FROM DATABASE SYSTEMS TO KNOWLEDGE-BASE SYSTEMS: AN EVOLUTIONARY APPROACH

A TUTORIAL ON INTEGRATING DATABASE AND KNOWLEDGE-BASE TECHNOLOGIES

JiaWei Han

Conference Tutorial at the Eleventh International Conference on Data Engineering (ICDE’95)

Taipei, Taiwan

March 7, 1995
1. From database systems to knowledge-base systems: Necessity and possibility.
2. Construction of knowledge-bases from databases: Knowledge discovery in databases.
3. Application of knowledge to databases: Deductive and object-oriented databases.
4. Integration of knowledge construction and knowledge application processes.
5. From database systems to data-intensive knowledge-base systems: An evolutionary approach.
Intelligent Database Systems: Motivation

- Why intelligent database systems?
  - Necessity: Intelligent organization, retrieval and understanding of the huge amounts of data in DBs.
  - Possibility: Mature relational technologies plus recent progress on ◇ Knowledge Discovery in Databases, and ◇ Deductive and Object-Oriented Databases.

- Key components of intelligent database systems:
  - A knowledge-rich data model: (DOOD?).
  - High-level, declarative, graphical user interfaces.
  - Support of tools for ◇ data browsing, ◇ knowledge mining, ◇ intelligent query answering, ◇ schema organization, etc.
  - Efficient implementation & high performance.
KNOWLEDGE DISCOVERY IN DATABASES
Outline: Knowledge Discovery in Databases (KDD)

- Data Mining (KDD): Demands and Potentials.
- Knowledge Discovery in Databases: General Philosophy.
- Knowledge Discovery in Relational Databases.
  - Primitive-level vs. generalization-based discovery.
  - Discovery of different kinds of knowledge: characteristic, discriminative, association, clustering (classification), deviation and evolution rules.
  - Attribute-oriented induction and its implementations.
- Knowledge Discovery in Advanced Database Systems.
- Application of Discovered Knowledge.
- Typical KDD Systems/Prototypes.
- Future Research Issues.
Data Mining: Demands and Potentials

- What is data mining (or knowledge discovery in databases, knowledge extraction, data archaeology, data dredging, information harvesting, pattern processing)?
  — Extraction of interesting knowledge (rules, constraints, regularities) from data in large databases.

- Computerization and automated data gathering lead to tremendous amounts of data stored in databases.

- From data to “information”, “knowledge”, “regularity”, “overview”, or “browsing”.

- Automatic knowledge-base construction: from databases to knowledge-bases.
Industry Computerization: An Evolution Path

• Wave I: (60s–70s) Data collection and database creation:
  — Creation of massive on-line databases.

• Wave II: (70s–80s) Data organization and usage:
  — Data organization, data retrieval and transaction processing in databases (relational database systems).

• Wave III: (80s–90s) Data distribution, diversification, sharing, and understanding:
  — Extended-relational, object-oriented, deductive and heterogeneous DBMS.
  — Application-oriented database systems (spatial, temporal, multimedia, active, scientific databases, knowledge-bases, office information-bases, global information-base, etc.).
  — Knowledge discovery in databases (data mining).
Knowledge Discovery in Databases: General Philosophy

- Quality of data in databases.
  - **pessimistic**: highly dynamic, incomplete, redundant, noisy and sparse.
  - **optimistic**: relatively stable, complete, nonredundant, rich and reliable.
  - Start with high quality data and evolve towards complex cases.

- Discovery with generalization vs. without it.
  - **without generalization** — FD, MVD, ICs and deduction rules.
  - **with generalization** — rules, regularities and statistics expressed at a general level.
  - Focus on generalization-based discovery.
• Autonomous vs. command-driven discovery.
  – autonomous: huge number, uninteresting rules can be discovered.
  – command-driven: specify what you want to find.
  ◊ An SQL-like or graphical knowledge discovery query interface is recommended.

• With background knowledge vs. without it.
  – without background knowledge: treat background knowledge as part of the discovery task.
  – with background knowledge: user preference, guidance & efficiency.
  ◊ Assume the availability of background knowledge, but refine or automatically generate such knowledge when necessary.
Knowledge Discovery in Relational DBs: Why and How?

• Why study knowledge discovery in relational databases?
  – Highly structured, most popularly used databases.
  – Retrieval and discovery on any portion of data.
  – Integration with set-oriented database operations.
  – Discovery of different kinds of rules.
  – Foundation for mining in other kinds of databases.

• What kind of KDD algorithms should we look for?
  – Disk-based, set-oriented, efficient, interactive.
  – Generating a reasonable number of rules, with structured representation & associated statistical information.
  – Integration of database, machine learning and statistical tools.
Knowledge Discovery in Relational DBs: An Outline

- Primitive-level knowledge discovery.
- Generalization-based knowledge discovery.
- Primitives for KDD tasks.
- Principles of attribute-oriented induction.
- Discovery of characteristic rules.
- Discovery of discriminant rules.
- Concept hierarchies: adjustment and generation.
- System implementation and experimental results.
- Discovery of class description rules.
- Discovery of multi-level association rules.
- Discovery of data evolution regularities.
- Discovery of data deviation rules.
Primitive-Level Knowledge Discovery

- Primitive-level KDD: Knowledge discovery without generalization of primitive-level concepts in databases into high level concepts.
- Discovery of FD (functional dependency) rules, MVD (multi-valued dependency) rules, association rules, etc.
  - Finding precise vs. approximate (strong) rules.
- Some typical work in this direction.
  - Discovery, analysis and presentation of strong rules (Piatetsky-Shapiro 1991).
A Primitive-Level KDD Method: The Quest Approach

- **Major measurements:** (rules in the form of $F(o) \rightarrow G(o)$).
  - **Confidence factor:** (rule strength) the fraction of objects in $O$ that satisfy $F$ also satisfy $G$.
  - **Syntactic constraints:** restrictions on predicates and methods that can appear in the rule.
  - **Support constraints:** (statistical significance of the pattern) fraction of objects in $O$ that satisfy $F$ and $G$.

- **Rules to be discovered:**
  - **Classification:** characterize each group in the training set.
  - **Association:** sets of objects appear together.
  - **Sequence:** a special kind of association rules, related to temporal component(s).
• Basic operations:
  String: an ordered sequence of \langle method, value \rangle pairs.
  – newstrings $\leftarrow$ Generate(seed, DB).
  – Measure(newstrings).
  – combstrings $\leftarrow$ Combine(newstrings).
  – seed $\leftarrow$ Filter(combstrings).
  – target + $\leftarrow$ Select(seed).

• Combining operations:
  – Classification: A decision-tree method (based on entropy and information gain, similar to ID-3).
  – Associations: based on minimum support and confidence factor.
  – Sequences: analogous to discovery of associations.

• Performance considerations:
  – Waste-ratio, balancing I/O and CPU costs.
Generalization-Based Knowledge Discovery

- Generalization-based KDD: Discovery of knowledge at generalized (high) concept levels.
- Motivation: Data summarization, browsing, and more regularities can be discovered at general levels.
- Discovery of different kinds of generalized rules.
  - Characteristic, discriminant, classification, and association rules.
  - Qualitative vs. quantitative rules.
  - Precise vs. approximate (strong) rules.
  - Disjunctive vs. conjunctive rules vs. generalized relations.
  - Stable, evolution & deviation rules.
Primitives for KDD Tasks

1. Task-relevant data:
   ◦ focusing on interested data set,
   ◦ collected by a relational query.

2. Expected results: kind of rules to be discovered.

3. Background knowledge:
   ◦ conceptual hierarchies (or lattices) and
   ◦ generalization operators.

\{ \text{freshman, \ldots, senior} \} \subset \text{undergraduate} \\
\text{city} \subset \text{province} \subset \text{country} (\text{student.birthplace})

- Provided by experts/users.
- Stored in DB implicitly, e.g., address.
- By aggr./approx./math. functions, e.g., avg, sum, etc.
- Generated automatically or semi-automatically.
- Adjusted automatically based on DB statistics.
DBMiner/DBLearn: Architecture

Graphical User Interface

SQL Server

Discovery Module

Data

Concept Hierarchy
DBMiner: Discovery Module

- Characterizer (Based on DBLearn)
- Discriminator (Based on DBLearn)
- Classifier (Class Description)
- Association Rule Finder
- Evolution Evaluator
- Deviation Evaluator
- Predictor (Potential Evaluator)
- Sequential Pattern Miner
- Future Module
Knowledge Discovery Interface: DBMiner Examples

1. Discovery of a characteristic rule.
   Query: Find a characteristic rule for Computer Science research grants from the database ‘NSERC94’ in relevance to discipline and amount categories and the distribution of count% and amount%.

   use NSERC94
   discover characteristic rule for "CS_Discipline_Grants"
   from award A, grant_type G
   where A.grant_code = G.grant_code and A.disc_code = "Computer"
   in relevance to disc_code, amount, percentage(count), percentage(amount)
2. Discovery of a discriminant rule.

