Sprachen und Systemarchitekturen

Seminar: Business Intelligence –

Teil II: Data-Mining & Knowledge-Discovery

von Jörg Ramser

Betreuer: Dipl. Inf. Jürgen Göres

23.01.2004

Gliederung

1 EINLEITUNG 3

2 DATA-MINING-PRIMITIVE 4

2.1 RELEVANTE DATEN 4
2.2 ART DES ZU ERKENNENDEN WISSENS 4
2.3 HINTERGRUNDWISSEN 6
2.4 RELEVANZMAßE FÜR ERGEBNISSE 7
2.5 PRÄSENTATION UND VISUALISIERUNG DER ERGEBNISSE 9

3 DATA-MINING-ANFRAGESPRACHEN 12

3.1 DMQL 12
3.1.1 Auswahl relevanter Daten 12
3.1.2 Auswahl der Art des zu erkennenden Wissens 13
3.1.3 Modellierung von Hintergrundwissen (Hierarchien) 15
3.1.4 Definition von Relevanzmaßen 16
3.1.5 Auswahl der Ergebnispräsentation und -visualisierung 16
3.1.6 Beispiel 17
3.1.7 Fazit 17
3.2 OLE DB FOR DATA MINING 17
3.2.1 Spezifikation von Data-Mining-Modellen 18
3.2.2 Bereitstellung von Trainings-Daten aus Tabellen 20
3.2.3 Durchsuchen des Data-Mining-Model 21
3.2.4 Anfragen zur Vorhersage auf auszuwertenden Daten 22
3.2.5 Fazit 23
3.3 GEGENÜBERSTELLUNG DMQL UND OLE DB FOR DATA MINING 23
3.4 ANDERE SPRACHEN 23

4 ARCHITEKTUREN VON DATA-MINING-SYSTEMEN 25

4.1 KLASSIFIZIERUNG VON DATA-MINING-ARCHITEKTUREN 25
4.2 BEISPIEL FÜR EINE STARK GEKOPPELTE ARCHITEKTUR 26

5 ZUSAMMENFASSUNG 28

6 ANHANG 30

REFERENZEN 31

1 Einleitung

In den vergangenen Jahren kam es unter anderem in der Forschung und in nahezu allen Unternehmensbereichen zu einem unaufhaltsam schnellen Wachstum der gesammelten Daten. In all diesen Daten ist möglicherweise wichtiges Wissen enthalten, das aber aufgrund der unüberschaubar großen Datenmengen von Menschen nicht entdeckt werden kann. Data-Mining dient zur Entdeckung dieses unbekannten, potentiell nützlichen Wissens. Nützlich ist dieses Wissen insofern, da beispielsweise Unternehmen hiermit in der Lage sind, sich Wettbewerbsvorteile gegenüber ihren Konkurrenten zu sichern. Sie nutzen dieses Wissen als Basis zur Entscheidungsfindung und erhoffen sich dadurch, in möglichst kurzer Zeit Tendenzen beziehungsweise Verhaltensmuster in den Datenbeständen zu finden. Dadurch können Korrelationen zwischen zwei oder mehreren relevanten Unternehmensgrößen der Gegenwart und Vergangenheit abgeleitet werden. Aber auch Risiken sind schnell erkennbar und somit vermeidbar.
Data-Mining-Systeme dienen zur Entdeckung dieses Wissens in der Form von Mustern beziehungsweise Regeln, sie durchforsten die Datenbanken automatisch und suchen dabei nach interessanten und nützlichen, aber nicht offensichtlichen Mustern. Data-Mining-Systeme werden heute in vielen verschiedenen Bereichen eingesetzt, zum Beispiel bei der medizinischen Diagnose, im Bereich Marketing und Verkauf zur Analyse des Kaufverhaltens der Kunden, sowie bei der Vergabe von Krediten und von Versicherungen.
Diese Arbeit beschäftigt sich im Rahmen des Seminars *Business Intelligence* - Teil II: Data Mining und Discovery mit Sprachen und Systemarchitekturen von Data-Mining-Systemen.
Die Menge aller in den Daten enthaltenen Muster kann leicht die Menge der Originaldaten überschreiten, dies würde zum einen schnell zu Performanzproblemen führen und zum anderen müsste der Anwender aus einer großen Menge uninteressanter Muster die für ihn relevanten finden. Der Anwender hat allerdings meistens eine mehr oder weniger genaue Vorstellung davon, welches Wissen er in der Datenbank finden will. Das Data-Mining-System sollte dem Anwender daher die Möglichkeit bieten, mit der Hilfe einer Data-Mining-Anfrage zu beschreiben, was er sucht. Eine solche Anfrage besteht nach [HFW+96] wiederum aus fünf Primitiven zur Kommunikation des Anwenders mit dem Data-Mining-System, welche in Kapitel 2 behandelt werden. Im ersten Teil des dritten Kapitels wird dann aufbauend auf diesen fünf Primitiven eine an SQL angelehnte Sprache namens DMQL vorgestellt. Der zweite Teil beschäftigt sich anschließend mit der von Microsoft vorgeschlagenen Sprache OLE DB for Data Mining. Sie wird bereits im Microsoft SQL Server 2000 unterstützt.
Wie bei allen Softwaresystemen ist auch beim Entwurf eines Data-Mining-Systems die Architektur von großer Bedeutung. Eine gute Architektur erleichtert es dem System Aufgaben effizient und schnell durchzuführen, Informationen zwischen verschiedenen Systemen auszutauschen und flexibel in Bezug auf die Anforderungen der Anwender zu sein. Aufgrund der großen Bedeutung der relationalen Datenbanken und Data-Warehouses macht es Sinn über die Integration von Data-Mining-Systemen in diese Systeme nachzudenken. Der erste Teil des vierten Kapitels stellt daher einer Klassifikation von Data-Mining-Architekturen in Bezug auf die Integration in Datenbank- und Data-Warehouse-Systeme vor. Im zweiten Teil des vierten Kapitels wird dann Beispielhaft die Architektur von Matheus et al. aufgezeigt.

2 Data-Mining-Primitive

Wenn ein Anwender mit Hilfe von Data-Mining-Systemen nach in seinen Daten enthaltenen Mustern sucht, dann hat er eine gewisse Vorstellung, wie diese Muster aussehen sollen. Es macht also keinen Sinn alle in den Daten enthaltenen Muster entdecken und präsentieren zu wollen. Mit einer Data-Mining-Anfrage kann der Anwender beschreiben, was er eigentlich sucht und was das System finden soll. Eine solche Anfrage besteht nach [HFW+96] wiederum aus den folgenden fünf Primitiven zur Kommunikation des Anwenders mit dem Data-Mining-System:

- Relevante Daten
- Art des zu erkennenden Wissens
- Hintergrundwissen
- Relevanzmaße
- Präsentation und Visualisierung der Ergebnisse

Die nächsten fünf Abschnitte werden sich nun im Detail mit diesen Primitiven beschäftigen.

2.1 Relevante Daten

Die erste Primitive ist die Spezifikation des Teils der Datenbank, der untersucht werden soll. Denn üblicherweise ist ein Anwender bei einem bestimmten Problem nur an einem Teil der Datenbank interessiert. Es macht also keinen Sinn immer die ganz Datenbank zu untersuchen, insbesondere deshalb nicht, da die Zahl der gefundenen Muster exponentiell mit der Datenbankgröße steigen kann. Dies würde zum einen schnell zu Performanzproblemen führen und zum anderen müsste der Anwender aus einer großen Menge uninteressanter Muster die für ihn relevanten finden. Ein Anwender sollte außerdem spezifizieren können, dass nur bestimmte Attribute in die Untersuchung einbezogen werden. Diese Attribute werden dann relevante Attribute genannt, beziehungsweise, falls das Data-Mining auf einem multidimensionalen Datenwürfel ausgeführt werden soll, relevante Dimensionen.
Die Spezifikation der relevanten Attribute oder Dimensionen kann eine schwierige Aufgabe für den Anwender darstellen, denn er hat meistens nur eine grobe Vorstellung, welche Attribute für ihn interessant sein könnten. Bei der Spezifikation der relevanten Daten übersieht ein Anwender beispielsweise schnell Daten, die eine starke semantische Beziehung zu den relevanten Daten haben. Zum Beispiel ist der Verkauf bestimmter Waren stark mit Ereignissen wie Weihnachten oder Silvester verbunden. Daher sollte ein Data-Mining-System Funktionalitäten zur Verfügung stellen, die eine präzisere Spezifikation der relevanten Daten unterstützen, um die Qualität der Ausgangsdaten zu erhöhen. Hierzu gehören zum Beispiel Funktionen, die Attribute abhängig von der Art des zu erkennenden Wissens in ihrer Relevanz bewerten, oder auch Techniken zur Unterstützung der Suche nach Attributen mit starker semantischer Verbindung.
Falls relationale Datenbanken als Datenquellen benutzt werden, kann die Menge der relevanten Daten mit Hilfe einer relationalen Anfrage, also mittels Selektion, Projektion, Join und Aggregation ermittelt werden. Die Ergebnismenge der Anfrage wird initiale Relation genannt, sie kann zur Optimierung des Data-Mining-Vorgangs geordnet oder gruppiert werden. Die Daten müssen vor dem eigentlichen Mining-Prozess eventuell noch bereinigt oder transformiert werden. Da die initialen Relationen in der Regel virtuelle Tabellen sind, werden sie in Anlehnung an die aus dem Bereich der relationalen Datenbanken bekannten Views auch Minable Views genannt.

2.2 Art des zu erkennenden Wissens

Um die Zahl der entdeckten aber für den Anwender uninteressanten Muster weiter zu reduzieren, ist es wichtig zu spezifizieren, nach welcher Art von Wissen der Anwender eigentlich sucht. Die Auswahl der Funktionalität, die das Data-Mining-System ausführen soll, ist somit die zweite Primitive. Zu den verschiedenen Wissensarten gehören:

Begriffsbeschreibungen (Datencharakterisierung, Datendifferenzierung)
Begriffsbeschreibungen (concept descriptions) bieten eine kurze und prägnante Zusammenfassung von Daten und unterscheiden sie damit von anderen Daten. Die Zusammenfassung einer

Datensammlung nennt man Datencharakterisierung (characterization); der Vergleich zwischen zwei oder mehr Datensammlungen nennt man Datendifferenzierung (discrimination). Begriffsbeschreibungen sollten nicht nur aus Funktionen wie count, sum und Mittelwert bestehen, sondern auch aus Beschreibungen der Datenverteilung, wie zum Beispiel die Varianz oder der Viertelwert. Die Begriffsbeschreibung kann zum Beispiel genutzt werden, um die Verkäufe eines Unternehmens in Europa mit den Verkäufen in Asien zu vergleichen. Gesucht sind Faktoren, die diese beiden Märkte klar trennen.

Assoziationsregeln
Das Entdecken von Assoziationsregeln (associations) ist das Entdecken von Verbindungen zwischen einer Menge von Items. Sie werden oft in Form von Regeln ausgedrückt. Diese Regeln zeigen Attributwertebedingungen, die häufig zusammen auftreten. Eine Assoziationsregel der Form X => Y sagt aus, dass Datentupel, die X erfüllen, mit hoher Wahrscheinlichkeit auch Y erfüllen. Das Entdecken von Assoziationsregeln findet häufig Anwendung in Transaktionsdatenanalysen (Transaktion hier nicht im Sinne einer ACID-Transaktion sondern im Sinne einer Geschäftstransaktion) für gerichtetes Marketing, Katalogdesign oder bei anderen Entscheidungssituationen, die im Rahmen verschiedener Geschäftsprozesse anfallen.

Klassifikation
Die Klassifikation (classification) analysiert eine Menge von Trainingsdaten (zum Beispiel eine Menge von Objekten, deren zugeordnete Klasse bekannt ist) und konstruiert für jede Klasse ein Modell basierend auf den Eigenschaften der Trainingsdaten. Als Ergebnis dieses Prozesses entstehen zum Beispiel Entscheidungsbäume oder Klassifikationsregeln; sie können für ein besseres Verständnis der Klassifizierung der vorhandenen und zukünftigen Daten genutzt werden. Zum Beispiel könnte man Symptome klassifizieren und damit die Krankheit als Klasse der Symptome bestimmen.

Vorhersage (prediction)
Diese Data-Mining-Funktion sagt mögliche Werte für fehlende Attribute oder die Werteverteilung von bestimmten Attributen einer Menge von Objekten voraus. Sie sucht mit Hilfe einer statistischen Analyse Attribute, die in Verbindung mit dem fehlenden Attribut stehen, und sagt dann auf Basis von Objekten, die Ähnlichkeiten zu dem ausgewählten Objekt aufweisen, die fehlenden Werte voraus. Zum Beispiel kann man das potentielle Gehalt eines Angestellten basierend auf der Verteilung der Gehälter von vergleichbaren Angestellten vorhersagen.

Clusterbildung
Bei der Clusterbildung (clustering) werden in den Daten enthaltene Gruppen von ähnlichen Datenobjekten – so genannte Cluster – identifiziert. Ähnlichkeit kann durch Abstandsfunktionen ausgedrückt werden; spezifiziert werden solche Funktionen durch den Anwender oder Experten. Ein gutes Verfahren zur Clusterbildung bildet Cluster, bei denen die Ähnlichkeiten zwischen Objekten innerhalb desselben Clusters hoch sind und die Ähnlichkeiten zwischen Objekten verschiedener Cluster niedrig sind.

Zeitliche Entwicklungsanalyse (Sequenzanalyse)
Die zeitliche Entwicklungsanalyse (evolution analysis) analysiert große Zeitreihen mit dem Ziel Regelmäßigkeiten und interessante Charakteristiken zu entdecken. Dazu gehört das Entdecken von sequentiellen Mustern, Periodizität, Trends und Sequenzen mit Ähnlichkeiten oder Abweichungen. Zum Beispiel kann man den Trend eines Aktienkurses auf der Basis des vergangen Verlaufs voraussagen.

