DBMS Implementation
CSE510

K. Selçuk Candan

Professor

Arizona State University

Question 1

- What is a database?

Question 1

- What is a database?
  - Set of data, organized in some fashion for easy retrieval

Question 2

- What is a Data Model?

Question 2

- What is a Data Model?
  - a formalism to describe “constraints” that describe “properties” of data
  - Examples: Relational, OO, Spatial, Fuzzy

Question 2

- What is a Data Model?
  - a formalism to describe “constraints” that describe “properties” of data
  - Examples: Relational, OO, Spatial, Fuzzy

- Schema?

Question 2

- What is a Data Model?
  - a formalism to describe “constraints” that describe “properties” of data
  - Examples: Relational, OO, Spatial, Fuzzy

- Schema?
  - a set of constraints that
    - describe the “properties” of data
    - describe the structure of the data.
    - organize the data
  - Schema is described within the formalism corresponding to the underlying data model

Classification of models/schemas

- Physical
  - Data structures
- Logical
  - Relational
  - OO, OR
- Conceptual
  - UML, ER, Extended ER

Question 3

- What is a DBMS?

Question 3

- What is a DBMS?
  - A software/hardware system for
    - storage
    - retrieval
    - manipulation

    of data

Why use a DBMS (*)*from the book's notes

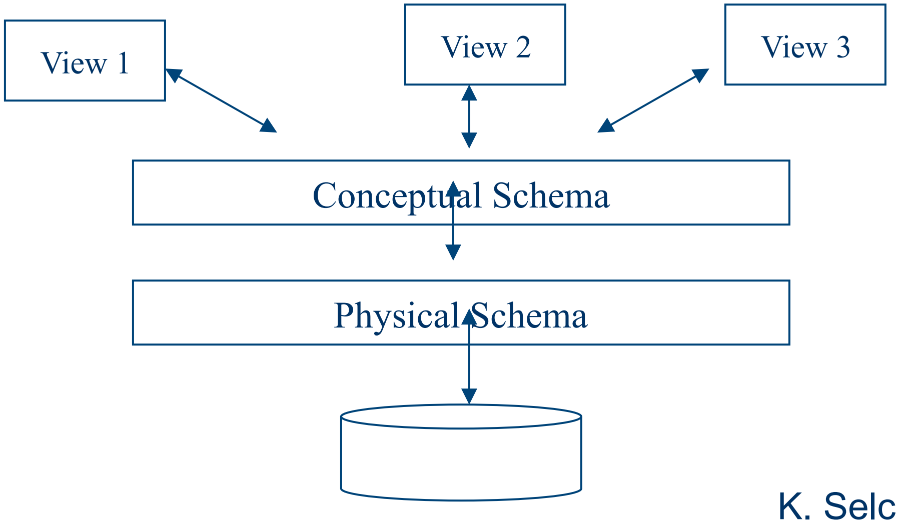
- Data independence and efficient access

- Reduced application development time.

- Data integrity and security.
- Uniform data administration
- Concurrent access, recovery from crashes.

Levels of Abstraction (*)

- Many *views*, single *conceptual (logical) schema* and *physical schema.*
  - Views describe how users see the data
  - Conceptual schema defines logical structure
  - Physical schema describes the files and the indexes used.



K. Selcuk Candan (CSE510)

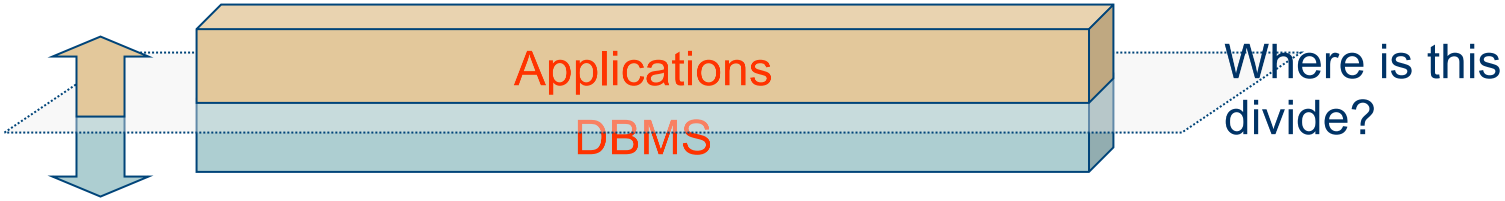
Data Independence (*)

- Applications insulated from how data is structured and stored.
- *Logical data independence:* Protection from changes in *logical* structure of data.
- *Physical data independence:* Protection from changes in *physical* structure of data.

One of the most important benefits of using a DBMS!

Question 4

- What is the difference between OS and DBMS?



Components of Data-Intensive Systems (*)

Three separate types of functionality:

- Data Management
- Application Logic
- Presentation

- The system architecture determines whether these three components reside on a single system ("tier") or are distributed across several tiers.

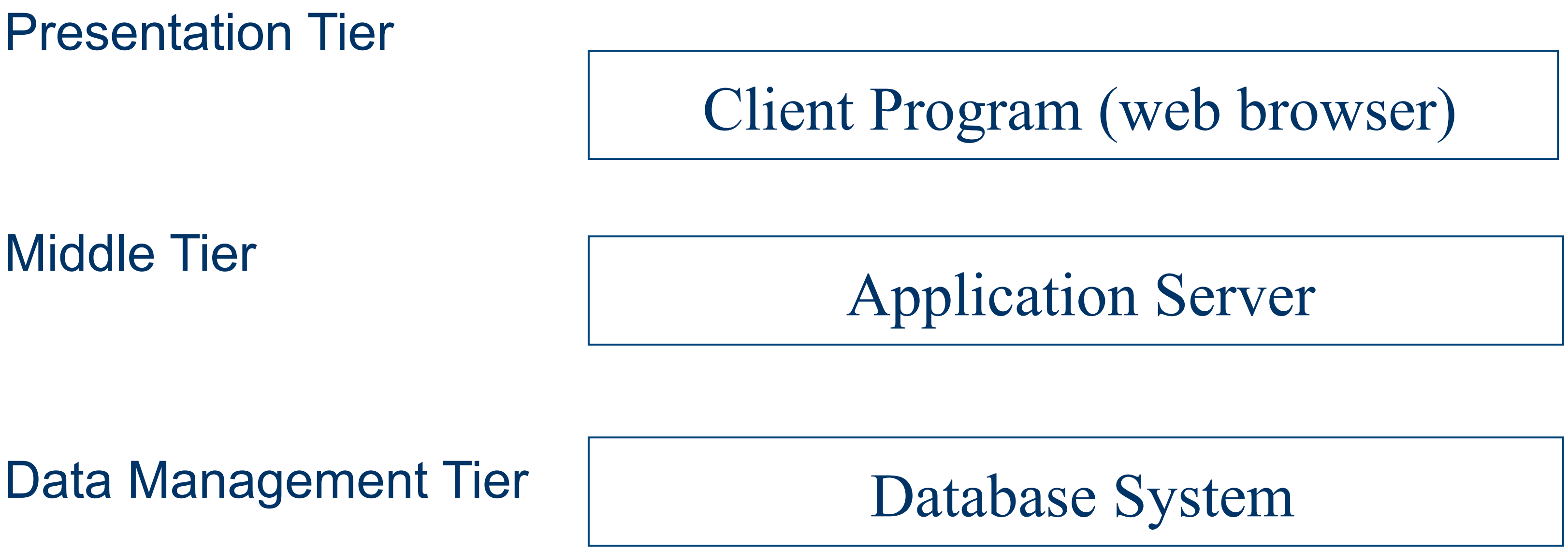
Single-tier Architecture (*)

- All functionality combined into a single tier usually on a mainframe
  - User access through dumb terminals
- Advantages:
  - Easy Maintenance and Administration
- Disadvantages:
  - Centralized computation of all of them is too much for a central system.

Client-Server Architectures (*)

- Work Division: Thin Client
  - Client implements only the graphical user interface
  - Server implements business logic and data management.
- Work Division: Thick Client
  - Client implements both the graphical user interface and business logic.
  - Server implements data management.

Three-Tiered Architecture (*)



Presentation Tier

Client Program (web browser)