Query: Find a discriminant rule from the database ‘NSERC94’ to distinguish the province ‘Ontario’ from ‘Newfoundland’ for Computer Science research grants in relevance to discipline and amount categories and the distribution of count% and amount%.

use NSERC94
discover discriminant rule for ‘Ont_Grants’
where 0.province = ‘Ontario’
in contrast to ‘Newfoundland_Grants’
where 0.province = ‘Newfoundland’
from award A, organization O, grant_type G
where A.grant_code = G.grant_code and A.org_code = 0.org_code and A.disc_code = ‘Computer’
in relevance to disc_code, grant_order, amount
3. Discovery of class description rules.
Query: Classify operating research grants from the database ‘NSERC94’ in relevance to discipline, organization and amount categories and find description rules for each class.

use NSERC94
classify and discover class description rule for
‘‘Operating_Grants’’
from award A, organization O, grant_type G
where A.grant_code = G.grant_code and A.org_code = O.org_code and A.grant_type = ‘‘Operating’’
in relevance to disc_code, organization, amount
A User-Friendly GUI for Interactive Knowledge Mining

- **Load:** (Load a KDD query from a file and edit it when necessary).
- **Save:** (Save a modified KDD query).
- **Run:** (Execute the knowledge discovery query).
- **Discovery type:** (Specify the kind of rules to be discovered).
- **Query formation:** (Form a KDD query interactively).
- **Threshold:** (Set/change a threshold for each attribute).
- **Hierarchy:** (Specify, modify, dynamically adjust, or automatically generate concept hierarchies).
- **Rule generation:** (Rule generation from the prime relation).
- **Output transformation:** (Transform the output relation with various kinds of options).
- **Graph display:** Display graphically different kinds of outputs.
Basic Attribute-Oriented Induction Strategies

1. **Data focusing**: Focusing at the set of relevant data.
2. **Generalization vs. specialization**:
   - Generalization only (usually, no negative instances in DBs).
   - Avoid over generalization by least commitment —
     generalization on the smallest decomposable components.
3. **Attribute removal**: Remove
   - nongeneralizable attrs with many distinct values (e.g., keys).
   - attributes representing lower level concepts (e.g., address).
4. **Concept tree ascension/generalization**:
   - climbing concept trees.
   - applying generalization operators.
5. **Count & aggr. value propagation & accumulation.**
6. **Attribute generalization control**:
   - Attribute threshold vs. desirable level.
Basic Attribute-Oriented Induction Algorithm

Input. (i) A DBQuery related to the DB and the learning task, (ii) Gen(a_i): concept hierarchies/gen. operators on attrs. a_i, (iii) T_i: thresholds for attributes a_i.

Output. A characteristic rule based on the learning request.

Method. The A-O induction is performed in the following four steps.

1. InitRel: Execute DBQuery to collect the task-relevant data.
2. PreGen: Prepare for generalization.
   2.1. Collect the distinct values for each attribute a_i and the number of occurrences of each distinct value. (For numeric values, automatic hierarchy generation can be performed by examining their value range.)
   2.2. Compute the desired level L_i for each attribute a_i based on its attribute threshold T_i (dynamic hierarchy
adjustment can be performed if desired).

2.3. Determining the mapping-pairs \( (v, v') \) for each attribute \( a_i \), where \( v \) is a distinct value of \( a_i \), and \( v' \) is its corresponding generalized value at level \( L_i \).

3. **PrimeGen**: Derive the *prime generalized relation*, \( \mathcal{R}_p \).

3.1. Generalize each tuple \( t \) in the initial relation into \( t' \) by replacing each \( v \) in \( t \) by \( v' \) based on \( (v, v') \) at PreGen.

3.2. Insert \( t' \) into the prime relation with \textit{count} accumulated.

4. **RuleGen**: Present \( \mathcal{R}_p \) or generalized rules (relations, tables) with the following alternative techniques.

4.1. Map \( \mathcal{R}_p \) into a set of *generalized feature table* for a (sub)set of attributes;

4.2. Further generalize \( \mathcal{R}_p \) to a relatively small set of tuples.

4.3. Using weight filters, rough sets, information-theoretic measurement (as ID3), or other measurements to reduce the number of attributes & simplify the rules. \( \square \)
**Discovery of a Characteristic Rule: An Example**

```
discover characteristic rule for GradStudents
from Student S
where S.status = "graduate"
in relevance to Name, Major, BirthPlace, GPA

DB
Discovery ↓ Request
(relevant data set in the university database)

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Major</th>
<th>Birth_Place</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anderson</td>
<td>M.A.</td>
<td>history</td>
<td>Vancouver</td>
<td>3.5</td>
</tr>
<tr>
<td>Fraser</td>
<td>M.Sc.</td>
<td>physics</td>
<td>Ottawa</td>
<td>3.9</td>
</tr>
<tr>
<td>Gupta</td>
<td>Ph.D.</td>
<td>math</td>
<td>Bombay</td>
<td>3.3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Liu</td>
<td>Ph.D.</td>
<td>biology</td>
<td>Shanghai</td>
<td>3.4</td>
</tr>
<tr>
<td>Monk</td>
<td>Ph.D.</td>
<td>computing</td>
<td>Victoria</td>
<td>3.8</td>
</tr>
<tr>
<td>Wang</td>
<td>M.Sc.</td>
<td>statistics</td>
<td>Nanjing</td>
<td>3.2</td>
</tr>
</tbody>
</table>

↓
(Attribute-Oriented Induction)

↓
(Prime Relation)
```
Further Generalization of Prime Relation (Example)

(prime relation)

<table>
<thead>
<tr>
<th>Major</th>
<th>Birth Place</th>
<th>GPA</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>art</td>
<td>B.C.</td>
<td>excellent</td>
<td>35</td>
</tr>
<tr>
<td>science</td>
<td>Ontario</td>
<td>excellent</td>
<td>10</td>
</tr>
<tr>
<td>science</td>
<td>B.C.</td>
<td>excellent</td>
<td>30</td>
</tr>
<tr>
<td>science</td>
<td>India</td>
<td>good</td>
<td>10</td>
</tr>
<tr>
<td>science</td>
<td>China</td>
<td>good</td>
<td>15</td>
</tr>
</tbody>
</table>

\[ \downarrow \]

(Further generalization)

<table>
<thead>
<tr>
<th>Major</th>
<th>Birth Place</th>
<th>GPA</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>any</td>
<td>Canada</td>
<td>excellent</td>
<td>75</td>
</tr>
<tr>
<td>science</td>
<td>Foreign</td>
<td>good</td>
<td>25</td>
</tr>
</tbody>
</table>

Rule extracted from the last table:

\[ \forall(x)\ \text{graduate}(x) \rightarrow (\text{Birth Place}(x) \in \text{Canada} \land \text{GPA}(x) \in \text{excellent}) \ [75\%] \lor \]
\[ (\text{Major}(x) \in \text{science} \land \text{Birth Place}(x) \in \text{foreign} \land \text{GPA}(x) \in \text{good}) \ [25\%] \]
From Prime Relation to Feature Table (Example)

(Prime relation)

<table>
<thead>
<tr>
<th>Major</th>
<th>Birth Place</th>
<th>GPA</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>art</td>
<td>B.C.</td>
<td>excellent</td>
<td>35</td>
</tr>
<tr>
<td>science</td>
<td>Ontario</td>
<td>excellent</td>
<td>10</td>
</tr>
<tr>
<td>science</td>
<td>B.C.</td>
<td>excellent</td>
<td>30</td>
</tr>
<tr>
<td>science</td>
<td>India</td>
<td>good</td>
<td>10</td>
</tr>
<tr>
<td>science</td>
<td>China</td>
<td>good</td>
<td>15</td>
</tr>
</tbody>
</table>

(Feature table)

<table>
<thead>
<tr>
<th>Major</th>
<th>Birth Place</th>
<th>GPA</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B.C.</td>
<td>Ont.</td>
<td>India</td>
</tr>
<tr>
<td>art</td>
<td>35</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>science</td>
<td>30</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>65</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

(Rules extracted from the feature table)

\(\forall (x) \text{ Major}(x) \in \text{science} \rightarrow \text{GPA}(x) \in \text{excellent} [62\%] \lor \text{GPA}(x) \in \text{good} [38\%]\)

\(\forall (x) \text{ Birth Place}(x) \in \text{B.C.} \rightarrow \text{Major}(x) \in \text{art} [54\%] \lor \text{Major}(x) \in \text{science} [46\%]\)
## Discovery of a Characteristic Rule: An Experiment

<table>
<thead>
<tr>
<th>disc_code</th>
<th>amount</th>
<th>count%</th>
<th>amount%</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>0-20Ks</td>
<td>11.64%</td>
<td>7.00%</td>
</tr>
<tr>
<td>AI</td>
<td>20Ks-40Ks</td>
<td>6.34%</td>
<td>6.99%</td>
</tr>
<tr>
<td>AI</td>
<td>40Ks-60Ks</td>
<td>1.37%</td>
<td>2.95%</td>
</tr>
<tr>
<td>AI</td>
<td>60Ks-</td>
<td>0.68%</td>
<td>2.41%</td>
</tr>
<tr>
<td>COMP_METHODS</td>
<td>0-20Ks</td>
<td>7.36%</td>
<td>4.31%</td>
</tr>
<tr>
<td>COMP_METHODS</td>
<td>20Ks-40Ks</td>
<td>5.48%</td>
<td>6.07%</td>
</tr>
<tr>
<td>COMP_METHODS</td>
<td>40Ks-60Ks</td>
<td>1.03%</td>
<td>2.11%</td>
</tr>
<tr>
<td>COMP_METHODS</td>
<td>60Ks-</td>
<td>0.86%</td>
<td>2.70%</td>
</tr>
<tr>
<td>DATABASES</td>
<td>0-20Ks</td>
<td>5.99%</td>
<td>3.08%</td>
</tr>
<tr>
<td>DATABASES</td>
<td>20Ks-40Ks</td>
<td>2.23%</td>
<td>2.61%</td>
</tr>
<tr>
<td>DATABASES</td>
<td>60Ks-</td>
<td>0.17%</td>
<td>0.48%</td>
</tr>
<tr>
<td>HARDWARE</td>
<td>0-20Ks</td>
<td>2.23%</td>
<td>1.39%</td>
</tr>
<tr>
<td>HARDWARE</td>
<td>20Ks-40Ks</td>
<td>1.20%</td>
<td>1.37%</td>
</tr>
<tr>
<td>HARDWARE</td>
<td>40Ks-60Ks</td>
<td>0.17%</td>
<td>0.31%</td>
</tr>
<tr>
<td>HARDWARE</td>
<td>60Ks-</td>
<td>0.66%</td>
<td>3.94%</td>
</tr>
<tr>
<td>SOFTWARE</td>
<td>0-20Ks</td>
<td>11.30%</td>
<td>6.73%</td>
</tr>
<tr>
<td>SOFTWARE</td>
<td>20Ks-40Ks</td>
<td>6.68%</td>
<td>7.66%</td>
</tr>
<tr>
<td>SOFTWARE</td>
<td>40Ks-60Ks</td>
<td>1.03%</td>
<td>2.21%</td>
</tr>
<tr>
<td>SOFTWARE</td>
<td>60Ks-</td>
<td>0.68%</td>
<td>2.13%</td>
</tr>
<tr>
<td>SYS_ORGANIZATION</td>
<td>0-20Ks</td>
<td>4.28%</td>
<td>2.83%</td>
</tr>
<tr>
<td>SYS_ORGANIZATION</td>
<td>20Ks-40Ks</td>
<td>2.74%</td>
<td>3.02%</td>
</tr>
<tr>
<td>SYS_ORGANIZATION</td>
<td>40Ks-60Ks</td>
<td>0.68%</td>
<td>1.39%</td>
</tr>
<tr>
<td>SYS_ORGANIZATION</td>
<td>60Ks-</td>
<td>0.34%</td>
<td>1.34%</td>
</tr>
<tr>
<td>THEORY</td>
<td>0-20Ks</td>
<td>11.82%</td>
<td>6.73%</td>
</tr>
<tr>
<td>THEORY</td>
<td>20Ks-40Ks</td>
<td>9.93%</td>
<td>11.46%</td>
</tr>
<tr>
<td>THEORY</td>
<td>40Ks-60Ks</td>
<td>2.40%</td>
<td>4.86%</td>
</tr>
<tr>
<td>THEORY</td>
<td>60Ks-</td>
<td>0.68%</td>
<td>1.93%</td>
</tr>
</tbody>
</table>