Die Art des zu erkennenden Wissens kann durch so genannte Mustervorlagen (templates) noch weiter eingeschränkt werden. Muster, die nicht mit der Vorlage übereinstimmen, werden als uninteressant aussortiert. Solche Vorlagen werden auch Metamuster oder Metaregeln genannt. Obwohl Metaregeln im Zusammenhang mit verschiedenen Wissensarten genutzt werden können, haben sie dennoch die größte Bedeutung bei der Entdeckung von Assoziationsregeln, denn hier werden potentiell die meisten uninteressanten Muster entdeckt.

2.3 Hintergrundwissen

Die dritte Primitive stellt die Modellierung von Hintergrundwissen (Wissen über die Anwendungsdomäne) dar. Dieses Wissen unterstützt das Data-Mining-System bei der Entdeckung von Mustern und hilft die gefundenen Muster zu bewerten. Die vorliegende Arbeit beschränkt sich auf eine einfache aber sehr leistungsfähige Art des Hintergrundwissens - die Begriffshierarchien (concept hierarchies). Begriffshierarchien erlauben es, die Entdeckung von Wissen auf verschiedenen Abstraktionsebenen ablaufen zu lassen.
Eine Begriffshierarchie bildet eine Menge von speziellen Begriffen auf allgemeinere Begriffe ab. Sie wird oft als eine baumartig organisierte Menge von Knoten dargestellt, wobei jeder Knoten für einen Begriff steht. Die Wurzel des Baumes wird auch all-Knoten genannt; er stellt den allgemeinsten Wert der Hierarchie dar. Eine Begriffshierarchie kann man in Ebenen unterteilen. Konventionell werden die Ebenen einer Begriffshierarchie von oben nach unten durchnummeriert, beginnend mit der Ebene Null für den all-Knoten. Dabei stellen die Begriffe höherer Ebenen die allgemeineren Begriffe dar. Die Blätter einer Begriffshierarchie repräsentieren die Originaldaten einer Dimension. Sie sind die speziellsten Werte (beziehungsweise Begriffe) der Dimension. Meist hat eine Begriffshierarchie die Form eines Baumes, sie kann aber auch eine gitterartige Struktur haben oder sich in partieller Ordnung befinden.

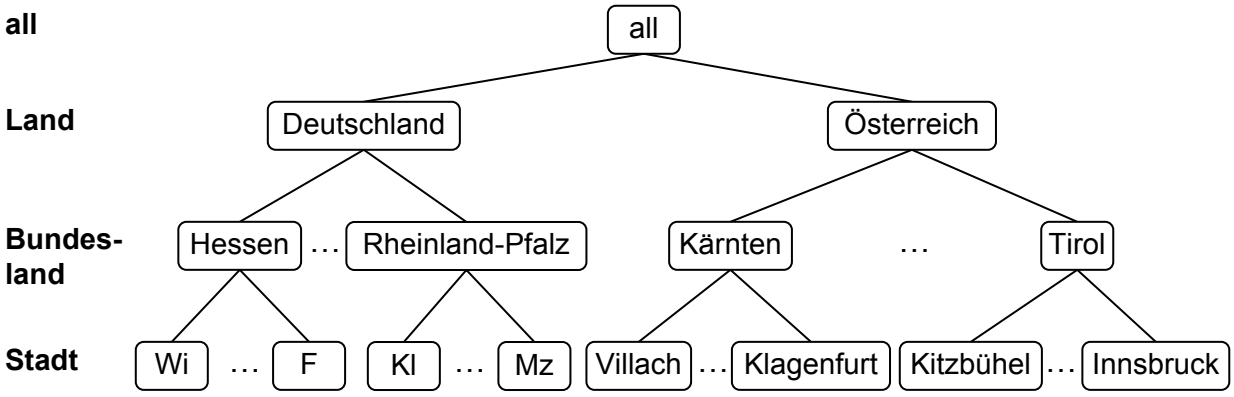

Abbildung 1: Begriffshierarchie

In Abbildung 1 ist eine solche Begriffshierarchie beispielhaft für die Dimension Ort gezeigt. Dort ist unter anderem die Abbildung vom spezielleren Begriff Stadt auf den allgemeineren Begriff Land zu sehen. In diesem Beispiel repräsentiert die erste Ebene den Begriff Land, während die zweite und dritte Ebene die Begriffe Bundesland und Stadt repräsentieren. Die Verallgemeinerung oder das Roll-Up der Daten wird durch das Ersetzen der Begriffe niederer Ebenen (zum Beispiel Stadt) durch Begriffe höherer Ebenen (zum Beispiel Land) erreicht. Dies erlaubt dem Anwender die Daten auf einer höheren Ebene der Abstraktion zu betrachten und erleichtert ihm das Verständnis der entdeckten Muster. Die Verallgemeinerung hat den weiteren Vorteil, dass die Daten verdichtet werden. Data-Mining auf der reduzierten Datenmenge verringert daher die Zahl der teuren Ein-/Ausgabeoperationen. Falls die Ergebnisdaten zu allgemein erscheinen, erlauben die Begriffshierarchien auch eine Datenverfeinerung, das so genannte Drill-Down, womit ein Absteigen in der Begriffshierarchie gemeint ist. Durch dynamisches Roll-Up und Drill-Down ist der Anwender in der Lage die Daten von verschieden Perspektiven zu betrachten und erlangt damit einen tieferen Einblick in versteckte Datenbeziehungen.
Begriffshierarchien können von Anwendern, Domänenexperten oder Wissensingenieuren entwickelt werden. Ihre Erstellung kann oft automatisch als Teil des Discovery-Prozesses erfolgen zum Beispiel auf der Basis der statistischen Verteilung der Daten. Begriffshierarchien sind typischerweise sowohl daten- als auch anwendungsbezogen. Für jede Dimension beziehungsweise jedes Attribut ist - abhängig vom Blickwinkel des Anwenders - mehr als eine Begriffshierarchie möglich. Nach [HaKa01] existieren vier verschiedene Typen von Begriffshierarchien.

Schemahierarchien
Eine Schemahierarchie (schema hierarchy) oder schemadefinierte Hierarchie ist gegeben durch eine totale oder partielle Ordnung auf den Attributen eines Datenbankschemas. Schemahierarchien stellen formell die semantisch vorhandenen Beziehungen zwischen den einzelnen Attributen dar. In Abbildung 1 ist eine solche Schemahierarchie zu sehen.

Mengengruppierende Hierarchien
Mengengruppierende Hierarchien (set-grouping hierarchies) gruppieren die Werte für ein gegebenes Attribut oder eine Dimension in verschiedene Wertebereiche. Zwischen den einzelnen Wertebereichsgruppen kann eine totale oder partielle Ordnung definiert werden. Solche mengengruppierenden Hierarchien können zur weiteren Verfeinerung von schemadefinierten Hierarchien genutzt werden. Abbildung 2 zeigt eine solche Hierarchie für das monatliche Einkommen in Euro.

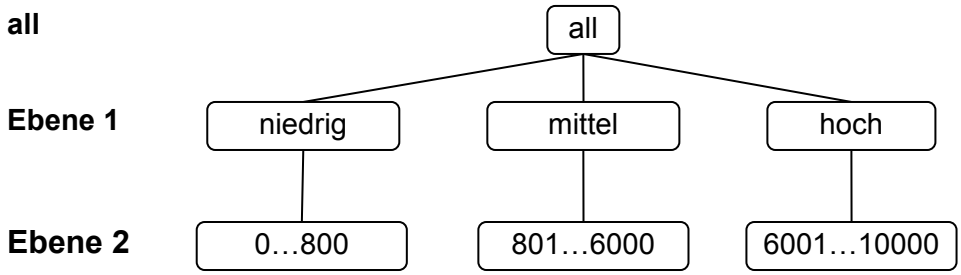

Abbildung 2: Beispiel für eine mengengruppierende Hierarchie

Aus Operationen abgeleitete Hierarchien
Aus Operationen abgeleitete Hierarchien (operation-derived hierarchies) ergeben sich aus Operationen, die auf ein gegebenes Attribut angewendet werden. Diese Operationen können von Anwendern, Experten oder dem Data-Mining-System definiert worden sein. Eine solche Operation könnte beispielsweise das Decodieren einer gegebenen URL sein. Aus der URL „informatik.uni-kl.de“ ergibt sich zum Beispiel die Hierarchie Fachbereich < Universität < Land. Ein anderes Beispiel für eine Operation könnte eine Data-Mining-Funktion wie Clusterbildung sein. Ein Beispiel wäre hier, wenn die Operation *cluster* die Wertebereiche für die Unterteilung des monatlichen Einkommens in drei Klassen selbstständig vornimmt (siehe Abbildung 2).

Regelbasierte Hierarchien
Regelbasierte Hierarchien (rule-based hierarchies) sind Hierarchien, die sich durch Ableitung aus einer Regelmenge ergeben. Als Regelmenge kommen zum Beispiel logische Regeln in Form von Prädikatenlogik in Frage. Abbildung 3 zeigt ein Regelsystem zur Einteilung des monatlichen Gehalts eines Arbeitnehmers nach Steuern in Euro in drei Kategorien.

$$\text{einkommen\_niedrig}(X) \leftarrow \text{steuersatz}(X, SS) \wedge \text{einkommen}(X, E) \wedge (1 - SS) \cdot E \leq 500$$

$$\text{einkommen\_mittel}(X) \leftarrow \text{steuersatz}(X, SS) \wedge \text{einkommen}(X, E) \wedge (1 - SS) \cdot E > 500 \wedge (1 - SS) \cdot E \leq 3500$$

$$\text{einkommen\_hoch}(X) \leftarrow \text{steuersatz}(X, SS) \wedge \text{einkommen}(X, E) \wedge (1 - SS) \cdot E > 3500 \wedge (1 - SS) \cdot E \leq 15000$$

Abbildung 3: Regelsystem für eine regelbasierte Hierarchie

2.4 Relevanzmaße für Ergebnisse

Obwohl die Spezifikation der relevanten Daten und der Art des zu erkennenden Wissens (Datencharakterisierung, Assoziationsregeln…) die Zahl der generierten Muster deutlich reduziert, generiert ein Data-Mining-Prozess noch immer eine große Zahl von Mustern. Typischerweise ist nur

ein Bruchteil dieser Muster für den Anwender von Interesse. Daher brauchen Anwender die Möglichkeit, die entdeckten Muster zu bewerten und aufgrund der Bewertung die Zahl der uninteressanten Muster einzuschränken. Dies kann durch die Definition von Relevanzmaßen geschehen. Die verschiedenen Wissensarten haben unterschiedliche Relevanzmaße. Zur Reduzierung uninteressanter Muster wird jedem Relevanzmaß ein Schwellenwert zugeordnet, welcher vom Anwender gewählt werden kann. Regeln, deren Relevanzmaße diese Schwellenwerte nicht erreichen, werden aussortiert und dem Anwender nicht als Wissen präsentiert. Diese Definition von Relevanzmaßen und den dazugehörigen Schwellenwerten ist die vierte Primitive.
Diese Arbeit beschränkt sich auf eine Auswahl objektiver Relevanzmaße.[1] Sie basieren auf der Struktur beziehungsweise der zugrunde liegenden Statistik der Muster. Man versucht mit ihnen die Zuverlässigkeit, die Einfachheit, die Nützlichkeit und die Neuheit von entdeckten Mustern zu beschreiben.

Nützlichkeit
Ein Relevanzmaß für entdeckte Muster ist die potentielle Nützlichkeit (utility) eines Musters. Der Grad der Nützlichkeit kann nach [AIS93] mit einer Funktion wie zum Beispiel support beschrieben werden. Der support einer Assoziationsregel ist der Anteil der relevanten Datentupel (oder Transaktionen), für die die Regel zutrifft. Für Assoziationsregeln mit einer Menge I von Items (uninterpretierte, diskrete Entities), einer Liste D von Transaktionen (hier nicht im Sinne einer ACID-Transaktion sondern im Sinne einer Geschäftstransaktion), wobei eine Transaktion eine Menge $t \subseteq I$ von Items ist, gilt: Eine Transaktion erfüllt eine Assoziationsregel $A \Rightarrow B$ (A, B Item-Mengen), falls $(A \cup B) \subseteq t$ ist. Der support einer Assoziationsregel ist dann

$$\text{support } (A \Rightarrow B) = \frac{\left|\{t \in D \mid (A \cup B) \subseteq t\}\right|}{|D|}$$

Für Datencharakterisierung und Datendifferenzierung kann der Anwender einen Schwellenwert Y für das Rauschen von Daten definieren. Für jede Charakterisierung, die weniger als 1/Y der relevanten Tupel repräsentiert, werden die zugehörigen Tupel als Rauschen (noise) betrachtet, sie werden dem Anwender nicht gezeigt.

