fhwedel
UNIVERSITY OF APPLIED SCIENCES

XML und Datenbanken

Simon Temp
Hamburg, 30 Nov. 2005

Vergleich von XML und relationalen Datenbanken

Integrationsarchitektur für XML und relationaler Datenbanken

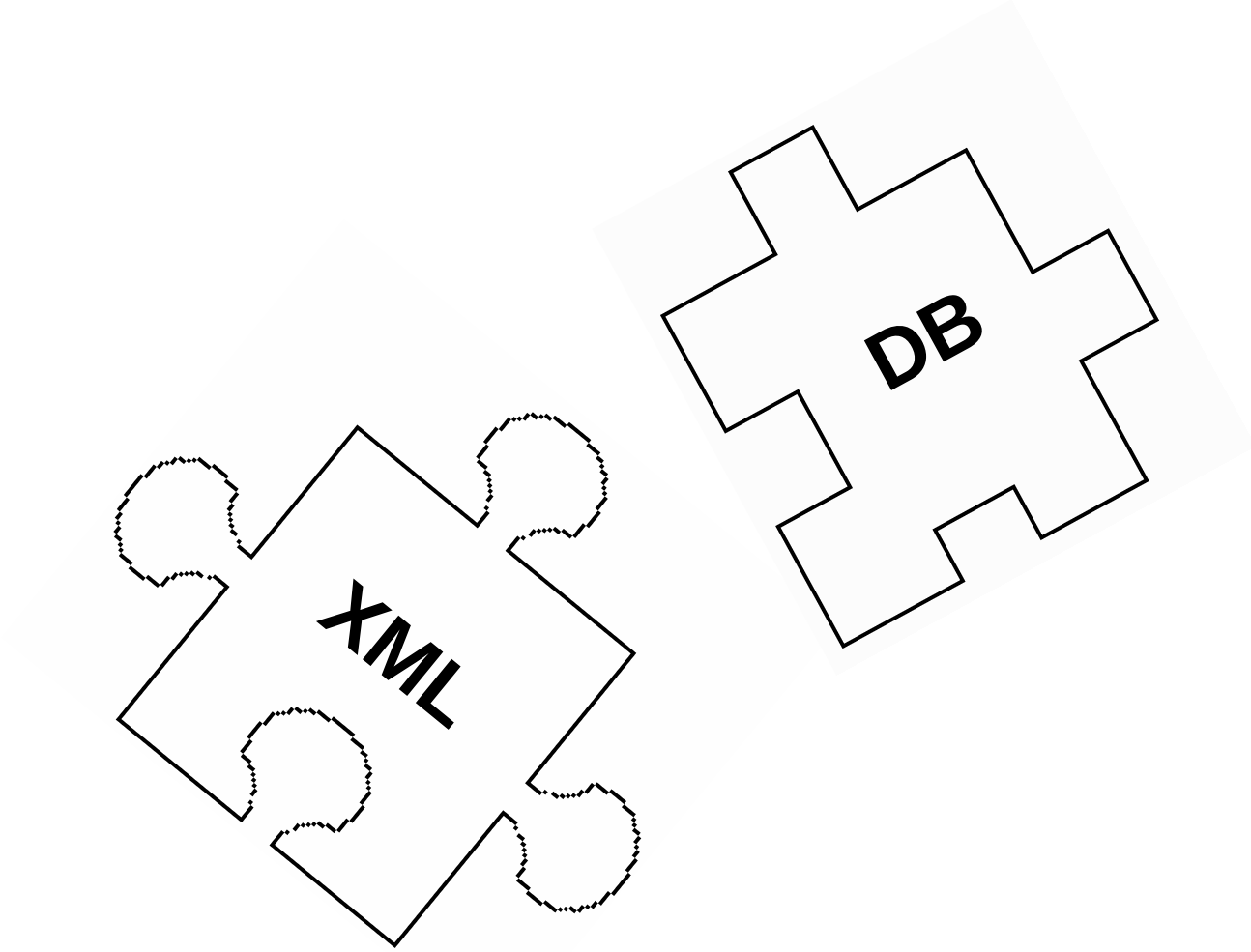
Integrationsstrategie von XML mit relationalen Datenbanken

„Mapping“ von XML und relationalen Daten

Herstellerunterstützung von XML in Datenbanken

Reine XML-Datenbanken

Vergleich von XML und relationalen Datenbanken



fhwedel
UNIVERSITY OF APPLIED SCIENCES

Speicherung und Sicherheit

	Datenbank	XML
Speicherung	Stark kontrollierte Speicherumgebung.	Erfolgt ausschließlich als normaler Text.
Sicherheit	Eigenes Sicherheitssystem oder eines integriert in das Betriebssystem. Beinhaltet die Kontrolle über die Daten und ihrer Struktur.	Kein integriertes Sicherheitssystem. Die Zugriffsberechtigung wird über Datei-, bzw. Verzeichnisrechte des Betriebssystems gesteuert oder durch die Anwendung selber.