### DBMiner Query:

use NSERC94
discover characteristic rule
for "CS_Discipline-Grants"
from award A, grant_type G
where A.grant_code = G.grant_code
   and A.disc_code = "Computer"
in relevance to disc_code, amount,
   percentage(count), percentage(amount)

+++ 13476738 100.00% 100.00%
## Extraction of Feature Tables from Prime Relation

* disc_code feature table by count %

<table>
<thead>
<tr>
<th>disc_code</th>
<th>amount</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0-20Ks</td>
<td>20Ks-40Ks</td>
</tr>
<tr>
<td>AI</td>
<td>11.64%</td>
<td>6.34%</td>
</tr>
<tr>
<td>COMP_METHODS</td>
<td>7.36%</td>
<td>5.48%</td>
</tr>
<tr>
<td>DATABASES</td>
<td>5.99%</td>
<td>2.23%</td>
</tr>
<tr>
<td>HARDWARE</td>
<td>2.23%</td>
<td>1.20%</td>
</tr>
<tr>
<td>SOFTWARE</td>
<td>11.30%</td>
<td>6.68%</td>
</tr>
<tr>
<td>SYS_ORGANIZA</td>
<td>4.28%</td>
<td>2.74%</td>
</tr>
<tr>
<td>THEDRY</td>
<td>11.82%</td>
<td>9.93%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>54.62%</td>
<td>34.59%</td>
</tr>
</tbody>
</table>

* disc_code feature table by amount %

<table>
<thead>
<tr>
<th>disc_code</th>
<th>amount</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0-20Ks</td>
<td>20Ks-40Ks</td>
</tr>
<tr>
<td>AI</td>
<td>7.00%</td>
<td>6.99%</td>
</tr>
<tr>
<td>COMP_METHODS</td>
<td>4.31%</td>
<td>6.07%</td>
</tr>
<tr>
<td>DATABASES</td>
<td>3.08%</td>
<td>2.61%</td>
</tr>
<tr>
<td>HARDWARE</td>
<td>1.39%</td>
<td>1.37%</td>
</tr>
<tr>
<td>SOFTWARE</td>
<td>6.73%</td>
<td>7.65%</td>
</tr>
<tr>
<td>SYS_ORGANIZA</td>
<td>2.83%</td>
<td>3.02%</td>
</tr>
<tr>
<td>THEDRY</td>
<td>6.73%</td>
<td>11.46%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>32.06%</td>
<td>39.18%</td>
</tr>
</tbody>
</table>
### Another Characteristic Rule: Grant Distribution

<table>
<thead>
<tr>
<th>amount</th>
<th>org_name</th>
<th>count%</th>
<th>amount%</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANY</td>
<td>Toronto</td>
<td>7.92%</td>
<td>12.60%</td>
</tr>
<tr>
<td>ANY</td>
<td>Waterloo</td>
<td>8.87%</td>
<td>10.45%</td>
</tr>
<tr>
<td>ANY</td>
<td>British Columbia</td>
<td>5.85%</td>
<td>7.15%</td>
</tr>
<tr>
<td>ANY</td>
<td>Simon Fraser</td>
<td>4.34%</td>
<td>4.97%</td>
</tr>
<tr>
<td>ANY</td>
<td>Concordia</td>
<td>4.91%</td>
<td>4.81%</td>
</tr>
<tr>
<td>ANY</td>
<td>Alberta</td>
<td>4.15%</td>
<td>4.26%</td>
</tr>
<tr>
<td>ANY</td>
<td>Calgary</td>
<td>3.77%</td>
<td>4.21%</td>
</tr>
<tr>
<td>ANY</td>
<td>McGill</td>
<td>3.02%</td>
<td>4.12%</td>
</tr>
<tr>
<td>ANY</td>
<td>Victoria</td>
<td>3.96%</td>
<td>3.91%</td>
</tr>
<tr>
<td>ANY</td>
<td>Queen’s</td>
<td>4.34%</td>
<td>3.90%</td>
</tr>
<tr>
<td>ANY</td>
<td>Carleton</td>
<td>3.40%</td>
<td>3.54%</td>
</tr>
<tr>
<td>ANY</td>
<td>Western Ontario</td>
<td>3.77%</td>
<td>3.25%</td>
</tr>
<tr>
<td>ANY</td>
<td>Ottawa</td>
<td>3.40%</td>
<td>2.87%</td>
</tr>
<tr>
<td>ANY</td>
<td>York</td>
<td>2.45%</td>
<td>2.41%</td>
</tr>
<tr>
<td>ANY</td>
<td>Saskatchewan</td>
<td>2.45%</td>
<td>2.36%</td>
</tr>
<tr>
<td>ANY</td>
<td>McMaster</td>
<td>2.26%</td>
<td>2.18%</td>
</tr>
<tr>
<td>ANY</td>
<td>Manitoba</td>
<td>2.64%</td>
<td>2.15%</td>
</tr>
<tr>
<td>ANY</td>
<td>Regina</td>
<td>2.26%</td>
<td>1.76%</td>
</tr>
<tr>
<td>ANY</td>
<td>New Brunswick</td>
<td>1.89%</td>
<td>1.24%</td>
</tr>
<tr>
<td>ANY</td>
<td>Guelph</td>
<td>1.51%</td>
<td>1.21%</td>
</tr>
<tr>
<td>ANY</td>
<td>Memorial Univ. of Nf</td>
<td>1.70%</td>
<td>1.18%</td>
</tr>
<tr>
<td>ANY</td>
<td>Dalhousie</td>
<td>1.32%</td>
<td>0.90%</td>
</tr>
<tr>
<td>ANY</td>
<td>Windsor</td>
<td>1.13%</td>
<td>0.78%</td>
</tr>
</tbody>
</table>

**DBMiner Query:**

```sql
use NSERC94
discover characteristic rule
for "CS_Organization_Grants"
from award A, organization O, grant_type G
where A.grant_code = G.grant_code and
        O.org_code = A.org_code
        and A.disc_code = "Computer"
        and G.grant_order = "Operating Grant"
in relevance to amount, org_name,
    percentage(count), percentage(amount)
set attribute threshold 1 for amount
unset attribute threshold for org_name
```

J. W. Han: ICDE’95 Conference Tutorial
Dynamic Adjustment of Concept Hierarchies

Original Concept Hierarchy

Adjusted Concept Hierarchy
Automatic Generation of Numeric Hierarchies

Count

0 10 20 30 40

10000 20000 30000 40000 50000 60000 70000 80000 90000 100000

Amount

2000~97000

2000~16000

16000~97000

2000~12000

12000~16000

16000~23000

23000~97000
A Comparison with Tuple-Oriented Induction

The Entire Version Space

The Factored Version Space
Attribute-Oriented Induction: Major Strength

- Low computational complexity: $\sim O(n \log p)$.
  $n = \#$ of tuples in the initial relation (i.e., relevant data).
  $p = \#$ of tuples in the output relation.
- Integration of KDD with database operations.
- Data generalization: high level summary and browsing.
- Using counts and statistics: Discovery of disjunctive rules, approximate rules, exceptional cases, and incremental learning.
- Discovery of different kinds of rules: characteristic, discriminant, classification, association, evolution, etc.
- Integration with other learning methods: Knowledge extraction on the prime generalized relation, using ID-3, rough sets, information theory, etc.
Discovery of Discriminant Rules

- Collect the relevant data respectively into the target class and the contrasting class.
- Extract the prime target relation by AO induction, and then generalize the concepts in the contrasting class to the same level as those in the prime target relation, forming the prime contrasting relation.
- To generate qualitative discriminant rules,
  ◦ compare the tuples in two prime relations, mark those overlap in both relations, and
  ◦ output only the unmarked tuples.
- To generate quantitative discriminant rules,
  ◦ compute the discriminating weight for each property,
  ◦ output those whose d-weight is close to 1 together with the d-weight.
## Discovery of Discriminant Rules: An Example

A KDD Query for discovery of discriminant rules.

```sql
use UniversityDB
discover discriminant rule for GradStudent
where status = "graduate"
in contrast to UnderGradStudent
where status = "undergrad"
from Student
```

### Database

<table>
<thead>
<tr>
<th>Major</th>
<th>Birth Place</th>
<th>GPA</th>
<th>Count</th>
<th>mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>art</td>
<td>Canada</td>
<td>excellent</td>
<td>35</td>
<td>*</td>
</tr>
<tr>
<td>science</td>
<td>Canada</td>
<td>excellent</td>
<td>40</td>
<td>*</td>
</tr>
<tr>
<td>science</td>
<td>Foreign</td>
<td>good</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>science</td>
<td>Canada</td>
<td>excellent</td>
<td>50</td>
<td>*</td>
</tr>
<tr>
<td>arts</td>
<td>Canada</td>
<td>average</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>science</td>
<td>Canada</td>
<td>average</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>art</td>
<td>Canada</td>
<td>excellent</td>
<td>20</td>
<td>*</td>
</tr>
</tbody>
</table>

