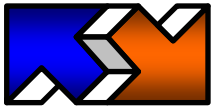


MIDDLEWARE SYSTEMS
RESEARCH GROUP

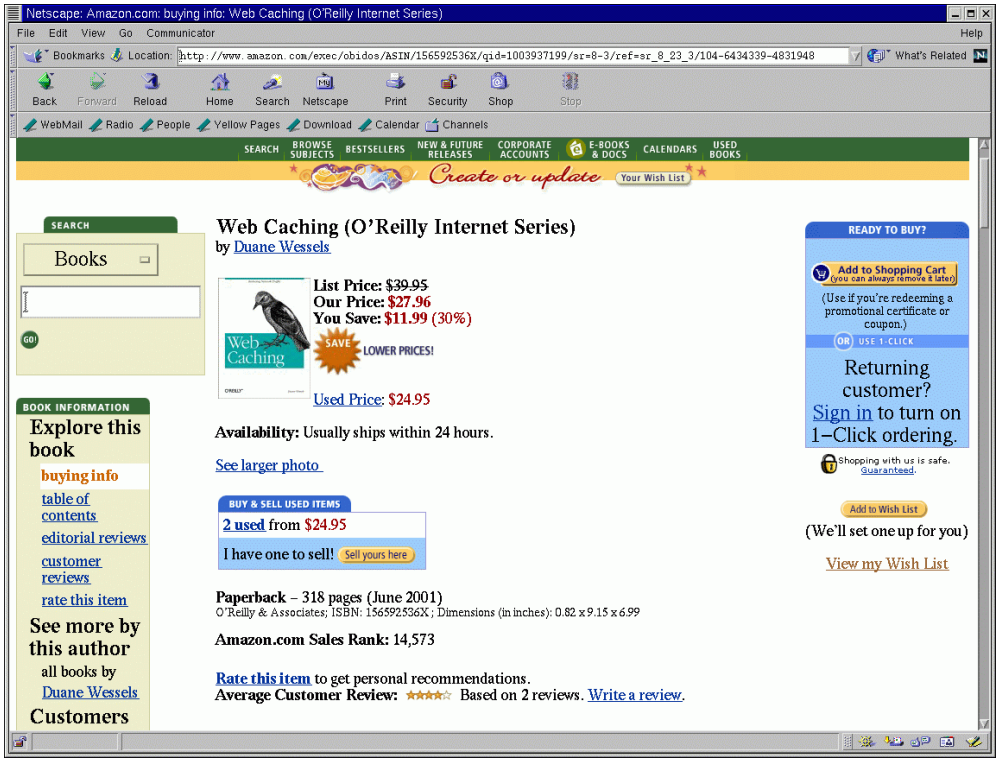
Publish/Subscribe

Hans-Arno Jacobsen

Bell University Laboratory Chair in Software Engineering
Middleware Systems Research Group
University of Toronto

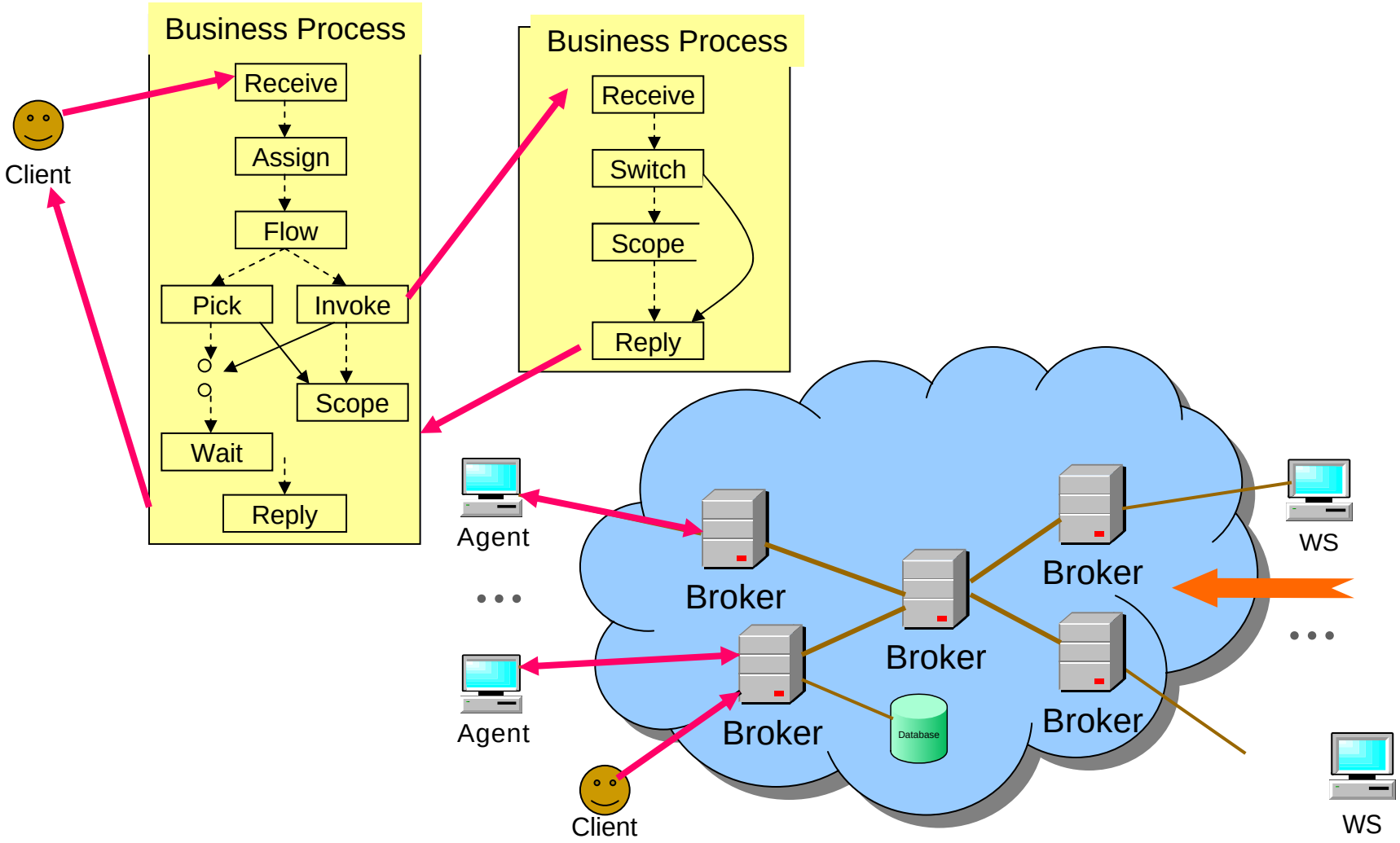
Amazon to Chapters to you ... .

Monday, October 10th in Cyberspace

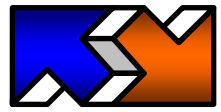


Thursday, November 15th, in Toronto

Business Process Execution & Web Service Composition



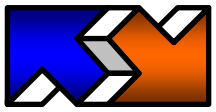
Other "Similar" Applications



- Selective information dissemination
- Location-based services
- Personalization
- Alerting services
- Application integration
- Job scheduling
- Monitoring, surveillance, and control
- Network and distributed system management
- Workforce management
- (Scientific) workload management
- Business activity monitoring
- Business process management, monitoring, and execution

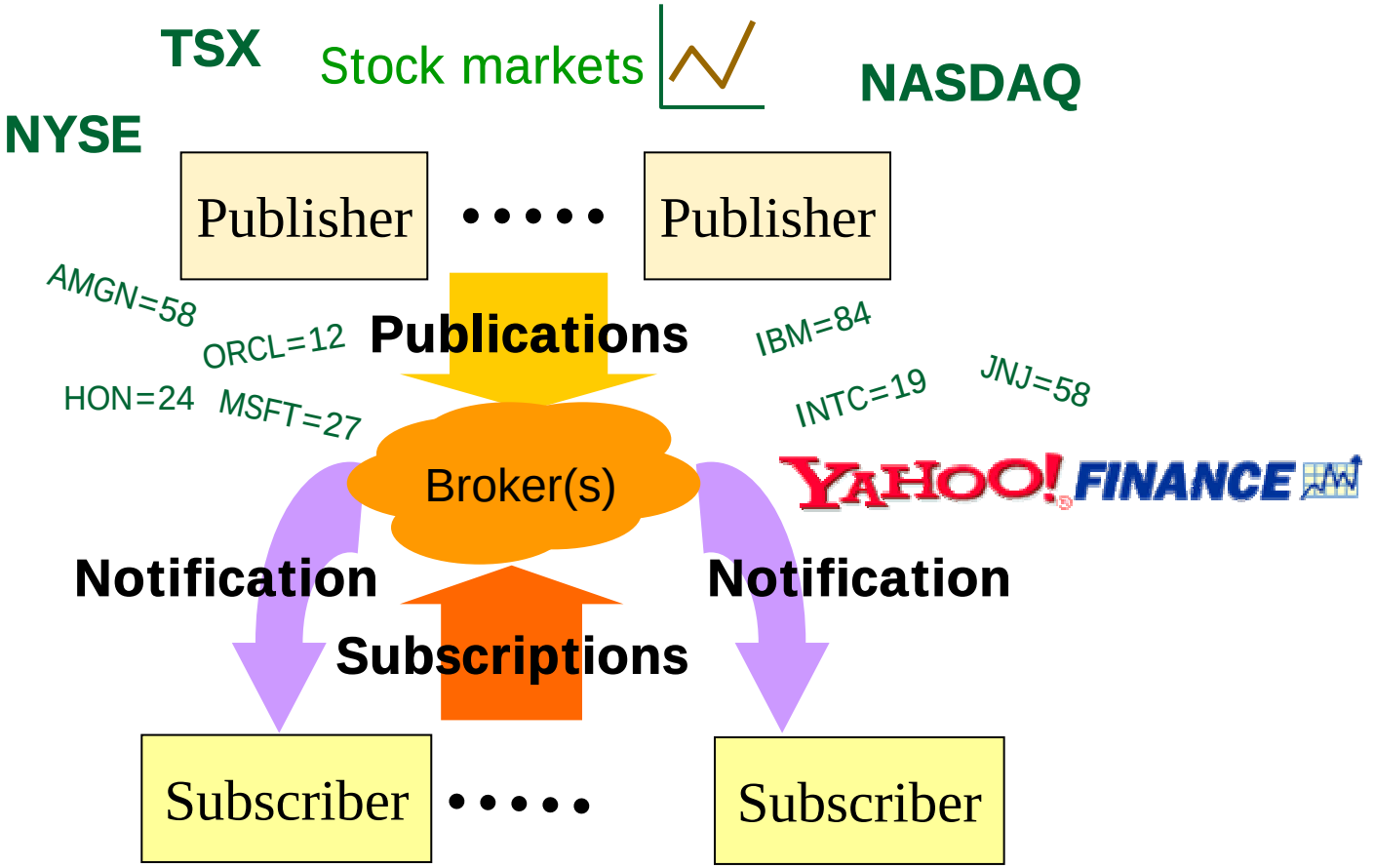
What Relates All These Applications?