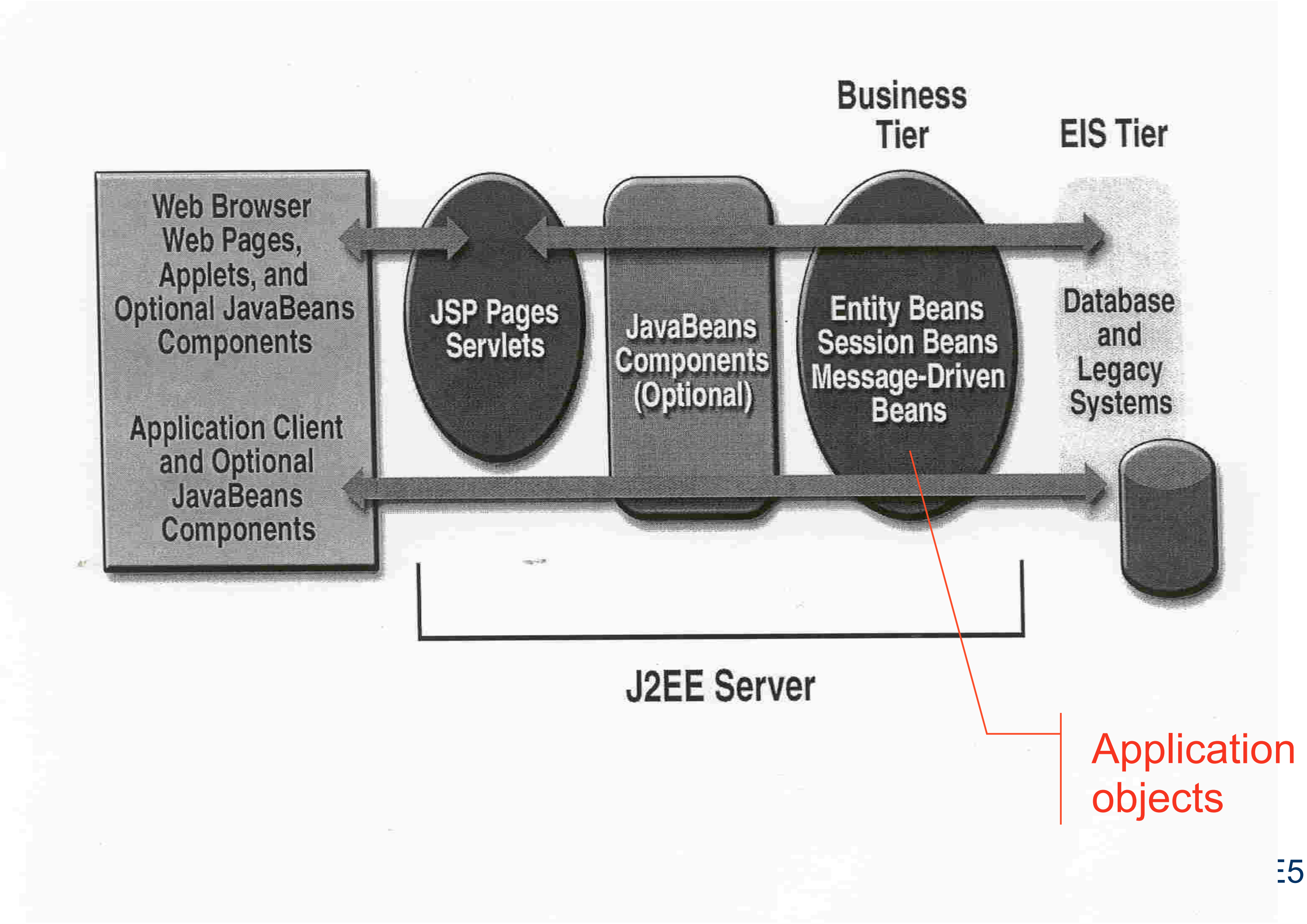
Middle Tier

Application Server

Data Management Tier

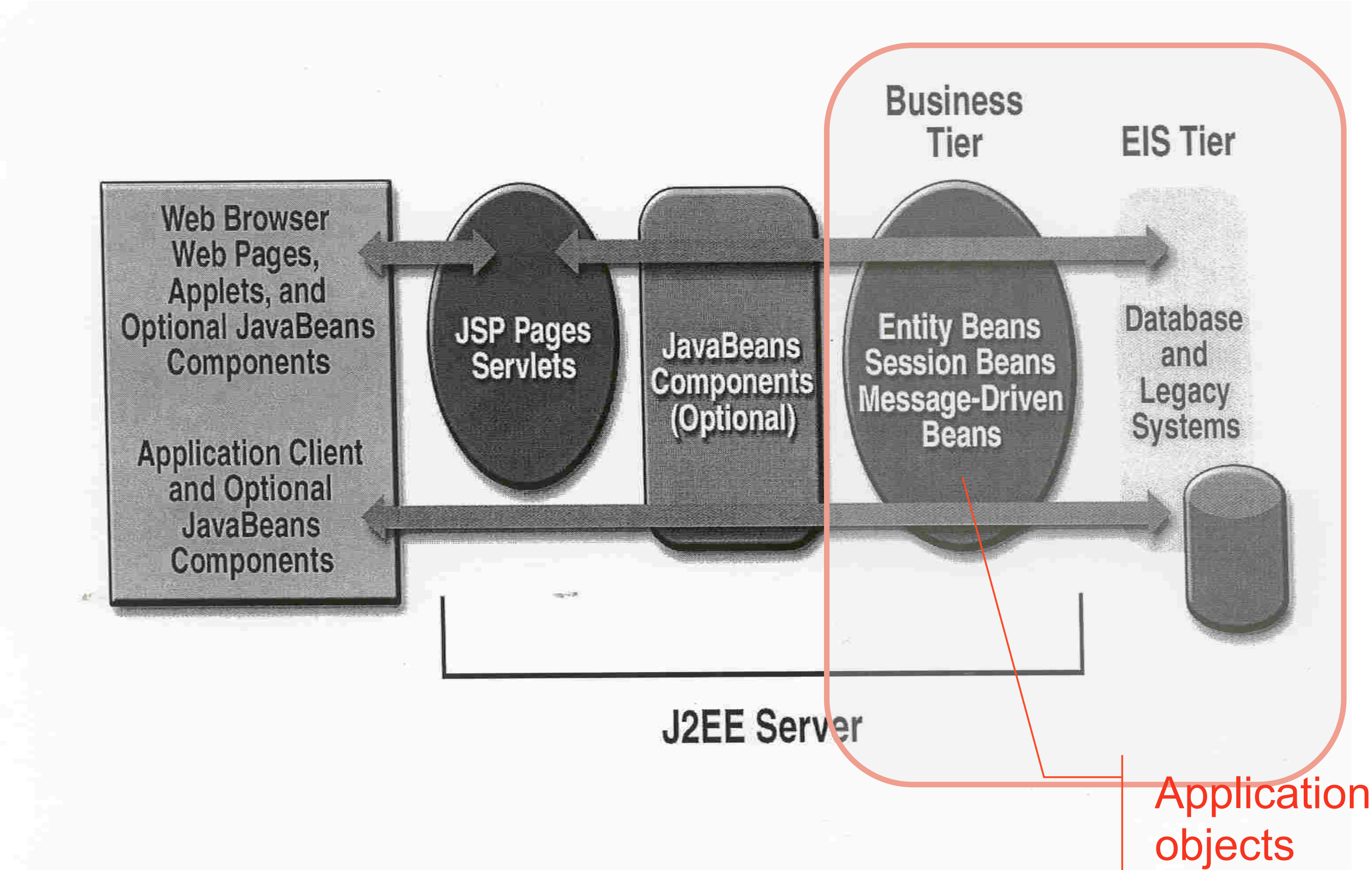
Database System

Multi-Tiered Architecture (e.g. J2EE)


Business Tier
EIS Tier
Web Browser Web Pages, Applets, and Optional JavaBeans Components
Application Client and Optional JavaBeans Components
JSP Pages Servlets
JavaBeans Components (Optional)
Entity Beans Session Beans Message-Driven Beans
Database and Legacy Systems
J2EE Server

Application objects

Multi-Tiered Architecture (e.g. J2EE)

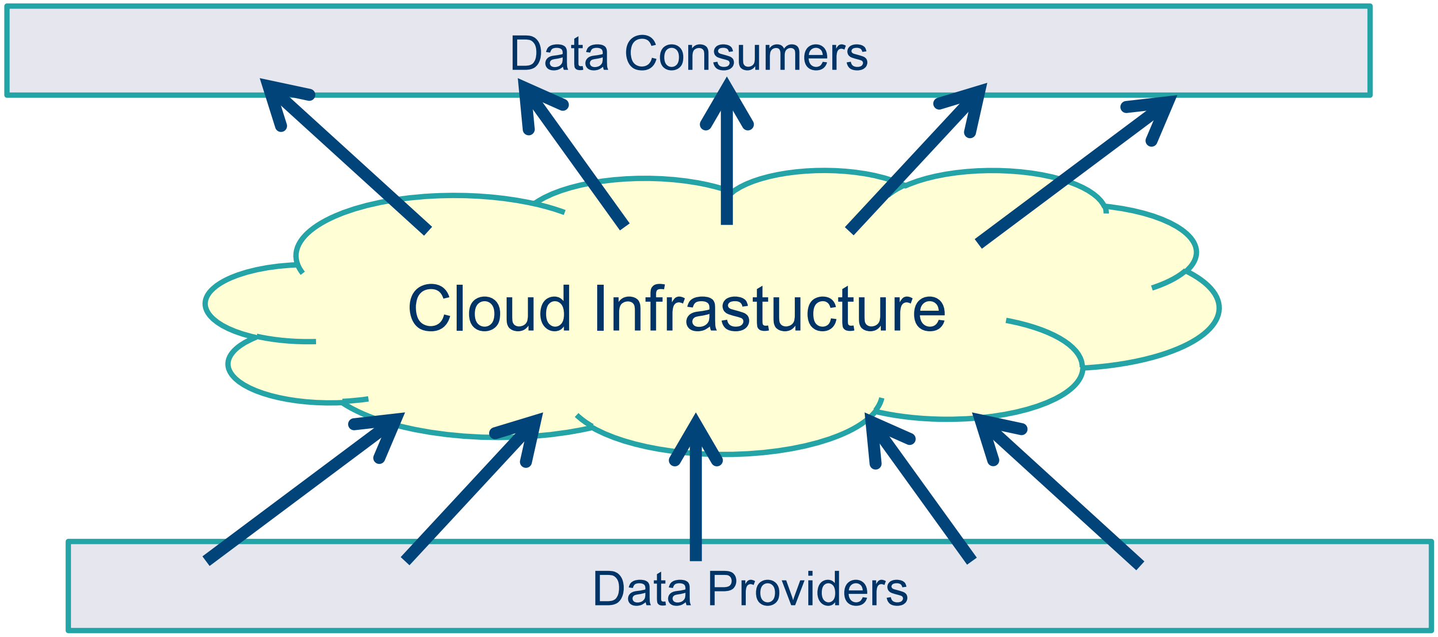

Business Tier
EIS Tier
Web Browser Web Pages, Applets, and Optional JavaBeans Components
Application Client and Optional JavaBeans Components
JSP Pages Servlets
JavaBeans Components (Optional)
Entity Beans Session Beans Message-Driven Beans
Database and Legacy Systems
J2EE Server

Application objects

Could or should we store complex objects (EJBs) in a DBMS?

Information and Software as Services

- Products, services and solutions delivered over a shared service infrastucture (or cloud)



Information and Software as Services

- Advantages
  - rapid deployment
  - lower nt cost
    - cor
  - instar
  - Web-

IDC:

SaaS revenue will reach $14.5B in 2011

Overall IT Cloud spending will reach $42B (9%) by 2012

F. Gens. "Clouds and Beyond: Positioning for the Next 20 years in Enterprise IT IDC Report, 2009.

L. Herbert, E. G. Brown, and S. Galvin, "Competing in the fast-growi saas market," 2008, no. 0,5110,44254,00, 2008. Forrester Report.

Information and Software as Ser

Key customer demands

- Competitive pricing
- Service assurances
- Flexibility: ability to move services back on-premise

- Pro ed over a shared service infrastucture (or cloud)

Key customer concerns

- Will it be secure?
- Will it work with in-house IT?
- Will it really cost less?
- Will it be configurable?
- Will it provide a complete solution?
- Will it work with other clouds if needed?

Policies

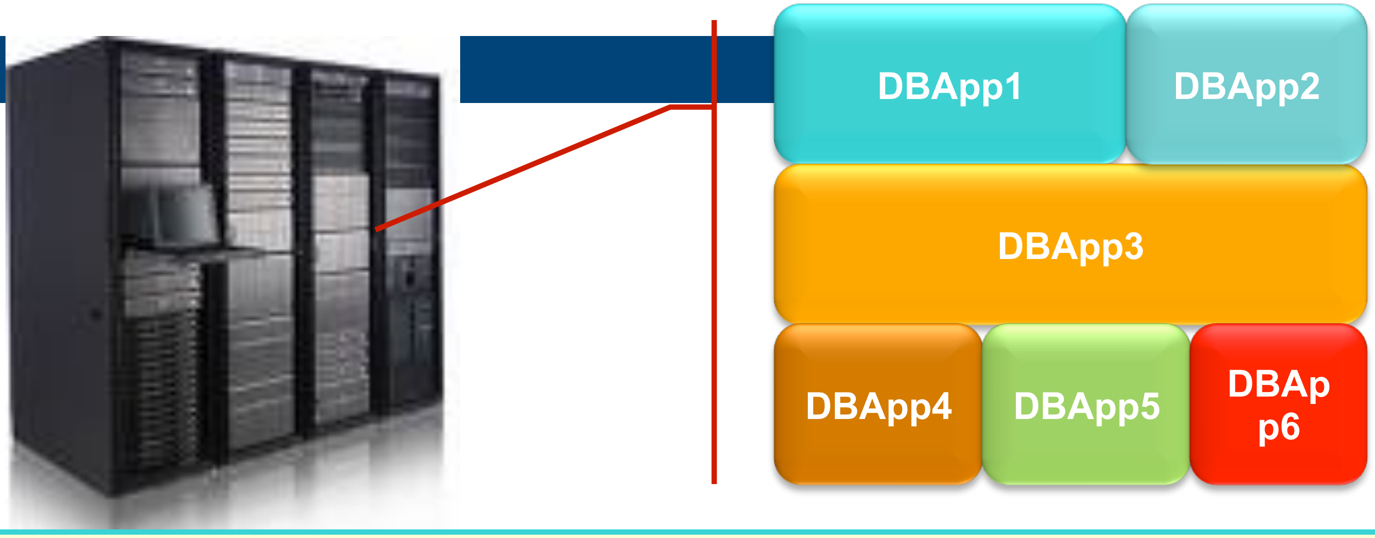
Resource Optimization

- Ad
  -
  -
  - I
  - Web-native

F. Gens. "Clouds and Beyond: Positioning for the Next 20 years in Enterprise IT IDC Report, 2009.

L. Herbert, E. G. Brown, and S. Galvin, "Competing in the fast-growi saas market," 2008, no. 0,5110,44254,00, 2008. Forrester Report.

Multi-tenant databases



DBApp1

DBApp2

DBApp3

DBApp4

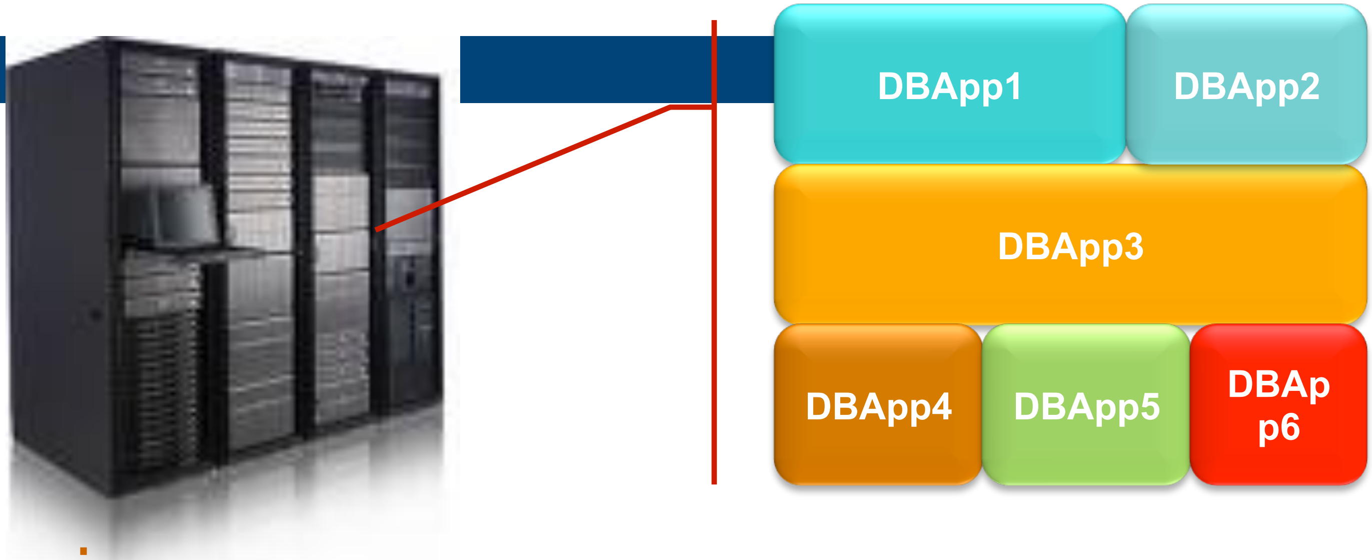
DBApp5

DBApp6

- Why important???
  - Salesforce has more than 55,000 enterprise customers, 1.5 million individual subscribers, 30 million lines of third-party code, and hundreds of terabytes of data all running on 1,000 machines
  - Amazon’s Web Services, in comparison, runs on about 100,000 machines

http://www.techcrunch.com/2009/03/23/the-efficient-cloud-all-of-salesforce-runs-on-only-1000-servers/