Zuverlässigkeit
Für einen Anwender ist es von großer Bedeutung zu wissen, wie „glaubwürdig“ beziehungsweise zuverlässig (certainty) die einzelnen Muster sind. Als Maß für die Zuverlässigkeit von Assoziationsregeln der Form „$A \Rightarrow B$“, schlägt [AIS93] confidence - die Stärke der Korrelation - vor. Bei gegebenen Itemmengen A, B ist confidence definiert als

$$\text{confidence } (A \Rightarrow B) = \frac{\left|\{t \in D \mid (A \cup B) \subseteq t\}\right|}{\left|\{t \in D \mid A \subseteq t\}\right|}$$

Falls confidence den Wert 1 besitzt, bedeutet das, dass die Regel für alle analysierten Daten zutrifft. Solche Regeln werden auch exakt genannt. Assoziationsregeln, die sowohl den Schwellenwert für confidence, als auch den Schwellenwert für support erfüllen, werden als stark bezeichnet und gehören zu den relevanten Mustern. Regeln mit niedrigem support werden entweder als Rauschen oder Ausnahmefälle betrachtet.
Für Klassifikationsregeln existiert als Zuverlässigkeitsmaß das Maß reliability (auch accuracy genannt). Klassifikationsregeln definieren ein Klassifikationsmodel CM zur Unterscheidung von Objekten O einer Zielklasse von Objekten einer Kontrastklasse. Sei CR(o), $o \in O$ die korrekte Klasses des Objektes o, dann ist reliability definiert als

$$\text{reliability (CM)} = \frac{\left|\{o \in O \mid CM(o) = CR(o)\}\right|}{|O|}$$

[1] subjektive Relevanzmaße werden in [PSM94], [MPSM96], [ST96] und [LHC97] vorgestellt

Ein niedriger reliability-Wert bedeutet, dass die Regel fälschlicherweise eine große Anzahl von Objekten der Kontrastklasse als Objekte der Zielklasse klassifiziert.

Einfachheit
Für den Menschen sind nur solche Muster von Interesse, die er auch einfach interpretieren kann. Je komplexer die Struktur einer Regel, desto schwieriger ist es sie zu interpretieren; sie wird dadurch uninteressanter. Es bietet sich daher an, Relevanzmaße für die Einfachheit (simplicity) der entdeckten Muster zu definieren. Objektive Maße für die Einfachheit von Mustern können als Funktionen der Musterstruktur dargestellt werden, [Mic83] schlägt hier zum Beispiel die Mustergröße in Bits oder die Regellänge als Anzahl der im Muster auftretenden Attribute beziehungsweise Operatoren (falls sich die Regel in disjunktiver beziehungsweise konjunktiver Normalform befindet) vor. Regeln deren Länge eine vom Anwender definierte Schwelle überschreiten, werden als uninteressant betrachtet. Für Muster in der Form von Entscheidungsbäumen kann die Einfachheit als Funktion der Blätter- beziehungsweise Knotenzahl dargestellt werden.

Neuheit (novelty)
Als neue Muster bezeichnet man solche Muster, die neue Informationen enthalten. Zum Beispiel werden Ausnahmedaten als neu betrachtet, falls sie von dem abweichen, was aufgrund eines statistischen Models oder der Vorstellung eines Anwenders erwartet werden konnte. Eine andere Strategie, um neue Muster zu identifizieren, besteht darin redundante Muster zu entfernen. Wenn eine entdeckte Regel durch eine andere impliziert wird, die bereits in der Datenbank oder der Menge der daraus ableitbaren Muster enthalten ist, dann sollte man diese redundante Regel entfernen. Data-Mining mit Begriffshierarchien führt oft zu einer großen Menge von redundanten Regeln. Eine Strategie zur Identifizierung redundanter Muster wird in [SA95] und [SA96] vorgestellt.

Data-Mining-Systeme sollten es dem Anwender erlauben flexibel und interaktiv Relevanzmaße und deren Grenzwerte zu definieren, zu testen und zu modifizieren.

2.5 Präsentation und Visualisierung der Ergebnisse

Um Data-Mining effektiv nutzen zu können, müssen verschiedene Möglichkeiten bestehen, um die entdeckten Muster darzustellen. Die fünfte Primitive ist daher die Auswahl des Anwenders zwischen verschiedenen Darstellungsmöglichkeiten wie zum Beispiel Regeln (Abbildung 4), Tabellen (Abbildung 5), Kreuztabellen (Abbildung 6), Kuchen- und Balkendiagrammen (Abbildung 7), Entscheidungsbäumen (Abbildung 8), Würfeln (Abbildung 8) oder anderen visuellen Präsentationsformen. In den beispielhaften Abbildungen steht Klasse A für geringes, Klasse B für mittleres und Klasse C für hohes Versicherungsrisiko.

Geschlecht(X,“w“) => Klasse(X,“A“)
Geschlecht(X,“m“) $\wedge$ Alter(X,“$\leq$35“) => Klasse(X,“C“)
Geschlecht(X,“m“) $\wedge$ Alter(X,“>35“) $\wedge$ Autotyp(X,“Coupé“) => Klasse(X,“B“)
Geschlecht(X,“m“) $\wedge$ Alter(X,“>35“) $\wedge$ Autotyp(X,“Van“) => Klasse(X,“A“)

Abbildung 4: Regeln

Geschlecht	Alter	Autotyp	Klasse	Anzahl
w	$\leq$35	Coupé	A	30291
w	$\leq$35	Van	A	248
w	>35	Coupé	A	41468
w	>35	Van	A	579
m	$\leq$35	Coupé	C	31241
m	$\leq$35	Van	C	738
m	>35	Coupé	B	45531
m	>35	Van	A	1772

Abbildung 5: Tabelle

		Geschlecht				Klasse		
		m		w				
		Autotyp		Autotyp				
		Coupé	Van	Coupé	Van	A	B	C
Alter	>35	45531	1772	41468	579	43819	45531	0
	≤35	31241	738	30291	248	30539	0	31979
Anzahl		76772	2510	71759	827	74358	45531	31979

Abbildung 6: Kreuztabelle

Abbildung 7: Kuchen- und Balkendiagramm

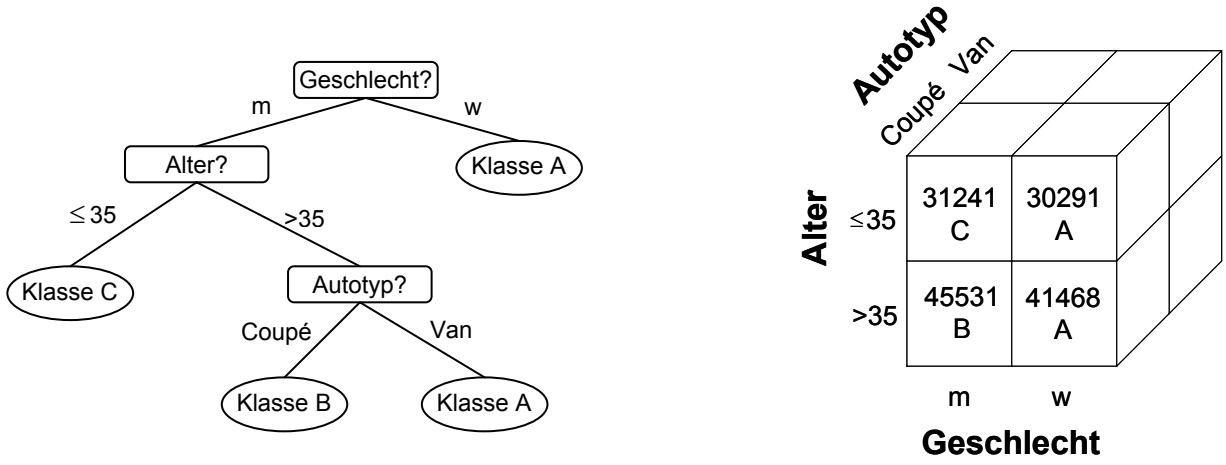

Abbildung 8: Entscheidungsbaum, Datenwürfel

Diese unterschiedlichen Präsentationsmöglichkeiten helfen Anwendern aus verschiedenen Anwendungsgebieten die relevanten Muster zu identifizieren und korrekt zu interpretieren. Die Tauglichkeit der einzelnen Präsentationsformen ist abhängig von der verwendeten Wissensform, so sind zum Beispiel Entscheidungsbäume eine gute Wahl für Klassifikationen. Die Nutzung von Begriffshierarchien spielt auch bei der Visualisierung der entdeckten Muster eine wichtige Rolle, sie erlauben die Darstellung des entdeckten Wissens auf höherer Ebene. Eine Darstellung auf höherer Ebene ist für den Anwender häufig leichter verständlich als eine Darstellung auf der Ebene der Ursprungsdaten. Mit Hilfe der Begriffshierarchien und Drill-Down- beziehungsweise Roll-Up-Operationen erhält der Anwender sogar die Möglichkeit die entdeckten Muster dynamisch auf mehreren Abstraktionsebenen zu betrachten, wobei diese Operationen jeweils ein Ab- beziehungsweise Aufsteigen in der Begriffshierarchie realisieren. Eine Verallgemeinerung beziehungsweise Spezialisierung kann aber auch ohne Drill-Down- oder Roll-Up-Operationen erreicht werden, im Falle der Verallgemeinerung durch das Entfernen von Attributen oder Dimensionen. Man stelle sich vor das Muster enthalte das Attribut Stadt, bei gegebener Hierarchie

Stadt < Bundesland < Land < Kontinent auf der Relation Ort. Durch das Entfernen des Attributs Stadt vom Muster werden die Daten auf der nächst höheren Ebene (Bundesland) dargestellt. Äquivalent dazu können Muster unter Hinzunahme von Attributen beziehungsweise Dimensionen auch auf einer spezielleren Ebene betrachtet werden. Um dem Anwender noch weitere Perspektiven auf die Ergebnisse zu geben, stehen auch die aus dem Bereich des OLAP bekannten Operationen Pivoting (oder Rotating) und Slice-and-Dice zur Verfügung. Ein Data-Mining-System sollte solche interaktiven Operationen sowohl für jede Dimension als auch für jeden einzelnen Wert einer Dimension anbieten. Zusätzlich sollten für jedes entdeckte Muster Relevanzmaße angezeigt werden, um den Anwender bei der Trennung von nützlichem und unnützem Wissen zu unterstützen.

3 Data-Mining-Anfragesprachen

Eine der Hauptforderungen an ein Data-Mining-System war die Unterstützung von ad-hoc-Anfragen und interaktivem Data-Mining. Zur Realisierung dieser Forderung bietet sich der Entwurf einer Data-Mining-Anfragesprache an. Wie wichtig die Entwicklung einer leistungsfähigen Data-Mining-Anfragesprache ist, hat uns auch die Geschichte der relationalen Datenbanken gezeigt. Relationale Datenbanken haben den Markt für Datenbanken jahrzehntelang beherrscht. Experten machen die frühe Einführung eines Standards für den großen Erfolg der relationalen Datenbanken verantwortlich. Dieser Standard stellte eine Grundlage dar, auf der relationale Datenbanken entwickelt wurden, denn obwohl jedes kommerzielle relationale Datenbanksystem über eine eigene graphische Benutzeroberfläche verfügt, liegt unter jeder dieser Oberflächen eine relationale Anfragesprache, die in den wichtigen Punkten dem Standard entspricht. Sie erleichtert den Austausch des DBMS und fördert somit die Kommerzialisierung und breite Akzeptanz von relationaler Datenbanktechnologie. Als Standard der relationalen Anfragesprachen gilt der SQL-Standard, der in der aktuellsten Version als SQL:1999 [SQL99] vorliegt. Obwohl noch kein Datenbank-System den Standard komplett erfüllt, stellt er eine gute Grundlage dar. Überträgt man diese Entwicklung nun auf die Entwicklung von Data-Mining-Systemen, so kommt die Hoffnung auf, dass die Verfügbarkeit einer guten Anfragesprache den Data-Mining-Systemen zu ähnlichem Erfolg verhelfen könnte.
Der Entwurf einer verständlichen und leistungsfähigen Data-Mining-Sprache stellt aufgrund des weiten Aufgabenspektrums von Data-Mining-Systemen eine große Herausforderung dar. Wie in Abschnitt 2.2 bereits erwähnt, gehören zu den Aufgaben eines Data-Mining-Systems unter anderen Datencharakterisierung, Datenklassifikation, Sequenzanalyse und Entdeckung von Assoziationsregeln. Aufgrund der Unterschiedlichkeit der einzelnen Funktionalitäten eines Data-Mining-Systems stellt jede Funktionalität ihre eigenen Anforderungen an eine Anfragesprache.
In Abschnitt 3.1 wird mit DMQL eine Sprache vorgestellt, die direkt aus den fünf Primitiven in Kapitel 2 hervorgegangen ist. In Abschnitt 3.2 wird dann die von Microsoft vorgeschlagene Sprache OLE DB for Data-Mining vorgestellt. Abschließend wird in Abschnitt 3.3 noch ein Überblick über weitere Sprachen gegeben.

3.1 DMQL

In Kapitel 2 wurden Primitive zur Kommunikation mit einem Data-Mining-System vorgestellt. Basierend auf diesen Primitiven wurde in [HFK+96] mit DMQL (Data-Mining-Query-Language) eine Data-Mining-Anfragesprache entworfen und in [HaKa01] weiterentwickelt. DMQL wird in einem kommerziellen System namens DBMiner [CCC+97] eingesetzt. DMQL erlaubt ad-hoc-Mining der verschiedenen Wissensarten auf Daten aus relationalen Datenquellen beziehungsweise aus Data-Warehouses. Des Weiteren erlaubt DMQL ein Mining auf verschiedenen Abstraktionsebenen. Hierzu verwendet die Sprache eine SQL-ähnliche Syntax, so dass eine Integration mit relationalen Systemen, die SQL unterstützen, vereinfacht wird. Die Syntax von DMQL ist in EBNF (Extended-Backus-Naur-Form) definiert. In EBNF bedeutet ein Ausdruck in eckigen Klammern („[]“), dass der Ausdruck an dieser Stelle gar nicht oder genau einmal auftreten kann, ein Ausdruck in geschweiften Klammern („{}“) kann an dieser Stelle entweder gar nicht oder beliebig oft auftreten. Fettgeschriebene Wörter stellen Schlüsselwörter dar. Im Anhang (Kapitel 6) ist die Syntax für DMQL im Überblick zu sehen. In den folgenden Abschnitten wird nun ausgehend von den fünf Primitiven die Syntax von DMQL im Einzelnen vorgestellt.

