution + heterogeneity**
4. C/S systems: clients – app servers – DB servers – DBs (efficient division, price/performance+)
5. P2P systems: ESs – GCS – LCSs – LISs (user processor elements: user interface handler, semantic data controller, global query optimizer and decomposer, distributed execution monitor)

* LIS: local internal schema
* GCS: global conceptual schema
* LCS: local conceptual schema
* ES: external schema

1. Multi-database systems: users – mediators – wrappers – DBMSs – DBs

Database design

**Problems in relation scheme**: repetition anomaly, update anomaly, insertion anomaly, deletion anomaly

**Relational Algebra**:

1. selection: select rows
2. projection: select columns
3. join: (分为vertical join or horizontal join，结果含有R和T的全部columns)
4. Cartesian product: (R中每一个元组和S中每一个元组前后相连) – 效率很低
5. semi-join: (F是一个predicate/condition，结果仅含有R中的元素)

**Design Methods**: top-down approach, and bottom-up approach.

**Correctness rules for fragmentation**: completeness, reconstruction, disjointness.

**PHF** (primary horizontal fragmentation): ( is a minterm predicate / selection formula / fragmentation predicate from given simple predicates) – completeness and minimality

**DHF** (derived horizontal fragmentation): ( 在表S里先PHF，然后根据PHF通过semi-joint分另一个关联的表R)

**VF** (vertical fragmentation) two methods: (primary key kept) grouping (bottom-up: attribute 🡪 fragments), splitting (top-down: relation 🡪 fragments)

Affinity (how close the attributes are) measures:

Attribute usage value: ( is a query, is PK+column)

Affinity **matrix**:

Clustering algorithm – Bond Energy Algorithm (BEA):

逐一选取n-i**列**（i是已放入CA的数目），尝试放入CA可选位置，使contribution最大。然后CA矩阵左上角亲和度最高，右下角亲和度最低，设计分片规则。复杂度 for m-way partitioning。【有时间的时候详看】

**HF** (hybrid fragmentation): VF then (P/D)HF or (P/D)HF then VF 🡪 tree structure / mixed fragmentation / nested fragmentation

Exercise: P126-128

Query Processing

High level user query 🡪 query processor 🡪 low-level data manipulation commands for D-DBMS

**Generic layering scheme** for distributed query processing: query decomposition (global schema), data localization (fragment schema), global optimization (allocation schema), distributed execution.

**Query optimization Objectives**: minimize a cost function (disk I/O, CPU, network), WAN/LAN, max throughput

**Complexities**:

|  |  |
| --- | --- |
| Operations | Complexity |
| Select, project (without dup) |  |
| Project with dup, group |  |
| Join, semi-join, division, set operations |  |
| Cartesian product |  |

**Types of** optimizers: exhaustive search (costly), heuristics (not optimal)

**Granularity (颗粒性)**: single query (not intermediate results), multiple queries at a time (decision space much larger)

**Optimization timing**: static (compile time, no intermediate results), dynamic (run time, re-optimize for multiple time execution), hybrid (compile using static first, then if estimate size > threshold 🡪 re-optimize at run time).

1. AND / OR / NOT – **conjunctive normal form**: (A and B and C) or (D and E) where NOT is eliminated – **normalization**
2. **Analysis**: attribute name or type error, semantic error (having redundant/useless parts) – query graph / connection graph (one result node, the rests are query relations) – the connection can be used for detecting errors

**Solutions**: reject the query, assume an implicit Cartesian product between the missing connection, infer the missing join predicate A=B using the schema

1. **Simplification (elimination of redundancy)**: remove duplicated predicates (e.g. )
2. **Rewriting (reconstructing)**: the relational algebra query can be translated into an operator tree (many ways: combination of project, select, join, etc.)

**Data localization**: the easiest way is to replace global query to localized queries in each site (fragmentation) – the operator tree is converted to many sub trees. Another way is called reduction techniques:

1. PHF: (若限定条件于分片规则矛盾🡪空关系)

(用来消除空关系)

1. VF: (没有公共属性ENO，外键，则空关系)
2. DHF: 上述两个结合

Query Optimization

The output of optimization is query execution plan (QEP):

Query input 🡪 generating search space (find equivalent joint trees, or considering linear tree or bushy tree 线性树代价最高) 🡪 search strategy (DP, greedy) 🡪 best QEP

**Distributed Cost Model**: 时间=CPU时间+磁盘IO时间+消息传输开始/完成的时间+网络传输时间

**Dynamic Query Optimization**: in execution, do query decomposition + query optimization

Recursion to decompose the query into smaller queries (mono-relation queries) using detachment (select into tempVar, select from tempVar) 🡪 tuple substitution (按照值域展开成多个用于执行，首选值域最小的sub-R)

Choose best access method to the Relation based on predicates. (irreducible 两点链/k点环)

**Static Query Optimization**: accurate cost model is the key (added CPU and IO, found in history). Nested-loop (蛮力遍历，要有索引，不然太慢) and merge-join(排序，然后相等连接就很快)

**Hybrid Query Optimization**: static first, when running and finding not good 🡪 dynamic. (Dynamic QEPs are produced during compile-time, with choose-plan operations available at runtime based on parameter passed in or cost)

**Join ordering**: collect all and join all

**Semi-join based algorithm** for joint ordering: join locally and combine all – 节约传输

**Distributed Query Optimization**:

1. Dynamic: 考虑中间site的数据量大小，使得总传输**量**最小 – 处理站点应当是存储最多数据的
2. Static: ship-whole or fetch-as-needed (using semi-join)
3. Hybrid: (hard to estimate costs) static plan with operation ordering and access method without considering sites 🡪 dynamically select/copy data allocating to a selected site

**Transactions – (*only have concepts*)**

**Definition**: a collection of actions that make consistent transformations of system states while preserving system consistency.