Approach #1: Tagging

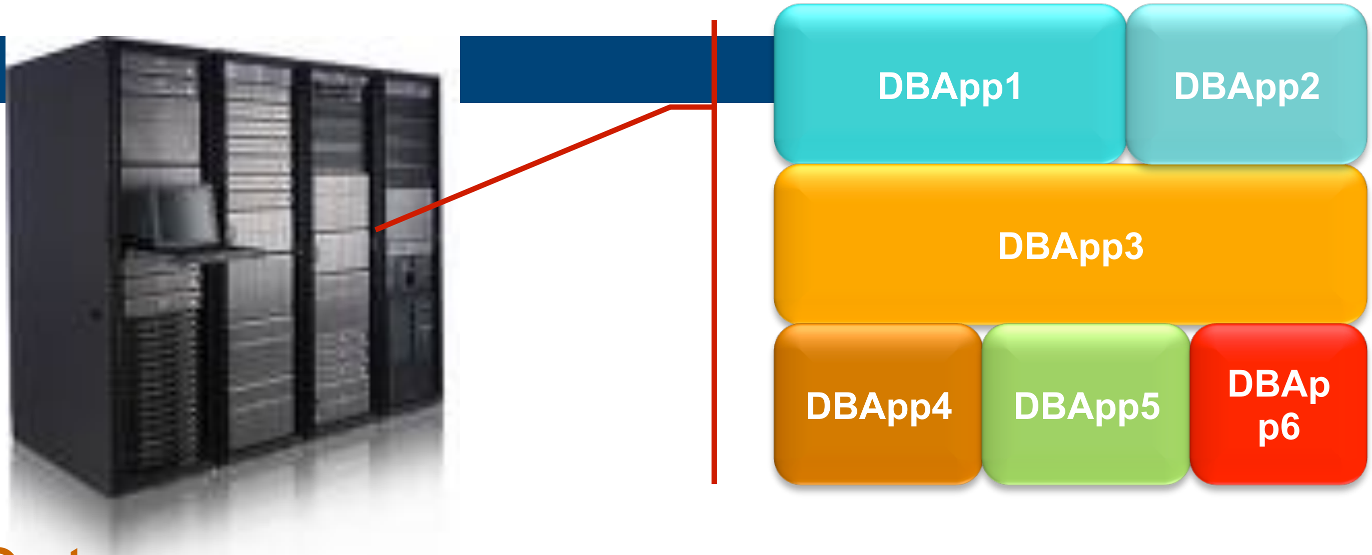


- Tagging
  - single common schema
  - data is row-partitioned across
    - “ownership” attribute

•**Salesforce’s secret ingredient**

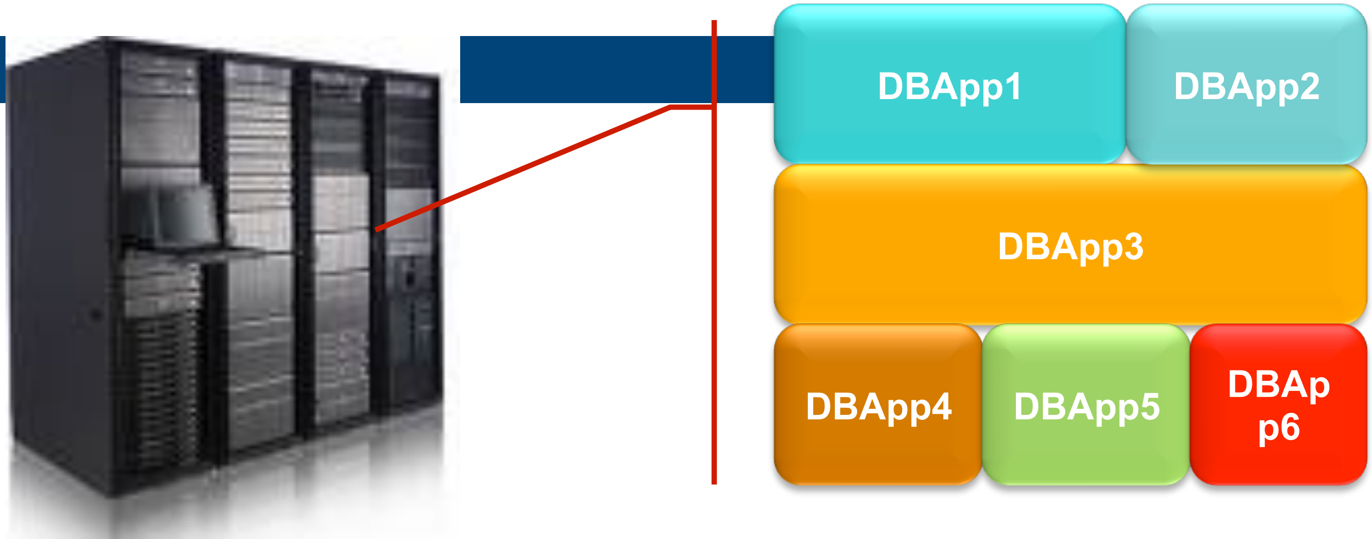
http://www.techcrunch.com/2009/03/23/the-efficient-cloud-all-of-salesforce-runs-on-only-1000-servers/

Approach #2: Data spaces



- Data spaces
  - similar to application server virtualization
  - no assumptions about the tenant schemas
  - full separation (semantic and performance isolation)

Approach #3: Flexible integration



- Integrated/consolidated schema
  - Similar, but different tenant schemas
    - mapping from tenant schemas to consolidated schema
  - data is row/column-partitioned across tenants

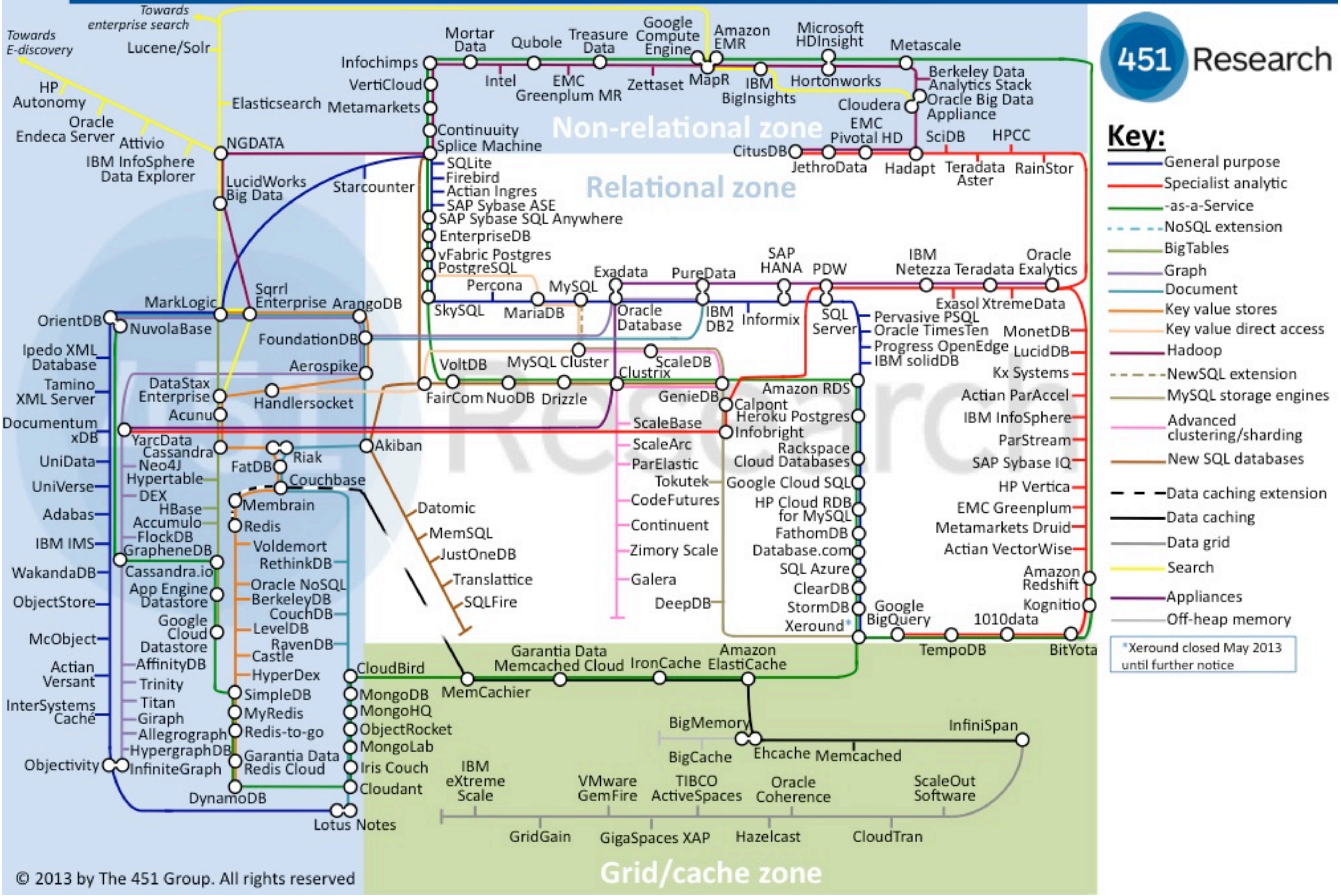
Question 5

- So what is out there?

DB Generations

- First generation: Hierarchical & network
- Second generation: Relational
  - Business App
- Third generation
    - Novel Applications
  - Objects oriented
  - Object-relational
- Fourth, Fifth, Sixth, …. generations.

Database Landscape Map – June 2013



“No SQL”?

- “Not Only SQL” or “Not Relational”
- Scaling a transaction-centric relational DBMS is difficult
  - ❑ ACID (Atomicity, Consistency, Isolation, and Durability) is costly
- CAP theorem* states that a system can have only two out of three
  - ❑ consistency,
  - ❑ availability, and
  - ❑ partition-tolerance

S. Gilbert and N. Lynch, “Brewer’s conjecture and the feasibility of consistent, available, and partition-tolerant web services”, ACM SIGACT News 33 , 2, pp 51-59, March 2002.

“Scalable SQL and NoSQL” Data Stores by R. Cattell 2008

Target features…

- “Shared nothing” horizontal scaling
  - replicating and partitioning data (often by a key – “sharding”) and load over many servers to support a large number of key lookups and small writes
- Vertical scaling
  - use of multiple cores or CPUs
- Use of distributed indexes and RAM for data storage,
- Weaker consistency
  - BASE (Basically Available, Soft state, Eventually consistent) instead of ACID
- Flexible schema (dynamically add new attributes) and application specific data structures
- Call level interface instead of SQL

Examples

- Key-value stores
  - store values and an index to find them
  - Project Voldemort, Memcached
- • Document stores
  - documents are indexed and a simple query mechanism is provided
  - SimpleDB, CouchDB, MongoDB
- • Extensible Record Stores
  - store extensible records that can be partitioned vertically and horizontally across nodes.
  - HBase, Cassandra, PNUTS
- Scalable RDBMSs
  - MySQL cluster, VoltDB, Clustrix, ScaleDB etc.

….read….

- Read the CACM blog “**The End of a DBMS Era (Might be Upon Us)** by Michael Stonebraker posted on June 30, 2009
- ..don’t forget to read the “user comments”….
- http://cacm.acm.org/blogs/blog-cacm/32212-the-end-of-a-dbms-era-might-be-upon-us/fulltext

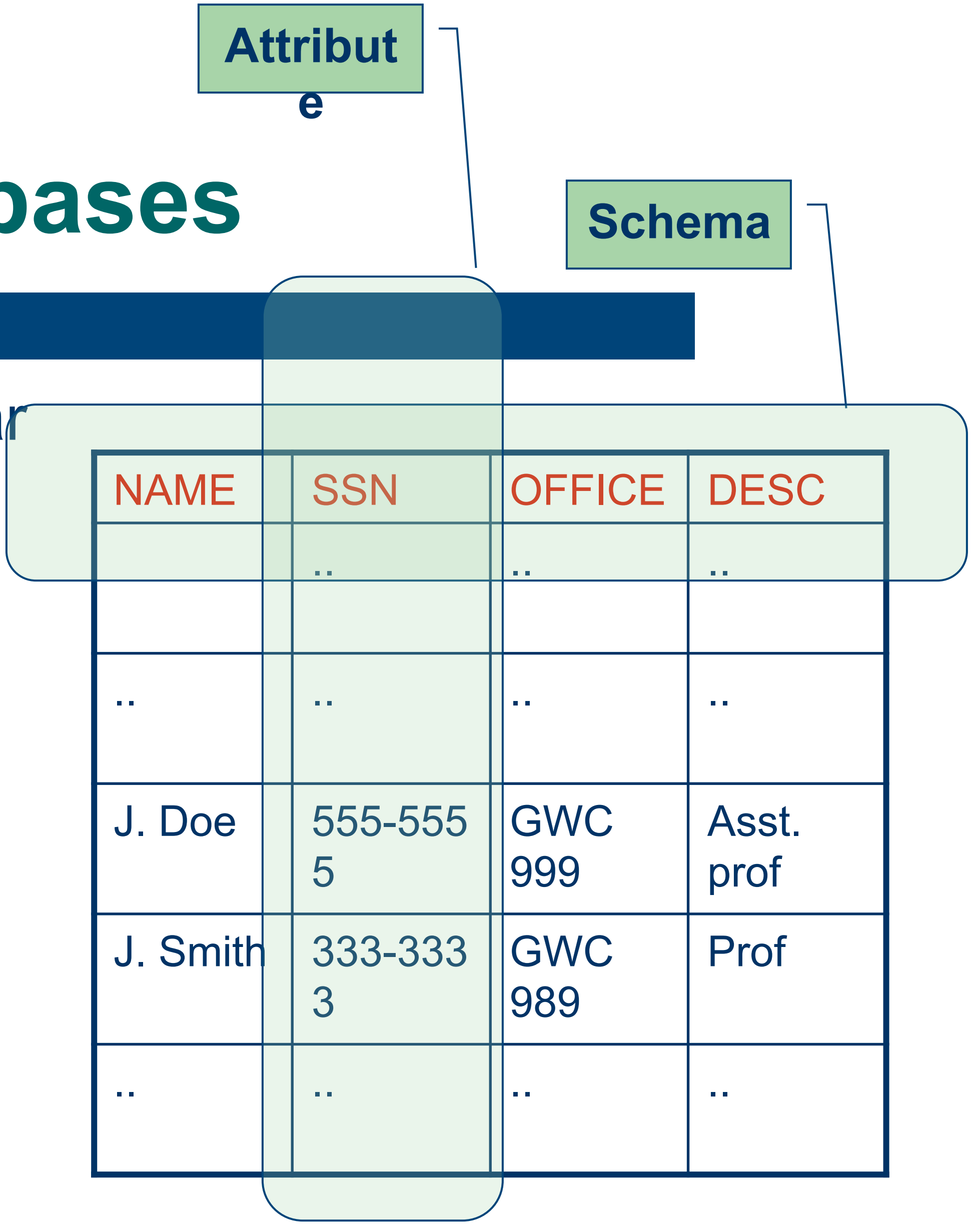
Relational databases

- Data is
  - Textual
  - numerical

- This is the main assumption for
  - storage
  - query processing
  - optimization