3.1.1 Auswahl relevanter Daten

Der erste Schritt bei der Definition einer Data-Mining-Anfrage ist die Auswahl der für die spezielle Data-Mining-Aufgabe relevanten Daten. DMQL unterstützt hierzu die Angabe von Datenbank und Tabelle beziehungsweise Data-Warehouse und Würfel als Quelle der Daten. Mit einer zu SQL praktisch identischen Syntax unterstützt DMQL die Spezifikation von Bedingungen für die weitere Einschränkung der relevanten Daten und die Möglichkeit zur Auswahl der relevanten Attribute beziehungsweise Dimensionen. Auf den so erhaltenen Daten stellt DMQL Operationen zum Ordnen und Gruppieren bereit. Die Syntax sieht im Einzelnen wie folgt aus:

- **use database** <database_name>: Die use-database-Klausel dient zur Angabe der Quelldatenbank.

- **use data warehouse** <data_warehouse_name>: Die use-data-warehouse-Klausel ermöglicht die Auswahl einer multidimensionalen Datenquelle in Form eines Data-Warehouse.

- **from** <relation(s)/cube(s)>: Die from-Klausel wählt die beteiligte Tabelle beziehungsweise den beteiligten Datenwürfel aus.

- [**where** <condition>]: Mit der optionalen where-Klausel können Bedingungen zur Selektion der Daten spezifiziert werden, wobei <condition> einen booleschen Ausdruck darstellt.

- **in relevance to** <attribute_or_dimension_list>: Die in-relevance-to-Klausel gibt die für die Untersuchung relevanten Attribute beziehungsweise Dimensionen an. Die Klausel entspricht damit einer Projektion beziehungsweise einer Select-Klausel in SQL.

- [**order by** <order_list>]: Mit der optionalen order-by-Klausel kann die Sortierreihenfolge der relevanten Daten gewählt werden. Sie dient lediglich zur Optimierung des Data-Mining-Vorgangs.

- [**group by** <grouping_list>]: Die optionale group-by-Klausel ermöglicht die Gruppierung von Daten anhand der in <grouping_list> enthaltenen Attribute beziehungsweise Dimensionen.

- [**having** <condition>]: Die having-Klausel ermöglicht die Angabe von Booleschen Ausdrücken zur Selektion der Daten auf der Ebene der Gruppen.

Die mit dem populären SQL praktisch identische Syntax vereinfacht es dem Data-Mining-System erheblich, für jede beteiligte relationale Datenbank eine entsprechende SQL-Anfrage zur Beschaffung der Quelldaten zu formulieren.

Das folgende Beispiel wählt als relevante Daten die Tupel der Studenten aus der Tabelle *Studenten* der *Uni-DB*, die noch kein Diplom haben aber schon länger als 13 Semester studieren.

```
use database Uni-DB
in relevance to S.Name, S.Geschlecht, S.Hauptfach, S.Geburtsort, S.Geburtsdatum,
   S.Wohnort, S.Notendurchschnitt, S.Semester
from Studenten S
where S.status <> 'diplomiert' and S.semester > 13
```

3.1.2 Auswahl der Art des zu erkennenden Wissens

Mit dem Mine-Knowledge-Specification-Statement wird die Art des zu erkennenden Wissens also die auszuführende Funktion ausgewählt. Die folgenden Abschnitte stellen jeweils die Syntax für die einzelnen Wissensarten wie Datencharakterisierung, Datendifferenzierung, Assoziationsregeln und Klassifikation vor. Allen Statements gemeinsam ist die optionale Klausel "**as** <pattern_name>", hiermit kann man den entdeckten Mustern einen Namen geben.

Datencharakterisierung

```
<Mine_Knowledge_Specification> ::=
      mine characteristics [as <pattern_name>]
      analyze <measure(s)>
```

Die Mine-Characteristics-Klausel wählt als Data-Mining-Funktion die Charakterisierung aus. Die analyze-Klausel spezifiziert im Falle der Datencharakterisierung Aggregatmaße, wie zum Beispiel count, sum, count% (prozentualer Anteil der relevanten Daten mit dieser speziellen Charakteristik). Diese Aggregatmaße werden für jede gefundene Datencharakteristik berechnet und ausgegeben.

Das folgende Beispiel spezifiziert als Art des zu erkennenden Wissens die Datencharakterisierung und gibt an, dass für jede angezeigte Charakteristik zusätzlich der relative Anteil der relevanten Tupel, die diese Charakterisierung tatsächlich erfüllen ausgegeben werden sollen. Die Charakteristik bekommt den Namen Langzeitstudenten.

```
mine characteristics as Langzeitstudenten
analyze count%
```

Datendifferenzierung

```
<Mine_Knowledge_Specification> ::=
      mine comparison [as <pattern_name>]
      for <target_class> where <target_condition>
      {versus <contrast_class_i> where <contrast_condition_i>}
      analyze <measure(s)>
```

Die Mine-Comparison-Klausel spezifiziert als zu entdeckendes Wissen Muster zur Datendifferenzierung. Diese Beschreibungen vergleichen eine gegebene Zielklasse von Objekten mit einer oder mehreren Kontrastklassen. Die Zielklasse wird mit "**for** <target_class> **where** <target_condition>" angeben, dabei ist <target_class> ein Name für die Zielklasse und die where-Klausel spezifiziert mittels des Booleschen-Ausdrucks <target_condition> die Daten, die zur Zielklasse gehören sollen. Mittels beliebig vielen versus-Klauseln kann man eine beliebige Anzahl von Kontrastklassen spezifizieren und zwar jeweils mit dem Namen <contrast_class_i> und dem Booleschen-Ausdruck <contrast_condition_i> als Selektionskriterium. Wie auch bei der Charakterisierung spezifiziert die analyze-Klausel Aggregatmaße wie count, sum, count%.

Diese Anfrage vergleicht die Klasse der Langzeitstudenten mit den Studenten, die keine Langzeitstudenten sind. Dabei gilt ein Student als Langzeitstudent, falls er länger als 13 Semester studiert, aber noch keinen Abschluss hat.

```
mine comparison
for Langzeitstudenten where S.status <> 'diplomiert' and S.semester > 13
versus Nicht_Langzeitstudenten where S.semester <= 13
analyze count
```

Assoziationsregeln

```
<mine_knowledge_specification> ::=
      mine associations [as <pattern_name>]
      [matching <metapattern>]
```

Die Mine-Associations-Klausel spezifiziert die Entdeckung von Assoziationsregeln. Der Anwender hat in der matching-Klausel die Möglichkeit Vorlagen (auch Metamuster, oder Metaregeln genannt) anzugeben. Vorlagen können benutzt werden, um den Entdeckungsprozess auf Muster zu beschränken, die den Vorlagen in Form syntaktischer Beschränkungen entsprechen. Dies ermöglicht dem Anwender Ahnungen oder Hypothesen als Vorlage zu formulieren und so durch diese Beschränkungen eventuell eine Bestätigung für die Ahnung oder Hypothese zu bekommen.
Das folgende Beispiel wählt als zu erkennendes Wissen Assoziationsregeln aus. Die matching-Klausel spezifiziert, dass in den gefundenen Regeln aus den Prädikaten Geschlecht, Alter und Autotyp das Prädikat Versicherungsklasse folgt. Die Variable X steht dabei für den zu Versichernden und die Variabeln G, A, AT, VR stehen für das Geschlecht, das Alter, den Autotyp und das Versicherungsrisiko des zu Versichernden. Die Regeln bekommen den Namen Versicherungseinstufung.

```
mine associations as Versicherungseinstufung
matching Geschlecht(X,G) ^ Alter(X,A) ^ Autotyp(X,AT) =>
      Versicherungsrisiko(X,VR)
```

Klassifikation

```
<Mine_Knowledge_Specification> ::=
      mine classification [as <pattern_name>]
      analyze <classifying_attribute_or_dimension >
```

Die Mine-Classification-Klausel spezifiziert, dass der Anwender Muster sucht, welche die relevanten Daten klassifizieren. Die analyze-Klausel gibt an, dass die Werte des in <classifying_attribute_or_dimension> angegebenen Attributs (beziehungsweise der Dimension) klassifiziert werden sollen. Für kategorische Attribute oder Dimensionen repräsentiert typischerweise jeder Wert eine Klasse (zum Beispiel „niedriges“, „mittleres“ oder „hohes „ Versicherungsrisiko). Für numerische Attribute oder Dimensionen wird jede Klasse durch einen Wertebereich definiert (so wie „0-800“, „801-6000“ und „6001-10000“ für monatliches Einkommen in Euro).
Das folgende Beispiel sucht nach Mustern, die die einzelnen Tupel der Kundendatenbank in die verschiedenen Versicherungsrisiko-Klassen einteilt. Dabei bekommen die Muster den Namen Risikoklassifizierung.

```
      mine classification as Risikoklassifizierung
      analyze versicherungsrisiko
```

Eine Data-Mining-Anfragesprache sollte außerdem die Spezifikation von weiteren Wissensarten unterstützen, unter anderem die zeitliche Entwicklungsanalyse (Sequenzanalyse) oder das Erkennen von Datenclustern.

3.1.3 Modellierung von Hintergrundwissen (Hierarchien)

Wie bereits in Abschnitt 2.3 beschrieben, erlauben es Begriffshierarchien, das Entdecken von Wissen auf mehreren Abstraktionsebenen ablaufen zu lassen. Hierfür kann ein Anwender mit dem Statement ”use hierarchy” entsprechend den Forderungen aus Abschnitt 2.3 eine oder mehrere Begriffshierarchien, die bereits definiert sein müssen, in die aktuelle Anfrage einbeziehen. Dabei gibt <hierarchy_name> den Namen der zu nutzenden Begriffshierarchie an und die for-Klausel spezifiziert für welches Attribut beziehungsweise welche Dimension die Hierarchie genutzt werden soll.

```
      uses hierarchy <hierarchy_name> for <attribute_or_dimension>
```

Für die Definition neuer Begriffshierarchien wird das Define-Hierarchy-Statement genutzt, wobei man mit der for-Klausel das Attribut oder die Dimension, auf der die Hierarchie definiert werden soll, angibt. Mit der on-Klausel wird die Relation beziehungsweise der Würfel, auf denen die Hierarchie definiert wird, angegeben. Die as-Klausel enthält dann die eigentliche Beschreibung der Hierarchie.

```
<Concept_Hierarchy_Definition_Statement> ::=
      define hierarchy <hierarchy_name>
      [for<attribute_or_dimension>]
      on <relation_or_cube_or_hierarchy>
      as <hierarchy_description>
```

In Abschnitt 2.3 wurden vier verschiedene Typen von Begriffshierarchien vorgestellt. Im Folgenden wird nun für die dort aufgeführten Beispiele die entsprechende Definition in DMQL betrachtet.

Schemahierarchie
Für die Schemahierarchie der Relation Ort mit der totalen Ordnung Stadt < Bundesland < Land ergibt sich

```
      define hierarchy hierarchie_ort on ort as
      [stadt, bundesland, land]
```

Hierbei ist natürlich die Ordnung der aufgelisteten Attribute wichtig.

Mengengruppierende Hierarchien
Die Definition für die in Abschnitt 2.3 aufgeführte mengengruppierende Hierarchie sieht in DMQL wie folgt aus:

```
define hierarchy hierarchie_einkommen for einkommen on angestellte as
ebene1: {niedrig, mittel, hoch} < ebene0:all
ebene2: {0…800} < ebene1:niedrig
ebene2: {801…6000} < ebene1:mittel
ebene2: {6001…10000} < ebene1:hoch
```

Aus Operationen abgeleitete Hierarchien
Die Definition für die in Abschnitt 2.3 aufgeführte aus Operationen abgeleitete Hierarchie für das Einkommen sieht in DMQL wie folgt aus:

```
define hierarchy brutto_einkommens_hierarchie for einkommen on angestellte as
{einkommens_Kategorie(1), einkommens_Kategorie(2), einkommens_Kategorie(3)}
       := cluster(default, einkommen, 3) < all(einkommen)
```

Regelbasierte Hierarchien
Die Definition für die in Abschnitt 2.3 aufgeführte regelbasierte Hierarchie sieht in DMQL wie folgt aus:

```
define hierarchy einkommens_hierarchie on angestellte as

level_1: einkommen_niedrig < level_0: all
if (1 - Steuersatz)*Einkommen ≤ 500

level_1: einkommen_mittel < level_0: all
if (1 - Steuersatz)*Einkommen > 500 and (1 - Steuersatz)*Einkommen ≤ 3500

level_1: einkommen_hoch < level_0: all
if (1 - Steuersatz)*Einkommen ≤ 15000 and (1 - Steuersatz)*Einkommen > 3500
```

3.1.4 Definition von Relevanzmaßen

Wie bereits festgestellt braucht der Anwender eine Möglichkeit, die Zahl der zurückgelieferten Muster weiter zu begrenzen; hierzu wurden in Abschnitt 2.4 Relevanzmaße definiert. Auch DMQL unterstützt die Definition von Relevanzmaßen und deren Schwellenwerten. Das with-Statement erlaubt das interaktive Setzen und Modifizieren der Relevanzmaße und ihrer Schwellenwerte.

```
with [<interest_measure_name>] threshold = <threshold_value>
```

Mit <interest_measure_name> wird der Name des anzuwendenden Relevanzmaßes angegeben. In Frage kommen zum Beispiel support, confidence, noise. Für die Verwendung eines support-Schwellenwertes von 5% sieht das das DMQL-Fragment wie folgt aus.

```
with support threshold = 0,05
```

3.1.5 Auswahl der Ergebnispräsentation und -visualisierung

Auch die letzte Primitive – die Auswahl der Präsentationsform – wird von DMQL unterstützt. Als Präsentationsformen kommen zum Beispiel Regeln, Tabellen, Kreuztabellen, Kuchen- oder Balkendiagramme, Entscheidungsbäume und Würfel in Frage. Realisiert wird diese Auswahlmöglichkeit durch das display-Statement.

```
display as <result_form>
```

Dabei kann die <result_form> eine der oben genannten Präsentations- oder Visualisierungsformen sein. Die Forderung nach interaktivem Mining verlangt, dass es dem Anwender ermöglicht wird, die entdeckten Muster auf verschiedenen Ebenen der Begriffshierarchie zu betrachten. Dieser Forderung wurde in DMQL mittels folgender Statements Rechnung getragen:

```
<Multilevel_Manipulation> ::=
        roll up on <attribute_or_dimension>
      | drill down on <attribute_or_dimension>
      | add <attribute_or_dimension>
      | drop <attribute_or_dimension>
```

Dabei realisieren die Statements Roll-Up-On und Drill-Down-On die in Abschnitt 2.5 geforderten Roll-Up- und Drill-Down-Operationen. Des Weiteren stehen mit den Statements Add und Drop die Möglichkeiten zum Hinzufügen und Entfernen von Attributen beziehungsweise Dimensionen zur Verfügung. Beachtet werden muss allerdings, dass ein hinzuzufügendes Attribut eines der Attribute sein muss, die bei der Spezifikation in der relevance-to-Klausel aufgeführt waren. Die Anwendung der Operatoren erfolgt auf den bereits entdeckten Mustern im Sinne einer schrittweisen Verfeinerung, um dynamisch die Ergebnismenge durch ein Auf- und Absteigen in den Begriffshierarchien auf verschiedenen Abstraktionsebenen betrachten zu können.