(Concurrency transparency, failure transparency)

**Termination conditions** of transactions: commit, abort, undo, rollback

**Characterization**: Read set (RS), write set (WS), base set(BS=RS+WS)

**Formalization** of the transaction concept: atomic operations + termination operation.

**Properties of transactions (ACID)**: atomicity (all-or-nothing), consistency (correctly transform from one state to another state), isolation (the state of db is always controlled by the transaction), durability (persistent after committing)

**Types** of transactions: flat transaction, nested transaction, workflows (有并发的任务)

**Execution schedule**: partial order ( 表示A在B之前执行) over operations of a set of transactions.

**Conflict equivalence**: relative order of execution of the conflicting operations from different histories are the same.

**Serializable Schedule**: each local history should be serializable, two conflicting operations should be in the same relative order in all local histories where they appear together.

Data Warehouse

**Data Warehouse (Concept SITN)** is a subject-oriented (organized around major subjects), integrated (multiple data sources), time-variant (time horizon is significantly longer) and nonvolatile (physically separately stored) collection of data in support of management’s decision-making process.

**Data Warehousing** is the process of constructing and using DW.

**DW conceptual modelling**: star schema (one fact table collected to a set of dimension tables), snowflake schema (many levels of dimensional hierarchy), fact constellations (many fact tables with many shared dimension tables).

*P.S. fact table contains measures and keys to each related dimension tables.*

**DW Models**: enterprise warehouse, data mart, virtual warehouse.

**DW design process**: top-down (starts with overall design and planning – mature), bottom-up (starts with experiments and prototypes – rapid).

**Data cube measures**: distributive (count, min, max), algebraic (avg, minN, standardDeviation), holistic (median, mode, rank).

**DW usage**: information processing, analytical processing, data mining.

OLTP: online transaction and processing (IT professional)

OLAP: online analytical processing (data scientist) –

Roll up, drill down, slice and dice, pivot (rotate), drill across, drill through

OLAM: online analytical mining (based on OLAP)

Why? high quality data in DW, available information from many places (ODBC, OLEDB, OLAP tools), OLAP-based exploratory data analysis, online selection of data mining functions.

**Why data mining**: data becoming extremely big, data cannot directly tell knowledge, it is necessary.

Associated rule mining

**Frequent Pattern**: a pattern that occurs frequently in a data set. (Intrinsic and important property of data set)

**Data mining 2-step process**: frequent itemsets, association rules

**Measures** X 🡪 Y: support () and confidence ()

(If data set too large, mine closed patterns and max-patterns instead)

**Computing frequent itemsets**: incremental process (freq. pattern growth), downward property (Apriori)

**The Apriori algorithm** (idea is: any subset of a frequent itemset must be frequent)

Def: min support = ?

Start from 1-dim (like {A} {B} …), remove invalid ones, to 2-dim ({A,B}, {A,C} …), remove invalid … (每一步都不能含有前面消除掉的invalid的子集合)

**Hash Tree**: "localize" each transaction to relevant leaf nodes of the tree thus reducing the computation (comparison) cost.

Classification

**Supervised learning** (classification): labelled training data (labels used for indicating the class)

**Classification**: predict class label (discrete or nominal) based on the training data (input + label) for new data.

**Decision tree (ID3)**: (column by column) constructed in a top-down + recursive + divide-and-conquer manner. (All training examples are at the root, them categorized from top to down)

Termination condition: all samples in the node belong to the same class.

**Naïve Bayes Classifier**: 🡺

Example, given classes: ,

Good: easy to implement, good prediction results in most cases

Bad: classes are independent 🡪 accuracy lose

**Rule-based classification**:

1. IF-THEN-ELSE rules, can be extracted from decision tree;
2. Using sequential covering algorithm (FOIL, AQ, CN2, RIPPER) to extract rules directly from training data (one rule is learnt each time, removing covered data, repeat).

Clustering

**Unsupervised learning** (clustering): labels unknown, and use measurements and observations to establish the existence of classes or clusters in the data. (Applications: tool to get insight to data, pre-processing step for other algorithms)

**Clustering approaches**: partitioning, hierarchical, density-based, grid-based, model-based, frequent pattern-based, user-guided or constraint-based, link-based clustering.

**K-means**: given k, partition into k subsets, computer centroid (center point) of the cluster of the current partitioning (e.g. mean point), assign each object to the nearest centroid, repeat computation.

* + Strength: – n is the number of objects, k is the number of clusters, t is the number of iterations.
  + Weakness: only applicable for continuous n-dimensional space; need to specify k; sensitive to noisy; not suitable for clusters with non-convex shapes.

**K-medoid**: find representative objects (medoids) in clusters.

PAM (Partitioning around medoids): start from an initial set of medoids, replaces one of them by a non-medoid, if it improves then use it. (like heuristic method, genetic algorithm) – Strength: very effective in small dataset, but not good in big data.

**Hierarchical methods** (agglomerative凝聚 vs divisive分歧): use distance matrix as clustering criteria. (Do not need as input, but need a termination condition.)

Agglomerative Nesting (AGNES): 从最近的开始，逐渐扩大，直到终止条件

Divisive Analysis (DIANA): the inverse order of AGNES