### Qualitative rule extraction

\( \forall(x) \) graduate\( (x) \) \leftarrow \begin{align*}
& \text{Major}(x) \in \text{science} \land \\
& \text{Birth\_Place}(x) \in \text{foreign} \land \\
& \text{GPA}(x) \in \text{good}.
\end{align*} \)

### Quantitative Rule extraction

\( \forall(x) \) graduate\( (x) \) \leftarrow \begin{align*}
& ((\text{Major}(x) \in \text{science} \land \text{Birth\_Place}(x) \in \text{foreign} \\
& \land \text{GPA}(x) \in \text{good}) [100\%]) \lor \\
& ((\text{Major}(x) \in \text{art} \land \text{Birth\_Place}(x) \in \text{Canada} \\
& \land \text{GPA}(x) \in \text{excellent}) [63.64\%]) \lor \\
& ((\text{Major}(x) \in \text{science} \land \text{Birth\_Place}(x) \in \text{Canada} \\
& \land \text{GPA}(x) \in \text{excellent}) [44.44\%]))
\end{align*} \)
### Discovery of a Discriminant Rule: An Experiment

| * | The target class |
|--------------------------------------------------|
| disc_code | grant_order | amount | count | mark |
| Computer | Strategic_Grants | 40Ks-60Ks | 1 | |
| Computer | Operating_Grants | 60Ks- | 6 | |
| Computer | Strategic_Grants | 60Ks- | 8 | |
| Computer | Other | 40Ks-60Ks | 5 | |
| Computer | Other | 60Ks- | 7 | |
| Computer | Other | 20Ks-40Ks | 10 | * |
| Computer | Operating_Grants | 40Ks-60Ks | 25 | |
| Computer | Other | 0-20Ks | 10 | * |
| Computer | Operating_Grants | 20Ks-40Ks | 62 | |
| Computer | Operating_Grants | 0-20Ks | 119 | * |

**DBMiner Query:**

Distinguish Computer Science NSERC research grants of the province “Ontario” from that of “Newfoundland” according to discipline, amount, and grant categories

```
use NSERC94
discover discriminant rule for "Ont_Grants"
where O.province = "Ontario"
in contrast to "Newfoundland_Grants"
where O.province = "Newfoundland"
from award A, organization O, grant_type G
where A.grant_code = G.grant_code and
   O.org_code = A.org_code
   and A.disc_code = "Computer"
   and G.grant_order = "Operating Grant"
in relevance to disc_code, grant_order, amount
```

| * | The contrasting class |
|--------------------------------------------------|
| disc_code | grant_order | amount | count | mark |
| Computer | Operating_Grants | 0-20Ks | 9 | * |
| Computer | Other | 20Ks-40Ks | 1 | * |
| Computer | Other | 0-20Ks | 1 | * |
* The final generalized relation *

<table>
<thead>
<tr>
<th>disc_code</th>
<th>grant_order</th>
<th>amount</th>
<th>count</th>
<th>mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer</td>
<td>Strategic_Grants</td>
<td>40Ks-60Ks</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Computer</td>
<td>Operating_Grants</td>
<td>60Ks-</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Computer</td>
<td>Strategic_Grants</td>
<td>60Ks-</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Computer</td>
<td>Other</td>
<td>40Ks-60Ks</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Computer</td>
<td>Other</td>
<td>60Ks-</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Computer</td>
<td>Operating_Grants</td>
<td>40Ks-60Ks</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Computer</td>
<td>Operating_Grants</td>
<td>20Ks-40Ks</td>
<td>62</td>
<td></td>
</tr>
</tbody>
</table>

* Discriminant rule for "Ont_Grants" vs. "Newfoundland_Grants":*

For all x, Ont_Grants(x) \(\leftarrow\) 

\[
((\text{disc_code} = \text{Computer}) \land (\text{grant_order} = \text{Operating_Grants}) \\
\quad \land (\text{amount} = 20Ks-40Ks)) \\
\lor ((\text{disc_code} = \text{Computer}) \land (\text{grant_order} = \text{Operating_Grants}) \\
\quad \land (\text{amount} = 40Ks-60Ks)) \\
\lor ((\text{disc_code} = \text{Computer}) \land (\text{grant_order} = \text{Other}) \\
\quad \land (\text{amount} = 60Ks-)) \\
\lor ((\text{disc_code} = \text{Computer}) \land (\text{grant_order} = \text{Other}) \\
\quad \land (\text{amount} = 40Ks-60Ks)) \\
\lor ((\text{disc_code} = \text{Computer}) \land (\text{grant_order} = \text{Strategic_Grants}) \\
\quad \land (\text{amount} = 60Ks-)) \\
\lor ((\text{disc_code} = \text{Computer}) \land (\text{grant_order} = \text{Operating_Grants}) \\
\quad \land (\text{amount} = 60Ks-)) \\
\lor ((\text{disc_code} = \text{Computer}) \land (\text{grant_order} = \text{Strategic_Grants}) \\
\quad \land (\text{amount} = 40Ks-60Ks))
\]
Discovery of Multi-Level Association Rules

- Why discover multi-level (ML) association rules?
  - Stronger associations are more likely to exist at high concept levels, e.g., milk and bread.
  - Cluster-based, interactive, and progressive mining.
- Mining ML-association rules: A top-down approach.
- Occurrence ratio \( \rho(A/S) \) for pattern \( A \) occurring in set \( S \).
  - Association “\( A \rightarrow B \)”: \( \rho(A \rightarrow B/S) = \rho(A \land B/S)/\rho(A/S) \).
- Algorithm. Find multi-level association rules, satisfying
  1. Generalize the task-relevant set \( S \) by A-O induction.
  2. Find \( A \) and \( B \) in \( S \), satisfying the conditions.
  3. Walking down the concept levels, find \( A' \in A, \ B' \in B \), satisfying the conditions.
  4. Output the nonsubsumed rules, together with ratios.
Discovery of ML Association Rules: An Example

• Find association rules between milk and bread in a shopping transaction DB.
  
  discover association rules
  between ‘‘milk’’ and ‘‘bread’’
  from ShoppingTransactionDB
  where category = ‘‘food’’

• Let the thresholds for support and confidence be 5% and 50%. One may get results as follows.

  ‘milk’ ∧ ‘bread’ [10%]    ‘milk’ → ‘bread’ [60%]
  ‘bread’ → ‘milk’ [80%].    ‘2% milk’ ∧ ‘wheat bread’ [20%]
  ‘2% milk’ → ‘wheat bread’ [70%]
  ‘GoldenGate wheat bread’ → ‘Dairyland 2% milk’ [50%]

• The multi-level association rules can be associated with a concept tree for clear presentation.
Discovery of Class Description Rules

- Why discover class description rules?
  Concept-based data clustering & class associated rules.
- Algorithm. Based on Cluster-2 but for large DBs.
  1. Generalize the task-relevant set $S$ by A-O induction.
  2. Map $S$ into a multi-dimensional table $T$, filtering out the entries with low occurrence frequency.
  3. Merge slots and generate candidate schemes.
  4. Select the one with the lowest combined sparseness and complexity from the candidate schemes, with rules associated.
  5. Iterate the above for each class to generate a hierarchy.
- A similar algorithm can be worked out based on ID-3.
Discovery of Class Description Rules: An Example

• A store DB: A-O induction leads to an even space table.

<table>
<thead>
<tr>
<th></th>
<th>USA</th>
<th>Japan</th>
<th>S. America</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>M</td>
<td>E</td>
</tr>
<tr>
<td><strong>elec equip</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dark</td>
<td>9</td>
<td>264</td>
<td>3</td>
</tr>
<tr>
<td>light</td>
<td>290</td>
<td>4</td>
<td>367</td>
</tr>
<tr>
<td><strong>clothing</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dark</td>
<td>11</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>light</td>
<td>7</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td><strong>furniture</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dark</td>
<td>153</td>
<td>242</td>
<td>13</td>
</tr>
<tr>
<td>light</td>
<td>12</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

• Frequency ratio in the event space table.

<table>
<thead>
<tr>
<th></th>
<th>USA</th>
<th>Japan</th>
<th>S. America</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>M</td>
<td>E</td>
</tr>
<tr>
<td><strong>elec equip</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dark</td>
<td>0.3%</td>
<td>8.8%</td>
<td>0.1%</td>
</tr>
<tr>
<td>light</td>
<td>9.67%</td>
<td>0.13%</td>
<td>12.23%</td>
</tr>
<tr>
<td><strong>clothing</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dark</td>
<td>0.37%</td>
<td>0.23%</td>
<td>0.17%</td>
</tr>
<tr>
<td>light</td>
<td>0.23%</td>
<td>0.07%</td>
<td>0.1%</td>
</tr>
<tr>
<td><strong>furniture</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dark</td>
<td>5.1%</td>
<td>8.07%</td>
<td>0.43%</td>
</tr>
<tr>
<td>light</td>
<td>0.4%</td>
<td>0.03%</td>
<td>0.2%</td>
</tr>
</tbody>
</table>
• Filter “noise” (< 1%) ⇒ a discrete event space table.

<table>
<thead>
<tr>
<th></th>
<th>USA</th>
<th></th>
<th>Japan</th>
<th></th>
<th>S. America</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>M</td>
<td>E</td>
<td>C</td>
<td>M</td>
</tr>
<tr>
<td>elec_equip dark</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>light</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>clothing  dark</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>light</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>furniture dark</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>light</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• The candidate classification schemes.
  - Scheme 1 [Classifying attribute = “Category”]:
    - ◊ [Category(x) ∈ \{elec_equip\}]
    - ◊ [Category(x) ∈ \{clothing\} ∧ Place\_made(x) ∈ \{S.America\} ∧ Price(x) ∈ \{cheap, moderate\}]
    - ◊ [Category(x) ∈ \{furniture\} ∧ Color(x) ∈ \{dark\_color\} ∧ Place\_made(x) ∈ \{USA, Japan\}]
  - Scheme 2 [Classifying attribute = “Place\_made”].
  - Scheme 3 [Classifying attribute = “Price”].