Relational databases

- Information is in tabular form
  - Example: Information about an employee
- Schema describes the content
- A key uniquely identifies a given tuple
- Each attribute has a domain

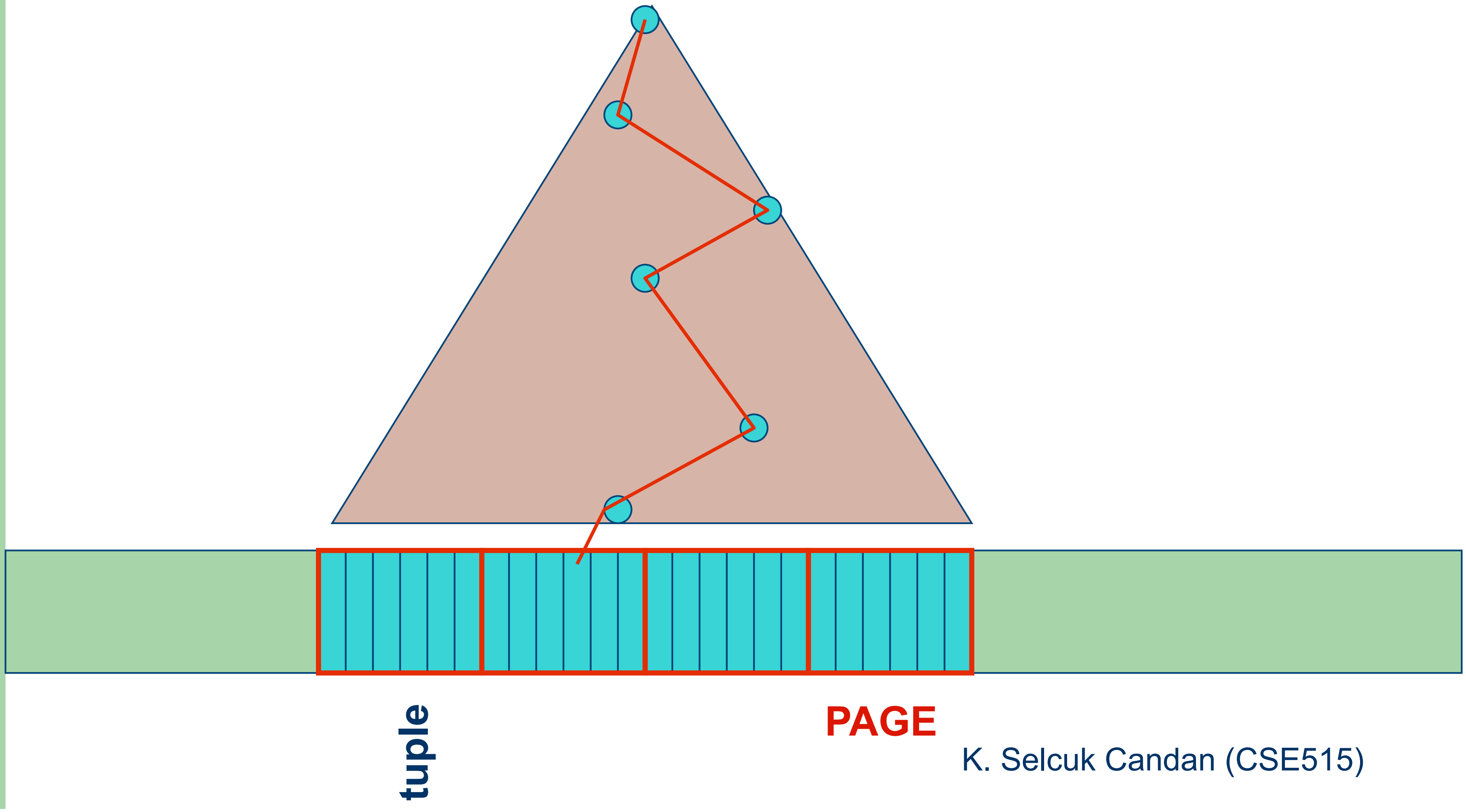


NAME	SSN	OFFICE	DESC
	..	..	..
..	..	..	..
J. Doe	555-5555	GWC 999	Asst. prof
J. Smith	333-3333	GWC 989	Prof
..	..	..	..

Index structures

- Disk is divided into logical units, called “pages”
- A relation/table is stored contiguously
- Each page contains a certain number of tuples
- Index structures created for access to data

Index structures



Algebra

- A set of data manipulation operators
- Relational algebra (operates on relations)
  - Select (σ)
  - Project (π)
  - Cartesian product (×), join
  - Union (U)
  - Intersection (∩)
  - Difference (-)

Calculus

- A query language should be declarative:
  - Say what we want
  - Don't say how we get it
    - otherwise, no optimization possible

```
{t.name | (t in Employee) and  (t.salary < 1000) and
            (exists t2  (t2 in Students) and (t2.gpa > 3.7)
                                               and (t.ssn = t2.ssn)
            )}
```

Query optimization

- Query: Find all student employees whose GPAs are greater than 3.7 and salaries are less than $1000; return their names

plan 1: $\Pi_{name}(\sigma_{gpa>3.7}(\sigma_{sal<1000}(\sigma_{ssn=ssn}(Employee \times Students))))$

plan 2: $\Pi_{name}(\sigma_{ssn=ssn}(\sigma_{gpa>3.7}(Students) \times \sigma_{sal<1000}(Employee)))$

  - same results
  - different execution costs

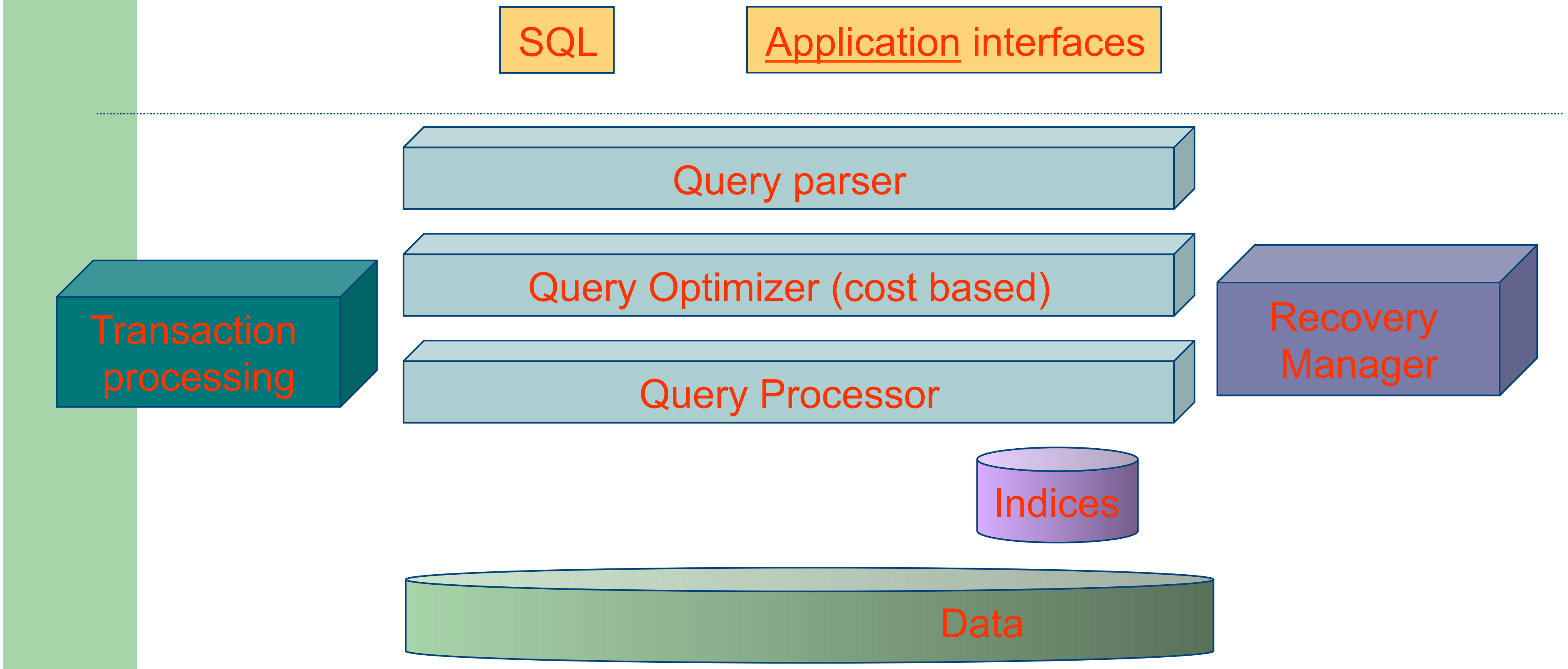
SQL

- Based on relational calculus

```
select <attribute_list>
from <relation_list>
where <condition>
```

```
select t.name
from employee t, student t2
where (t.salary < 1000) and
          (t2.gpa > 3.7) and
          (t.ssn = t2.ssn)
```

How does a Relational DBMS look like?