- **Asynchronous state transitions** captured as **events** that
  - drive and underlay
    - all applications and
    - infrastructures implementing these applications
- Require middleware support for **event processing**
- Publish/Subscribe is ideally suited to fulfill these requirements



Publish/Subscribe

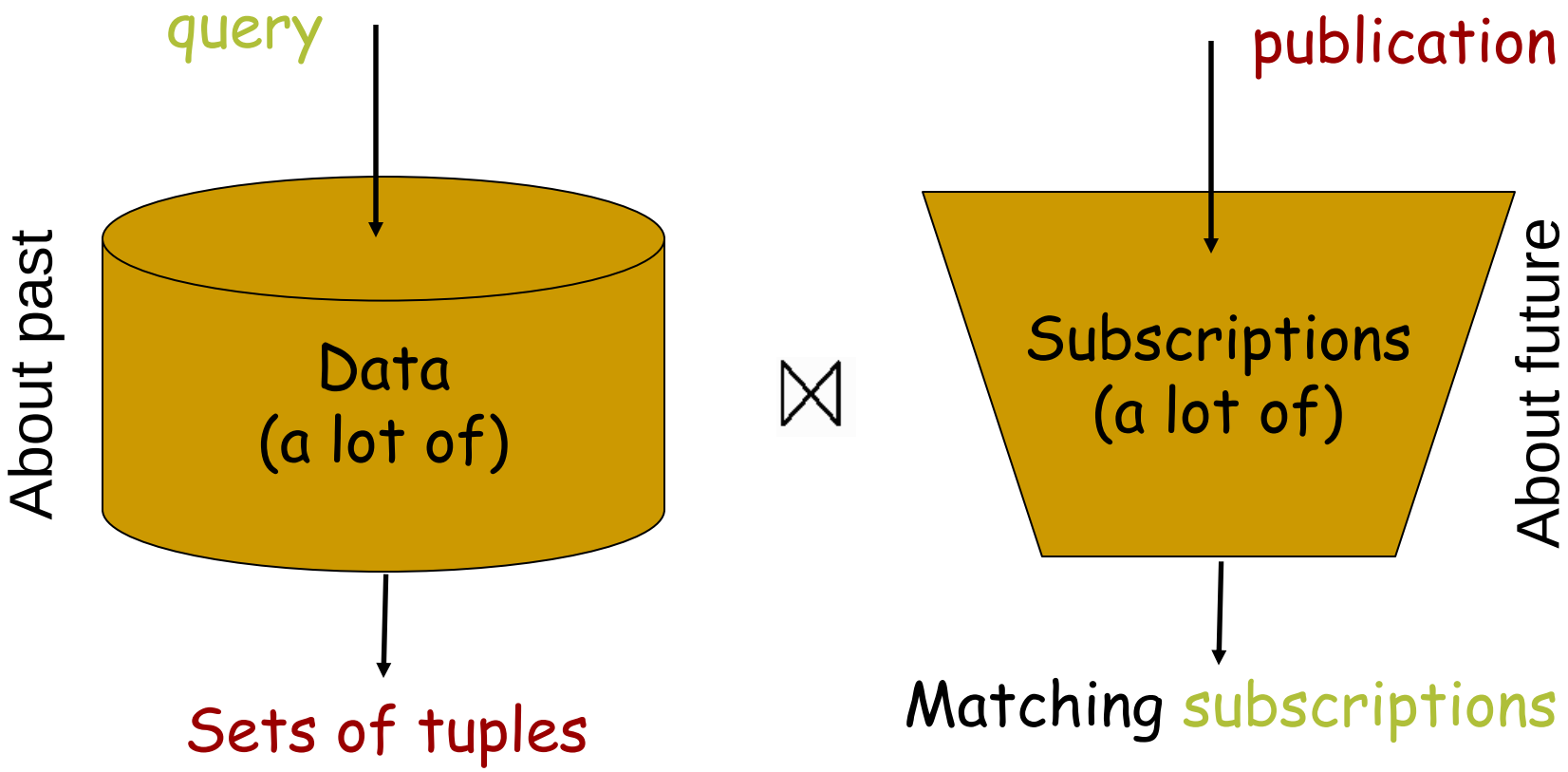
Publish/Subscribe



Subscriptions:
IBM > 85
ORCL < 10
JNJ > 60

That's Like Data Base Querying ☹ !!

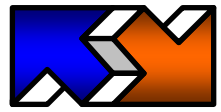
MIDDLEWARE SYSTEMS RESEARCH GROUP



Query and ***subscription*** is very similar.
Set of tuples and ***publication*** is very similar.

However, the two problem statements are inverse.

Key Benefit of Publish/Subscribe



- Decoupling of publishers and subscribers
  - Publishers do not need to know subscribers
  - Publishers and subscribers do not need to be up simultaneously
  - Amenable for physical distribution

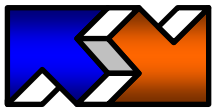
Benefits of Publish/Subscribe

- **independence** of participants
- lends itself well to distributed system development
  - de-coupled development & processing
  - (dynamic) system evolution
- **interaction** with **large number** of entities facilitated
- naturally supports **non-continuous** operations
- potential for scalability & fault-tolerance
- open for (legacy) **system integration** on either end

Of course it is not a ***one size fits all*** paradigm, but a good solution for certain kinds of problems.

Publish/Subscribe Matching Problem

- Given a set of subscriptions, *S*, and a publication, *e*, return all *s* in *S* matched by *e*.
- *e* is referred to as **event** or **publication**
- Splitting hairs
  - *Event* is a state transition of interest in the environment
  - *Publication* is the information about *e* submitted to the publish/subscribe system
- Simple problem statement, widely applicable, and lots of open questions



Problem Instantiations I

- Text / search strings (information filtering)
- Semi-structured data / queries
  - **attribute-value pairs / attribute-operator-value-predicates**
  - XML, HTML
- Tree-structured data / path expressions
  - XML ./ XPath expressions
- Graph-structured data / graph queries
  - RDF / RDF queries (e.g., SPARQL)
- Regular languages / regular expressions
- Tables / SQL queries

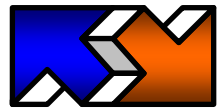
Problem Instantiations II

- Different matching semantics
  - **Crisp**
  - Approximate,
  - Similar
  - n-of-m (n of m predicates match)
  - Probability of match

Problem Instantiations III

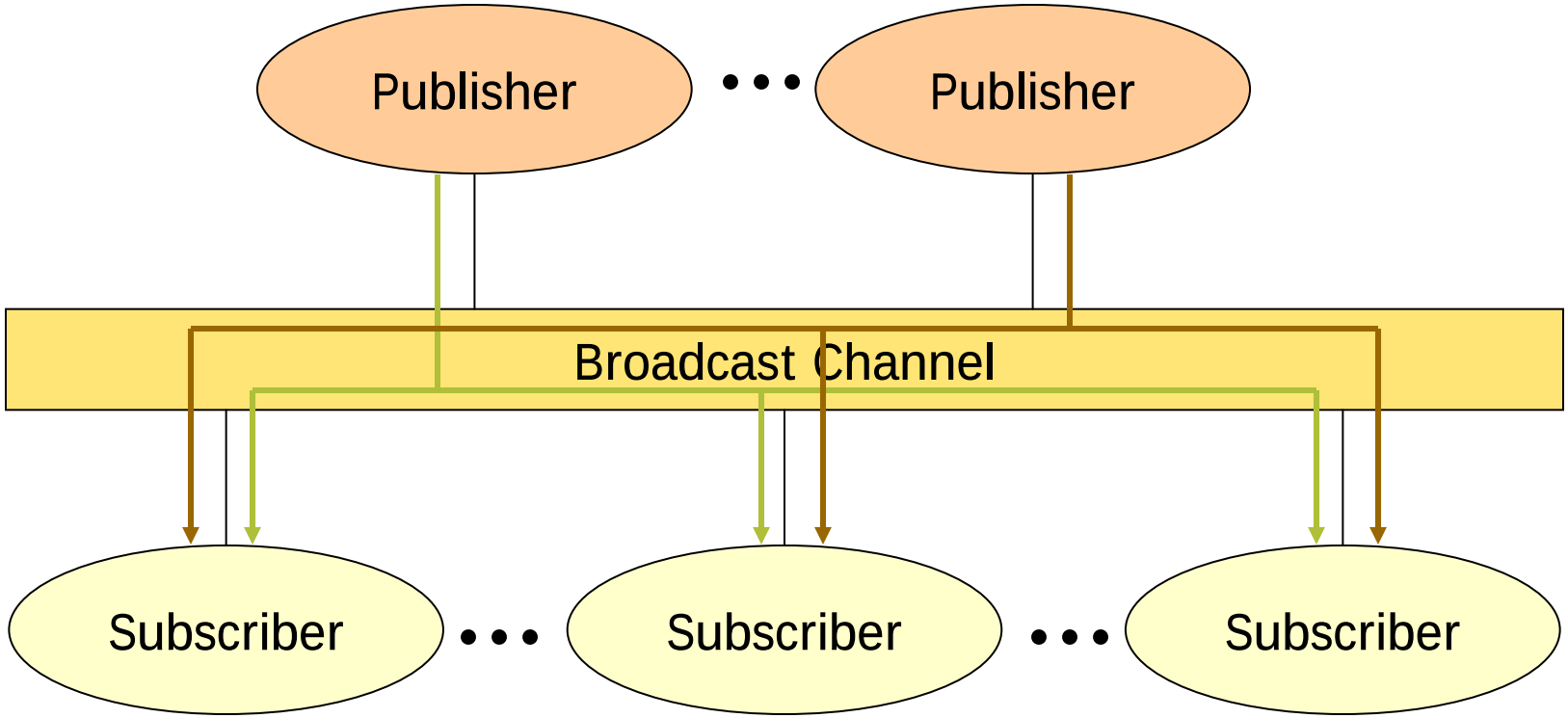
- Centralized and **distributed** instantiation
- Networking architecture
  - Internet (as overlay network)
  - Peer-to-peer style interface (DHT, table-lookup)
  - With mobile publishers, subscribers, brokers
  - Ad hoc network