3.1.6 Beispiel

Im folgenden abschließenden Beispiel sollen für Langzeitstudenten charakteristische Werte in Bezug auf Geschlecht, Geburtsort, Hauptfach, Geburtdatum, Wohnort und Notendurchschnitt gefunden werden. Hierzu werden als Quelldaten die Tupel der Studenten, die noch kein Diplom haben, aber schon länger als 13 Semester studieren aus der Relation Studenten der Datenbank uni_db ausgewählt. Als Wissensart wird die Datencharakterisierung ausgewählt, wobei zu jeder entdeckten Charakterisierung auch der relative Anteil der relevanten Daten (count%), der die Charakterisierung erfüllt, angezeigt wird. Zusätzlich wird noch zur Reduzierung des Rauschens noise als Relevanzmaß mit einem Schwellenwert von 5% spezifiziert. Als Anzeigeform wird die Tabelle gewählt.

```
use database uni_db
use hierarchy Wohnort_hierarchie for S.Wohnort
mine characteristics as Langzeitstudenten
analyze count%
in relevance to to S.Geschlecht, S.Hauptfach, S.Geburtsort,
      S.Geburtsdatum, S.Wohnort, S.Notendurchschnitt
from Student S
where S.status <> 'diplomiert' and S.semester > 13
with noise threshold = 0.05
display as table
```

3.1.7 Fazit

DMQL wurde aufbauend auf den fünf Primitiven

- Auswahl relevanter Daten
- Auswahl der Art des zu erkennenden Wissens
- Modellierung von Hintergrundwissen (Hierarchien)
- Definition von Relevanzmaßen
- Auswahl der Ergebnispräsentation und -visualisierung

zur Kommunkation mit dem Data-Mining-System entwickelt, so dass auch alle fünf Primitive unterstützt werden. Die große Nähe zu SQL vereinfacht für den Anwender das Erlernen von DMQL und erleichtert das Formulieren einer SQL-Anfrage an relationale Datenbanken zum Erlangen der relevanten Daten. Viele wissenschaftliche Arbeiten im Bereich Data-Mining beziehen sich auf DMQL so dass die Bedeutung dieser Sprache in diesem Bereich relativ groß ist. Da es aber in nur einem kommerziellen System (DBMiner) eingesetzt wird und dieses System in der aktuellen Version E2.0 laut Anwendungsdokumentation [DBM99] den Zugriff auf DMQL für Anwender beziehungsweise Anwendungsprogramme und auch die Einbindung von Data-Mining-Algorithmen fremder Hersteller nicht unterstützt, lässt sich die Bedeutung dieser Sprache im kommerziellen Bereich als eher gering einschätzen.

3.2 OLE DB for Data Mining

OLE DB for Data Mining (DM) ist eine von Microsoft in [Mic00] vorgeschlagene OLE-DB-Erweiterung zur Unterstützung von Data-Mining-Operationen auf OLE-DB-Daten-Providern. OLE

DB for DM wird bereits im SQL Server 2000 von Microsoft unterstützt. Das Ziel dieser Spezifikation besteht darin einen Industriestandard für Data-Mining bereitzustellen, so dass verschiedene Data-Mining-Algorithmen verschiedener Hersteller in Anwendungsprogramme eingefügt werden können. Microsoft bezeichnet solche Softwarepakete, die Data-Mining-Algorithmen zur Verfügung stellen, als Data-Mining-Provider (im folgenden Provider genannt) und die Anwendungsprogramme, die diese Funktionen nutzen als Data-Mining-Consumer (im folgenden Consumer genannt). OLE DB for DM stellt eine API zwischen Consumer und Provider zur Verfügung.
OLE DB for DM führt das virtuelle Objekt Data-Mining-Model (DMM) mit darauf definierten Operationen ein. OLE DB for DM behandelt das DMM als einen speziellen Tabellentyp. Sobald Daten in diese spezielle Tabelle eingefügt werden, werden sie von einem bei der Erstellung spezifizierten DM-Algorithmus eines Providers bearbeitet und die daraus resultierenden Muster, Regeln, Formeln, Klassifikationen, Verteilungen werden anstelle der eingefügten Daten als DMM-Inhalt (DMM-Content) gespeichert. Das Füllen des DMM entspricht einem Trainieren des DMM mit den Eingabedaten als Trainingsdaten. Später kann das DMM durchsucht oder zur Ableitung von so genannten Vorhersagen (Predictions) fehlender Werte der eingefügten Daten genutzt werden. Vorhergesagt werden können zum Beispiel die Werte eines Attributes, einer Klasse als Ergebnis einer Klassifikation (zum Beispiel der Höhe des Versicherungsrisikos) oder eines Items (zum Beispiel ein bestimmtes Produkt, das im Zusammenhang mit anderen Produkten gekauft wird) als Ergebnis einer Assoziation.
Zu untersuchende Daten werden logisch als eine Menge von verschachtelten Tabellen (nested tables) einer relationalen Datenbank repräsentiert. Dabei ist die untergeordnete Tabelle jeweils als Attribut der übergeordneten Tabelle zu betrachten, zum Beispiel eine Datenbank, die aus Kunden mit ihren Bestellungen und den bestellten Artikeln (jeweils als untergeordnete Tabellen in Form eines Attributes der Kundentabelle) besteht. Alle Daten, die zu einem einzelnen Entity gehören (im Falle der Kundendatenbank ist ein Entity ein Kunde mit all seinen Bestellungen) werden Case genannt und die Menge aller relevanten Cases wird Case-Set genannt. Um diese Beziehung zu repräsentieren, benutzt OLE DB for DM die verschachtelten Tabellen des Data-Shaping-Service, der in den Microsoft Data Access Components (MDAC) Produkten enthalten ist. Dabei können für verschiedene Analysezwecke aus denselben physischen Daten verschiedene Case-Sets resultieren. Falls der Anwender zum Beispiel Muster über bestimmte Produkte entdecken will, würde er jedes Produkt als ein Entity betrachten. Dann wäre ein Produkt mit dem Attribut Kunden, die dieses Produkt gekauft haben, ein Case.
Das DMM wird mit einem Create-Mining-Model-Statement erstellt, welches in seiner Anwendung dem Create-Table-Statement von SQL ähnelt. Das Füllen des DMM erfolgt mit dem Insert-Into-Statement, wobei dieses in seiner Anwendung dem entsprechenden SQL-Statement gleicht. Beim Erstellen des DMM wird zusätzlich zur Struktur des DMM der anzuwendende Data-Mining-Algorithmus spezifiziert. Mittels eines Prediction-Joins werden dann die aktuellen Daten mit den Daten des DMM verknüpft. Auf den so verknüpften Daten können mit dem Select-Statement (auch dieses ähnelt dem entsprechenden SQL-Statement) die Vorhersagewerte ausgegeben werden. Die Ähnlichkeiten zwischen DMM's (und zugehörigen Operationen) und relationalen Tabellen (und zugehörigen SQL-Operationen) sind nicht zufällig, denn Microsoft erwartet, dass die Funktionalitäten der DM-Systeme in Zukunft komplett in relationale Datenbanken integriert werden und dass in diesem Zusammenhang das DMM in Zukunft als Bestandteil der RDBS vergleichbar mit Views der relationalen Datenbanken integriert werden wird.
Die folgenden Abschnitte zeigen nun die Erstellung, das Training, das Durchsuchen des DMM und das Abfragen der Vorhersagewerte im Detail.

3.2.1 Spezifikation von Data-Mining-Modellen

Ein neues Modell wird durch Anwendung des Create-Mining-Model-Statements erstellt, es definiert mit folgender Syntax die DMM Spalten und den Data-Mining-Algorithmus, der in diesem DMM genutzt werden soll.

```
CREATE MINING MODEL <mining model name> (<Column definitions>)
USING <Service>[(<service arguments>)]
```

Die Möglichkeit der Auswahl des Data-Mining-Algorithmus entspricht der zweiten Primitive – der Wahl des zu erkennenden Wissens. Die Syntax, um die DMM-Spalten zu definieren, ist ähnlich

der Syntax wie sie beim Create-Table-Statement genutzt wird. Da die Spalten eines DMM einige spezielle Informationen benötigen, wurden lediglich ein paar Erweiterungen zur Syntax des SQL-Standards hinzugefügt.
Spalten vom Typ Attribut und vom Typ Tabelle können Eingabespalten, Ausgabespalten oder beides sein. Der Provider baut nun ein DMM auf, das aufgrund seines Inhalts in der Lage ist, die Werte der Ausgabespalten auf Basis der Eingabespalten vorherzusagen. Als Ausgabespalten kommen nicht nur die eigentlichen Informationen, wie "geschätztes Alter ist 21" in Frage, sondern auch statistische Informationen wie Zuverlässigkeit (reliability) und Standardabweichung. Des Weiteren kann eine Vorhersage auch aus einer Menge von Tupeln bestehen, so wie "die Menge der Produkte, die ein bestimmter Käufertyp üblicherweise kauft". Eine Vorhersage kann auch als Balkendiagramm dargestellt werden. Ein Balkendiagramm bietet verschiedene mögliche Vorhersagewerte an, jedes mit einer Wahrscheinlichkeit oder anderen Statistiken belegt. Die Definition der einzelnen Spalten umfasst folgende Angaben:

- Name (obligatorisch)
- Datentyp (obligatorisch), neben Standarddatentypen wie LONG, DOUBLE, TEXT, DATE, BOOL existiert auch der spezielle Datentyp Table, um eingebettete Tabellen darstellen zu können
- Inhaltstyp (content-type) zum Beispiel: Schlüsselspalte (key), diskrete Werte (discrete) und weitere.
- Beziehung zu einer Attributspalte, angegeben durch die RELATED TO oder OF Klausel. Sie wird benutzt, um andere Attribute zu klassifizieren. Eine gegebene Beziehung muss für alle Tupel konsistent sein.
- die Schlüsselwörter PREDICT und PREDICT_ONLY, um dem Algorithmus anzuzeigen, dass diese Spalte eine Ausgabespalte ist. PREDICT zeigt an, dass diese Spalte zusätzlich auch als Eingabespalte verwendet werden kann.
- Weitere Angaben wie Not Null

Abbildung 9 zeigt ein Beispiel für das Create-Mining-Model-Statement. Es soll ein DMM namens [Age Prediction] erstellt werden, die Schlüsselspalte soll [Customer ID] die Ausgabespalte [Age] sein, des Weiteren die Spalten [Gender] und [Age Probability] (als Wahrscheinlichkeit für die Richtigkeit von [Age]) und als untergeordnete Tabelle [Product Purchases] mit den Käufen des Kunden. Die untergeordnete Tabelle soll als Schlüsselspalte [Product Name] und als weitere Spalten [Quantity] und [Product Type] haben. Da mit der RELATED-TO-Klausel eine Beziehung zwischen [PRODUCT TYPE] und [PRODUCT NAME] aufgebaut ist, müssen diese Werte immer konsistent zueinander sein, so dass das Produkt mit dem Namen Rindfleisch immer zur Produktkategorie Lebensmittel gehören muss.

```
CREATE MINING MODEL [Age Prediction]
(
    [Customer ID]         LONG      KEY,
    [Gender]              TEXT      DISCRETE,
    [Age]                 DOUBLE    DISCRETIZED() PREDICT,
    [Age Probability]     DOUBLE    PROBABILITY of [Age]
    [Product Purchases]   TABLE
    (
        [Product Name]    TEXT      KEY,
        [Quantity]        DOUBLE    NORMAL CONTINUOUS,
        [Product Type]    TEXT      DISCRETE RELATED TO [Product Name]
    )
)
USING [Decision Trees]
```

Abbildung 9: Create-Mining-Model-Statement

Zusätzlich zum Create-Mining-Model-Statement gibt es noch die Möglichkeit ein DMM mittels eines XML-Dokumentes zu definieren. Die dazugehörige Sprache heißt Predictive-Model-Markup-Language (PMML) und das dazugehörige Statement ist

```
CREATE MINING MODEL <mining model name> FROM PMML <xml string>
```

Für weitere Informationen sei in dieser Arbeit lediglich auf die DTD für PMML in [Mic00] verwiesen.