Datenrepräsentation

	Datenbank	XML
Datenmodell	Relationales Datenmodell, welches auf tabellarischen Daten besteht, mit Zeilen und Spalten.	Hierarchisches Datenmodell, vergleichbar mit einer Dokumentenstruktur mit Element- und Attributknoten.
Datentyp	Typische Datentypen, einschließlich der Unterstützung von binären Daten, werden zur Verfügung gestellt.	XSD Schemen sind ausgestattet mit einer vergleichbaren Menge von Datentypen.
Datenbeziehungen	Spaltenverknüpfungen können innerhalb und zwischen den Tabellen definiert sein, entsprechend den DDL Regeln	Referenzen können klar oder an sich zwischen Elementen definiert werden

Plausibilitäts- und Integritätsprüfung

	Datenbank	XML
Schema	Die lose Struktur von relationalen Schemen ermöglicht eine Menge an Flexibilität, z.B. wie Daten existieren und zusammenhängen können.	XML Schemen sind hingegen durch ihren hierarchischen Aufbau eingeschränkter. Mehrere komplizierte Schematechniken, wie die „XML Schema“ Sprache, bieten jedoch eine derartige Funktionalität.
Integrität	Umfangreiche Unterstützung für die Sicherstellung von Beziehungsbedingungen, sowie die Fähigkeit Änderungen, die in Beziehung stehen, über kaskadierende Löschungen oder Updates durchzuführen.	Beziehungen können simuliert werden, jedoch nicht im Umfang wie es RDBMS unterstützen. Es erfordert einen zweiten Schritt durch einen XML Parser.
Gültigkeitsprüfung	Gültigkeitsprüfungen können ergänzend unterstützt werden durch Triggers und gespeicherten Prozeduren.	XSD Schemen können für die Gültigkeitsprüfung der Elemente und Attribute entworfen werden. Zusätzliche Technologien, wie XSLT und Shematron, können zum verfeinern der Gültigkeitsregeln benutzt werden.

Anfragen und Indizierung

	Datenbank	XML
Anfragesprache	Die meisten kommerziellen RDBMSs unterstützen den industriellen Standard SQL. Viele führen properitäre Erweiterungen ein.	Einzelne XML Dokumente werden üblich mit DOM und SAX APIs abgefragt, oder durch XPath. XQuery ist am vergleichbarsten zu SQL, welches auch eine Kreuz-Dokumentensuche unterstützt.
Ergebnis-manipulation	SQL unterstützt eine Anzahl an Ausgabeparametern, die das Abfrageergebnis anpassen (group, sort etc.).	XSLT und XQuery können zur formatierten XML Ausgabe verwendet werden. Beide Sprachen können komplexe Datenmanipulationen vornehmen.
Abfrage über mehrere Datenquellen	Mehrere Datenbankplattformen erlauben Anfragen über mehrere Datenquellen, solange jede Datenquelle das Protokoll unterstützt, das verwendet wird, um die „Query“ zu stellen.	XPath unterstütz, im Gegensatz zu XQuere und XSLT, derartige Anfragen nicht.

Anfragen und Indizierung

	Datenbank	XML
Indexierung	Differenzierte Indexe werden unterstützt, anpassbar auf das Spaltenniveau. Indexe eines RDBMS können für Abfragen optimiert werden.	Die XML Technologie unterstütz keine vergleichbare Indexerweiterung. Ein XML-Dokumente-Index wird oft von Anwendungen erzeugt und gewartet.