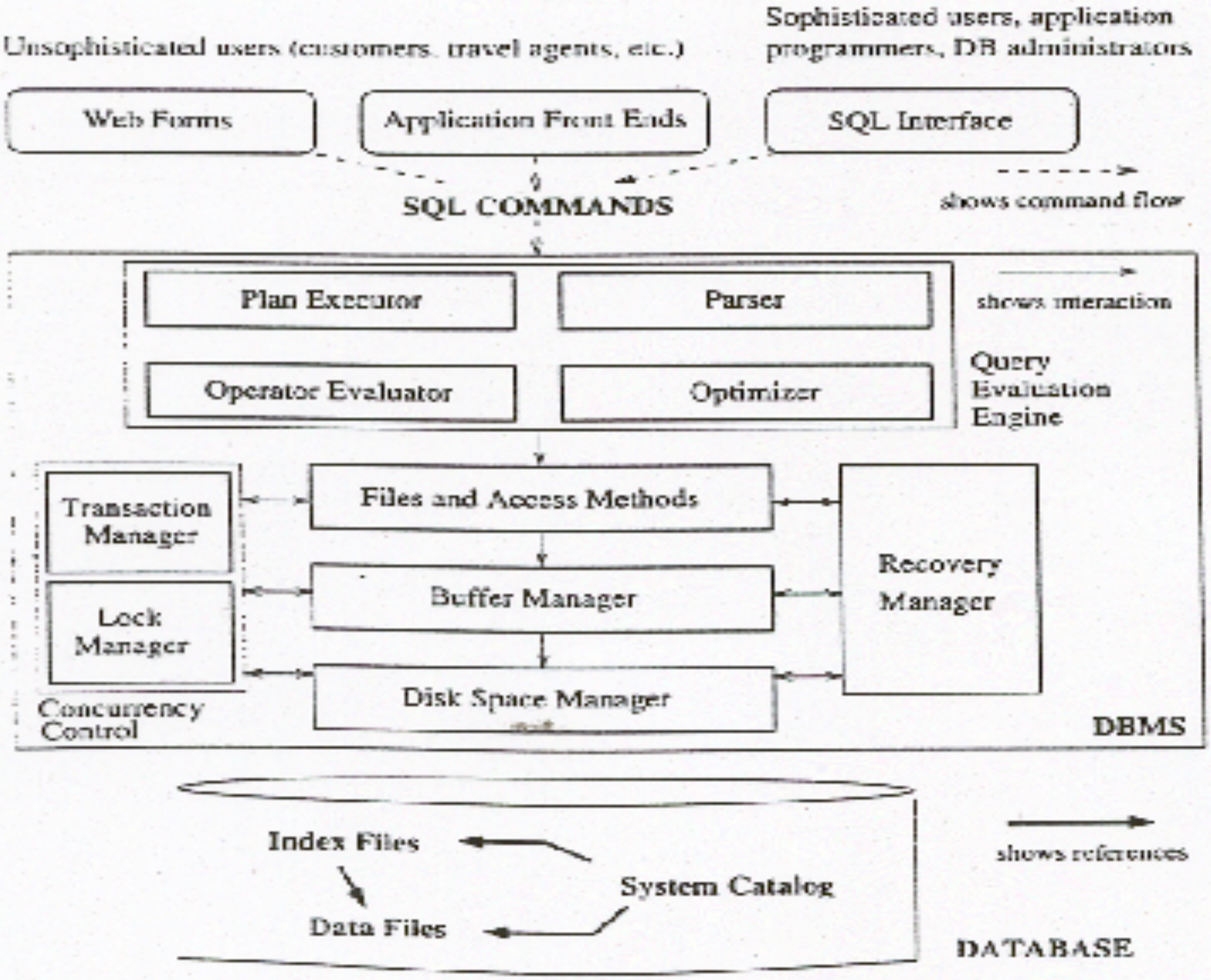
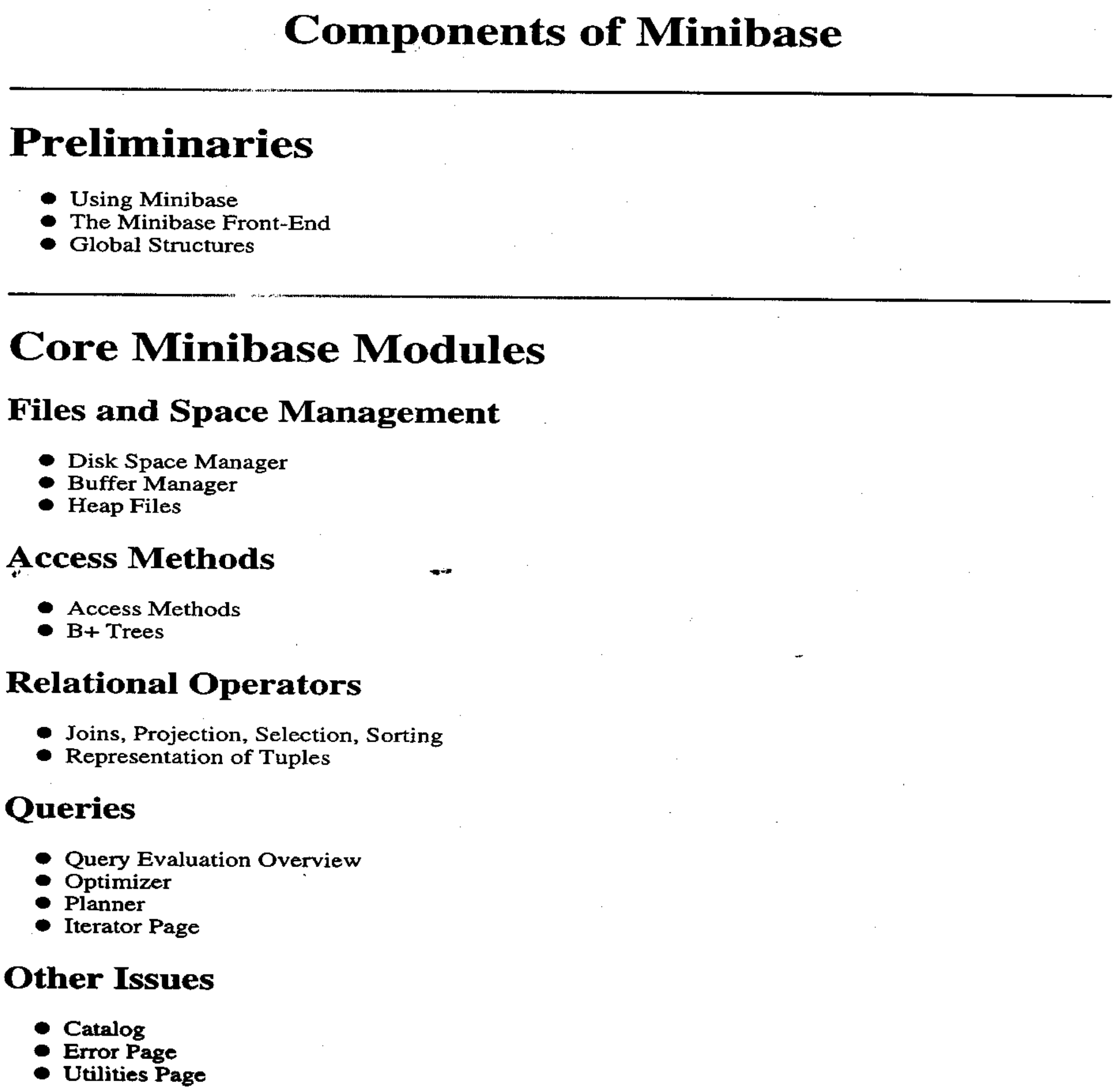


Figure 1.3 Architecture of a DBMS



Components of Minibase

Preliminaries

- Using Minibase
- The Minibase Front-End
- Global Structures

Core Minibase Modules

Files and Space Management

- Disk Space Manager
- Buffer Manager
- Heap Files

Access Methods

- Access Methods
- B+ Trees

Relational Operators

- Joins, Projection, Selection, Sorting
- Representation of Tuples

Queries

- Query Evaluation Overview
- Optimizer
- Planner
- Iterator Page

Other Issues

- Catalog
- Error Page
- Utilities Page

Relational Databases

- Business applications
- DM is relational
- Queries are exact/declarative
- Updates are important
- Concurrency is important
- Distributivity is important

Shortcomings

- Many data doesn’t fit into tuples.
- Large data needs to be kept separately.
- No support for imprecise data or approximate matches

Computational Completeness

- Relational Databases are not computationally complete
  - store, retrieve, simple computations
  - thus, there is a need for a host language

Computational Completeness

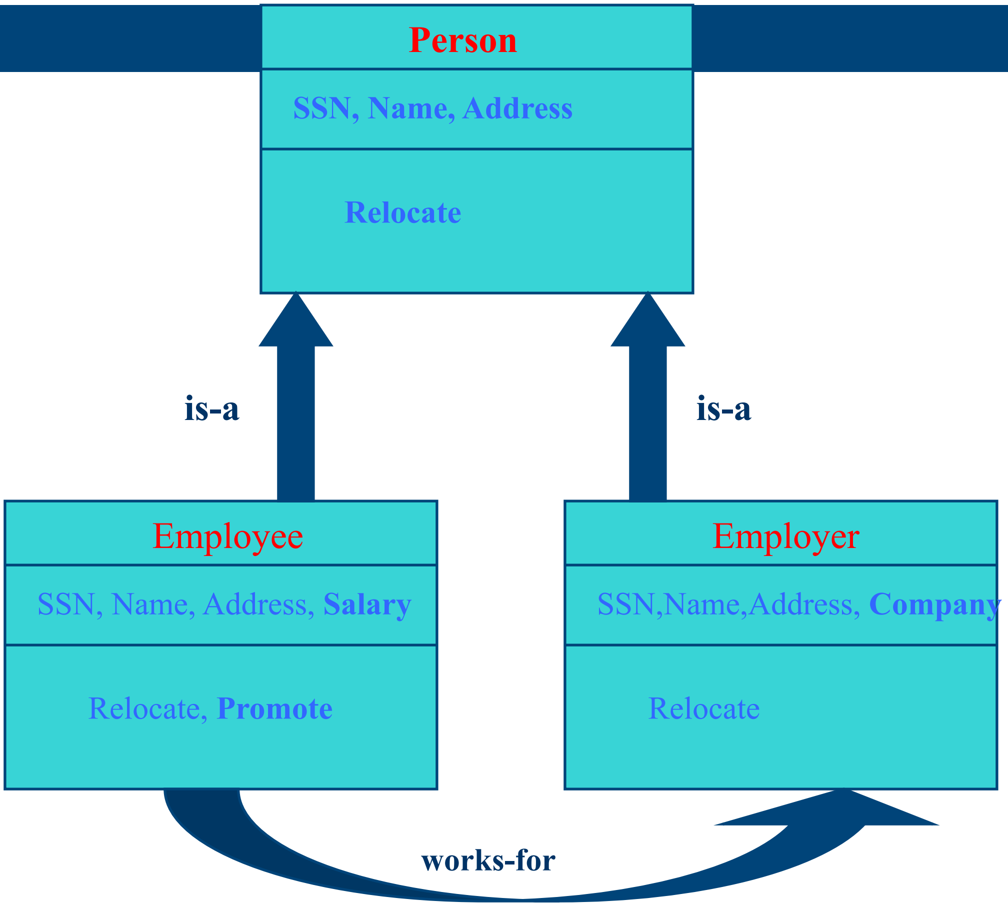
- Relational Databases are not computationally complete
  - store, retrieve, simple computations
  - thus, there is a need for a host language
- OODBs are computationally complete
  - "method"s can be used to perform arbitrary operations on the objects
    - store, retrieve,
    - manipulate/modify,
    - complex computations

Object-oriented data model

- No tuples, no rows

- maps “entities” to data structures
- maps “behaviors” to functions
- relationships can be described as
  - object references or
  - separate entities.

Example, OO Model



Person

SSN, Name, Address

Relocate

Employee

SSN, Name, Address, **Salary**

Relocate, **Promote**

Employer

SSN,Name,Address, **Company**

Relocate

OODB

- Object oriented databases provide
  - Higher computational power
  - Aggregation hierarchies
  - Inheritence hierarchies
- They model the real world better!
  - Everything is an object
- You can define your own external methods

E.image_similar_to (c.image)

Object-Oriented Databases

- Business Applications
- Multimedia Applications (new data types)
- DM is object oriented
- Data is exact
- Queries are exact
- Queries are procedural (…there are aso some declarative lang.)
- Concurrency/updates/distributivity are important

Shortcomings:

- Too much overhead.
- Optimization is very hard.

Object-Relation Databases

- Benefits from Both:
  - relational technology
    - tuples
    - SQL
  - object technology
    - user-defined functions
    - user-defined ADTs
      - data blades
      - data cartridges
      - extenders

Object Relational Databases

- Business applications
- Other Applications
- DM is Object Oriented + relational
- Queries are exact
- Queries are declarative
- Concurrency/updates/distributivity are important

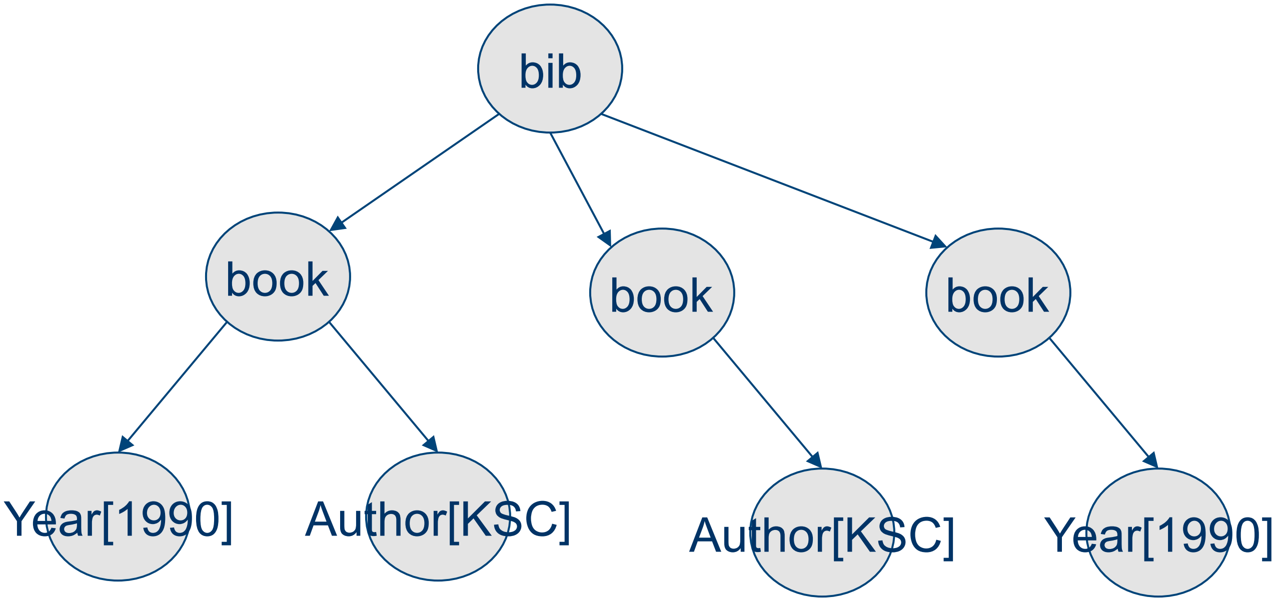
Shortcomings

- Optimization?

Semi-Structured Databases

- Most data models assume a “schema” which describes the elements in the database.
- SSDs do not assume a “schema”
  - schemaless
  - self-describing
- Each item in the database describes its own schema

XML document example



What is semi-structured data

- schemaless
- self-describing

Are these the same??

What is semi-structured data

- schemaless
- self-describing

Are these the same??

- Is a web-page semi-structured?
- Is the web semi-structured?

Extensible Markup Language (XML)

- Developed by ***W3C Generic SGML Editorial Review Board***
- XML is is a subset of SGML (Standard Generalized Markup Language-ISO 8879).
- SGML/XML is a method for creating
  - interchangeable,
  - structured documents
- Document structure is defined using *Document Type Definition (DTD)*

Document Type Definitions (DTD)

```
<!ELEMENT article (section+)>

<!ATTLIST article
          title CDATA #REQUIRED>

<!ELEMENT section (title,(subsection|
  CDATA )+)>

<!ELEMENT subsection (title,
  (subsubsection| CDATA )+)>

<!ELEMENT subsubsection(title, CDATA)>
```

Why semistructured??

- Semi-structured data may have
  - missing attributes
  - attributes which repeat itself

Why semistructured??