Publish/Subscribe Models

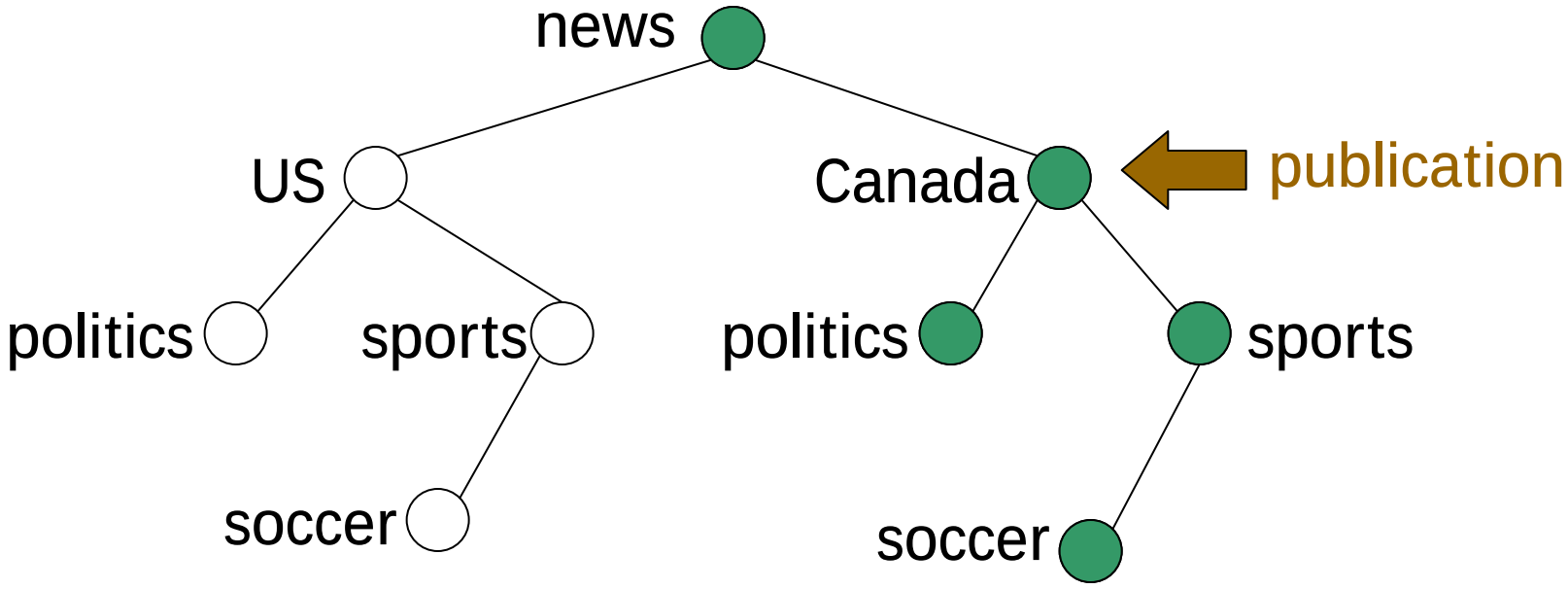


- Channel-based model
  - Subscribe & publish to a channel
- Topic-based model
  - … topics and topic hierarchy
- Type-based model
  - … typed objects
- Content-based model
  - … to content of messages
- Subject-spaces (State-based model)
  - Maintain state in publications and subscriptions

Channel-based



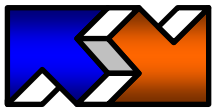
Topic-based publish/subscribe



The Content-based Model

- Language and Data model
  - Conjunctive Boolean functions over predicates
  - Predicates are attribute-operator-value triples
    - **`[class,=,trigger]`**
  - Subscriptions are conjunctions of predicates
    - **`[class,=,trigger],[appl,=,payroll],[gid,=,g001]`**
  - Publications are sets of attribute-value pairs
    - **`[class,trigger],[appl,printer],[gid,g007]`**
- Matching semantic
  - A subscription matches if all its predicates are matched

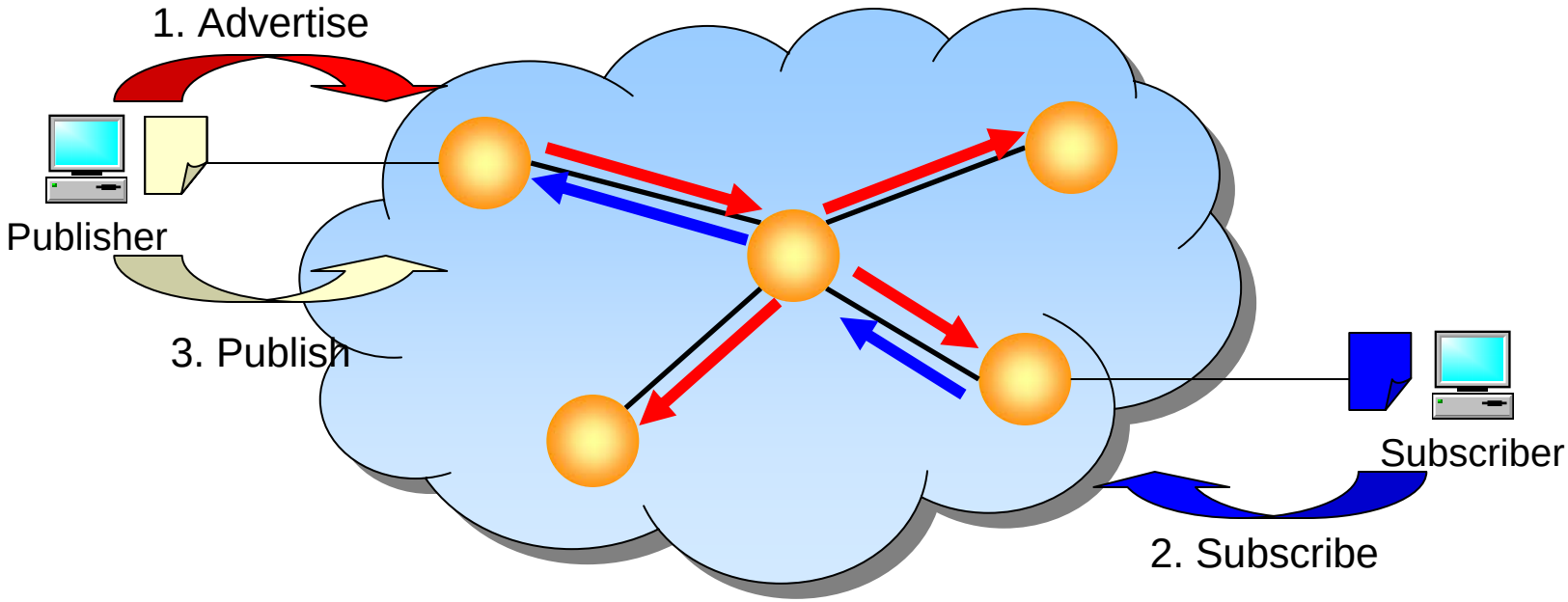
Content-based routing



- Distributed publish/subscribe
- **Network of** publish/subscribe **brokers**
- Subscriptions & publications are injected into network at closest edge broker
- Routing protocol distributes subscriptions throughout network
- Network **routes relevant publications to interested subscribers**
- **Routing is based on content**; it is not based on addresses, which are not available
- Subscriptions may change dynamically

Content-based Routing

MIDDLEWARE SYSTEMS
RESEARCH GROUP



A: [class, =, stock], [name, =, HP], [price, >, 50]
S: [class, =, stock], [name, =, *], [price, >, 50]
P: [class, stock], [name, *], [price, 50]

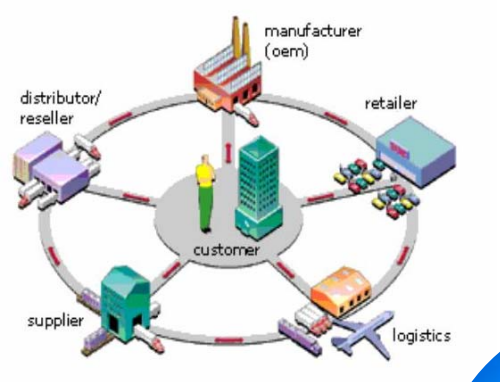
Event-Based | Content Routing | Flexible

Decoupled | Declarative | Responsive

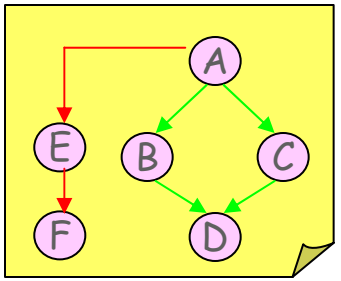
Applications

MIDDLEWARE SYSTEMS RESEARCH GROUP

Supply chain and logistics



Workflows, business processes and job scheduling



Event-Based

In flight

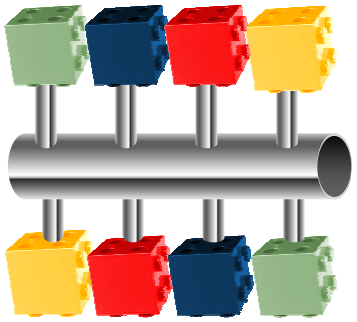
Delivered

Order

Job A done

Trigger

Fault



Callback

Invoke Loan

Transform

Razor SKU

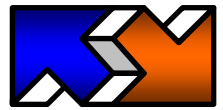
Light

Temperature

Service oriented architecture

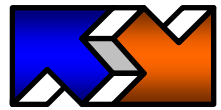
RFID and sensor networks

Publish/Subscribe in Industry



- Standards
  - CORBA Event Service
  - CORBA Notification Service
  - OMG Data Dissemination Service
  - Java Messaging Service
  - WS Eventing
  - WS Notification
  - INFO-D (Grid Forum)
- Emerging technologies
  - RSS aggregators
    - PubSub.com, FeedTree
  - Real-time data dissemination
    - TIBCO, RTI Inc., Mantara Software
  - Application integration
    - Softwired
  - Hardware-based brokers
    - Sarvega (Intel), Solace Systems, DataPower (IBM)