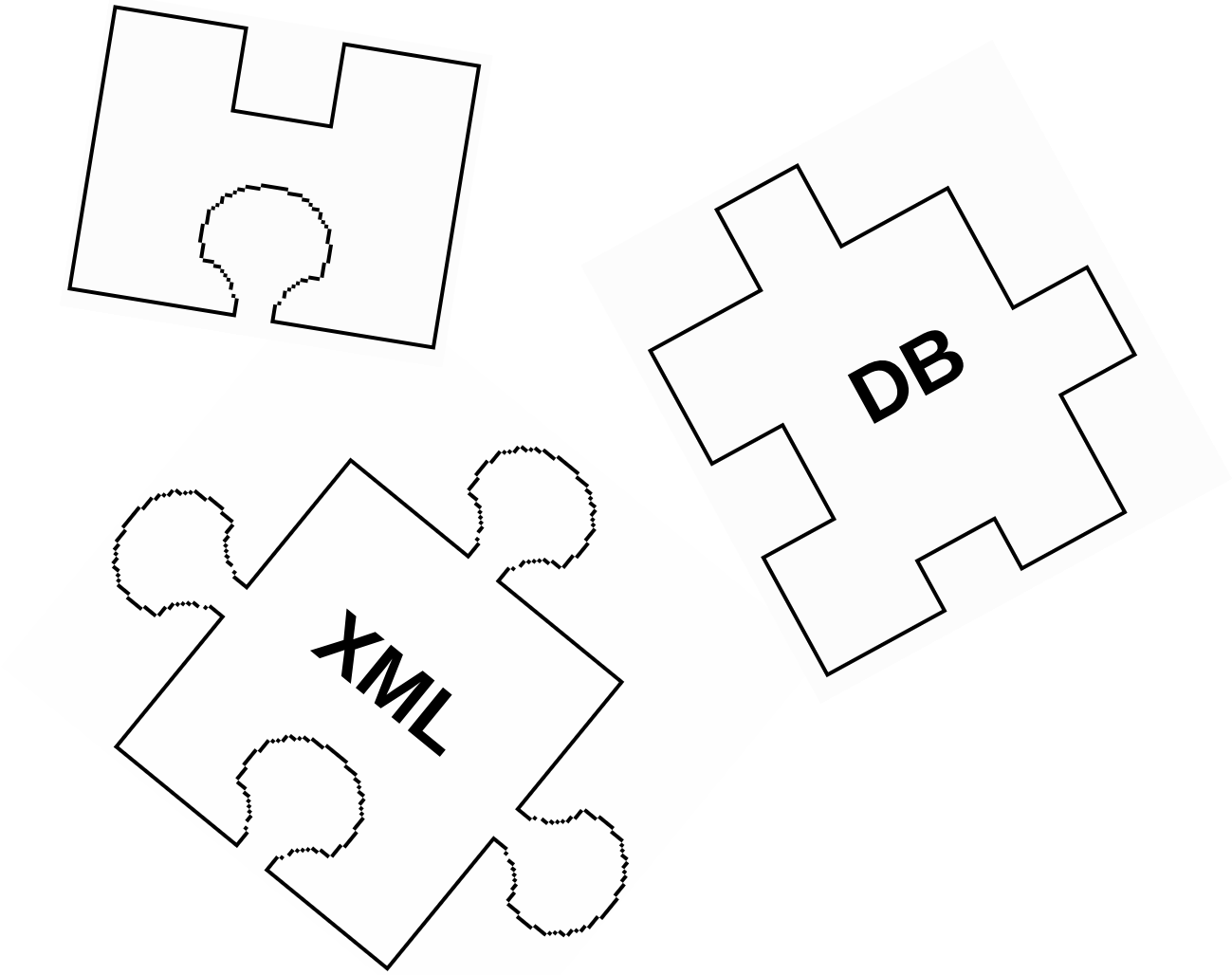
Zusätzliche Eigenschaften

	Datenbank	XML
Transaktionen	Datenbanken liefern Transaktions- und Rollback-Unterstützung, sowie ACID Eigenschaften. Einige RDBMSs beherrschen sogar zweiphasige Transaktionstechniken, die Transaktionen über mehrere Datenbanken erlauben.	XML bietet noch keinen industriellen Standard für Transaktionstechniken (second-generation Web service).
Mehrbenutzerzugriff	Um die Integrität der Daten zu gewährleisten, beim Mehrbenutzerbetrieb, übernimmt die Datenbank die Kontrolle über die zu aktualisierenden Daten (optimistisch oder pessimistisch).	Der physische Dateienzugriff wird von der Anwendung durch den XML Parser kontrolliert. Es existiert kein vergleichbares Sperrverfahren.
Plattform-unabhängigkeit	RDBMS sind herstellerspezifische Produkte, die properitäre Speicherformate besitzen. Die Daten können jedoch meist leicht von einem zum anderen Hersteller umgestellt werden.	Die XML Spezifikationen sind Industriestandards und sind von keiner kommerziellen Plattform abhängig.