Scheme 1 is chosen as it has the lowest combined value of sparseness and rule complexity.
Discovery of Data Evolution Regularities

- Why discover general data evolution regularities?
  ◦ Low level irregular, but often high level regular.

- Characteristic rules of evolving data:
  ◦ Describe the firms whose sales increased 20% in 1994.

- Discriminant rules:
  ◦ Compare an up-turn firm with a down-turn one.

- Association rules: ◦ Find those going up together.

- Trends (general patterns) of data evolution:
  ◦ How the stocks fluctuate in the past 12 months?

Method:

1. Find and group evolving data based on the KDD query.
2. Perform A-O induction on the corresponding groups.
3. Use the corresponding KD techniques.
Discovery of Data Deviation Rules

- Why discover general data deviation rules?
  - Difference is norm. Only those deviating from the general characteristics substantially are deviations.
- Data deviation analysis: Measurement of data deviation from major characteristics.
- Method.
  - Find general characteristics and normal deviation by A-O induction.
  - Collect those deviating from the normal and perform A-O induction to find their general characteristics.
- Example. Find the mutual funds which performed much better (or worse) than average last year.
Knowledge Discovery in Advanced Database Systems

- Knowledge discovery in object-oriented databases.
- Knowledge discovery in spatial databases.
- Integration of knowledge discovery and deductive database techniques.
- Knowledge discovery in active databases and dynamic environments.
- Knowledge discovery in multi-(heterogeneous) databases.
- Schema integration and schema evolution by knowledge discovery techniques.
- Resource and knowledge discovery in global information systems.
Knowledge Discovery in Object-Oriented Databases

- Implementation of generalization operators for object identifiers, complex structured data, class hierarchies, class composition hierarchies, methods, spatial and multimedia data, and active data.

- A-O induction in OODBs.
  1. Data focusing (executing an OODB query).
  2. Generalization on the smallest decomposable units.
  3. Attribute removal (similar standard).
  4. Attribute generalization (DBGen operator application).
  5. Count and/or aggregate function value accumulation.
  6. Attribute generalization control.
  7. Rule formation.
Implementation of Generalization Operators in OODBs

- **Object Identifier**: Gen. each OID to the lowest subclass it belongs to, and then “climb-up” the class hierarchy.

- **Complex data objects**: structured values, tuples, trees, records, nested structures, etc.

  Hobby: \{tennis, hockey, violin\} ⇒ \{sports(2), music(1)\}.

  Education: “((B.Sc. \ldots), \ldots, (Ph.D. in Computer Science, UCLA, Aug., 1987))” ⇒ (Ph.D. in CS, UCLA, 1987).

  ◇ gen. each attr., ◇ aggregation (sum, avg, \ldots), ◇ retain important attr., ◇ structure flattening, ◇ overview, ◇ typing, ◇ multi-direction ⇒ heterogeneous type.

- **Inherited and derived data**: Treat as stored one.

- **Methods**: Derive behavioral data by method application.

- **Class composition hierarchy**: Generalize closely-related components.
Generalization on Spatial and Multi-Media Data

- Recognition and extraction of essential features or general patterns.
- Spatial data: clustering, region merging, using spatial data structures, spatial computation, approximation (ignoring scattered regions) & aggregation.
- Text & hypertext: title, author, abstract, table of contents, keywords, etc.
- Graphics & images: number, color and types of objects contained, object size, aggregation, approximation.
- Music: title, author, melody (patterns repeatedly occur), tempo, major musical instruments played, etc.
- Other audio/video information: title, author, date, content summary, etc.
Knowledge Discovery in Spatial DBs

- Two kinds of concept hierarchies: thematic vs. spatial.
- Thematic hierarchy:
  - agriculture (food (grain (corn, rice, ...), vegetable, fruit), others(...)).
- Spatial hierarchy:
  - Spatial data structure hierarchies (quad-tree and R-tree).
  - Semantics-based spatial hierarchies (geo-region classification).
  - Spatial clustering (e.g., neighborhood or adjacent_to).
- Two primitive spatial data mining techniques:
  - Nonspatial dominant generalization.
  - Spatial dominant generalization.
Nonspatial-Dominant Spatial Data Mining

- A spatial data mining query:
  ```
  discover region
  from precipitation-map
  where province = "B.C." and period = "spring"
  and year = 1990
  in relevance to precipitation and region.
  ```
- Method (nonspatial-dominant spatial data mining).
  - Nonspatial generalization on “precipitation” (e.g., averaging and concept tree ascension to “wet”).
  - Collect spatial object pointers into the generalized and merged nonspatial data entries.
  - Spatial generalization on “region” by region merging and approximation (ignoring small areas or irregular distributions).
Spatial-Dominant Spatial Data Mining

- A spatial data mining query:
  
  discover characteristic rule
  from temperature-map
  where province = "B.C." and period = "summer"
  and year = 1990
  in relevance to region and temperature.

- Method (spatial-dominant spatial data mining).
  
  ◇ Nonspatial & spatial data collection: Query processing.
  ◇ Spatial generalization on “region” according to certain spatial hierarchy.
  ◇ Nonspatial pointers are collected in the generalized and merged spatial object entries.
  ◇ Nonspatial generalization on “temperature”: weighted average and concept tree ascension.
More on Spatial Data Mining

• Interleaved generalization between spatial and nonspatial data.
  ◦ Non-spatial generalization to certain level.
  ◦ High level spatial merge/join/approximation.

• Generalization on multiple thematic maps.
  ◦ Thematic maps: “road_map”, “altitude”, etc.
  ◦ Generalization on nonspatial attributes, spatial merge & spatial overlay or in reverse order.

• Generalization using temporal spatial data.
  ◦ Generalization over a sequence of data maps collected in a sequence of time.
  ◦ Comparison or summarization of data evolution or deviation regularities.
Integration of KDD and Deductive Database Techniques

- The integration not only leads to discovery of new knowledge in deductive databases but also enhances the power of knowledge discovery mechanisms.
  - Data derived by applying deduction rules, (then knowledge discovery on the derived data).
  - Rules representing partially generalized data. (A rule is (part of) a generalized rule.).
  - Deduction rule-specified concept hierarchy.
  - Rule-directed knowledge discovery. (e.g., use a desired pattern, such as ‘$A \land B \rightarrow C$’ or ‘buy milk → buy bread’, for knowledge-directed data mining).

- Knowledge merge: Merge discovered rules into a rule-base.
Knowledge Mining in Active Databases

- In a dynamic environment, data are generated rapidly, continuously and in huge volumes —
  - Data sampling technique: Sample interesting pieces of information dynamically and systematically.
- Level gap: Data are presented at low, primitive levels. It is desired to analyze the system and express the control primitives at a relatively high level —
  - Knowledge discovery technique: Efficient and effective data generalization to discover useful knowledge or regularity at high levels.
- Systems often require prompt, real-time & intelligent reactions in response to situation changes —
  - Active database technology: automatic and prompt reaction and control of the environment.
Intelligent Reactions to Dynamic Environments

- Regularity extraction.
  - Use data sampling and KDD techniques to discover current status rules, stable rules, and evolution rules.
  - Store regular data summaries at a high level.
- High-level active rule specification and triggering.
- Integration of active DB and KDD techniques.
  - Use active DB techniques to activate a KDD process based on the importance (e.g., critical conditions) and freshness (compared with the similar situation in KBs).
  - Active dedicated sampling & knowledge discovery process at critical points.
  - Progressively refined knowledge discovery process.
Knowledge Discovery in Multi-Databases

- A major challenge in multi-DBs: semantic heterogeneity.
  - Multi-databases: low level heterogeneity but high-level homogeneity (e.g., school grading systems).
  - The role of generalization-based knowledge discovery: raise concept levels and ease semantic heterogeneity.
- Knowledge discovery and query transformation.
  - Not only an exchangeable “export schema” but also a common high level “language” (“vocabulary”).
  - Each local database system provides two-way transformation between low and high level data.
  - The transformation contributes to high level knowledge exchange (for KDD) and query/result interpretation (for interoperability).
Schema Integration and Schema Evolution

- How data influences schema evolution? A schema should reflect the current status of data in the database.
  - Objects with many common attributes and properties should be grouped together.
  - Frequently accessed and complex structured data should have detailed classification.
  - Different classes should be maximally distinguished from each other.

- The role of data mining in schema construction.
  - Finds general characteristics, regularity, distribution and evolution of data in a database.
  - Provides multi-views for different users, but the physical one should be chosen for efficient accessing.
Knowledge Discovery in Global Information Systems

- Internet: stores huge volumes, highly unstructured, multimedia, fast expanding information for diverse users.
- A multiple layered database approach for resource and knowledge discovery in the global information-base.
  - Layer-0: An unstructured, massive, primitive, diverse global information-base.
  - Layer-1: A relatively structured, massive, distributed database by data analysis, transformation and generalization techniques.
  - Higher-layers: Further generalization to form progressively smaller, better structured, and less remote databases for efficient browsing, retrieval, and information discovery.
An MLDB Architecture for Global Information Systems

Global Information Base

Layer 0

Layer 1

Layer 2

Layer 3
Resource and Knowledge Discovery Queries for GIS

- WebMiner and WebQL (a query language).
- A resource discovery query.

```
select * 
from document 
related to computing science 
where ‘‘Ted Thomas’’ in authors 
and one of keywords like ‘‘data mining’’
```

- A knowledge discovery query.

```
describe authors.affiliation, publication, 
publication_date 
from document 
related to Computing Science 
where one of keywords like ‘‘data mining’’ 
and access_frequency = ‘‘high’’
```
Strength of the MLDB Approach for GIS Management

- Transform unstructured, diverse global information-base into relatively structured, progressively smaller DBs.
- Facilitate the application of well-developed database technology.
- Support high-level resource and knowledge discovery queries.
- High level information browsing from different angles.
- Interoperability: Heterogeneity (low levels) vs. homogeneity (high levels).
- Query processing: localized, layered, saving the transmission of bulky data.
- Schema formation, evolution, and integration.
- Incremental updating and consistency checking.
Knowledge Discovery Applications

- Querying database knowledge.
- A multi-layered, knowledge-intensive data model for database content browsing.
- Cooperative query answering.
- Design of database systems: Schemas evolution and object migration.
- High-level integration of multi-DBs: Low level heterogeneity vs. high level homogeneity.
- Semantic query optimization: Using data-relevant knowledge and general statistics.
- Performance prediction: Using the information available for the similar groups of data.
Performance Prediction Using Data Mining Techniques

• Performance prediction: Predict one’s performance based on that of similar classes of data in the database.