3.2.2 Bereitstellung von Trainings-Daten aus Tabellen

Nachdem nun die Struktur des DMM definiert ist, kann das Insert-Into-Statement genutzt werden, um das leere Modell mit Trainingsdaten zu füllen. Dies entspricht der ersten Primitive der Auswahl der relevanten Daten. Auch dieses Statement ähnelt in seiner Anwendung stark dem Insert-Into-Statement aus dem SQL-Standard. Das Einfügen von Trainingsdaten führt dazu, dass der Data-Mining-Algorithmus auf die eingefügten Trainingsdaten angewendet wird und den DMM-Inhalt erzeugt. Das DMM wird für gewöhnlich keine der eingefügten Daten speichern, sondern nur den DMM-Inhalt. Die Basis-Syntax für das Einfügen von Trainingsdaten ist:

```
INSERT [INTO] <mining model name>
  [<mapped model columns>]
  <source data query>
```

Wie in den folgenden Abschnitten beschrieben wird, ist die Syntax der Anfrage auf den Quelldaten (source-data-query) abhängig von der Art der Quelle. Unabhängig davon, welche Syntax benutzt wird, wird standardmäßig die Zuordnung von den Quellspalten zu den Zielspalten im DMM über die Reihenfolge gemacht. Es besteht aber auch die Möglichkeit die Zuordnung explizit über die mapped-model-columns-Klausel zu machen.
Da nicht bei jeder Syntax für die Quelldatenanfrage die Möglichkeit besteht, die benötigten Spalten auszuwählen, existiert mit dem Schlüsselwort SKIP in der Into-Klausel die Option, Spalten, die in der Formulierung der Quellanfrage enthalten sein müssen, aber für das DMM keine Bedeutung haben, zu überspringen und nicht einzubringen. Im Folgenden werden die drei wichtigsten Möglichkeiten für die Auswahl der Quelldaten vorgestellt.

Openrowset
Die meisten Data-Mining-Provider werden nicht in RDBMS integriert sein, so dass die Daten in das Data-Mining-System importiert werden müssen. Die Openrowset-Funktion unterstützt den Import von Daten mit der folgenden Syntax

```
OPENROWSET('provider_name','provider_string','query_syntax')
```

Der 'provider_name' ist dabei der OLE-DB-Provider-Name, der 'provider_string' ist der OLE-DB-Connection-String für diesen Provider und die 'query_syntax' ist eine Anfrage, die ein Rowset zurückliefert. Der DM-Provider richtet dann eine Verbindung zur Datenquelle ein und führt die in 'query_syntax' spezifizierte Anfrage aus. Ein Beispiel für die Anwendung der Openrowset-Funktion ist:

```
OPENROWSET ('SQLOLEDB', 'catalog=Sales',
     'SELECT [Customer ID], [Gender], [Age]
      FROM [Customers] ORDER BY [Customer ID] ')
```

Select-Statement
Falls der Provider Select-Anfragen nach dem SQL-Standard unterstützt, können auch diese als Quelldaten (source-data) genutzt werden.

Shape-Statement
Um geschachtelte Tabellen einzufügen, werden mehrere einzelne Anfragen mittels des Shape-Statements zu verschachtelten Tabellen zusammengefügt. Die Syntax des Shape-Statements sieht folgendermaßen aus:

```
SHAPE {<master query>}
      APPEND ({ <child table query> }
      RELATE <master column> TO <child column>)
      AS < column table name>
[
      APPEND ({ <child table query> }
      RELATE <master column> TO <child column>)
      AS < column table name>
      …
]
```

Dabei enthält <master query> die Anfrage für die übergeordnete Tabelle – zum Beispiel in Form des SQL-Statements Select – und <child table query> enthält die Anfrage für die untergeordnete Tabelle. Mit der Relate-To-Klausel werden die Spalten spezifiziert, die die untergeordnete mit der übergeordneten Tabelle verbinden. Mit der As-Klausel wird den so verbundenen Spalten ein neuer Name gegeben. Das Shape-Statement kann mit dem Openrowset-Statement verknüpft werden. Abbildung 10 zeigt ein Beispiel für das Insert-Into-Statement.

```
INSERT INTO [Age Prediction]
(
      [Customer ID], [Gender], [Age], [Age Probability],
      [Product Purchases]
)
SHAPE
{
      SELECT [Customer ID], [Gender], [Age]
      FROM [Customers]
      ORDER BY [Customer ID]
}
      APPEND
      ({
            SELECT [CustID], [Product Name], [Quantity], [Product Type]
            FROM [Sales]
            ORDER BY [CustID]}
            RELATE [Customer ID] TO [CustID]
      })
      AS [Product Purchases]
```

Abbildung 10: Insert-Into-Statement

3.2.3 Durchsuchen des Data-Mining-Model

Zusätzlich zur Möglichkeit die Spaltenstruktur des DMM anzuzeigen, besteht auch die Option, den Inhalt des DMM zu durchsuchen. Der Inhalt eines DMM ist eine Menge von Regeln, Formeln, Klassifikationen, Verteilungen, Knoten oder beliebigen anderen Informationen, die auf der Basis eines Data-Mining-Algorithmus erzeugt wurden. Der Inhalt des DMM ist abhängig von dem speziellen Algorithmus, der bei der Erstellung des DMM genutzt wurde, es kann sogar sein, dass für einen bestimmten Algorithmus keine Darstellung verfügbar ist. Das Durchsuchen des DMM-Inhalts kann einen wichtigen Einblick in die Daten geben.
Die beliebteste Art den DMM-Inhalt darzustellen ist der gerichtete Graph (als allgemeinere Form des Baums). Der Entscheidungsbaum ist ein klassisches Beispiel für einen gerichteten Graphen. Jeder Knoten im Baum hat Beziehungen zu anderen Knoten. Knoten, die keine Wurzel sind, können einen oder mehrere Vaterknoten haben und keinen oder mehrere Kinderknoten.
Die gesamte Struktur und Inhalt in Form eines gerichteten Graphen des DMM ist im Rowset MINING_MODEL_CONTENT beschrieben. Die folgende Select-Anfrage an das Data-Mining-Model liefert das Rowset mit den Knoten des Baums zurück.

```
SELECT * FROM <mining model>.CONTENT
```

Die Navigation durch einen Baum ist bis auf kleine Änderungen der OLE DB for OLAP Spezifikation entnommen, die eine große Anzahl von Navigationsoperationen unterstützt. Das Attribut NODE_RULE des Rowset MINING_MODEL_CONTENT enthält den Inhalt des DMM auch in Form einer XML-Beschreibung. Der XML-String bietet eine einfache Möglichkeit die kompletten DMM-Informationen zu erhalten, zu speichern und zu manipulieren. Mit dieser Struktur wird dem

Consumer die Möglichkeit gegeben das entdeckte Wissen auf verschiedene Arten darzustellen – ganz im Sinne der fünften Primitive.

3.2.4 Anfragen zur Vorhersage auf auszuwertenden Daten

Vorhersagen auf einem DMM erlauben es, Attribute, die eventuell in neuen Cases fehlen, vorherzusagen. Um eine solche Anfrage ausführen zu können, benötigt man ein gefülltes DMM (also ein bereits trainiertes) und die Cases mit den fehlenden Werten für die Vorhersage. Vorhersageanfragen auf einem DMM werden mittels eines Select-Statements und eines Prediction-Join durchgeführt. Die Syntax sieht folgendermaßen aus:

```
SELECT <SELECT-expressions>
FROM <mining model name> PREDICTION JOIN <source data query> ON <join condition>
[WHERE <WHERE-expression>]
```

Die source-data-query-Klausel spezifiziert eine Menge von Cases. Diese Cases enthalten die Attribute, deren Werte durch ein Verknüpfen der Cases mit dem gelernten Wissen im DMM vorhergesagt werden sollen. Die Anwendung dieser Klausel wurde bereits im Abschnitt 3.2.2 behandelt. Der Vorhersageprozess geschieht durch ein Abgleichen der aktuellen Cases aus dem <source data query> mit der Menge aller möglichen Cases aus dem DMM mit dem Namen <mining model name> mittels eines Prediction-Join. Dieser wurde eingeführt, weil die Kombination der aktuellen Cases mit allen möglichen Cases des DMM keine so einfache Semantik wie der normale SQL-Join hat. Dies kommt zum einen daher, dass das DMM nicht die eigentlichen Daten speichert und zum anderen, weil eine Möglichkeit gebraucht wird mit fehlenden Werten umzugehen. Die Select-Expressions-Klausel ist eine Menge durch Komma getrennter Ausdrücke, wobei als Ausdrücke entweder einfache Spaltenreferenzen oder berechnete Ausdrücke, wie zum Beispiel die support-Funktion (siehe Abschnitt 2.4) in Frage kommen. Mit der on-Klausel können die join-Bedingungen auch explizit durch mit “und“ verknüpfte Ausdrücke angegeben werden, falls die join-Bedingungen nicht implizit durch die Namensgebung eindeutig sind. In den Ausdrücken darf nur auf Gleichheit geprüft werden. Mit der where-Klausel unterstützt OLE DB for DM eine vereinfachte Form der vom SQL-Standard bekannten where-Klausel. Durch die <where-expression> kann man zum Beispiel spezifizieren, dass die angezeigten Vorhersagen einen bestimmten Schwellenwert für den support überschreiten müssen. So hat man die Möglichkeit die Zahl der uninteressanten aber zurückgelieferten Muster im Sinne der vierten Primitive zu beschränken.

```
SELECT
     T1.[Customer ID], M1.[Age]
FROM
     [Age Prediction] AS M1 PREDICTION JOIN
     (
     SHAPE
     {
             SELECT [Customer ID], [Gender],
             FROM [Customers]
             ORDER BY [Customer ID]
     }
     APPEND
     ({
             SELECT [CustID], [Product Name], [Quantity]
             FROM [Sales]
             ORDER BY [CustID]
     }
     RELATE [Customer ID] TO [CustID] AS [Product Purchases]
     ) AS T1
ON
     M1.[Gender] = T1.[Gender] AND
     M1.[Product Purchases].[Product Name] = T1.[Product Purchases].[Product Name] AND
     M1.[Product Purchases].[Quantity] = T1.[Product Purchases].[Quantity]

WHERE PredictProbability(M1.Age) > 0.8
```

Abbildung 11: Beispielanfrage

Abbildung 11 zeigt eine Beispielanfrage, in der mittels Prediction-Join das DMM [Age Prediction] mit den Quelldaten, deren Ausgabespalte [Age] keinen Wert enthält, verknüpft wird. Die Quellda-

ten werden, da sie aus verschachtelten Tabellen bestehen, mittels eines Shape-Statements spezifiziert. Die On-Klausel enthält dabei die Join-Bedingungen und spezifiziert somit die Spalten, die zur Vorhersage genutzt werden sollen. Beim Verknüpfen wird mittels des Data-Mining-Algorithmus und dem trainierten DMM der Wert für das Alter [Age] vorhergesagt. Mittels Select-Klausel werden als anzuzeigende Spalten die Spalten T1.[Customer ID] und M1.[Age] ausgewählt.

3.2.5 Fazit

Der Microsoft SQL Server 2000 bietet mit OLE DB for DM einen skalierbaren Ansatz für das Einfügen von Data-Mining-Funktionalität (in Form von Data-Mining-Providern) in das Datenbanksystem. Unterstützt wird durch OLE DB for DM die erste Primitive zur Auswahl der relevanten Daten, die zweite Primitive zur Auswahl der Art des zu erkennenden Wissens, die vierte Primitive zur Definition von Relevanzmaßen und die fünfte Primitive der Auswahl der Darstellungsmöglichkeit. Auch hier wurde durch die Nähe zu SQL eine Sprache entwickelt die leicht zu erlernen und leicht in bestehende Datenbanken zu integrieren ist.

3.3 Gegenüberstellung DMQL und OLE DB for Data Mining

Nachdem DMQL und OLE DB for DM als Beispiele für Data-Mining-Anfragesprachen etwas genauer vorgestellt wurden, sollen in einer kleinen Gegenüberstellung (Abbildung 12) die Vor- und Nachteile der einzelnen Sprachen aufgezeigt werden.

	DMQL	OLE DB for DM
Vorteile	▪ unterstützt alle fünf Primitive ▪ orientiert sich am SQL-Standard	▪ unterstützt bis auf die Definition von Hintergrundwissen alle Primitive ▪ ermöglicht die Auswahl beliebiger Data-Mining-Algorithmen verschiedener Hersteller ▪ Orientiert sich am SQL-Standard
Nachteile	▪ es existiert kein System, das den Zugriff auf und die Nutzung von DMQL zulässt ▪ ermöglicht nicht die Auswahl beliebiger Data-Mining-Algorithmen verschiedener Hersteller	▪ unterstützt nicht die Definition von Hintergrundwissen

Abbildung 12: Gegenüberstellung DMQL und OLE DB for Data Mining

3.4 Andere Sprachen

Neben DMQL und OLE DB for DM gibt es noch weitere Bemühungen Data-Mining-Sprachen zu entwickeln beziehungsweise zu standardisieren. Im Folgenden werden ein paar Beispiele angeführt.

MSQL ist eine Data-Mining-Anfragesprache die von Imielinski und Virmani in [IV99] vorgeschlagen wurde. Auch MSQL nutzt eine SQL-ähnliche Syntax und unterstützt Sortierung und Group-By. Da eine enorme Zahl von Regeln durch die Data-Mining-Algorithmen entdeckt werden können, stellt MSQL Primitive wie GetRules und SelectRules für Regelentdeckung und Regelselektion bereit.

In [TUA+98] schlagen Tsur, Ullman, Abitboul, Clifton und andere eine Sprache vor, die die Datalog-Syntax benutzt, um Anfragen zu formulieren. Damit wird sowohl der eigentliche Data-Mining-Prozess als auch das Testen von Regeln vereinfacht.