Publish/Subscribe in Academia



- Research projects
  - Elvin (Australia)
  - Gryphon (IBM)
  - Hermes (Cambridge)
  - SIENA (Boulder)
  - REBECA (Darmstadt)
  - ToPSS (UofT)
  - PADRES (UofT)

The Toronto Publish/Subscribe System Family (ToPSS)

- Matching algorithms
  - Language expressiveness, scalability, speed
- Routing protocols
  - Network architectures, scalability
- Higher level abstractions
  - Workflow execution
  - Monitoring

ToPSS (matching)
A-ToPSS (approximate)
CS-ToPSS (composite subs)
S-ToPSS (semantic)
L-ToPSS (location-based)
Rb-ToPSS (rule-based)
X-ToPSS (XML matching)
persistent-ToPSS (subject spaces)

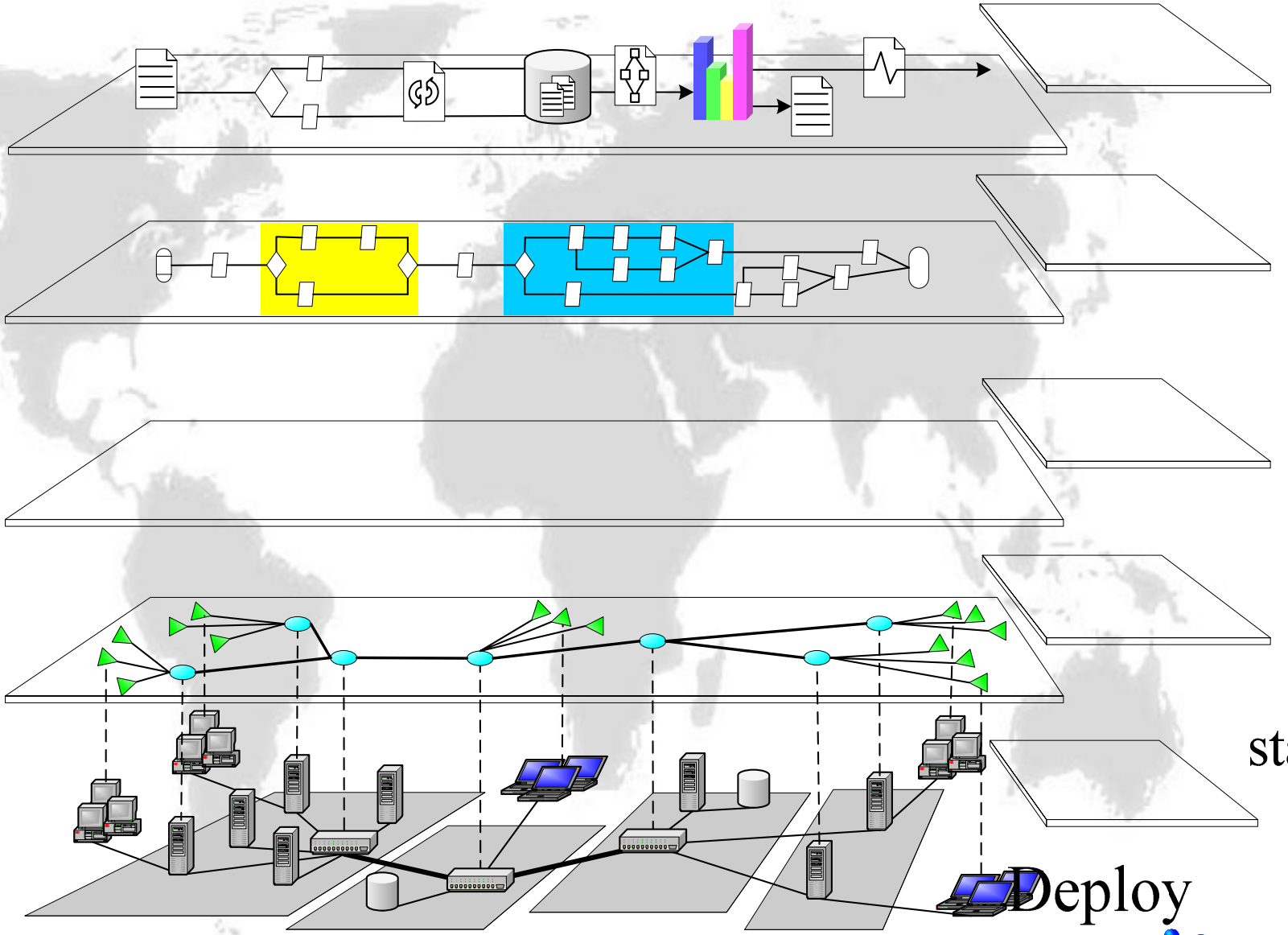
M-ToPSS (mobile)
P2P-ToPSS (peer-to-peer)
LB-ToPSS (load balancing)
Federated-ToPSS (federation of ToPSS brokers)
Ad hoc-ToPSS (ad hoc networking)

Historic-ToPSS (historic data)
FT-ToPSS (fault tolerance)
JS-ToPSS (job scheduling)
BPEL-ToPSS (BPEL execution)

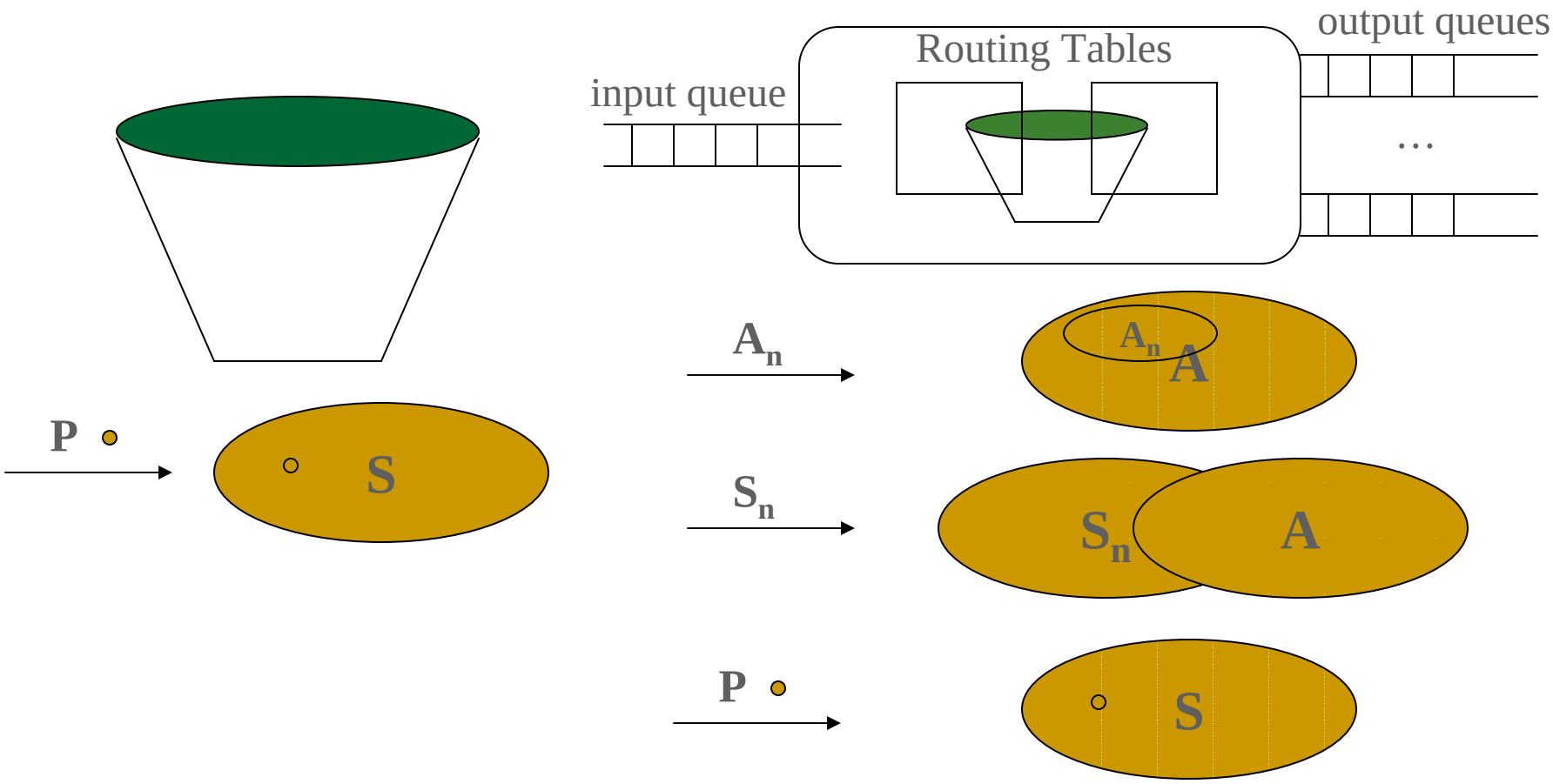
Overall Project Vision

MIDDLEWARE SYSTEMS
RESEARCH GROUP

A Real-Time Event-driven Enterprise



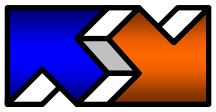
Matching and Content-based Routing



Matching Algorithms

- For solving the pub/sub matching problem

- Tree-based algorithms
- Graph-based algorithms
- Automaton-based algorithms (NFA, DFA)
- **Two-staged algorithms**
  - predicate matching
  - subscription matching