• Performance prediction using data mining techniques.
  – Determine the major factors influencing one’s performance.
    * Expert’s judgement.
    * Automatic discovery of the influencing factors: (distribution and deviation analysis).
  – Discovery of the performance of similar class of existing data.
  – Comparative performance prediction.

• Example: The amount of research grants that an applicant may obtain.
Typical Knowledge Discovery Systems

- **KDW** (GTE Labs; Piatetsky-Shapiro, et al.): multi-strategy, strong rules, statistical approaches, etc.
- **INLEN** (George Mason; Michalski, Kershberg, et al.): Integration of multiple learning strategies.
- **DBMiner/DBLearn**: Find characteristic, discriminant, class description, and multi-level association rules in large relation DBs.
- **Explora** (Germany; Klösgen): Internal DB, data analyzer.
- **IMACS** (AT & T, Brachman et al.): Internal DB, knowledge representation and construction.
- **Quest** (IBM Almaden; Agrawal, Imielinski, et al.): Find association, classification, sequential patterns and
similar sequences in large databases.

- **SKICAT** (Fayyad, et al.) Large-scale sky survey.
- **Others:**
  - 49er (Zytkow)
  - Spotlight/Opportunity explorer (Anand and Kahn)
  - Datalogic/R (Ziarko et al.)
  - Health KEFIR (Matheus, et al.)
  - KnowledgeSeeker (de Ville)
  - RECON (Kerber)
  - CoverStory (IRI)
A View of Knowledge Discovery Methods in KDD Systems

- Machine learning: AQ15, ID3/C4.5, Cluster-2, Cobweb, etc.
- Knowledge representation & reasoning: e.g., IMACS.
- Integration of database and machine learning: Quest, DBMiner(DBLearn), etc.
- Mathematical approaches: Rough sets, fuzzy sets, information theory. e.g., Datalogic/R, 49er.
- Statistical approaches, e.g., KnowledgeSeeker.
- Inductive logic programming (Muggleton & Raedt’94).
- Visualization and interactive mining, e.g., Keim et al.’94.
- Integration with deductive DB, e.g., Shen et al.’94.
- Multi-strategy data mining: INLEN, KDW+, etc.
Research Issues on Knowledge Discovery in Databases

• Further exploration of KDD methodologies:
  – Integration with deduction techniques
  – Statistical tools & probabilistic reasoning
  – Machine learning: concept formation, learning by analogy, explanation-based learning, etc.
  – Inductive logic programming

• Implementation: ◊ A set of well-tuned, customized knowledge discovery operators, ◊ visualization tools, ◊ multi-strategy integration.

• Knowledge discovery in advanced database systems:
  ◊ multiple heterogeneous DBs, ◊ multimedia DBs, ◊ global information systems.

• KDD Applications: ◊ Intelligent query answering, ◊ database content browsing, ◊ multi-resolution model.
DEDUCTIVE AND OBJECT-ORIENTED DATABASES
Deductive Database Technology

- Relational database technology: Why success?
  - High-level, declarative query interface (SQL).
  - Shift programmer’s burden to system implementation and optimization.
  - Others: concurrency, recovery, security, etc.

- Deductive database: Relational technology + logic programming. [i.e., an extension of relational DB technology towards general, logic-based (declarative) programming languages].

- Major issues: Compilation, constraint-based query evaluation, implementation of recursion, function, aggregation, negation, etc.
Deductive Database Systems/Prototypes

- Aditi (U. of Melbourne): magic Sets, constraints.
- Coral++ (U. of Wisconsin): magic templates, DOOD features, etc.
- LDL++ (UCLA & MCC): magic sets, functions.
- LogicBase (Simon Fraser): chain-based evaluation, functions, constraints.
- NAIL! (Stanford & AT&T): magic sets, imported functions.
- XSB (SUNY-Stony Brook): set-oriented, top-down memoing.
Why Chain-Based Evaluation?

1. Most popularly studied recursions can be compiled into highly regular chain or pseudo-chain programs.
2. Compilation may capture some intricate bindings which can hardly be captured otherwise.
3. It handles recursions with function symbols elegantly.
4. Regularity, simplicity and exactness of the compiled chain programs leads to efficient query evaluation by exploration of the available query constraints, integrity constraints, recursion structures, etc.
5. It facilitates the incorporation of several interesting evaluation techniques: Chain-following, chain-split, constraint pushing, partial evaluation, etc.
Compilation: Capture of Difficult Bindings

- A “complex” linear recursion.
  \[ r(X, Y, Z) \leftarrow e_0(X, Y, Z). \]
  \[ r(X, Y, Z) \leftarrow a(X, Y), r(X_1, Z, Z_1), b(X_1, Z_1). \]

- Binding propagation for query “? – r(X, Y, c)”.
  \[ r^{ffb}(X, Y, Z) \leftarrow r^{fbf}(X_1, Z, Z_1), a^{ff}(X, Y), b^{bb}(X_1, Z_1). \]
  \[ r^{fbf}(X_1, Z, Z_1) \leftarrow a^{fb}(X_1, Z), r^{fff}(X_2, Z_1, Z_2), b^{bb}(X_2, Z_2). \]

- Subgoals reordering cannot solve the problem, but query-independent compilation works fine.
  \[ r^{ffb}(X, Y, Z) \leftarrow e_0^{ffb}(X, Y, Z). \]
  \[ r^{ffb}(X, Y, Z) \leftarrow t^b(Z), a^{ff}(X, Y). \]
  \[ t^b(Z) \leftarrow e_0^{fbf}(U, Z, V), b^{bb}(U, V). \]
  \[ t^b(Z) \leftarrow a^{fbf}(Z, Z_1), t^b(Z_1) \]
Chain-Based Evaluation Method

• Chain: An infinite set of highly regular relational expressions.

\[
\text{ancestor}(X, Y) \leftarrow \text{parent}(X, Y).
\]

\[
\text{ancestor}(X, Y) \leftarrow \text{ancestor}(X, W), \text{parent}(W, Y).
\]

\[
\text{ancestor}(X, Y) = \bigcup_{i=1}^{\infty} (\text{parent}^i(X_{i-1}, X_i), X = X_0, Y = X_i).
\]

• Chain-based query evaluation method:
  - chain-following, chain-split, existence checking, constraint-based evaluation, etc.

• Function-bearing recursion — structured data objects, arithmetic functions, and recursive data structures (lists, trees, sets, etc.):
  Transform functions into functional predicates.
I. Chain-Following Evaluation

- Starts with a highly selective end of a chain (start end), proceeds towards the other end of the chain (finish end), and then possibly to other chains.
- Simulates partial transitive closure processing (single chain recursion) and the counting method (multiple chain recursion).
- Ex. length can be compiled into a double-chain recursion.

\[
\text{length}([], 0).
\]
\[
\text{length}([X|L_1], \text{succ}(N_1)) \leftarrow \text{length}(L_1, N_1).
\]

For the query "? = length([a, b, c], N)",

\[
\text{length}^{bf}(L, N) \leftarrow L =^{bb} [], N =^{fb} 0.
\]
\[
\text{length}^{bf}(L, N) \leftarrow \text{cons}^{fb}(X, L_1, L),
\]
\[
\text{length}^{bf}(L_1, N_1), \text{succ}^{bf}(N_1, N).
\]
II. Chain-Split Evaluation

- **Chain-split**: Based on (1) evaluation efficiency or (2) chain-level finite evaluability.
- A chain generating path is split into: (1) immediately evaluable portion, and (2) buffered portion: the buffered values should be patched in the “down-chain” evaluation.
- **Example.** $append$ — a single-chain recursion.
  
  $append^{bf}$: chain-following, $\{append^{bb}, append^{ff}\}$: chain-split, $append^{bff}$: not finitely evaluable.

  \[
  append([], L, L).
  \]

  \[
  append([X|L_1], L_2, X|L_3) \leftarrow append(L_1, L_2, L_3).
  \]

  **For query** “? – $append(U, V, [a, b])$”,

  \[
  append^{ff}(U, V, W) \leftarrow U =^{fb} [], V =^{fb} W.
  \]

  \[
  append^{ff}(U, V, W) \leftarrow cons^{ff}(X_1, W_1, W),
  append^{ff}(U_1, V, W_1), cons^{bf}(X_1, U_1, U).
  \]
III. Existence Checking Evaluation

- Avoid exhaustive search if the finish end is not inquired.
- Example: member — a single-chain recursion.
  \[
  \text{member}(X, [X|L_1]). \\
  \text{member}(X, [Y|L_1]) \leftarrow \text{member}(X, L_1). \\
  \text{member}^{bb}(X, L) \leftarrow \text{cons}^{bb}(X, L_1, L). \\
  \text{member}^{bb}(X, L) \leftarrow \text{cons}^{ff}(Y, L_1, L), \text{member}^{bb}(X, L_1).
  \]
- Evaluate “? – member(a, [b, a, c, d])” by existence checking.
  - Evaluating the exit rule derives no answer.
  - Evaluating the recursive rule derives “\(L_1 = [a, c, d]\)”, and
    “member(a, [a, c, d])”. It is evaluated to \text{true} since
    “member(a, [a, c, d])” satisfies the exit rule. It terminates
    because one \text{true} answer validates the query.
IV. Constraint-Based Evaluation

- Constraint push for efficiency and termination.
- Example. Airflight reservation.

\[
\text{travel}([Fno], \text{Dep}, \text{Arr}, \text{Fare}) \leftarrow \text{edb\_flight}(Fno, \text{Dep}, \text{Arr}, \text{Fare}).
\]

\[
\text{travel}([Fno|FnoList], \text{Dep}, \text{Arr}, \text{Fare}) \leftarrow \\
\text{edb\_flight}(Fno, \text{Dep}, \text{Int}, F_1), \text{travel}(FnoList, \text{Int}, \text{Arr}, F_2), \\
\text{Fare} = F_1 + F_2.
\]

- The compiled single-chain recursion:

\[
\text{travel}(L, D, A, F) \leftarrow \\
\text{edb\_flight}(Fno, D, A, F), \text{cons}(Fno, [], L), \text{sum}(F, 0, F).
\]

\[
\text{travel}(L, D, A, F) \leftarrow \\
\text{edb\_flight}(Fno, D, I, F_1), \text{sum}(F_1, S_1, F), \\
\text{cons}(Fno, L_1, L), \text{travel}(L_1, I, A, S_1).
\]
Query: Fly from Vancouver to Zurich, . . . .

