Distributed Databases

Databases, Aarhus University

Ira Assent

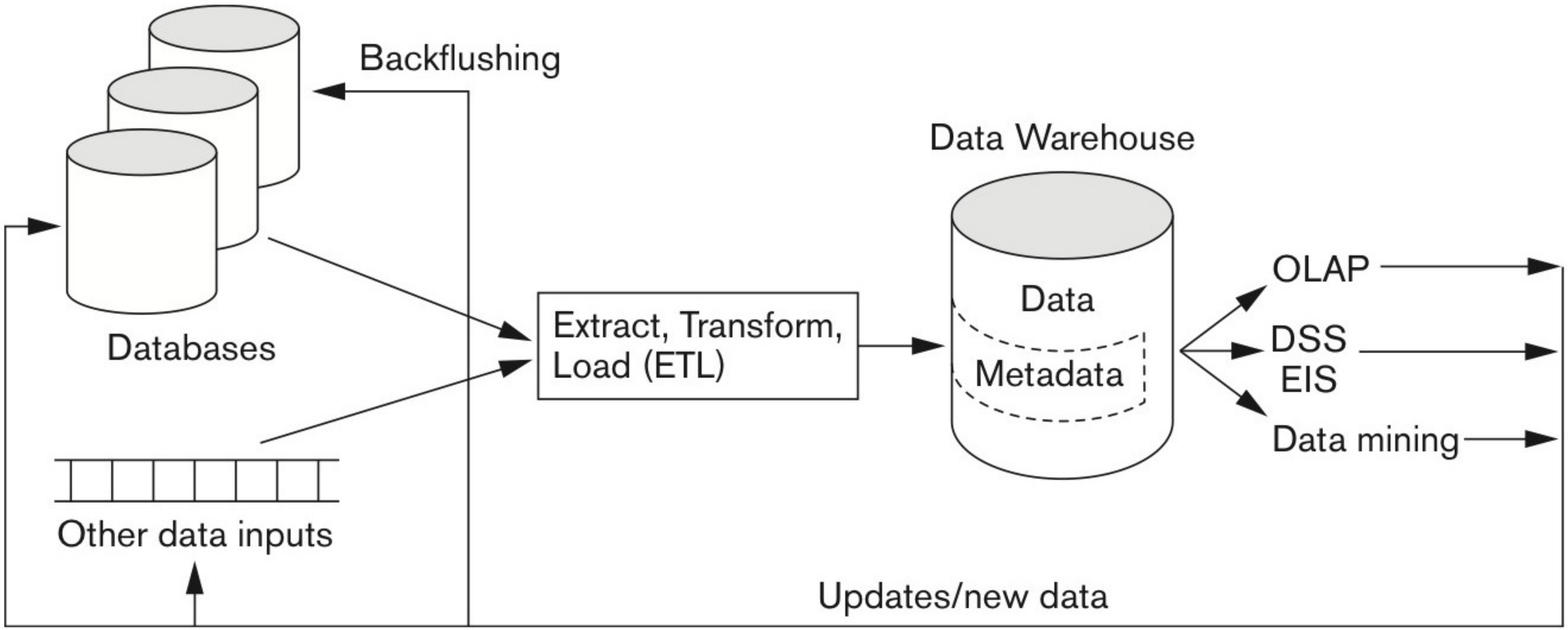
Intended learning outcomes

- Be able to
  - Describe characteristics of distributed databases
  - Explain query decomposition and optimization in DDBs
  - Describe and apply core techniques of MapReduce

ira@cs.au.dk

Recap: data warehouses

- Data Warehouse processing involves
  - Cleaning and reformatting of data
  - OLAP
  - Data Mining

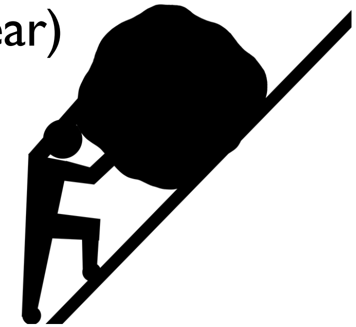


ira@cs.au.dk

ETL – data acquisition

→ A large part of the effort in Data Warehousing lies in **ETL**

- **Extract, transform, load**
- Often greatly underestimated time and effort
- Process of inserting data from the transactional database(s)
  - Different source databases with different schemas
    - Different semantics (e.g. different "years": fiscal vs. calendar year)
- Cleaning
  - Validity and quality of the data
    - Erroneous and incomplete data: difficult to automate
      - E.g. domain constraints
  - Corrected data can be backflushed to the transactional database (e.g. incorrect customer address)
- Converted to data model of Data Warehouse
- Loading of large data volumes is challenging in itself
  - Typically incrementally: go offline for a particular time at regular intervals