MIDDLEWARE SYSTEMS
RESEARCH GROUP

Predicate Matching

Predicate matching problem

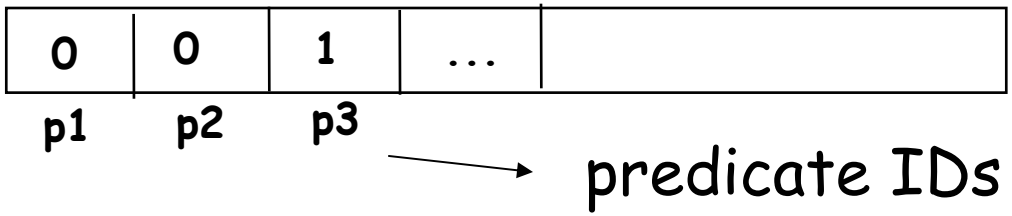
- Given a **set P** of predicates and an **event e**, ***identify all predicates p of P which evaluate to true under e***.
- Example:

  e = {…, (price, 5), (color, white) …}

  p1: price > 5; p2: color = red; p3: price < 4

  p1 is false p2 is false p3 is true

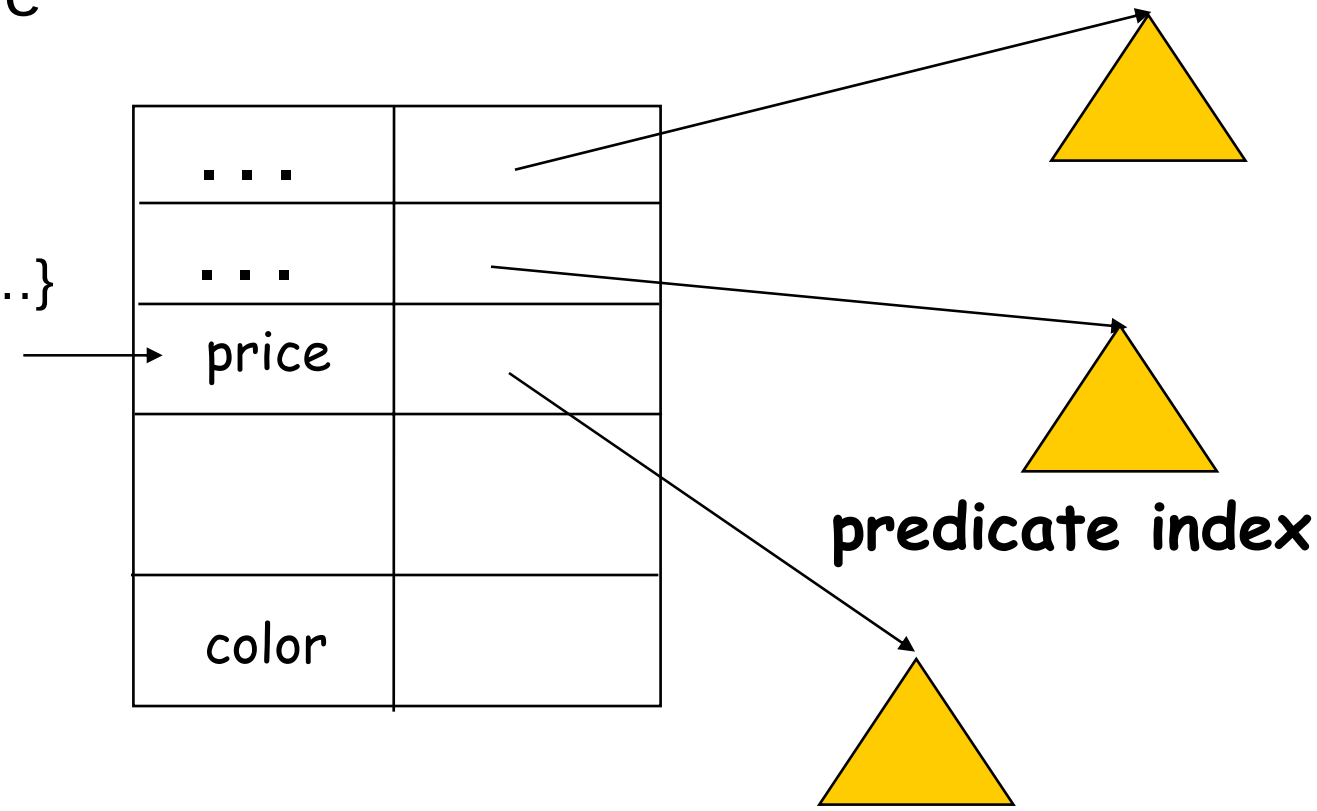
  predicate bit vector:



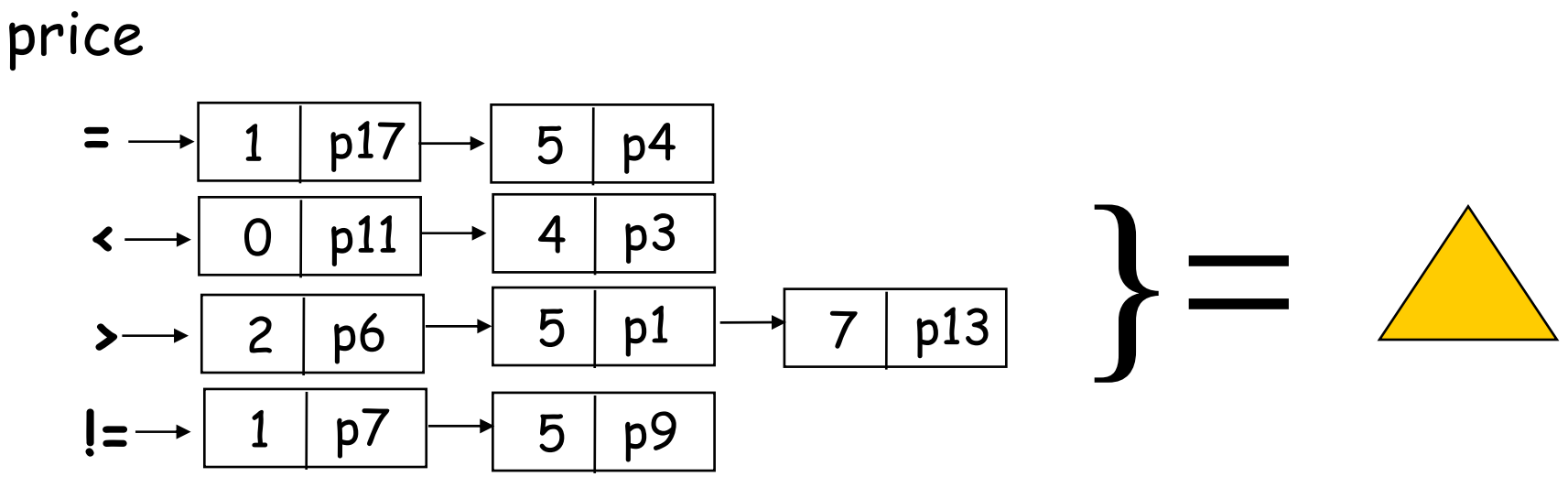
Data structure overview

- hash table on attribute name

e = {…, (price, 5), …}



Predicate Index

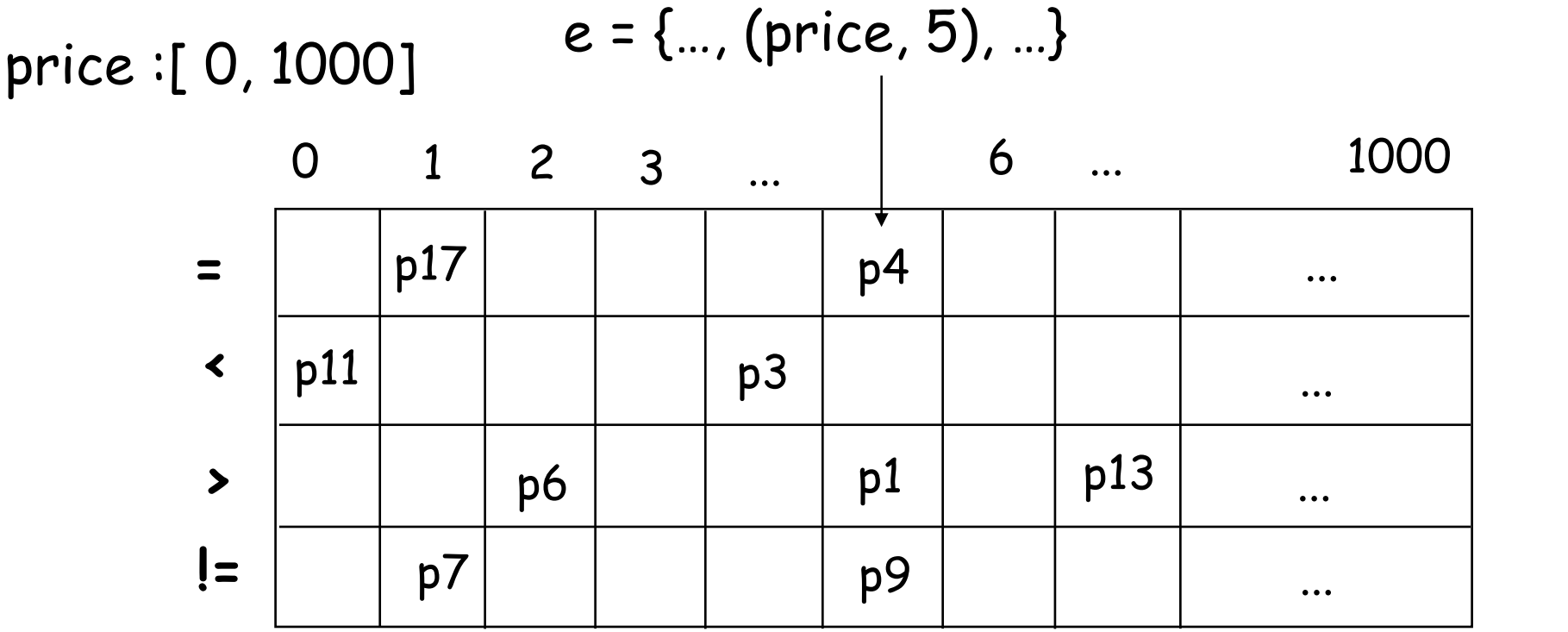


- one ordered linked list per operator
- insert, delete, match are O(n)-operations (per attribute name in ***e*** and per operator)
- alternatively, use a B-tree or B+-tree etc.