? = travel(FnoList, vancouver, zurich, F),
F ≥ 500, F ≤ 800, length(FnoList, N), N ≤ 4.

- Constraint push at the start end (arrival): All the constraints associated with this end.

- Constraints push at the finish end (departure).

Analysis of argument monotonicity and query:

◊ Push in “Fare ≤ 800”, but not “Fare ≥ 500”.
◊ If “Fare = 800”, push in “Fare ≤ 800”.
◊ Mapping monotonicity: e.g. longitude(Dep) (flight direction), length(FnoList, N), etc.

Query analysis: monotonicity

→ termination restraint template, “Fare ∇ C”
→ concrete termination restraint, “Fare ∇ 800”.

- Final selection using the remaining constraints.
**Nested Linear Recursions: The N-queens Recursion**

\[ \text{nqueens}(N, Qs) \leftarrow \text{range}(1, N, Ns), \text{queens}(Ns, [], Qs). \]

\[ \text{range}(M, N, [M|Ns]) \leftarrow M < N, M_1 \text{ is } M + 1, \text{range}(M_1, N, Ns). \]

\[ \text{range}(N, N, [N]). \]

\[ \text{queens}(\text{Unplaced}, \text{Safe}, Qs) \leftarrow \]
\[ \quad \text{select}(Q, \text{Unplaced}, \text{Unplaced}_1), \text{not attack}(Q, \text{Safe}), \]
\[ \quad \text{queens}(\text{Unplaced}_1, [Q|\text{Safe}], Qs). \]

\[ \text{queens}([], Qs, Qs). \]

\[ \text{attack}(X, Xs) \leftarrow \text{attack}(X, 1, Xs). \]

\[ \text{attack}(X, N, [Y|Ys]) \leftarrow X \text{ is } Y + N; X \text{ is } Y - N. \]

\[ \text{attack}(X, N, [Y|Ys]) \leftarrow N_1 \text{ is } N + 1, \text{attack}(X, N_1, Ys). \]

\[ \text{select}(X, [X|Xs], Xs). \]

\[ \text{select}(X, [Y|Ys], [Y|Zs]) \leftarrow \text{select}(X, Ys, Zs). \]
Chain-Based Query Analysis and Evaluation

- Query evaluation: ◇ is independent of predicate ordering and rule ordering in a program, ◇ computes the correct and complete set of answers.

- For the $n$-queens recursion, it evaluates the $bf$ bindings efficiently.
  ◇ $\cdot \ ? - n$queens$(5, Qs)$.

- It evaluates other kinds of bindings efficiently as well:
  ◇ $\cdot \ ? - n$queens$(N, [2, 4, 1, 3])$. ◇ $\cdot \ ? - n$queens$(N, [3, X, Y, 2])$.

- For unsafe queries, it returns a warning message without evaluation,
  ◇ $\cdot \ ? - n$queens$(N, [2|L])$.

- Such analysis and evaluation cannot be performed by other logic programming or deductive DB techniques.
Query Analysis for Query: ? − nqueens^{bf}(4, Qs).

\[\begin{align*}
n\text{queens}^{bf}(N, Qs) & \leftarrow \text{range}^{bf}(1, N, Ns), \text{queens}^{bf}(Ns, [], Qs). \\
\text{range}^{bf}(M, N, MNs) & \leftarrow \\
& M <^{bb} N, M_1 = M +^{fb} 1, \text{range}^{bf}(M_1, N, Ns), \text{cons}^{bf}(M, Ns, MNs). \\
\text{range}^{bf}(M, N, MNs) & \leftarrow M =^{bb} N, \text{cons}^{bf}(N, [], MNs). \\
\text{queens}^{bf}(U, S, Qs) & \leftarrow \text{select}^{bf}(Q, U, U_1), \\
& \text{not attack}^{bb}(Q, S), \text{cons}^{bf}(Q, S, S_1), \text{queens}^{bf}(U_1, S_1, Qs). \\
\text{queens}^{bf}(U, S, Qs) & \leftarrow U =^{bb} [], S =^{bf} Qs. \\
\text{select}^{bf}(X, YYs, YZs) & \leftarrow \text{cons}^{ff}(X, YZs, YYs). \\
\text{select}^{bf}(X, YYs, YZs) & \leftarrow \\
& \text{cons}^{ff}(Y, Ys, YYs), \text{select}^{bf}(X, Ys, Zs), \text{cons}^{bf}(Y, Zs, YZs). \\
\text{attack}^{bb}(X, Xs) & \leftarrow \text{atkk}^{bb}(X, 1, Xs). \\
\text{atkk}^{bb}(X, N, YYs) & \leftarrow (X = Y +^{fb} N; X = Y -^{fb} N), \text{cons}^{bf}(Y, Ys, YYs) \\
\text{atkk}^{bb}(X, N, YYs) & \leftarrow \\
& \text{cons}^{bf}(Y, Ys, YYs), N_1 = N +^{fb} 1, \text{atkk}^{bb}(X, N_1, Ys).
\end{align*}\]
I. Evaluating $\text{range}(1, 4, MNs)$: Chain-split evaluation

$$range^{bbf}(M, N, MNs) \leftarrow$$
$$M <^b N, M_1 = M +^f 1, range^{bbf}(M_1, N, Ns), cons^{bbf}(M, Ns, MNs).$$
$$range^{bbf}(M, N, MNs) \leftarrow M =^b N, cons^{bbf}(N, [], MNs).$$

• 1st iteration:
  $$range(1, 4, MNs) \rightarrow \text{“range}(2, 4, Ns), \ cons(1, Ns, MNs)”. $$

• 2nd iteration:
  $$range(2, 4, Ns) \rightarrow \text{“range}(3, 4, Ns_2), \ cons(2, Ns_2, Ns)”.$$ 

• 3rd iteration:
  $$range(3, 4, Ns_2) \rightarrow \text{“range}(4, 4, Ns_3), \ cons(3, Ns_3, Ns_2)”.$$ 

• 4th iteration: $range(4, 4, Ns_3) \text{ matches only the exit rule:}$
  $$Ns_3 = [4].$$ 

• Result: $MNs = [1, 2, 3, 4].$
II: Evaluating \textit{queens}([1, 2, 3, 4], [], Qs): Exit-Chain Evaluation

- The exit-chain evaluation should be performed since only the exit rule is finitely evaluable immediately.

\[ \text{queens}^{bbf}(U, S, Qs) \leftarrow \]
\[ \text{select}^{bbf}(Q, U, U_1), \text{not attack}^{bb}(Q, S), \]
\[ \text{cons}^{bbf}(Q, S, S_1), \text{queens}^{bbf}(U_1, S_1, Qs). \]
\[ \text{queens}^{bbf}(U, S, Qs) \leftarrow U =^{bb} [], S =^{bf} Qs. \]
\[ \text{select}^{bbf}(X, YYs, YZs) \leftarrow \]
\[ \text{cons}^{bbf}(Y, Ys, YYs), \text{select}^{bbf}(X, Ys, Zs), \text{cons}^{bbf}(Y, Zs, YZs). \]

- Similar to Step I, set-oriented iterative evaluation leads to the answer set:
\[ Qs = \{ [2, 4, 1, 3], [3, 1, 4, 2] \}. \]
Query Analysis for Query: $\text{?} \leftarrow \text{nqueens}^{fb}(N, [2, 4, 1, 3])$.

\[
n\text{queens}^{fb}(N, \text{Qs}) \leftarrow \text{queens}^{fbb}(N, [], \text{Qs}), \text{range}^{fbb}(1, N, N, N).
\]

\[
\text{queens}^{fbb}(U, S, \text{Qs}) \leftarrow
\text{queens}^{fbb}(U_1, S_1, \text{Qs}), \text{cons}^{fbb}(Q, S, S_1),
\text{not attack}^{bb}(Q, S), \text{select}^{fbb}(Q, U, U_1).
\]

\[
\text{queens}^{fbb}(U, S, \text{Qs}) \leftarrow
\text{queens}^{fbb}(U_1, S_1, \text{Qs}), \text{cons}^{fbb}(Q, S, S_1),
\text{not attack}^{bb}(Q, S), \text{select}^{fbb}(Q, U, U_1).
\]

\[
\text{queens}^{fbb}(U, S, \text{Qs}) \leftarrow U =^{fb} [], S =^{fb} \text{Qs}.
\]

\[
\text{select}^{fbb}(X, YYs, Y Zs) \leftarrow \text{cons}^{bbf}(X, Y Zs, YYs).
\]

\[
\text{select}^{fbb}(X, YYs, Y Zs) \leftarrow
\text{cons}^{fbb}(Y, Zs, Y Zs), \text{select}^{fbb}(X, Y s, Zs), \text{cons}^{bbf}(Y, Y s, YYs),
\text{range}^{fbb}(M, N, M Ns) \leftarrow
\text{cons}^{fbb}(M, N s, M Ns), M_1 = M +^{bb} 1, \text{range}^{fbb}(M_1, N, N s), M <^{bb} N.
\]

\[
\text{range}^{fbb}(M, N, M Ns) \leftarrow M =^{bf} N, \text{cons}^{bbf}(N, [], M Ns).
\]
Evaluating query “? - nqueens(N, [2, 4, 1, 3])”

- Evaluating “queens$^b_f(b)(Ns, [], [2, 4, 1, 3])” means to evaluate:
  
  $$queens([], [2, 4, 1, 3], [2, 4, 1, 3]),$$
  $$cons(Q, S, [2, 4, 1, 3]), not\ attack(Q, S), select(Q, U, []).$$
  
  which leads to $Q = 2, S = [4, 1, 3]$ and $U = [2]$.