Von H. Wang und C. Zaniolo [WaZa03] wurde mit ATLaS der Vorschlag gemacht SQL mit minimalen Erweiterungen in eine Data-Mining-Anfragesprache zu verwandeln Hierzu wurden neue

Tabellenfunktionen und benutzerdefinierte Aggregatfunktionen (User-Defined-Aggregates, UDA) eingeführt, die es ermöglichen Data-Mining-Funktionalitäten zu den Datenbanksystemen hinzuzufügen. Der Vorteil dieses Ansatzes ist, dass er prinzipiell auf SQL aufbaut, der Nachteil ist, dass die UDAs nur in dem Entwurf von SQL-3 [SQL96] enthalten waren, aber nicht in SQL:1999 [SQL99] unterstützt werden und somit nicht durch alle SQL:1999-Systeme unterstützt werden.
CRISP-DM (CRoss-Industry Standard Process for Data Mining) ist ein Versuch Data-Mining-Anfragesprachen zu standardisieren (siehe hierzu [CCK+00]). Dieser Ansatz betrachtet Data-Mining weniger von der technischen Seite, sondern mehr von der Seite der Anwender und ihren Problemen aus der Geschäftswelt, bei deren Lösung Data-Mining helfen soll. CRISP-DM ist ein internationales Projekt, bestehend aus Firmen, die im Bereich Data-Warehouses und Data-Mining tätig sind. Sie arbeiten, daran eine Plattform und einen strukturierten Prozess für effektives Data-Mining zu bieten. Das Projekt definiert einen Data-Mining-Prozess, der allgemein in verschiedenen Industriezweigen anwendbar ist. Er behandelt Dinge wie

- Abbildung von Problemen aus der Geschäftswelt auf Data-Mining-Probleme
- Erfassen und Verstehen von Daten
- Identifizieren und Lösen von Problemen innerhalb der Daten
- Anwenden von Data-Mining-Techniken
- Interpretieren von Data-Mining-Ergebnissen im wirtschaftlichen Kontext
- Anwendung und Wartung der Data-Mining-Ergebnisse
- Erlangen und Transfer von Expertenwissen, um bei zukünftigen Projekten davon profitieren zu können

Das Projekt bietet einen strukturierten Prozess, um Data-Mining durchzuführen und bietet einen Leitfaden bei potenziellen Problemen, die in Data-Mining-Projekten auftreten können.

4 Architekturen von Data-Mining-Systemen

Wie bei den meisten Softwaresystemen ist die Architektur beim Entwurf eines Data-Mining-Systems von großer Bedeutung. Eine gute Architektur erleichtert es dem System, Data-Mining-Aufgaben effizient und schnell durchzuführen, Informationen zur Zusammenarbeit mit anderen Systemen auszutauschen und flexibel in Bezug auf die Wünsche der Anwender zu sein.

4.1 Klassifizierung von Data-Mining-Architekturen

In den letzten Jahrzehnten wurden durch Forschung und Entwicklung im Bereich der Informationsverarbeitung Datenbanken und Data-Warehouses (DB/DW) zu den bedeutendsten Informationssystemen. Gewaltige Datenmengen sind bereits in diesen Systemen gespeichert. Darüber hinaus existieren im Zusammenhang mit solchen Systemen viele Werkzeuge zur Datenanalyse und Informationsverarbeitung. Dazu gehören Werkzeuge für den (Web-)Zugriff, die Integration verschiedener, heterogener Datenquellen, Reportfunktionen und OLAP-Analysewerkzeuge. Aufgrund des Fortschritts dieser Systeme bietet es sich an DM-Systeme, die sich erst in der Anfangsphase ihrer Entwicklung befinden, in solche Systeme zu integrieren. Zur Betrachtung der Frage, wie stark diese Integration sein sollte, werden in [HaKa01] verschiedene Ansätze von überhaupt keiner Integration bis hin zur vollen Integration vorgestellt.

Keine Kopplung (No coupling)
Keine Kopplung bedeutet, dass ein DM System vollkommen unabhängig von DB/DW-Systemen ist, das heißt es besitzt ein eigenes Speichermanagement und verwaltet die für das Data-Mining benötigten Daten ebenfalls unabhängig zum Beispiel mit der Hilfe eines Dateisystems. Ein solcher Ansatz hat sowohl Vor- als auch Nachteile. Ein Vorteil ist, dass durch die eigene Speicherverwaltung die Möglichkeit besteht, die Algorithmen zur Speicherverwaltung in Bezug auf die gerade ausgeführte Data-Mining-Aufgabe zu optimieren. Ein weiterer Vorteil ist, dass ein solches System sehr einfach zu realisieren ist. Zu den Nachteilen zählt, dass man die Effizienz und die Flexibilität, die DB/DW-Systeme durch getestete, gut skalierbare Algorithmen und Datenstrukturen beim Speichern, Zugreifen, Organisieren und Verarbeiten besitzen, nicht nutzt. Darüber hinaus kann man weitere Funktionalitäten wie Logging und Recovery für die Unterstützung dauerhafter Speicherung und die Unterstützung der DB/DW-Systeme für hochgradig nebenläufige Zugriffe nicht nutzen. Des Weiteren sind in DB/DW-Systemen bereits Daten von hoher Qualität vorhanden; ohne die Nutzung dieser Systeme müsste sich das DM-System selbst um die Extraktion und Aufbereitung der Daten kümmern. Zuletzt ist ganz allgemein die Integration eines DM-Systems in die vorhandenen informationsverarbeitenden Strukturen ohne jegliche Kopplung aufwendiger. Ein Ansatz ohne Kopplung ist daher nicht sehr vorteilhaft.

Lose Kopplung (Loose Coupling)
Bei der losen Kopplung nutzt das DM-System die DB/DW-Systeme zum Zugriff auf die Daten, verarbeitet dann die Daten mit den DM-Algorithmen und legt die Ergebnisse wieder in DB/DW-Systemen ab. Lose Kopplung ist besser als überhaupt keine Kopplung, da es die Vorteile der DB/DW-Systeme beim Zugriff auf die Daten nutzt. Dies führt zur Verbesserung der Flexibilität und Effizienz des DM-Systems. Da der Data-Mining-Vorgang an sich nicht die vorhandenen Datenstrukturen und Anfrageoptimierung der DB/DW-System nutzt, findet dieser meistens im Hauptspeicher statt. Daher ist es für solche Systeme schwierig, hohe Skalierbarkeit und gute Performanz bei großen Datenmengen zu erreichen.

Mittelstarke Kopplung (Semitight coupling)
Bei der mittelstarken Kopplung können neben der bloßen Nutzung der DB/DW als Datenquelle auch effiziente Implementierungen einiger wesentlicher DM-Primitive durch das DB/DW-System angeboten werden. Beispiele für solche Primitive sind Indexierung, Mehrwege-Joins, Sortierung, Aggregation und Vorausberechnung einiger wesentlicher statistischer Werte (zum Beispiel min, max, sum, count und Standardabweichung). Darüber hinaus können auch Zwischenergebnisse der Data-Mining-Algorithmen in DB/DW-Systemen abgelegt und wieder verwendet werden. Dies führt zu einer höheren Skalierbarkeit und Performanz bei großen Datenmengen.

Starke Kopplung (tight coupling)
Bei der starken Kopplung ist das DM-System komplett in das DB/DW-System integriert und stellt somit eine funktionale Komponente des Informationssystems dar. Das heißt die komplette Anfrageverarbeitung findet im DB/DW-System statt. Wie bei einer DB-Anfrage wird die Anfrage dann auf der Basis der vorhandenen Datenstrukturen und Indexe optimiert. Die Vorteile dieses Ansatzes sind, dass das DM-System alle Vorteile der DB/DW-Systeme in vollem Umfang nutzen kann und dass dieser Ansatz eine einheitliche informationsverarbeitende Umgebung bietet. Aufgrund dieser Vorteile ist dieser Ansatz sehr wünschenswert. Der Nachteil dieses Ansatzes ist, dass die Implementierung sehr komplex ist und dass die aktuellen Anfrageoptimierer der DB/DW-Systeme die komplexen DM-Anfragen nicht optimal verarbeiten, so dass unter Umständen ein externer Optimierer nötig ist [NeTs99]. Einen guten Kompromiss stellt daher die mittelstarke Kopplung dar.

4.2 Beispiel für eine stark gekoppelte Architektur

Als Stellvertreter für eine stark gekoppelte Architektur wird in Abbildung 13 die Multikomponenten Architektur von Matheus et al. vorgestellt (zu finden in [EKS97]). Eine vergleichbare Architektur ist in [Haka01] zu finden und wird in einem System namens DBMiner [CCC+97] verwendet.

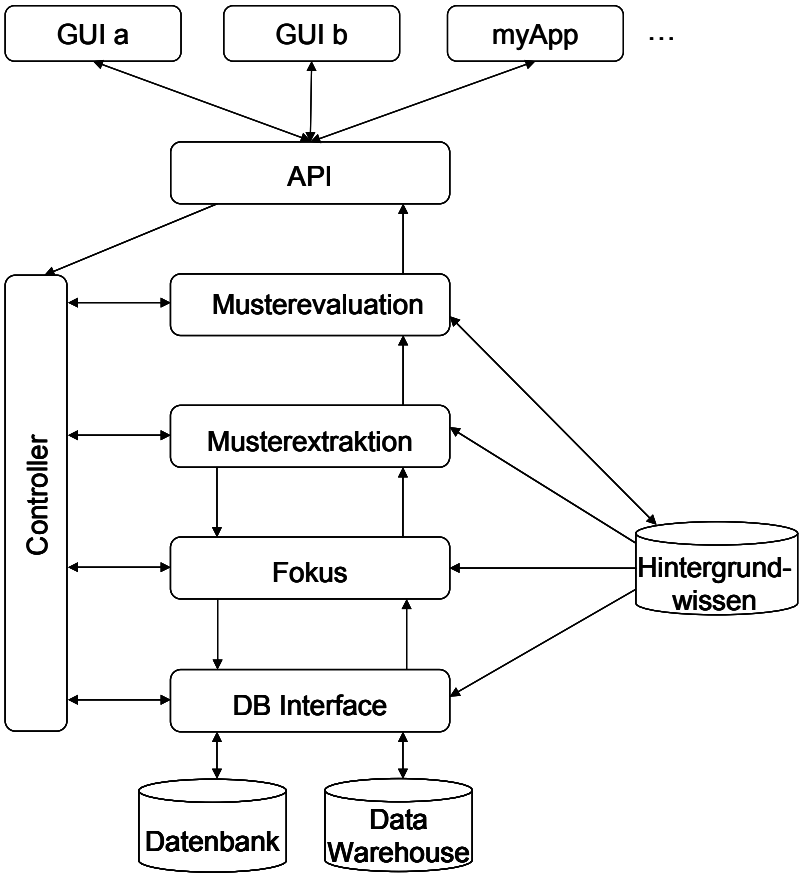

Abbildung 13: Stark gekoppelte Architektur

Die einzelnen Komponenten werden im Folgenden vorgestellt:

Datenbank, Data Warehouse
Eine oder mehrere Datenbanken oder Data Warehouses, aus denen die Quelldaten, die das potentielle Wissen enthalten, stammen.

DB Interface
Das DB Interface ermöglicht den Zugriff auf die Datenbanken und die Optimierung der Anfragen. Es unterstützt auch mehrdimensionale Indizes so wie R-Bäume oder Kd-Bäume.

Fokus
Die Fokus-Komponente ermöglicht die Auswahl der relevanten Daten und Attribute beziehungsweise Dimensionen. Hierbei ist es auch wichtig, dass die Komponente den Anwender bei der Auswahl der für eine bestimme Data-Mining-Funktion sinnvollen und wichtigen Daten und Attribute/Dimensionen unterstützt.

Musterextraktion
Das eigentliche Entdecken der verschiedenen Wissensarten findet in der Komponente der Musterextraktion statt. Diese Komponente ist aufgeteilt in Subkomponenten für die verschiedenen Wissensarten wie zum Beispiel Assoziation, Charakterisierung, Clusterbildung, Abweichungsanalyse und so weiter.

Musterevaluation
In diesem Modul werden die interessanten Muster von den uninteressanten und redundanten Mustern getrennt. Hierzu werden Relevanzmaße und deren Schwellenwerte auf die Muster angewandt. Abhängig von der konkreten Implementierung können die Module für die Musterevaluation und Musterextraktion zu einem Modul zusammengefasst werden oder aber die beiden Module sind getrennt, arbeiten aber eng zusammen. Wichtig ist nur, dass die uninteressanten Muster möglichst früh aussortiert werden, um die Performanz zu verbessern.

Hintergrundwissen
Hier wird eine Datenbasis für das Hintergrundwissen zur Verfügung gestellt, welches nötig ist um die Suche nach Mustern zu leiten beziehungsweise die gefundenen Muster zu bewerten. Zu den Arten von Hintergrundwissen zählen zum Beispiel die Begriffshierarchien, die es ermöglichen, die Quelldaten auf verschiedenen Abstraktionsebenen zu betrachten. Zusätzlich können noch jegliche Arten von Metainformationen hier abgelegt werden, zum Beispiel Schwellenwerte, Metamuster, Metadaten zum Zugriff auf heterogene Datenquellen und weitere.

Controller
Die Controllerkomponente übernimmt die Steuerung der restlichen Komponenten und leitet die Interaktion zwischen ihnen. Über die GUI und die Controllerkomponente hat der Anwender jederzeit die Möglichkeit interaktiv in den Data-Mining-Prozess einzugreifen.

Application-Programming-Interface (API)
Über das Application-Programming-Interface besteht die Möglichkeit zur Kommunikation mit dem Data-Mining-System. Hier kann zum Beispiel eine Data-Mining-Anfrage an das System gestellt werden.