Observations

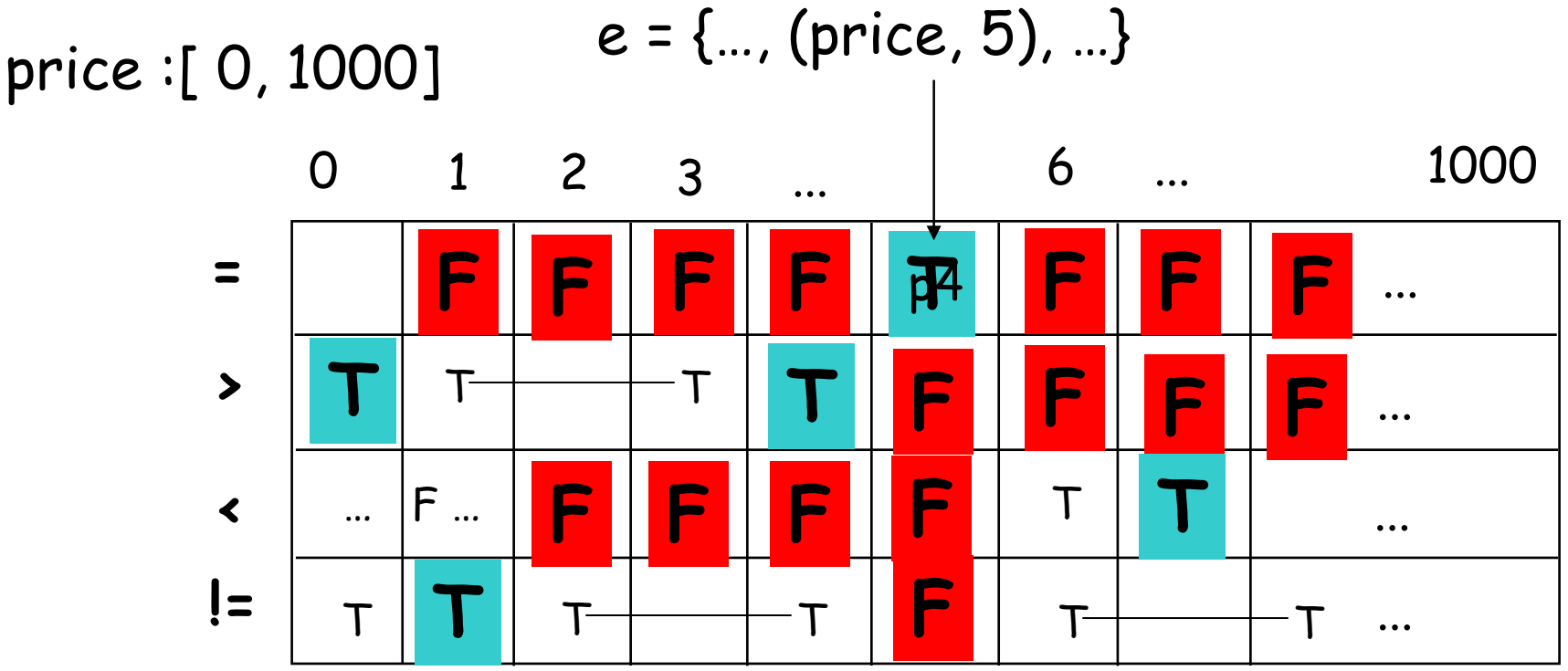
- countable domain types with small cardinality
  - integer intervals
  - collections (enums)
  - a set of tags
- Examples
  - price : [0, 1000], models variety of prices
  - color, city, state, country, size, weight
  - all tags defined in a given DTD
  - predicate domain is often context dependant, but limited in size
    - prices of cars vs. prices of groceries

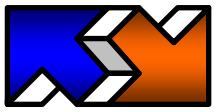
Predicate Matching for Finite Domains



p1: price > 5; p3: price < 4; p4: price = 5; p9: price != 5...