- Semi-structured data may have
  - missing attributes (null values)
  - attributes which repeat itself (multivalued attributes)

Is this any different from OODB?

Why semistructured??

- Semi-structured data may have
  - missing attributes (null values)
  - attributes which repeat itself (multivalued attributes)

Is this any different from OODB?

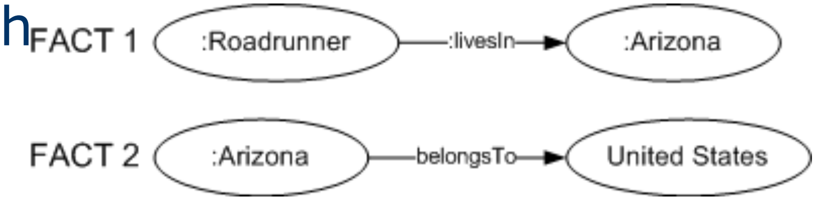
- Power of saying “or” in the schema!
  - DTD1 + DTD2 -> DTD_new
  - Even easier to integrate when DTDs are not given!!

What is really different about semi-structured data

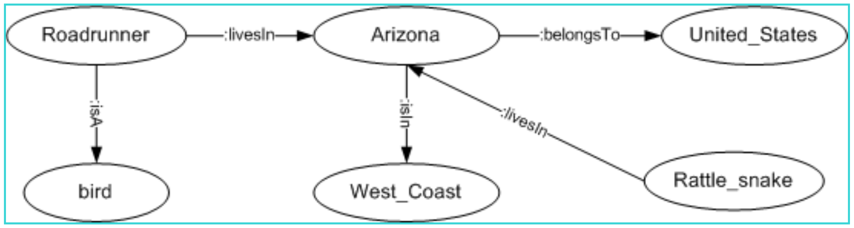
- Structure is not given; hence,
  - we may want to ask queries about the structure
  - we may need query languages to identify the structure
  - we may need to evaluate queries without explicit structure
  - we may need to answer queries based on approximate structural matching

Graph Databases (RDF)

- The Resource Description Framework (RDF) provides a way to express facts about entities and encode their relationsh



- Triplets of subject-predicate-object can encode basic facts.

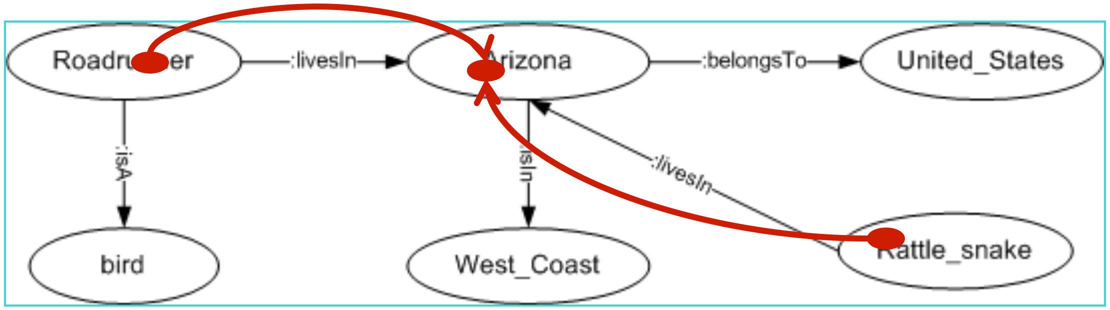


RDF: SPARQL in a nutshell

- Core of SPARQL include triple patterns

```
?s      livesIn      Arizona
```

Retrieve triples whose predicate is “livesIn” and object is “Arizona”:

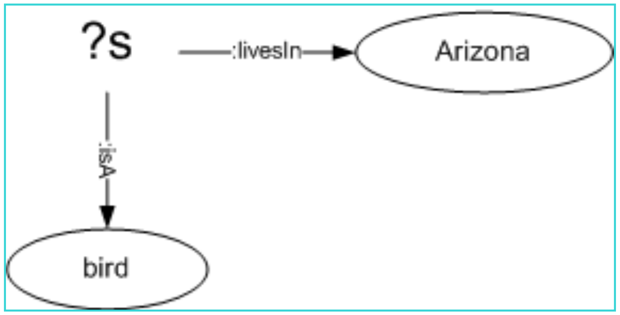


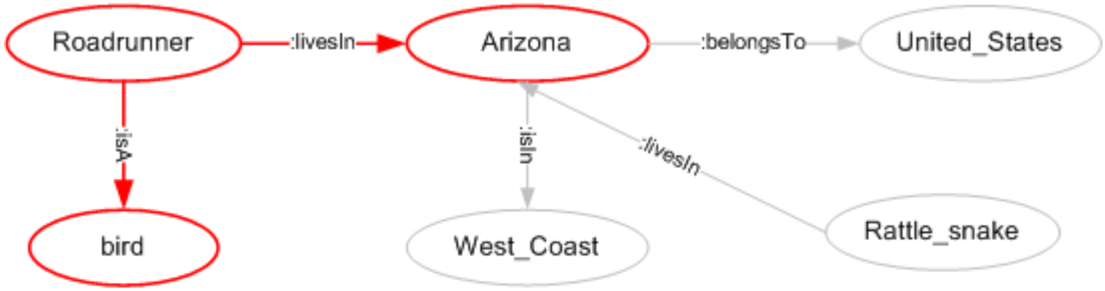
```
<Roadrunner   livesIn   Arizona>
<Rattle_snake   livesIn   Arizona>
```

RDF: Basic Graph Patterns (BGP)

- If triple patterns share variables, they form a query graph to be matched against parts of the RDF graph.

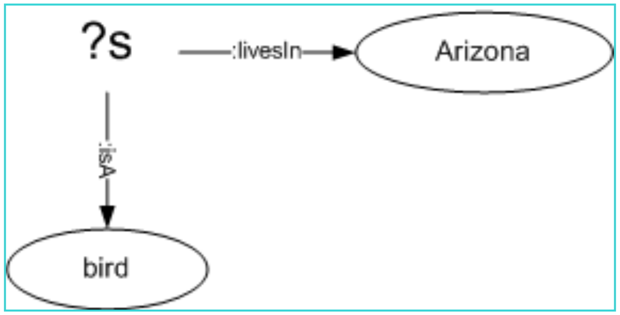
```
SELECT *
WHERE {
    ?s     livesIn     Arizona .
    ?s     isA          bird
}
```



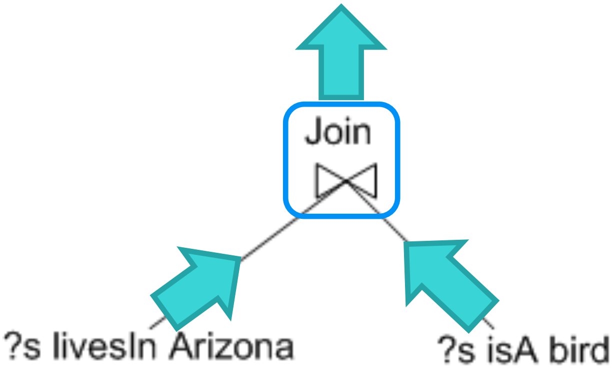


RDF: Query Processing

```
SELECT *
WHERE {
    ?s     livesIn     Arizona .
    ?s     isA          bird
}
```



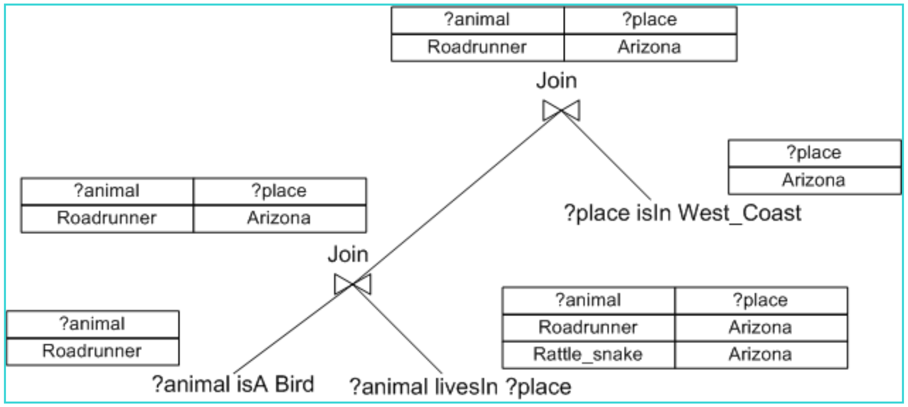
- A BGP can be processed by joining its component triple patterns on the common variable.



RDF: Query Plan

```
SELECT *
WHERE {
    ?animal   isA       bird .
    ?animal   livesIn   ?place .
    ?place    isIn      West_Coast
}
```

- More complex BGPs give rise to more complex join plans.



?animal	?place
Roadrunner	Arizona

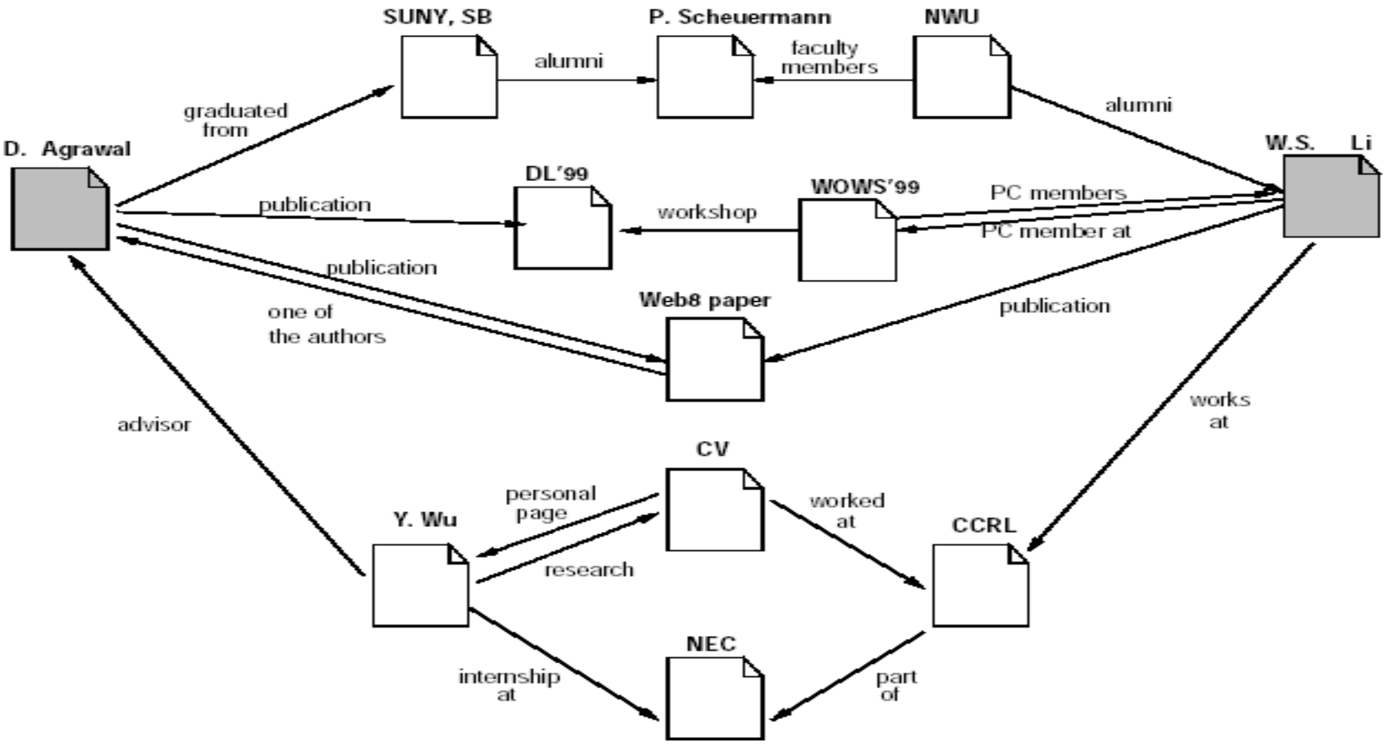
?place
Arizona

?animal	?place
Roadrunner	Arizona

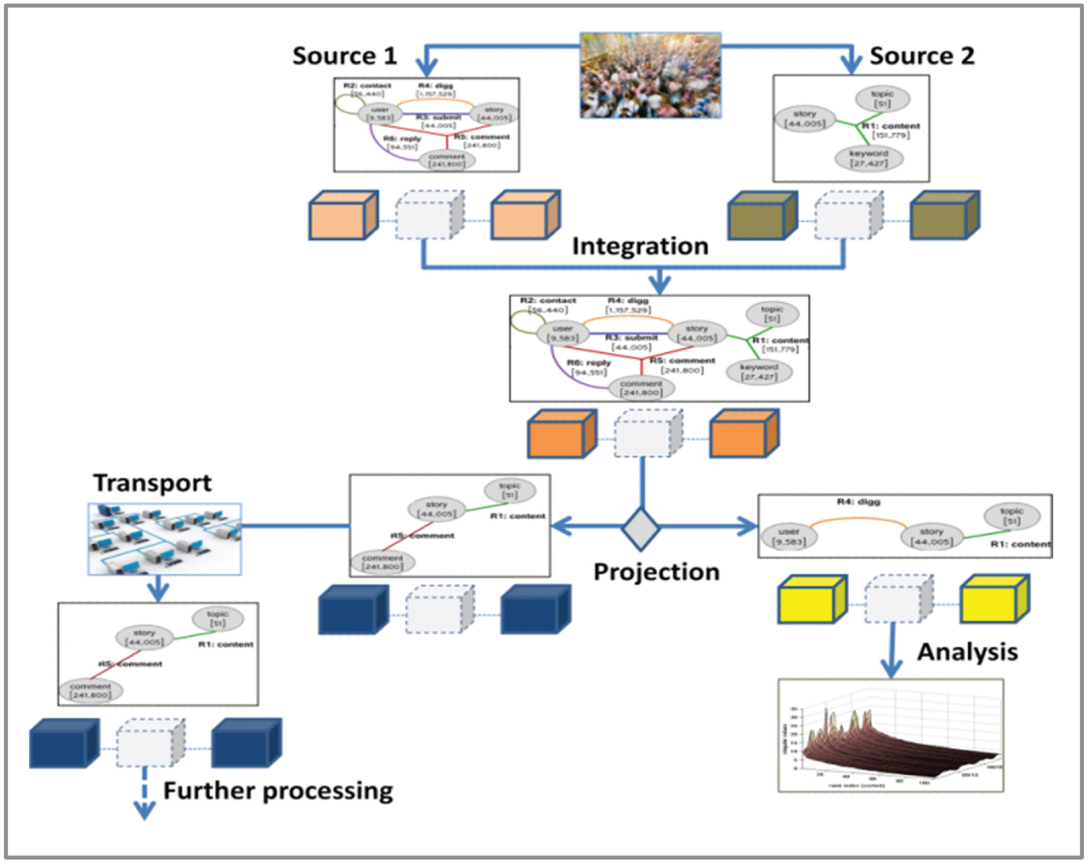
?animal
Roadrunner

?animal	?place
Roadrunner	Arizona
Rattle_snake	Arizona

Graph Databases (web, social networks, etc.)