Graphical User Interface (GUI)
Über diese Komponente kommuniziert der Anwender mit dem Data-Mining-System. Hier kann der Anwender also eine konkrete Data-Mining-Anfrage stellen. Die GUI sollte dann mit Hilfe des Controllers über das API den Anwender durch den ganzen Data-Mining-Vorgang leiten. Dazu gehört die Auswahl der relevanten Daten, die Eingabe von Hintergrundwissen, die Auswahl von Relevanzmaßen und ihren Schwellenwerten und zuletzt sollte die GUI die Präsentation und das Durchsuchen des entdeckten Wissens unterstützen. Wie jede GUI sollte sie den Anwender bei allen auftauchenden Fragen unterstützen.

5 Zusammenfassung

In Kapitel 2 dieser Arbeit wurden die fünf Data-Mining-Primitive zur Kommunikation des Anwenders mit dem Datenbanksystem vorgestellt. Ziel dieser Primitive ist es, eine Data-Mining-Anfrage zur Spezifikation einer bestimmten Data-Mining-Aufgabe zu formulieren. Die fünf Primitive sind:

Auswahl relevanter Daten
Da der Anwender bei einem bestimmten Problem nur an einem Teil der Daten interessiert ist, kann er mittels dieser Primitive den Teil der vorhandenen Daten spezifizieren, auf den der Data-Mining-Algorithmus angewendet werden soll.

Auswahl des zu erkennenden Wissens
Diese Primitive erlaubt dem Anwender durch die Spezifikation der zu erkennenden Wissensart, die vom System entdeckten Muster auf die für ihn interessanten Muster zu beschränken. Zu diesen Wissensarten gehören unter anderem Begriffsbeschreibungen (Datencharakterisierung, Datendifferenzierungen), Assoziationsregeln, Klassifikation, Vorhersage, Clusterbildung und zeitliche Entwicklungsanalyse (Sequenzanalyse).

Modellierung von Hintergrundwissen
Mit der Modellierung von Hintergrundwissen kann der Anwender das Data-Mining-System bei der Suche nach Mustern unterstützen und erlaubt es die bereits entdeckten Muster zu bewerten. Diese Arbeit hat sich dabei auf eine einfache aber sehr leistungsfähige Art des Hintergrundwissens – die Begriffshierarchien – beschränkt. Begriffshierarchien erlauben es den Entdeckungsprozess auf verschiedenen Abstraktionsebenen ablaufen zu lassen.

Definition von Relevanzmaßen
Um die Zahl der entdeckten aber für den Anwender uninteressanten Muster weiter zu reduzieren, ist es möglich Relevanzmaße zu Bewertung der entdeckten Muster zu definieren. Durch die Angabe dazugehörender Schwellenwerte werden Muster, die diesen Schwellenwert nicht erreichen, dem Anwender erst gar nicht angezeigt.

Auswahl der Ergebnispräsentation und -visualisierung
Um Data-Mining effektiv nutzen zu können ist es notwendig die entdeckten Muster in einer für den Anwender möglichst verständlichen und intuitiven Form darzustellen. Die fünfte Primitive ermöglicht daher die Auswahl der Darstellungsform. Beispiele hierfür sind Regeln, (Kreuz-)Tabellen, Kuchen- und Balkendiagramme, Entscheidungsbäume und Datenwürfel. Des Weiteren wird es dem Anwender mit Hilfe der Begriffshierarchien ermöglicht, die entdeckten Muster auf verschiedenen Abstraktionsebenen zu betrachten.

Kapitel 3 hat sich zuerst mit der Notwendigkeit einer standardisierten Anfragesprache für Data-Mining beschäftigt. Anschließend wurde mit DMQL eine Anfragesprache vorgestellt, die direkt aus den in Kapitel 2 vorgestellten fünf Primitiven hervorgegangen ist. DMQL wird zwar in einem kommerziellen System namens DBMiner eingesetzt, der Anwender selbst hat aber keinen Zugriff auf DMQL, was wiederum die Vorteile im Sinne einer Standardisierung zunichte macht. Als zweite Sprache wurde OLE DB for Data Mining vorgestellt. Sie soll die Möglichkeit unterstützen, Data-Mining-Algorithmen verschiedener Hersteller in verschiedene Anwendungsprogramme einzubinden. OLE DB for Data Mining wird bereits im SQL Server 2000 von Microsoft angeboten. Unterstützt wird durch OLE DB for DM die erste Primitive zur Auswahl der relevanten Daten, die zweite Primitive zur Auswahl der Art des zu erkennenden Wissens, die vierte Primitive zur Definition von Relevanzmaßen und die fünfte Primitive der Auswahl der Darstellungsmöglichkeit.
Kapitel 4 hat sich schließlich mit Architekturen von Data-Mining-Systemen und deren Integration in Datenbanken beziehungsweise Data-Warehouses beschäftigt. Der Grad der Integration wird in vier Klassen unterteilt, nämlich:

Keine Kopplung
Dies bedeutet, dass keinerlei Verbindung zwischen dem Data-Mining-System und der Datenbank beziehungsweise dem Data-Warehouse besteht. Wodurch die Vorteile dieser Systeme nicht genutzt werden können.

Lose Kopplung
Bei diesem Ansatz wird die Datenbank beziehungsweise das Data-Warehouse nur zum Zugriff auf die Daten genutzt. Die Datenbank-/Data-Warehouse-Systeme besitzen dabei keinerlei Data-Mining-Funktionalität oder Optimierungsansätze in Bezug auf den Entdeckungsvorgang.

Mittelstarke Kopplung
Bei der mittelstarken Kopplung werden wichtige DM-Primitive wie Indexierung, Mehrwege-Joins, Sortierung und andere direkt in der Datenbank beziehungsweise im Data-Warehouse implementiert, um eine höhere Skalierbarkeit zu erreichen.

Starke Kopplung
Bei der starken Kopplung stellt das Data-Mining-System eine funktionale Komponente eines Datenbanksystems dar; es ist demzufolge komplett integriert. Im Sinne einer einheitlichen informationsverarbeitenden Umgebung ist dieser Ansatz sehr wünschenswert.

Abschließend wurde dann im zweiten Teil des vierten Kapitels eine stark gekoppelte Architektur mit all ihren Komponenten vorgestellt.

Schließlich bleibt noch zu bemerken, dass die relativ junge Disziplin des Data-Mining wahrscheinlich noch eine weite Entwicklung vor sich hat. Es bleibt dabei abzuwarten, ob sich im Rahmen dieser Entwicklung eine Data-Mining-Anfragesprache in Zukunft als Standard durchsetzen wird und wenn ja welche?
In Anbetracht der ständig wachsenden Rolle von Data-Mining im betriebswirtschaftlichen Umfeld scheint es wahrscheinlich, dass in Zukunft Data-Mining-Funktionalität direkt in die Datenbanksysteme der großen Hersteller integriert werden wird. Dies würde dem Anwender die Möglichkeit geben ohne großen Aufwand Data-Mining-Funktionalitäten auf seinen vorhandenen Datenbeständen nutzen können.

6 Anhang

Dieser Anhang gibt einen Überblick über die DMQL-Syntax.

```
<DMQL> ::=
       <DMQL_Statement>;{<DMQL_Statement>}

<DMQL_Statement> ::=
         <Data_Mining_Statement>
       | <Concept_Hierarchy_Definition_Statement>
       | <Visulization_and_Presentation>

<Data_Mining_Statement> ::=
       use database <database_name>
       | use data warehouse <data_warehouse_name>
       {use hierarchy <hierarchy_name> for <attribute_or_dimension>}
       <Mine_Knowledge_Specification>
       in relevance to <attribute_or_dimension_list>
       from <relation(s)/cube(s)>
       [where <condition>]
       [order by <order_list>]
       [group by <grouping_list>]
       [having <condition>]
       [with [<interest_measure_name>] threshold=<threshold_value>
       [for <attribute(s)>]}

<Mine_Knowledge_Specification> ::=
       <Mine_Char> | <Mine_Discr> | <Mine_Assoc> | <Mine_Class>

<Mine_Char> ::=
       mine characteristics [as <pattern_name>]
       analyze <measure(s)>

<Mine_Discr> ::=
       mine comparison [as <pattern_name>]
       for <target_class> where <target_condition>
       {versus <contrast_class_i> where <contrast_condition_i>}
       analyze <measure(s)>

<Mine_Assoc> ::=
       mine associations [as <pattern_name>]
       [matching <metapattern>]

<Mine_Class> ::=
       mine classification [as <pattern_name>]
       analyze <classifying_attribute_or_dimension>

<Concept_Hierarchy_Definition_Statement> ::=
       define hierarchy <hierarchy_name>
       [for<attribute_or_dimension>]
       on <relation_or_cube_or_hierarchy>
       as <hierarchy_description>
       [where <condition>]

<Visualization_and_Presentation> ::=
       display as <result_form> | {<Multilevel_Manipulation>}

<Multilevel_Manipulation> ::=
         roll up on <attribute_or_dimension>
       | drill down on <attribute_or_dimension>
       | add <attribute_or_dimension>
       | drop <attribute_or_dimension>
```

Referenzen

[AIS93] R. Agrawal, T. Imielinski und A. Swami:
Database mining: A performance perspective, IEEE Trans. Knowledge and Data Engineering, 5:914-925,
1993

[CCC+97] J. Han, J.Y. Chiang, S. Chee, J. Chen, Q. Chen, S. Cheng, W. Gong, M. Kamber, K. Koperski, G. Lu, N. Stefanovic, L. Winstone, B. B. Xia, O. R. Zaiane, S. Zhang, H. Zhu:
DBMiner: A System for Data Mining in Relational Databases and Data Warehouses,
1997

[CCK+00] Pete Chapman (NCR), Julian Clinton (SPSS), Randy Kerber (NCR), Thomas Khabaza (SPSS), Thomas Reinartz (DaimlerChrysler), Colin Shearer (SPSS) und Rüdiger Wirth (DaimlerChrysler):
CRISP-DM 1.0, Step-by-step data mining guide,
2000
(Elektronisch verfügbar unter:
http://www.crisp-dm.org/CRISPWP-0800.pdf)

[DBM99] o.V.:
DBMiner 2.0 (Enterprise), User Manual,
1999
elektronisch verfügbar unter:
www.dbminer.com (abgerufen am 01.01.2004)

[EKS97] M. Ester, H.-P. Kriegel, und J. Sander:
Spatial Data Mining: A Database Approach. In Proc. int. Symp. on Large Spatial Databases (SSD'97), Berlin, Deutschland, Seiten 47-66

1997

[HaKa01] Han, J., Kamber, M:
Data Mining - Concepts and Techniques, Seiten 146 – 159, Morgan Kaufmann Publishers,
2001,

[HFK+96] J. Han, Y. Fu, K. Koperski, W. Wang and O. Zaiane:
DMQL: A Data Mining Query Language for Relational Databases, in Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery, Seiten 27-34, Montreal, QB,
1996,

[IV99] T. Imielinski und A. Virmani:
MSQL: A query language for database mining. Data Mining and Knowledge Discovery, 3:373-408,
1999

[LHC97] B. Liu, W. Hsu, und S. Chen:
Using general impressions to analyze discovered classification rules. *In Proc. 1997 Int. Conf. Knowledge Discovery and Data Mining (KDD'97), Newport Beach, CA,*
1997

[Mic00] o.V.:
OLE DB for Data Mining Specification, Version 1, Microsoft Corporation,
2000
(Elektronisch verfügbar unter:
http://download.microsoft.com/download/dasdk/Install/1/WIN98Me/EN-US/oledbdm.exe)

[Mic83] R. S. Michalski:
A theory and methodology of inductive learning. In Michalski et al., editors, Machine Learning: An Artificial Intelligence Approach, Vol. 1, Seiten 83 – 134. San Mateo, CA: Morgan Kaufmann
1983

[MPSM96] C. J. Matheus, G. Piatetsky-Shapiro, and D. MCNeil:
Selecting and reporting what is interesting: The KEFIR application to healthcare data. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, Seiten 495 – 516. Cambridge, MA:AAAI/MIT Press,
1996

[NeTs99] S. Nestorov und S. Tsur:
Integrating Data Mining with Relational DBMS : A Tightly-Coupled Approach. *Next Generation Information Technologies and Systems , 4th International Workshop, NGITS'99 Zikhron-Yaakov, Israel,*
1999

[PSM94] G. Piatesky-Shapiro und C.J. Matheus:
The interestingness of deviations*. In Proc. AAAI'94 Workshop Knowledge Discovery in Databases (KDD'94), Seattle, WA,*
1994

[SA95] R. Srikant und R. Agrawal:
Mining generalized association rules. In Proc. *1995 Int. Conf. Very Large Data Bases (VLDB`95), Zürich, Schweiz,*
1995

[SA96] R. Srikant und R. Agrawal.:
Mining quantitive association rules in large relational tables. *In Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96), Montreal, Canada,*
1996

[ST96] A. Silberschatz und A. Tuzhilin:
What makes patterns interesting in knowledge discovery systems. IEEE Trans. Knowledge and Data Engineering, 8:970-974,
1996

[SQL96] ISO/IEC JTC1/SC21 N10489, ISO//IEC 9075:
Committee Draft (CD), Database Language SQL,
1996.

[SQL99] o.V.: *ANSI/ISO/IEC 9075-2:1999*
Information technology - Database languages - SQL - Part 2: Foundation (SQL/Foundation) ISO,
1999

[TUA+98] D. Tsur, J.D. Ullman, S. Abitoul, C.Clifton, R. Motwani und S. Nestorov:
Query flocks: A generalization of association-rule mining. In Proc. 1998 ACM-Sigmod Int. Conf. Management of Data (SIGMOD'98), Seiten 1-12, Seatle, WA,
1998

[WaZa03] H. Wang, C. Zaniolo:
ATLaS: A Native Extension of SQL for Data Mining, *Second SIAM International Conference on Data Mining (SDM'03), San Francisco, 2003*