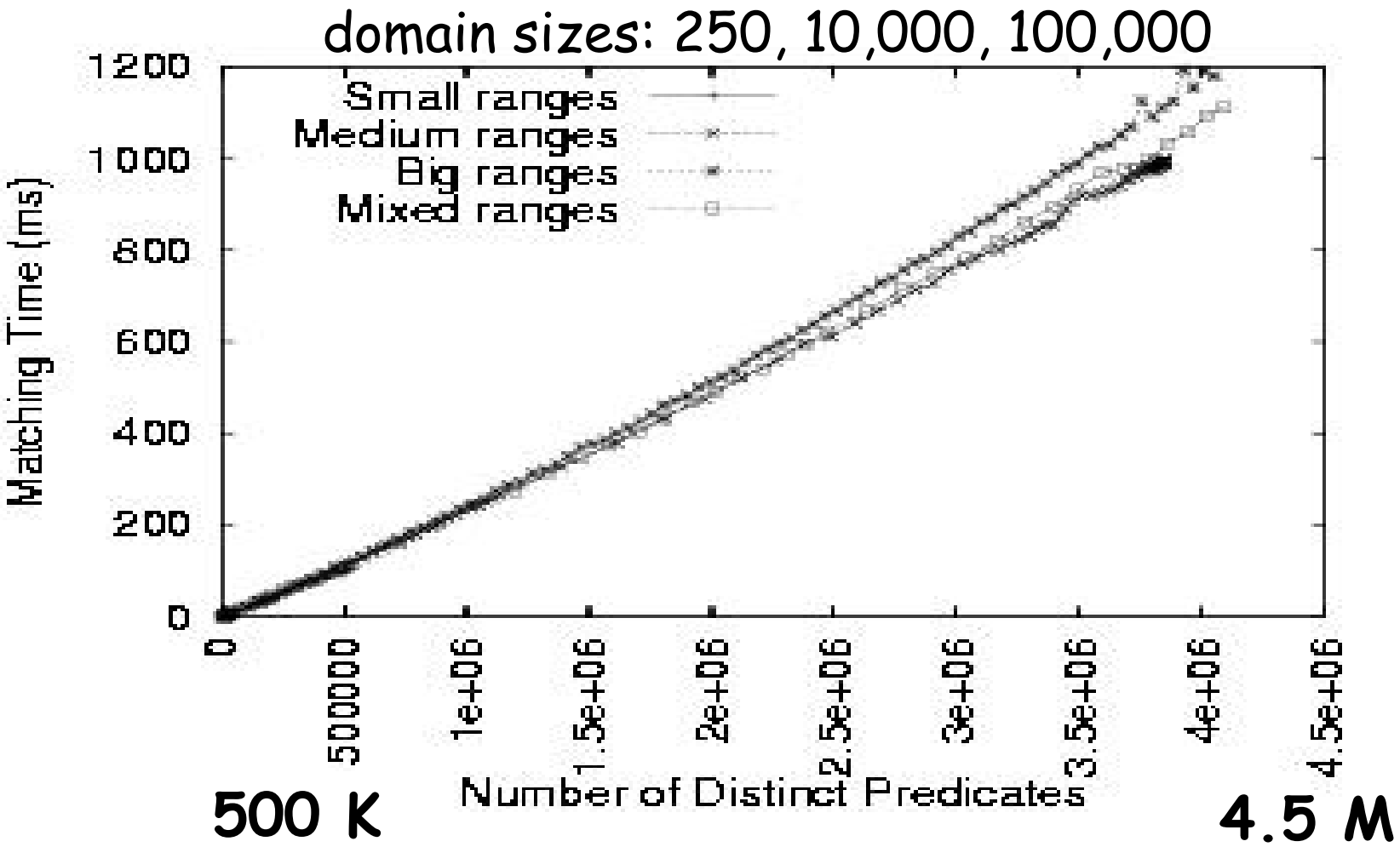
Predicate Matching Symmetries



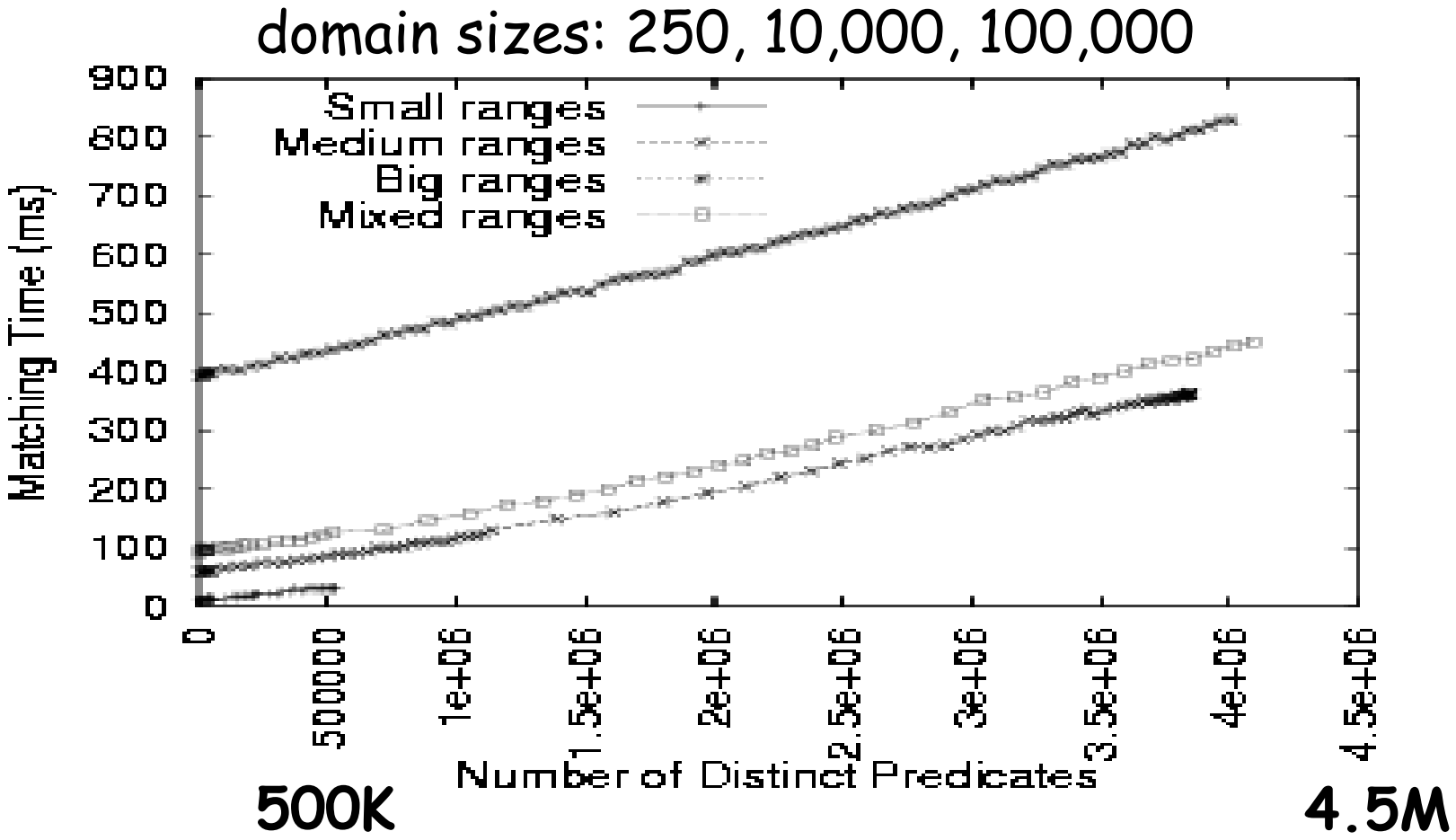


Experiments and evaluation

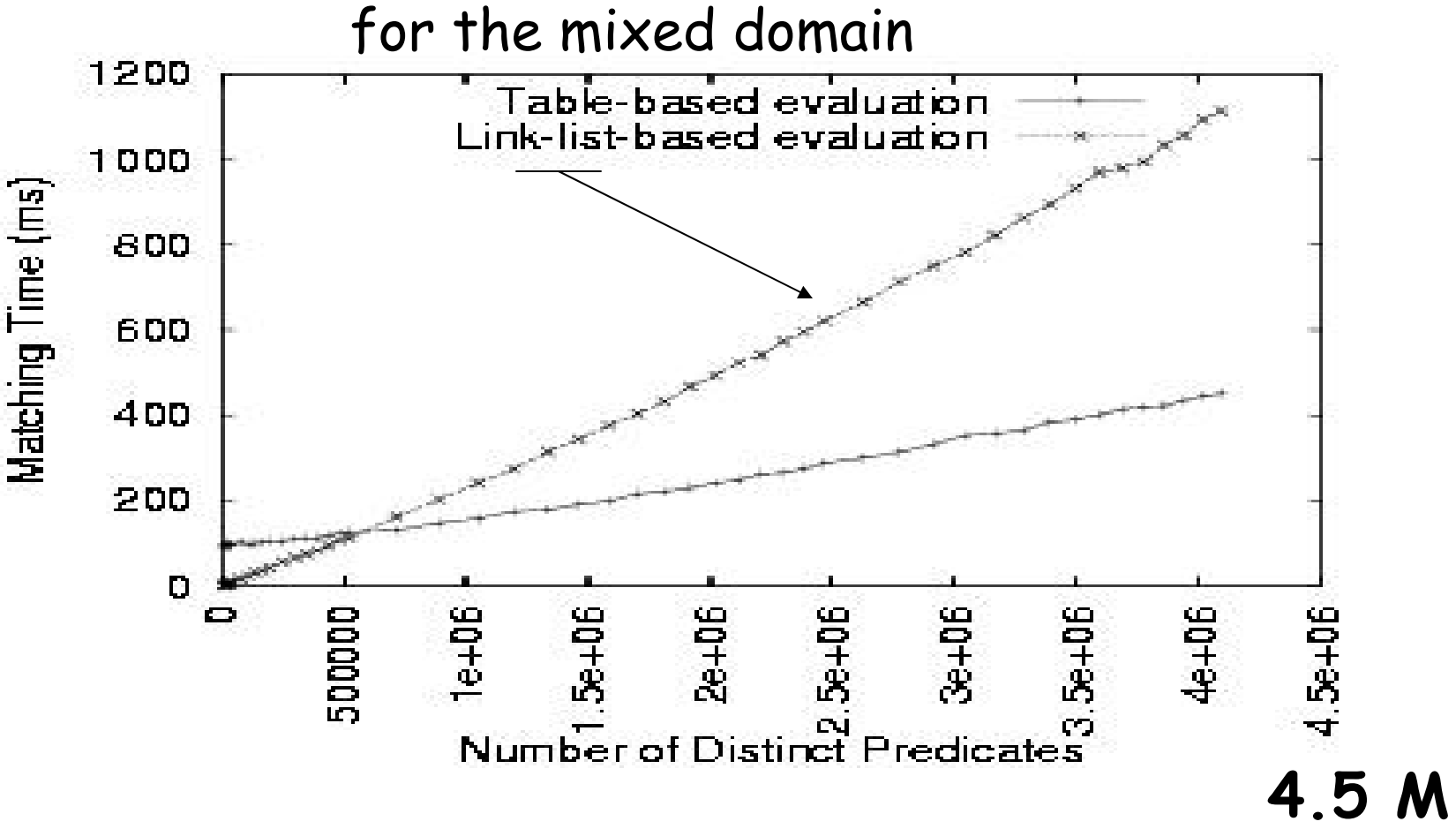
Predicate Matching Performance (list-based scheme)



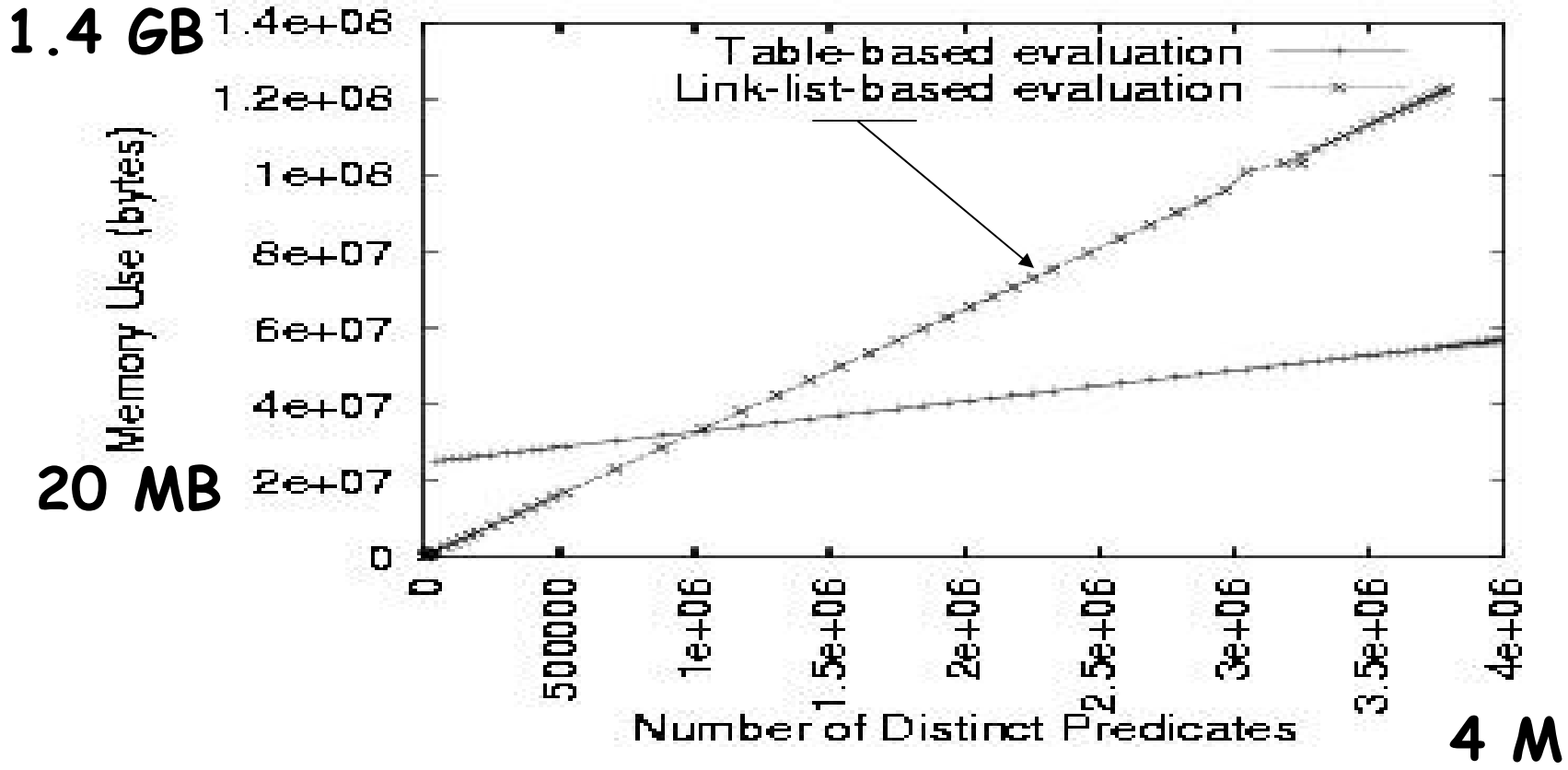
Predicate Matching Performance (table-based scheme)

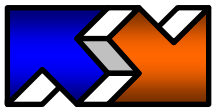


Predicate Matching Performance (tables-based vs. list-based scheme)



Predicate Matching Data Structure Size (table-based vs. list-based scheme)



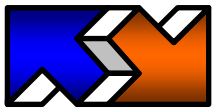


Subscription Matching

Multiple one-dimension indexes

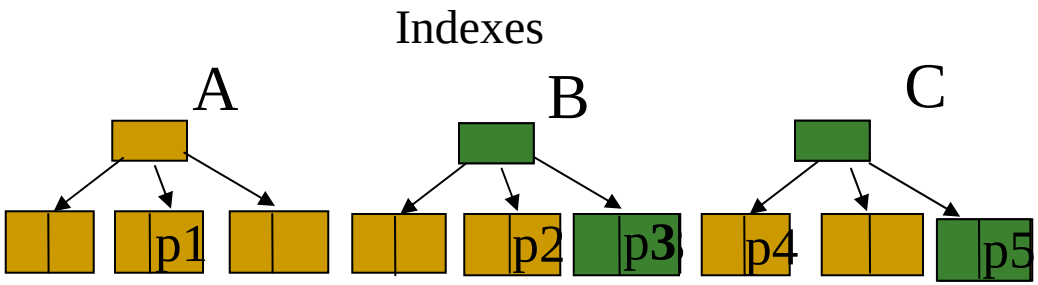
- One-dimension indexes.
  - hash tables
  - B-trees
  - Interval Skip Lists
- Counting algorithm
- Hanson algorithm
- Propagation algorithm