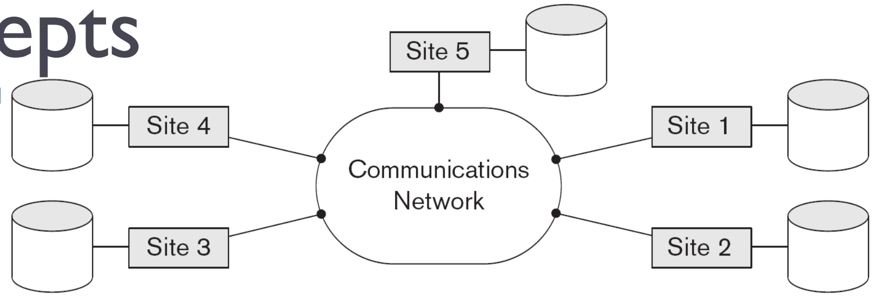
ira@cs.au.dk

Navigating a Data Warehouse

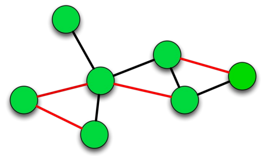
- Functionality to navigate and study data
  - **Roll-up**: Data is summarized with increasing generalization
  - **Drill-Down**: Increasing levels of detail are revealed
  - **Pivot**: Cross tabulation is performed
  - **Slice** and **dice**: Performing projection operations on the dimensions
  - **Sorting**: Data is sorted by ordinal value
  - **Selection**: Data is available by value or range
  - **Derived attributes**: Attributes are computed by operations on stored derived values

ira@cs.au.dk

Distributed Database Concepts



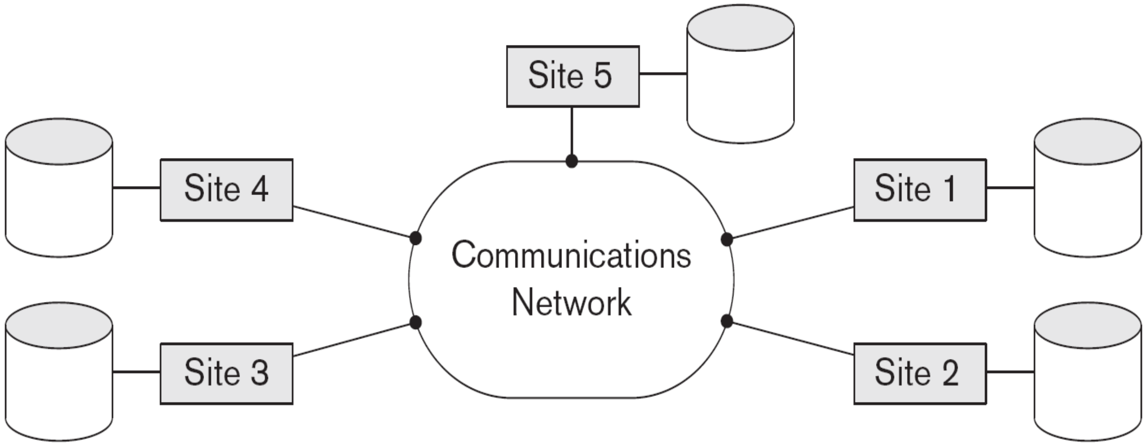
- Distributed computing system
  - Number of processing sites / nodes
  - Interconnected by computer network
  - Cooperate in performing certain assigned tasks
  - Partition big, unmanageable problem into smaller pieces
  - Solve efficiently in coordinated manner
  - Easy way of using more computing power, but with some overhead in terms of coordination / communication
- Distributed Database (**DDB**)
  - Process unit of execution (transaction) in a distributed manner
  - DDB definition
    - **collection of multiple logically related databases**
    - distributed over computer network
    - distributed database management system as a **software system**
      - manages a distributed database
      - makes the distribution transparent to the user