- The 2nd iteration proceeds as follows,
  
  $$queens([2], [4, 1, 3], [2, 4, 1, 3]),$$
  $$cons(Q, S, [4, 1, 3]), not\ attack(Q, S), select(Q, U, []).$$
  
  which leads to $Q = 4, S = [1, 3]$ and $U = \{[2, 4], [4, 2]\}$.

- The 4th iteration derives,
  
  $$Ns = \{[1, 2, 3, 4], [1, 2, 4, 3], \ldots, [4, 3, 2, 1]\}.$$  

- The evaluation of “range$^b_f(b)(1, N, Ns)” derives $N = 4.$
Level-Crossing Constraint Pushing in Nested Recursions

- Constraint derived in a lower level recursion by monotonicity analysis may be able to be pushed into a higher level recursion.
- Example. In $n\text{queens}(N, [2, 4, 1, 3])$, a constraint “$\text{mono}\_\text{inc}\_\text{list}(N)$” (list containing elements with monotonic increasing value), derived from range recursion, can be pushed into $\text{queens}$ (because “$\text{mono}\_\text{inc}\_\text{list}(YYs) \Rightarrow \text{mono}\_\text{inc}\_\text{list}(YZs)$” in select).
- The transformed $\text{queens}$ recursion with level-crossing constraint pushed in,

$$
\text{queens}(U, S, Qs) \leftarrow \\
\text{select}(Q, U, U_1), \text{not attack}(Q, S), \text{cons}(Q, S, S_1), \\
\text{mono}\_\text{inc}\_\text{list}(U), \text{queens}(U_1, S_1, Qs).
$$
queens(U, S, Qs) ← U = [], S = Qs.
select(X, YYs, YZs) ← cons(X, YZs, YYs).
select(X, YYs, YZs) ←
    cons(Y, Ys, YYs), cons(Y, Zs, YZs),
    mono_inc_list(YYs), select(X, Ys, Zs).

• With this constraint pushing, only one answer “[1, 2, 3, 4]”
is generated from queens (instead of n! lists).

• Complexity for evaluating nqueens\(^{fb}\) is substantially
  reduced from \(O(n!)\) to \(O(n^2)\).

• Similar level-crossing constraint pushing can be explored
  in a permutation sort program, which reduces the
  algorithm complexity from \(O(n!)\) to \(O(n^2)\).
Evaluation of Regular Nonlinear Recursions

- Query-independent compilation and chain-based query evaluation: applicable to regular nonlinear recursions.
- Most popularly studied nonlinear recursions are regular (with regular variable passing patterns).
  \[ \text{hanoi}(1, A, B, C, [A \text{ to } B]). \]
  \[ \text{hanoi}(N + 1, A, B, C, \text{Moves}) \leftarrow \text{hanoi}(N, A, C, B, \text{Ms}_1), \text{hanoi}(N, C, B, A, \text{Ms}_2), \text{append}(	ext{Ms}_1, [A \text{ to } B|\text{Ms}_2], \text{Moves}). \]

- Example: Evaluation of “hanoi” and “qsort”.
  \[ ? \leftarrow \text{hanoi}(3, a, b, c, \text{Moves}). \]
  \[ ? \leftarrow \text{hanoi}(N, a, b, c, [a \text{ to } b, a \text{ to } c, b \text{ to } c, a \text{ to } b, c \text{ to } a, c \text{ to } b, a \text{ to } b]). \]
  \[ ? \leftarrow \text{qsort}([4, 9, 5], Ys). \]
  \[ ? \leftarrow \text{qsort}([X|Ys], [4, 5, 9]), X > 4. \]
Compilation and Query Evaluation in LogicBase

- Query-independent deduction rule compilation:
  1. Classification and simplification of recursions.
  2. Compilation and normalization.

- Chain-based query evaluation.
  1. Test query’s finite evaluability and termination: if not, stop.
  2. Perform query binding analysis and determine:
     - the start point of the chain processing,
     - the predicate evaluation order, and
     - chain-split, chain-following, constraint-pushing, etc.
  3. Query evaluation plan generation.
  4. Set-oriented query evaluation.
Performance Comparison: Three Query Evaluation Methods

• A cost model:
  – cost for evaluating functions.
  – cost for accessing relations.
  – cost for call of logic rules (non-recursive and recursive).

• Three query evaluation strategies are compared:
  – top-down set-at-a-time evaluation (similar to Prolog).
  – magic sets method.
  – chain-based method as in LogicBase.

• Performance result shows the flexibility and high performance of chain-based evaluation method, especially with constraint pushing.
Performance Comparison: Cost for \( n\text{queens}^{bf} \)
Performance Comparison: Cost for $nqueens^{fb}$

- chain
- chain with constraint pushing

![Graph showing cost vs size for different strategies in the nqueens problem.](image)
Performance Comparison: Cost for $\text{permutation\_sort}^{bf}$

![Graph showing cost vs. size for different methods: chain, chain with constraint pushing, magic sets, and top-down.](chart.png)
Strength and Limitations of LogicBase

• Strength:
  – Systematic analysis of compiled recursions and queries and efficient query evaluation.
  – Generalizable to ◇ HiLog and F-logic programs, ◇ aggregation, ◇ modularly stratified negation, etc.
  – Can be integrated with other models of database systems for advanced applications.

• Limitation: Confined to the recursions compilable into chain forms or regular nonlinear recursions.
  ◇ But where are the application examples of complex recursions which cannot be implemented by the method?
Applications of Deductive Database Systems

- Deduction and recursion on large databases: Ancestor, parts/subparts, network connection, air-flight reservation, bill-of-materials, scheduling, inventory control, data dredging, etc.

- Deduction in spatial DBs, engineering DBs, multi-databases, etc. — for highly regulated industry (needs modularity, declarativity, knowledge independence, concurrent design activities, etc.)

- Logic programming by deductive DB approaches: order-independency, data intensive & set-oriented, completeness & efficiency.

- Integration of deductive database with object-oriented database technologies.
Towards Deductive and Object-Oriented DBs (DOODs)

- Why Deductive and Object-Oriented Databases?
  - An integration of ◇ data management, ◇ object management, and ◇ knowledge management.
- Strength and weakness of deductive DBs.
  - Declarative interface, high level queries, optimization.
  - Complex objects, methods, class hierarchies, OID, etc.
- OODBs: complementary to deductive DBs.
- Towards an integration of deductive databases and object-oriented databases.
  1. Extended relational DBs.
  2. Object-oriented features.
  3. High-level programming and query interfaces (assisted with deduction rules).
A DOOD Model and Its Implementation

- Theoretical foundations of deductive and object-oriented databases: F-Logic, HiLog, etc.
- A deductive and object-oriented data model: First-order semantics and higher-order syntax.
- Implementations: Compilation and query evaluation plan generation in deductive and object-oriented databases.
  - Deduction rules: Similar to relational views, declarative, maximal compilation when possible.
  - Methods: Input/output mode restraints, parameter passing, cost estimation, dynamic integration.
  - Intergated evaluation of rules and methods: Query plan generation and selection.
INTEGRATION AND SUMMARY
Integration of DOOD and KDD Techniques

- KDD: Construction of knowledge-bases from databases.
- DOOD: Application of knowledge to databases.
- Integration of knowledge construction and application.
  - Rule-guided induction.
  - Induction from data and schema.
  - Integration of induction and deduction techniques.
  - Knowledge discovery and KDD tools in DOODs.
  - Knowledge merging: merging deduction rules and discovered rules.
  - Applications of discovered knowledge in DOOD (semantic query optimization, intelligent query answering, data purification, etc.).
DBMS to KBMS: An Evolutionary Approach

• Why start with database systems?
  – Theoretical foundations,
  – System implementations, and
  – Technology applications.

• Why move towards knowledge-base systems?
  – Enhanced functionalities.
  – Intelligent interfaces.
  – Application demands for data-intensive knowledge-base systems.

• Why adopt an evolutionary approach?
  Integration of extended-relational database technologies with ◻ deduction, ◻ object-orientation, ◻ knowledge discovery, and ◻ active database technologies.
References for the Tutorial

Note: This list of references is not for a full coverage of the research topics in the area but only for understanding the materials in the tutorial.

**DBMS: General Introduction.** (Also, SIMGOD, VLDB, ICDE, PODS, ACM/ODS, IEEE/TKDE, and many other conf, proceedings and journals).


**Machine Learning: An Introduction** (Also, machine learning conference proceedings).


**Knowledge Discovery in Databases.** (Also, KDD workshop/conf. proceedings and IJCAI/AAAI/ML/DB conf. proceedings).

From DBMS to KBMS: An Evolutionary Approach

Integration and Summary


- Attribute-Oriented Induction for Knowledge Discovery in Databases.


- Deductive and Object-Oriented Databases (in most DB conf, proc.’s/journals and DOOD conf, proc.’s).


From DBMS to KBMS: An Evolutionary Approach

Integration and Summary


*LogicBase*

2. J. Han, Chain-based evaluation in deductive databases, *IEEE Transactions on Knowledge and Data Engineering*, 1995 (to appear).
7. J. Han, L. Liu, and Z. Xie, LogicBase: A deductive database system prototype, In *Proc. 3rd Int'l Conf. on Information and Knowledge Management*, pp. 226-233, Gaithersburg, Maryland, Nov, 1994.