Graph Manipulation vs. Analysis



Graph Databases (web, social networks, etc.)

- Common operations:
  - Shortest paths
  - Clustering/partitioning
  - Association mining
  - Authority, centrality, pagerank computation
- Focus on scalability
  - Pregel / Hama
  - GraphBase
  - Horton / Trinity

Deductive Databases

- Business applications
- Data model is logic-based, predicate-based
- Truth-based queries

Spatial/Temporal Databases

- Scientific/Geographic Applications
- Data model is vector/interval-based
- Queries:
  - range-based
  - nearest-neighbor based
- Queries are declarative/visual

Image Databases

- Multimedia applications
- Data model is feature vector based
  - multiple features
    - color
    - texture
  - each feature is represented as a space
- Queries
  - Query-by-example
  - Similarity based ranking
  - Feedback-to-user, feedback-from-user
  - continuous queries

Data Mining

- Business Applications/ Scientific Applications
- Data model is relational
- Queries:
  - identify “rules”
  - identify “classes”
  - Identify “clusters”
  - identify “outliers
  - in general statistical

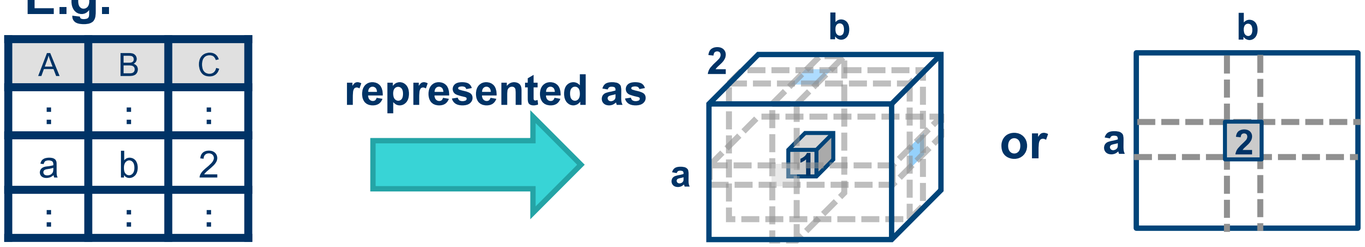
Others..

- Column-databases
  - Vertica, MonetDB, Hbase/BigTable, InfiniDB, GreenPlum, and Cloudera
- Key-value stores
  - Azure, Dynamo, levelDB, Voldemort, MemBase, RaptoreDB, and G-Store
- Array databases
  - SciDB
- MapReduce (Hadoop, Dryad, etc.)

Tensor Representation of Data

- Most media, sensor, social network data are
  - multi-dimensional and
  - Multi-modal

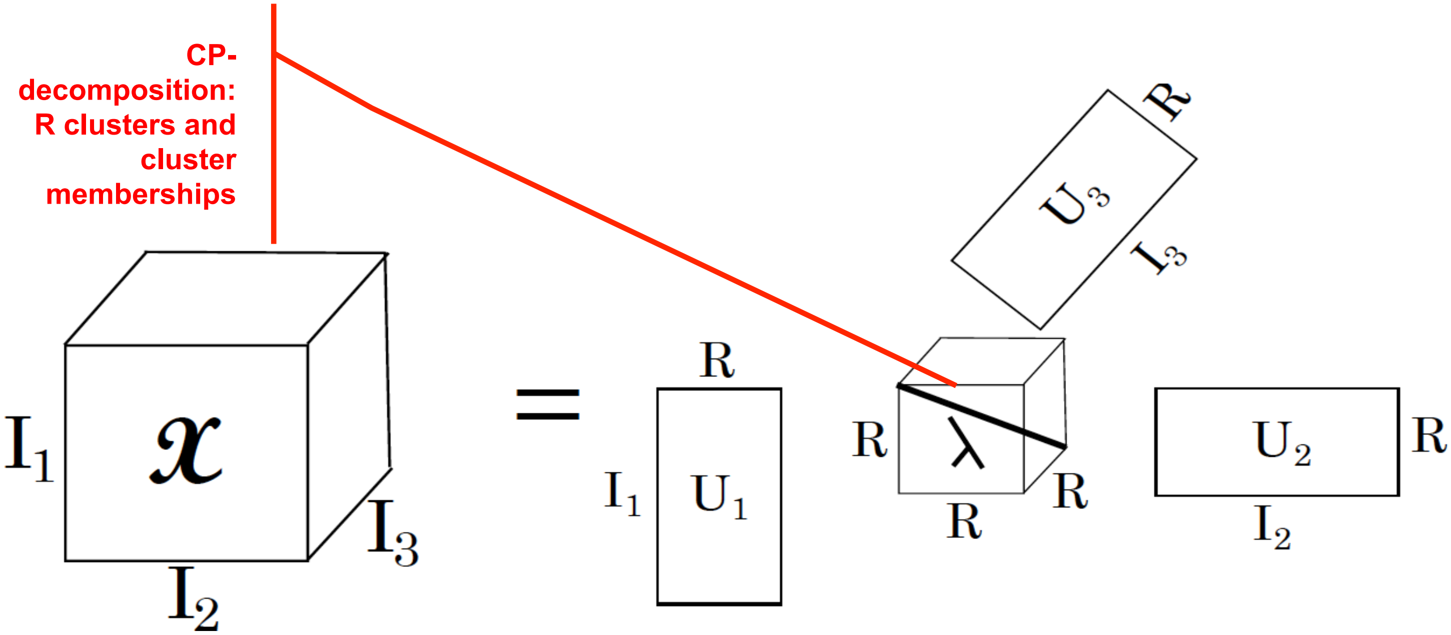
E.g.



A	B	C
:	:	:
a	b	2
:	:	:

Tensor Representation of the Data

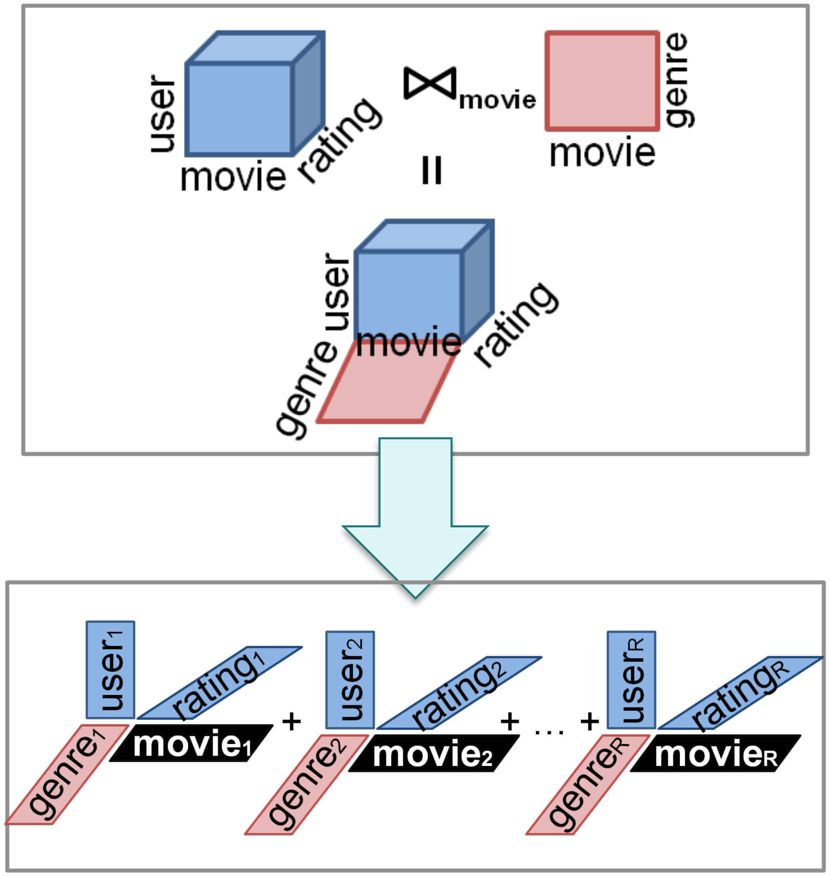
- Tensor decomposition [CP,Tucker] can be used for
  - understanding spectral characteristics of the data and
  - clustering the data based on inter-dependencies.



CP-decomposition: R clusters and cluster memberships

Example: A Query Involving Tensor Manipulation and Decomposition

STEP 1:



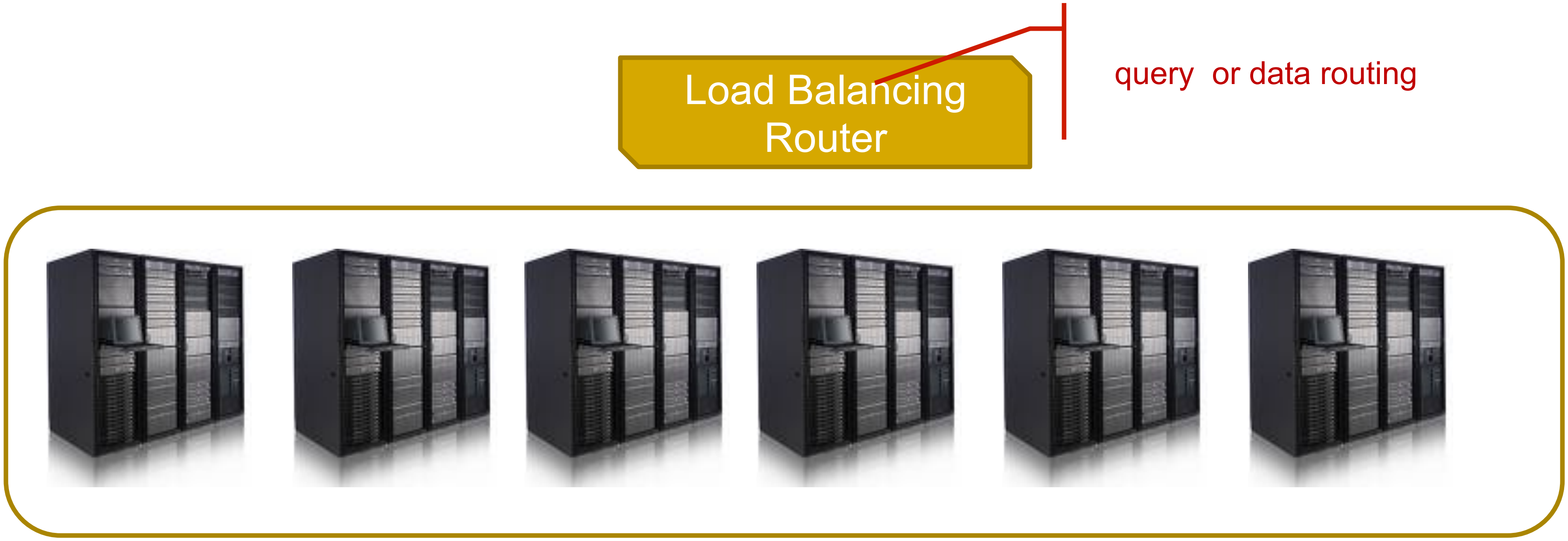
JOIN (costly)

STEP 2:

DECOMPOSITION (much costlier!!)

How to scale "data processing"?