ira@cs.au.dk

Transparency

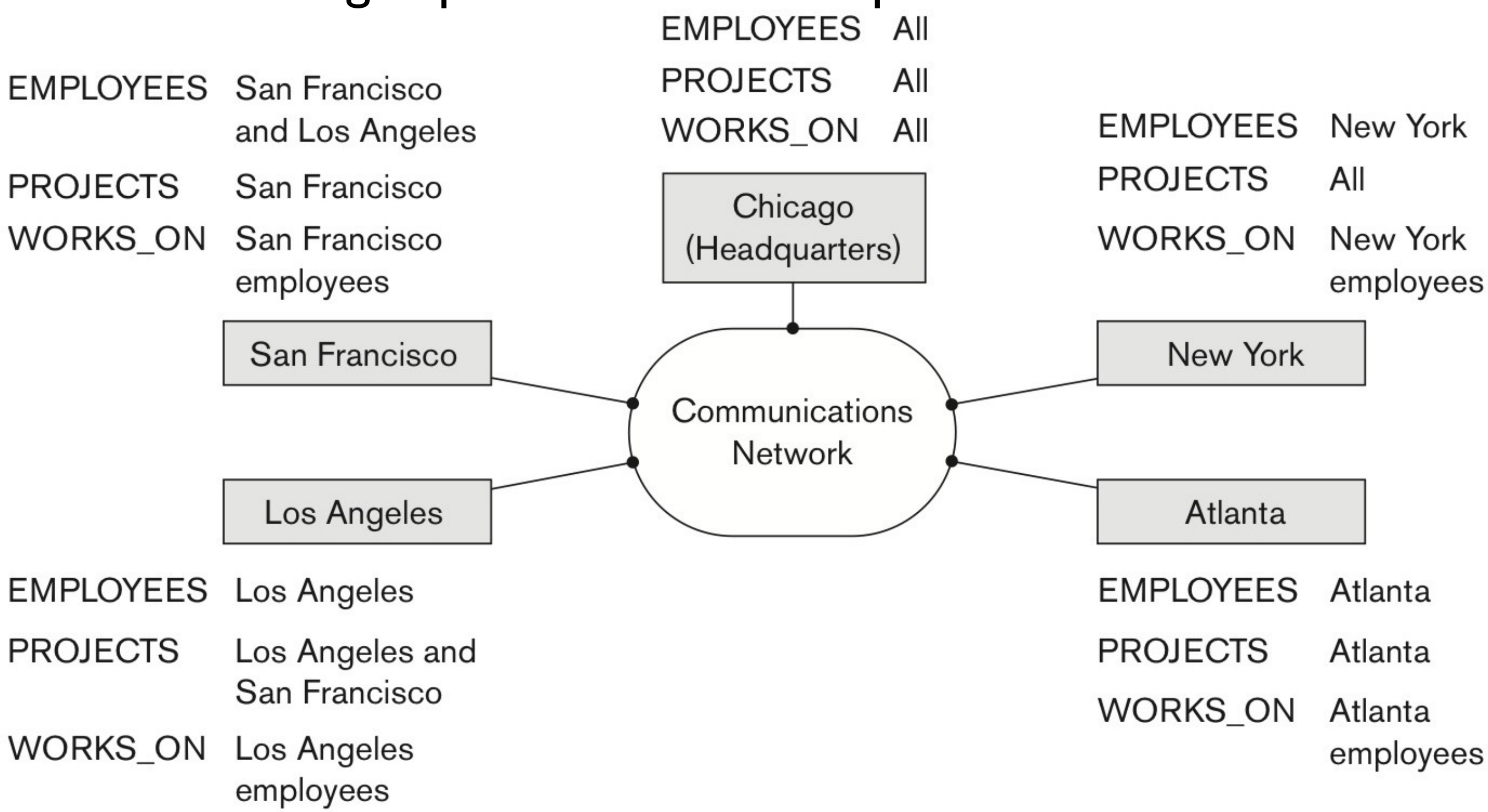
- Transparency ⚠
  - **Hide implementation details** from end users
  - Offers flexibility to user / developer
- Users do not have to worry about operational details of the network
  - Location transparency: access from any location, data can be located on any site
  - Naming transparency: access to any named object (files, relations, etc.) from any site
    - Requires unambiguous names regardless of location, without specifying location
  - More challenging than in centralized databases



ira@cs.au.dk

Textbook distributed example

- EMPLOYEE, PROJECT, and WORKS_ON tables fragmented horizontally and stored with some replication
  - Fragmentation transparency
    - Allows storing tuples or attributes at different sites
  - Replication transparency
    - Allows storing copies of data at multiple sites



ira@cs.au.dk

What is the main reason for fragmentation and replication?