Counting algorithm



- Subscriptions consist of a set of predicates
  - S1: (2< A<4) & (B=6) & (C >4) ⇒ pA : (2< A<4), pB (B=6), pC:(C>4)
  - S1: (2< A<4) & (C=3)⇒ pA : (2< A<4) pC: (C=3)
- A Subscription matches the event if all its predicates are satisfied.
- **Idea:** Count the number of satisfied predicates per subscription

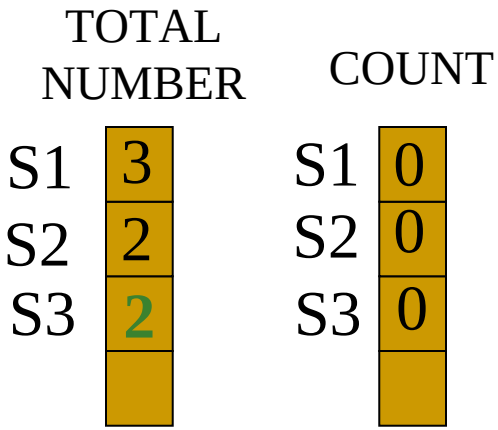
Data structures for the counting algorithm

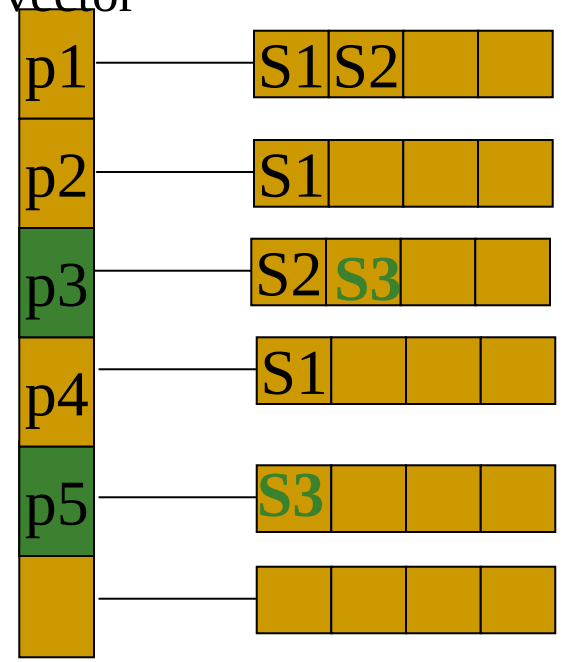


S1: p1,p2,p4

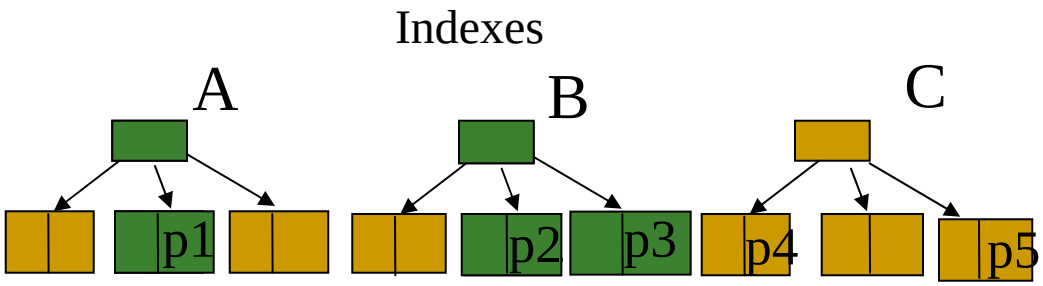
S2: p1,p3

S3: p3,p5





Counting algorithm



E: (A,5),(B,6)

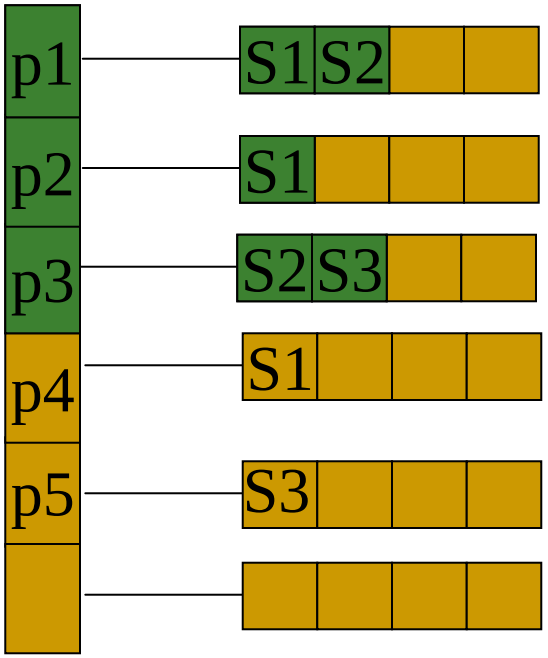
TOTAL NUMBER

S1	3
S2	2
S3	2

COUNT

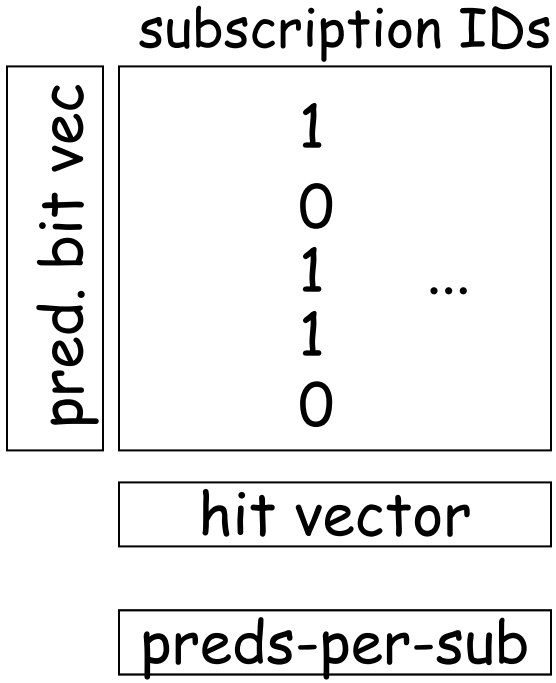
S1	2
S2	2
S3	1

Predicate vector

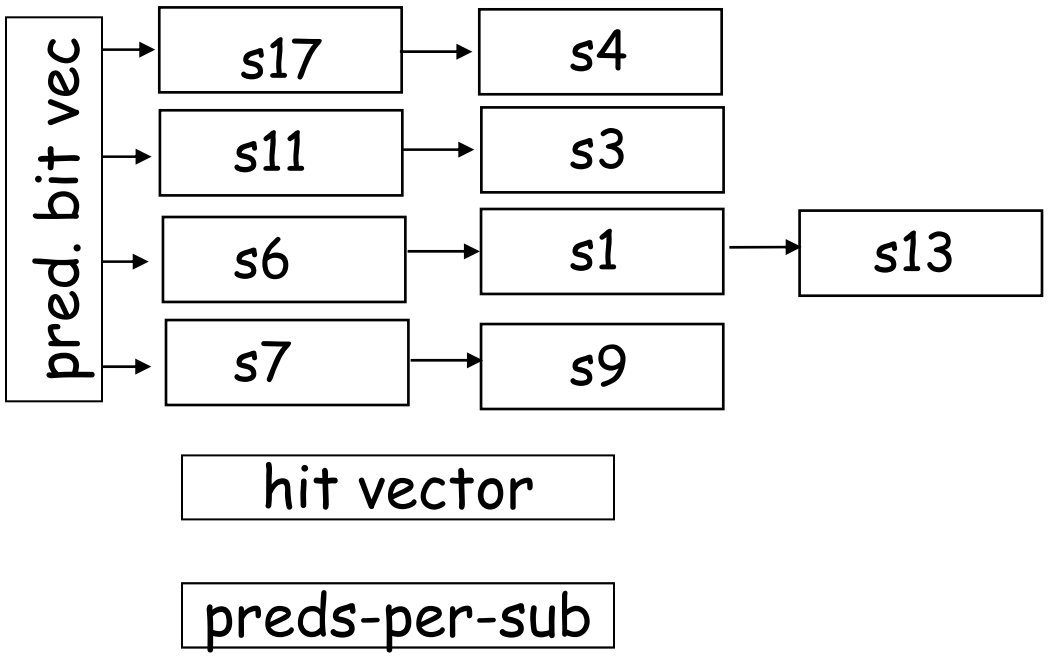


Subscription Matching

- bit-matrix based

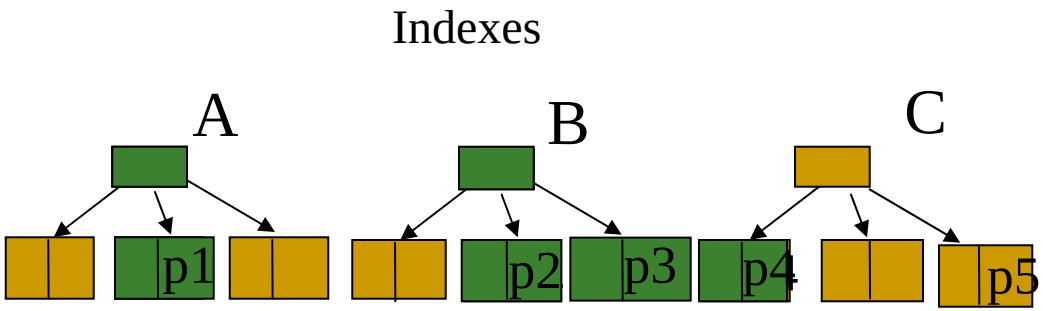


- list based



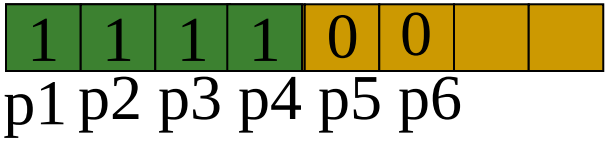
+ sub-pred association
to support deletion

Clustering Algorithm



E: (A,5),(B,6),(C,3)
satisfies p1,p2,p3,p4

Bit Vector



Access Predicate vector

S1: **p1**,p2,p4
S2: **p1**,p3,p5
S3: **p1**,p4