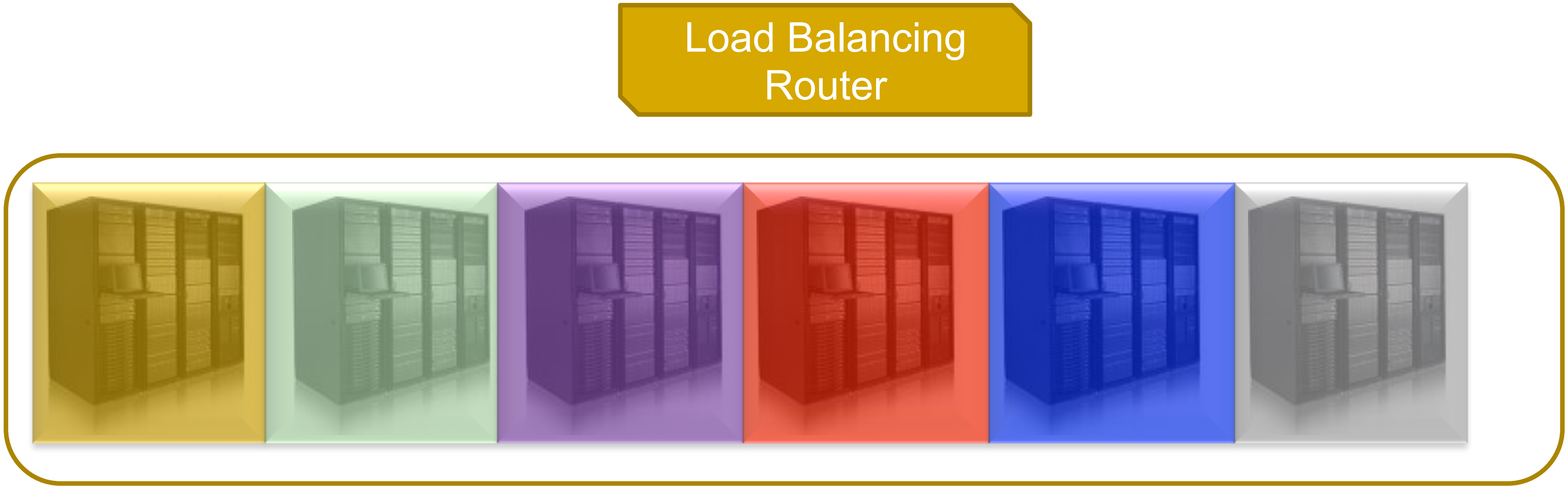
- Data distribution/parallelism….



Load Balancing Router

query or data routing

Data distribution/parallelism

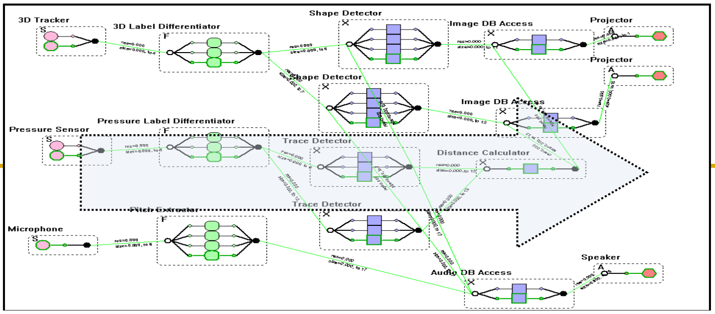


- Horizontal process partitioning
  - Data sets are partitioned, mapped, and processed by individual nodes

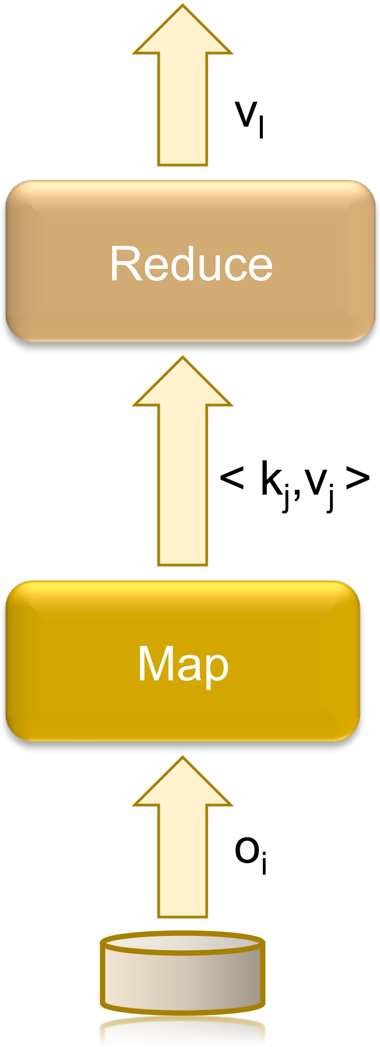
Data distribution/parallelism

Load Balancing Router

- Vertical process partitioning
  - Data is processed in a pipelined manner on the processing nodes in succession



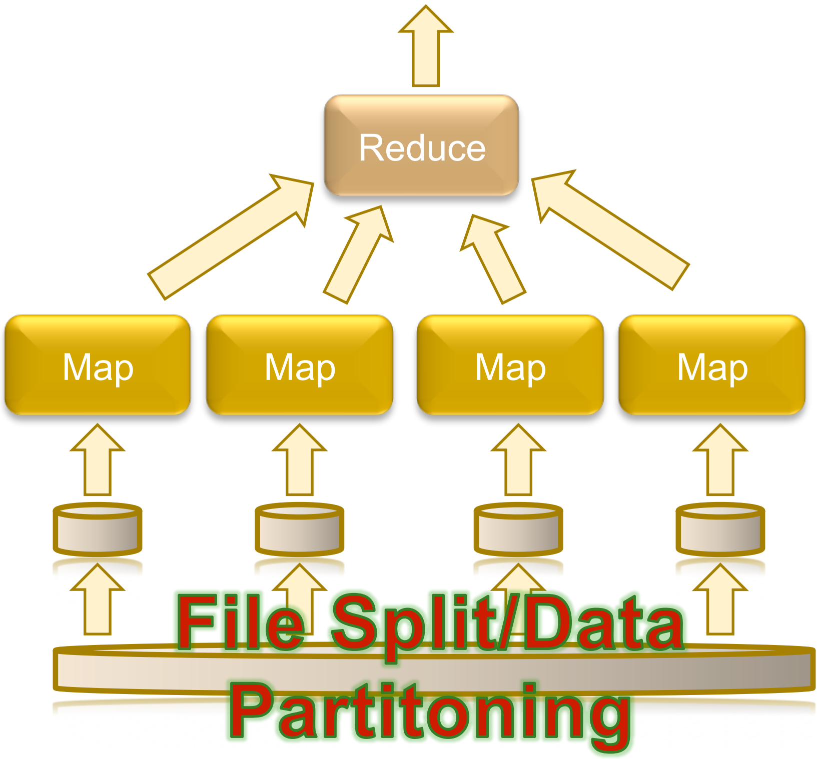
Example: MapReduce framework



- Reduce
  - reads key/value pairs
  - processes/transforms and returns different key/value pairs
  - acts as an aggregation, clustering, or classification operation

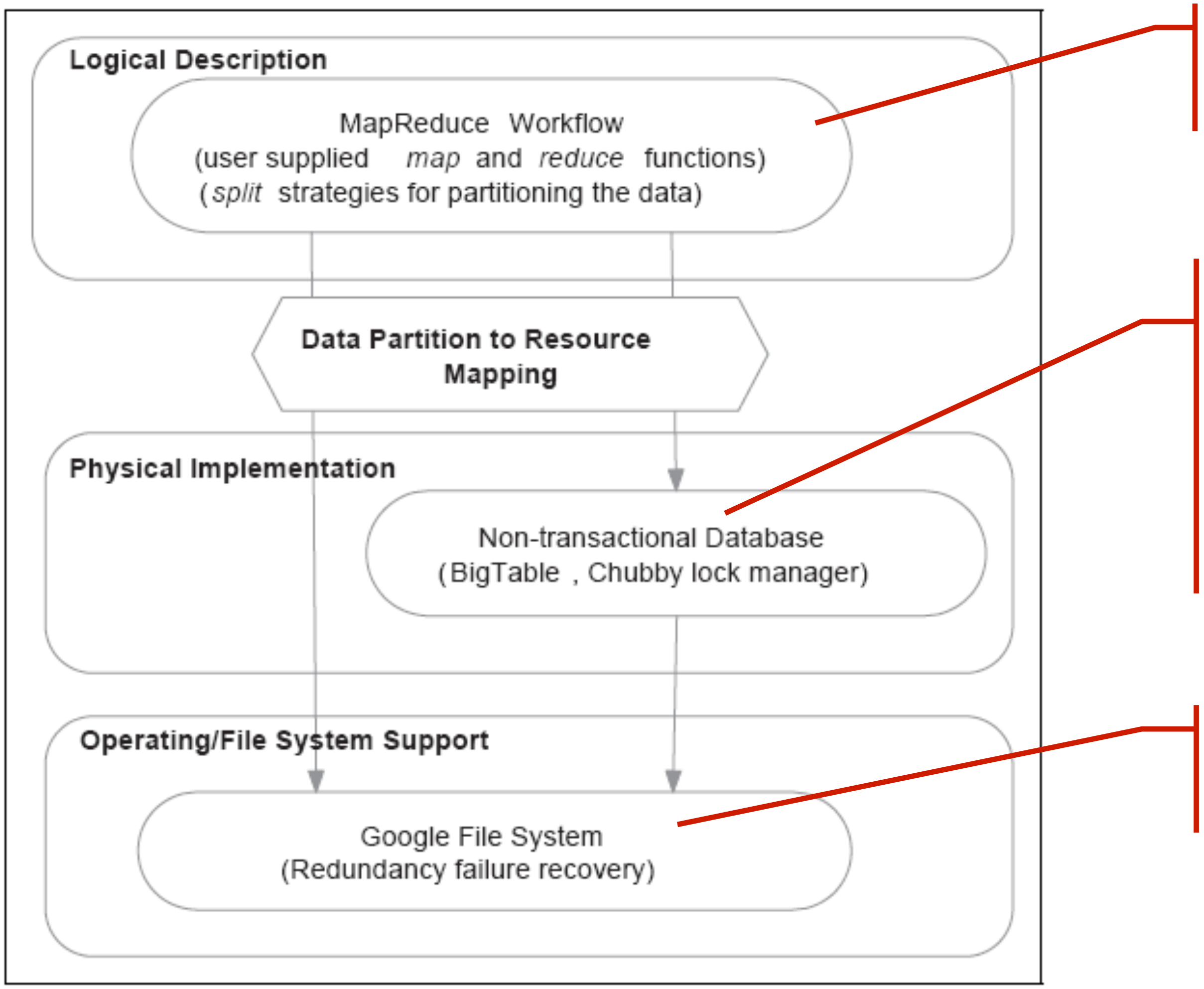
- Map
  - reads list of objects
  - extracts and returns key/value pairs
  - acts as a feature extractor

Example: MapReduce for hybrid parallelism



- map and reduce primitives can be combined into a workflow
- data parallelism: distribute "map"s across multiple machines
  - partition the input data into a set of splits
  - splits are processed in parallel by different machines
- common split strategies:
  - random
  - key range based

MapReduce framework



Logical Description

MapReduce Workflow
(user supplied *map* and *reduce* functions)
(*split* strategies for partitioning the data)

Data Partition to Resource Mapping

Physical Implementation

Non-transactional Database
(BigTable , Chubby lock manager)

Operating/File System Support

Google File System
(Redundancy failure recovery)

- Pig Latin
- Hive QL

- Google BigTable
- Hbase
- Greenplum's RDBMS
- Microsoft Dryad and Scope
- Amazon's Dynamo
- Clustera
- ETH Zurich's Database on S3
- Apache CouchDB

- Google File System
- Hadoop Distributed File Systems

MapReduce – based systems

	Key-base Access		SQL Supported				Transaction Supported		
	HBase/ Bigtable	Dynamo	Scope	Pig Latin	Dryad	Clustera	PNUTS	Database on S3	Cassandra
Data Model	Distributed column-based data store	Key-value pair only	Serialized file	Nested data model	Serialized file or file with schema	Serialized file	Simple relation model	Relational model	Distributed column-based data store
SQL	Key-based selection or scan, supports bloom filtering, map-reduce	Key-based access	Key-based selection, bloom filter	Supports limited SQL queries	Supports full fledged SQL queries	Supports full fledged SQL queries	Supports limited SQL queries	Supports full fledged SQL queries	Key-based selection, bloom filter
Data Storage	Several indexing scheme implementation in progress	Use consistent hashing to locate a record	Serialized file, hierarchical metadata	Serialized file	Serialized file, record	Serialized file, metadata in database	Table, record	B-tree indexes	Secondary indexing?
Update	Append only	Supported	Append only	Not supported	supported	Supported	supported	supported	Supported
Transaction	No concept of transaction	No concept of transaction	No concept of transaction	No concept of transaction	No concept of transaction	No concept of transaction	Application transaction but stops short of full serializability	Maximize consistency and other transactional guarantees	Single row transaction
Job Schedule	Not Supported	Supported	Supported	Supported	Supported	Supported	Not Supported	Not supported	Supported
Scalability	Highly scalable	Highly scalable	Highly scalable	Highly scalable	Highly scalable	Highly scalable	Highly scalable	Highly scalable	Highly scalable
Availability/ Consistency	Strong consistency, highly available	Eventual consistency, highly available	Eventual consistency, highly available	Read only	Eventual consistency, highly available	Eventual consistency	Pre-record consistency, highly available	Eventual consistency, highly available	Eventual consistency, highly available
Powered-by	Yahoo! & Microsoft	Amazon	Microsoft	Yahoo!	Microsoft	Wisconsin-Madison	Yahoo!	ETH Zurich	Facebook