VOTE

A. Improves durability and recovery
B. Improves isolation and atomicity
C. Improves reliability and availability
D. Improves concurrency

ira@cs.au.dk

Fragmentation and replication

- Increased reliability and availability
  - Reliability refers to system live time
    - System is running efficiently most of the time
  - Availability is probability that the system is continuously available (usable or accessible) during a time interval
  - If one node/site in distributed database system fails then others are available to do the job
- Improved performance
  - A distributed DBMS fragments the database to keep data closer to where it is needed most
  - This reduces data management (access and modification) time significantly
- Easier expansion (scalability)
  - Allows new nodes (computers) to be added anytime without changing the entire configuration

ira@cs.au.dk

Horizontal fragmentation

- Also called sharding
- Horizontal subset of relation with tuples which satisfy selection condition
- Consider Employee with **selection condition** (DNO = 5)
  - Tuples satisfying condition constitute horizontal fragment (shard) of Employee
  - Can be stored at particular site
- Selection condition may be composed of conditions using AND or OR

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

Derived horizontal fragmentation

- Derived horizontal fragmentation
  - Partitioning of primary relation (DEPARTMENT) to other secondary relations (EMPLOYEE) via foreign keys
  - Related data fragmented in the same way
    - E.g. anything relating to department 5, in all tables, is found at the same site

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEP_5

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joy									
Ah									
Jar									

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

ira@cs.au.dk

Vertical fragmentation

- Subset of a relation created by a subset of columns
  - vertical fragment contains values of **selected columns**
  - no selection condition used in vertical fragmentation
- Consider the Employee relation
  - vertical fragment can be created by keeping the values of Lname, Ssn, and Bdate
- Because there is no condition for creating a vertical fragment, each fragment must include the primary key attribute of the parent relation
  - all vertical fragments of a relation are connected

Lname	Ssn	Bdate
Smith	123456789	1965-01-09
Wong	333445555	1955-12-08
Zelaya	999887777	1968-01-19
Wallace	987654321	1941-06-20
Narayan	666884444	1962-09-15
English	453453453	1972-07-31
Jabbar	987987987	1969-03-29
Borg	888665555	1937-11-10

ira@cs.au.dk

Representation in relational algebra

- **Horizontal fragmentation**
  - All employees from Department 5

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

ira@cs.au.dk

Representation

- **Horizontal fragmentation**
  - Each horizontal fragment on a relation can be specified by $\sigma_{Ci}(R)$ operation in the relational algebra

$$\sigma_{Dno='5'}(\text{Employee})$$

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

  - Complete horizontal fragmentation
    - Set of horizontal fragments whose conditions $C_1, C_2, \ldots, C_n$ include all tuples in R
    - That is, every tuple in R satisfies (C_1 OR C_2 OR … OR C_n)
  - Disjoint complete horizontal fragmentation: No tuple in R satisfies (C_i AND C_j) where $i \neq j$
  - Reconstruct R from horizontal fragments using UNION

ira@cs.au.dk

How do we specify a vertical fragmentation?

A. $\Pi_{L_i}(R)$

B. $\sigma_{L_i}(R)$

C. $L_i \bowtie L_j$

D. $L_i \cap L_j$

E. $L_i \cup L_j$

Relation R, attributes list $L_1,\ldots,L_m$

Lname	Ssn	Bdate
Smith	123456789	1965-01-09
Wong	333445555	1955-12-08
Zelaya	999887777	1968-01-19
Wallace	987654321	1941-06-20
Narayan	666884444	1962-09-15
English	453453453	1972-07-31
Jabbar	987987987	1969-03-29
Borg	888665555	1937-11-10

ira@cs.au.dk

Vertical fragmentation representation

- A vertical fragment on a relation can be specified by $\Pi_{L_i}(R)$ operation in relational algebra
  - $\Pi_{\text{Lname, SSn, Bdate}}$ (Employee)
  - In SQL attributes listed in SELECT clause
    - SELECT Lname, Ssn, Bdate FROM EMPLOYEE

Lname	Ssn	Bdate
Smith	123456789	1965-01-09
Wong	333445555	1955-12-08
Zelaya	999887777	1968-01-19
Wallace	987654321	1941-06-20
Narayan	666884444	1962-09-15
English	453453453	1972-07-31
Jabbar	987987987	1969-03-29
Borg	888665555	1937-11-10

- Complete vertical fragmentation
  - Set of vertical fragments whose projection lists $L_1, L_2, \ldots, L_n$ include all attributes in R and share only primary key of R
    1. $L_1 \cup L_2 \cup \ldots \cup L_n$ = ATTRS (R)
    2. $L_i \cap L_j$ = PK(R) for any i, j,

    where ATTRS (R) set of attributes of R, PK(R) primary key of R
  - Reconstruct R using FULL OUTER JOIN

ira@cs.au.dk

Mixed (Hybrid) fragmentation representation

- Combination of vertical and horizontal fragmentation
  - SELECT-PROJECT operations $\Pi_{L_i}(\sigma_{C_i}(R))$

1. C = True (Select all tuples) and L ≠ ATTRS(R): vertical
2. C ≠ True and L = ATTRS(R): horizontal
3. C ≠ True and L ≠ ATTRS(R): truly mixed

1. $\Pi_{\text{Lname, Ssn, Bdate}}(\sigma_{\text{True}}(\text{Employee})) = \Pi_{\text{Lname, Ssn, Bdate}}(\text{Employee})$
2. $\Pi_{\text{Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super\_ssn, Dno}}(\sigma_{\text{Dno='5'}}(\text{Employee})) = \sigma_{\text{Dno='5'}}(\text{Employee})$
3. $\Pi_{\text{Lname, Ssn, Bdate}}(\sigma_{\text{Dno='5'}}(\text{Employee}))$

1.

Lname	Ssn	Bdate
Smith	123456789	1965-01-09
Wong	333445555	1955-12-08
Zelaya	999887777	1968-01-19
Wallace	987654321	1941-06-20
Narayan	666884444	1962-09-15
English	453453453	1972-07-31
Jabbar	987987987	1969-03-29
Borg	888665555	1937-11-10

EMPLOYEE 2.

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	380		
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	250		

3.

Lname	Ssn	Bdate
Smith	123456789	1965-01-09
Wong	333445555	1955-12-08
Narayan	666884444	1962-09-15
English	453453453	1972-07-31

Schema

- **Fragmentation schema**
  - A definition of a set of fragments (horizontal or vertical or horizontal and vertical)
  - includes all attributes and tuples in the database
  - satisfy the condition that the whole database can be reconstructed from the fragments
  - applying some sequence of UNION (or OUTER JOIN) and UNION operations
- **Allocation schema**
  - describes the distribution of fragments to sites of distributed databases
  - fully or partially replicated or partitioned
  - Replicated means stored at more than one site

ira@cs.au.dk

Query Processing in Distributed Databases

- Employee at site 1: 10,000 rows, row size100 bytes, table size10^6 bytes

Fname	Minit	Lname	SSN	Bdate	Address	Sex	Salary	Superssn	Dno

- Department at Site 2: 100 rows, row size 35 bytes, table size 3,500 bytes

Dname	Dnumber	Mgrssn	Mgrstartdate

- Query at Site 3: Retrieve employee name and name of their department

$$\Pi_{\text{Fname,Lname,Dname}} \text{ (Employee } \bowtie_{\text{Dno = Dnumber}} \text{ Department)}$$

- Suppose each result tuple 40 bytes long
- Problem: Employee and Department relations not present at site 3
  - Strategies:
    1. Transfer Employee and Department to Site 3
    2. Transfer Employee to Site 2, execute join at Site 2, send result to Site 3
    3. Transfer Department to Site 1, execute join at Site1, send result to Site 3
- Optimization criterion: minimizing data transfer

Which strategy is best?

$\Pi_{Fname,Lname,Dname}$ (Employee $\bowtie_{Dno\ =\ Dnumber}$ Department)

A. Strategy 1

B. Strategy 2

C. Strategy 3

D. Depends on query site

1. Transfer Employee and Department to Site 3
2. Transfer Employee to Site 2, execute join at Site 2, send result to Site 3
3. Transfer Department to Site 1, execute join at Site1, send result to Site 3

Employee at Site 1: 10,000 rows, row size 100 bytes, table size 10^6 bytes
Department at Site 2: 100 rows, row size 35 bytes, table size 3,500 bytes
each result tuple 40 bytes

ira@cs.au.dk

Query transfer cost calculation

Q: $\Pi_{Fname,Lname,Dname}$ (Employee $\bowtie_{Dno = Dnumber}$ Department)

1. Transfer Employee and Department to Site 3
   - Total transfer = 1,000,000 + 3500 = 1,003,500 bytes
2. Transfer Employee to Site 2, execute join at Site 2, send result to Site 3
   - Query result size = 40 * 10,000 = 400,000 bytes
   - Total transfer = 400,000 + 1,000,000 = 1,400,000 bytes
3. Transfer Department to Site 1, execute join at Site1, send result to Site 3
   - Total transfer = 400,000 + 3500 = 403,500 bytes preferred

- But, if query/result site is e.g. Site 2: Strategy 2 only uses 400,000 bytes, if query/result site is Site 1, Strategy 3 only uses 3500, which would make them preferred

Employee at Site 1: 10,000 rows, row size 100 bytes, table size 10^6 bytes
Department at Site 2: 100 rows, row size 35 bytes, table size 3,500 bytes
each result tuple 40 bytes

ira@cs.au.dk

Further optimization using semi-join

Q: $\Pi_{\text{Fname,Lname,Dname}}$ (Employee $\bowtie_{\text{Dno = Dnumber}}$ Department)

- Idea: instead of sending full tables, only send join attributes, then request remaining attribute values for those rows where a join match has been identified
- Semijoin:
  - Semi-join R $\ltimes_{A=B}$ S = π_R(R $\bowtie$ S)

1. Project to Dnumber of Department at Site 2, transfer column to Site 1
   - For Q, 4 * 100 = 400 bytes transferred, if Dnumber is 4 bytes
2. Join transferred Dnumber column with Employee at Site 1, transfer projection to Fname, Lname, Dno from resulting file to Site 2
   - For Q, 34 * 10,000 = 340,000 bytes transferred, if names 15 bytes each
3. Join transferred file with Department, present result at Site 2

> Employee at site 1: 10,000 rows, row size 100 bytes, table size 10^6 bytes
> Department at Site 2: 100 rows, row size 35 bytes, table size 3,500 bytes
> each result tuple 40 bytes, query/result: site 2

ira@cs.au.dk

Example query decomposition

- Guard conditions at site 2 for EMPD5, PROJ5, WORKS_ON5
  - all tuples with Dnum=5 for different tables: DEP5 from DEPARTMENT and related data from other tables (derived fragments)
  - E.g.

    EMPD5

    attribute list: Fname, Minit, Lname, Ssn, Salary, Super_ssn, Dno

    guard condition: Dno = 5

    DEP5

    attribute list: * (all attributes Dname, Dnumber, Mgr_ssn, Mgr_start_date)

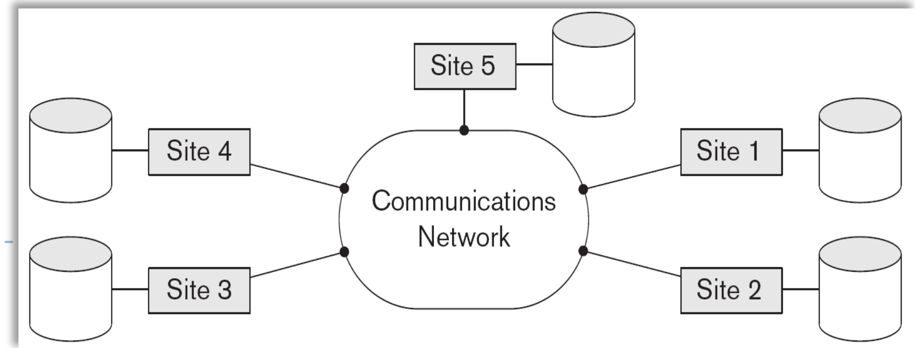
    guard condition: Dnumber = 5

- Query at site 2
  - retrieve names and hours per week for each employee who works on project of department 5

```
SELECT Fname, Lname, Hours
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE Dnum=5 AND Pnumber=Pno AND Essn=Ssn;
```

In this example, all data available locally

DDBMS maps query to fragments



Example insert decomposition

- Insert <'Bo','B','Juul','345671239','2-APR-84','8000 Aarhus', M,33000,'987654321','5'>

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

- Decomposed by DDBMS into row insert requests
  - Full tuple at site 1
  - Projected tuple at site 2

EMPD5
attribute list: Fname, Minit, Lname, Ssn, Salary, Super_ssn, Dno
guard condition: Dno = 5

<'Bo','B','Juul','345671239','33000,'987654321','5'>

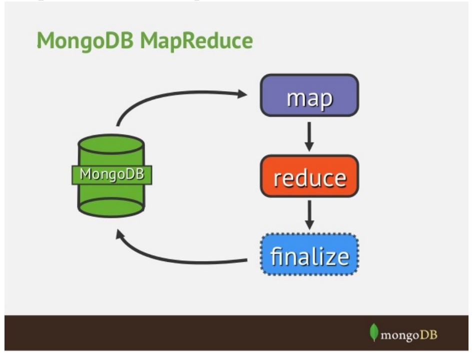
EMPD5

Fname	Minit	Lname	Ssn	Salary	Super_ssn	Dno

ira@cs.au.dk

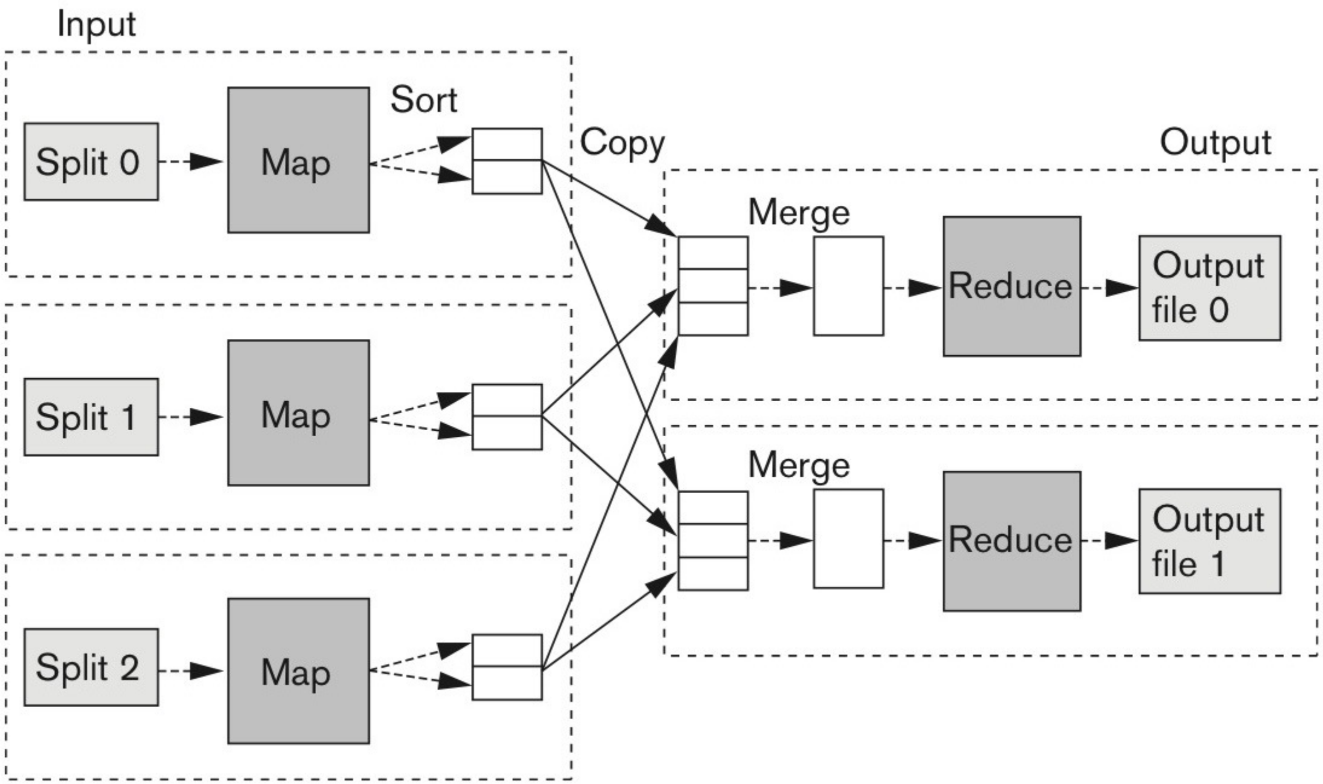
Distributed computation using MapReduce

- Hadoop (Apache) main components: MR and HDFS
- Goal: efficient and scalable distributed processing
- MR: MapReduce programming model (Google)
  - User writes program in function style of map and reduce tasks
  - Automatically parallelized and executed on large clusters
  - Runtime system handles many issues in parallelization
    - Fault tolerance, data distribution, load balancing, task communication
    - No distributed systems experience required by user/programmer
  - Data model: key-value pair
- Applies map operation to each record; produces intermediate key-value pairs; applies reduce to all values with same key
  - Map(k1,v1)→list(k2,v2);
  - Reduce(k2,list(v2))→list(k3,v3);
- Available e.g. in MongoDB
- Other NoSQL systems



ira@cs.au.dk

Overview of MapReduce execution

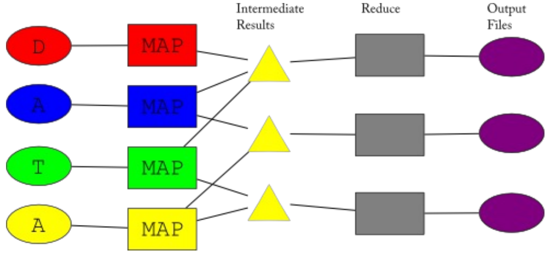


- Split data into chunks that are handled in distributed manner
- Map operation on each record; produce intermediate key-value pairs
  Map(k1,v1)→list(k2,v2)
- Reduce operation on shared key and list of its values to produce output chunk
  Reduce(k2,list(v2))→list(k3,v3)
- Easy to distribute chunks, balance workload, handle node failures

ira@cs.au.dk

Inverted Index example

- Goal: build inverted index based on words in document
  - i.e., look up a word and find where it occurs
  - Useful for example in document search, document grouping, etc.
  - D123: Trump tweets. D124: Trump watches TV. D125: POTUS tweets…



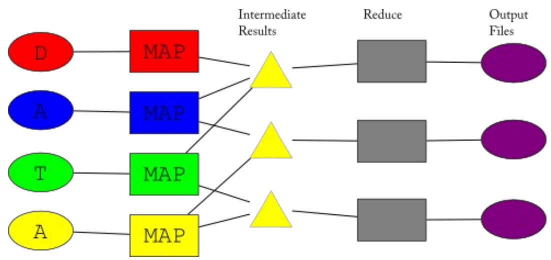
- Map function parses each document
  - Emits (word, document_id) pairs
    - E.g. (Trump, 123), (tweets, 123), (Trump, 124), (watches, 124), (TV, 124),…
  - Reduce function takes all pairs for a given word, sorts them by document_id, emits (word, list(document_id) pair
    - E.g. (Trump, (123,124))
  - Inverted Index is set of all pairs
    - E.g. {(Trump, (123,124)), (tweets, (123,125)),(watches, (124)),…}

ira@cs.au.dk

Sort-Merge Join in MapReduce

Example join (Employee $\bowtie$ $_{\text{Dno = Dnumber}}$ Department)

- Map function reads blocks from both tables and sorts join attribute values
  - Here Dno, Dnumber
  - Emits ((tag, key), value) pairs
    - E.g. ((0, 5),1) where 0 refers to the smaller table Department, 5 refers to Dnumber=5, 1 refers to row 1 in Department
    - i.e., the key is here a pair of (tag, key) so that the output is ordered with the rows from the smaller table with that key and then the rows from the larger table with that key
    - Reduce function takes all pairs for a given join value (Dno/Dnumber) and generates join for all rows
    - Emits (row1, row2) tuples where row1 and row2 are from table 0 and 1, respectively, and share the same join value
    - E.g. (1,1); (1,2); (1,5); …
      - Essentially a list over the rows that match in the join



ira@cs.au.dk

When is this join efficient?

A. For any equi-join
B. For any equi-join with single join matches in one table only
C. For any equi-join with single matches in both tables
D. For any equi-join with single matches in both tables on exactly one join attribute per table

ira@cs.au.dk

When is this join efficient?

A. For any equi-join
B. For any equi-join with single join matches in one table only if many matches in both then main memory buffers may be exceeded…
C. For any equi-join with single matches in both tables
D. For any equi-join with single matches in both tables on exactly one join attribute per table

ira@cs.au.dk

Employee * Project

Strategies:

1. Transfer Employee and Project to query site, perform join
2. Transfer Employee to site 2, execute join at site 2, send result to query site
3. Transfer Project to site 1, execute join at site, send result to query site

What is the best strategy?

1. Strategy 1
2. Strategy 2
3. Strategy 3
4. Depends on query site

Employee: 1000 rows, row size 20 bytes
Project: 3000 rows, row size 50 bytes
Each result tuple 65 bytes
Join selectivity: 9 projects per employee

ira@cs.au.dk

Employee * Project

Query result of 9000 tuples if every employee on average contributes to 9 projects

1. Transfer Employee and Project to query site, perform join e.g. at site 3
   - Query site 3: total bytes transferred = 20,000 + 150,000 = 170,000 bytes
   - But, if query at site 2: only need to transfer Employee: 20,000 bytes
   - But, if query at site 1: only need to transfer Project: 150,000 bytes
2. Transfer Employee to site 2, execute join at site 2, send result to e.g. site 3
   Query result size = 65 * 9000 = 585,000 bytes
   - Query site 3: total transfer size = 585,000 + 20,000 = 605,000 bytes
   - But, if query at site 2, only need to transfer Employee 20,000 bytes
   - But, if query at site 1: 605,000 bytes
3. Transfer Project relation to site 1, execute join at site, send result to e.g. site 3
   - Query site 3: total transfer size = 585,000 + 150,000 = 735,000 bytes
   - If query site 2: 735,000 bytes
   - If query site 1: 150,000

Employee: 1000 rows, row size 20 bytes
Project: 3000 rows, row size 50 bytes
Each result tuple 65 bytes
Join selectivity: 9 projects per employee

Intended learning outcomes

- Be able to
  - Describe characteristics of distributed databases
  - Explain query decomposition and optimization in DDBs
  - Describe and apply core techniques of MapReduce

ira@cs.au.dk

What is this all about?

Guidelines for your own review of today's session

- A distributed system is defined as…
  - We have discussed the following types…
- Distributed database systems have the benefit of…
  - However, we need to account for…
- Fragmentation and replication are…
  - They are used to…
  - We can define them in relational algebra as follows…
  - In SQL, this corresponds to…
- When evaluating distributed queries we need to…
  - Typical strategies are…
  - The semi-join defines…
- MapReduce addresses the need for…
  - The main steps in MapReduce do the following…
  - We can use it to compute e.g.…

ira@cs.au.dk