Zusätzliche Eigenschaften

	Datenbank	XML
Schemaabhängigkeit	Datenbanken benötigen ein Schema. Die Schematechnik wird von der Datenbank-Software mitgeliefert.	Sind optional für XML Dokumente. Beim Einsetzen von Schemen kann aus verschiedenen Techniken gewählt und sogar vermischt werden.
Schema-wiederverwendung	Das Schema ist an die Datenbank gebunden und kann nicht ohne weiteres wieder verwendet werden.	Ein Schema existiert unabhängig vom XML Dokument und kann somit auf mehrere XML Dokumente angewendet werden.
Verschachtelungen	Tabellenspalten besitzen im Allgemeinen keine innere Verschachtelung.	XML Elemente können verschachtelte Prozesselemente enthalten.

Vergleich von XML und relationalen Datenbanken



Vergleich von XML und relationalen Datenbanken

Integrationsarchitektur für XML und relationaler Datenbanken

Integrationsstrategie von XML mit relationalen Datenbanken

„Mapping“ von XML und relationalen Daten

Herstellerunterstützung von XML in Datenbanken

Reine XML-Datenbanken

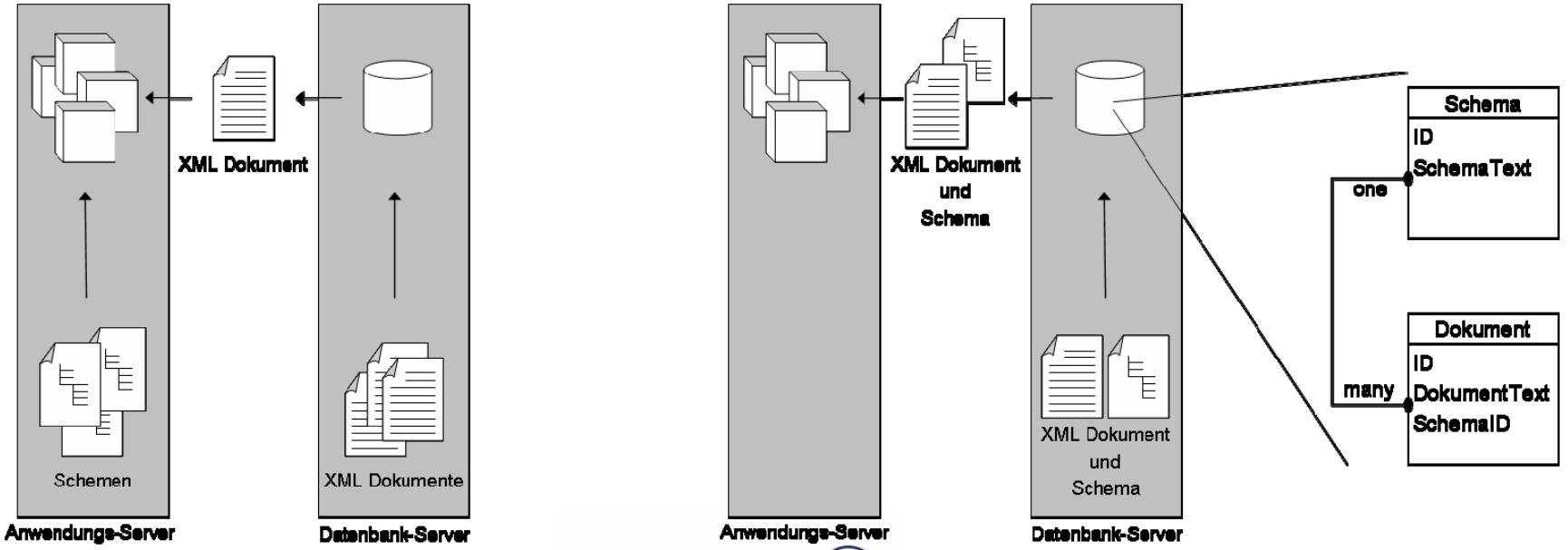
Integrationsarchitektur für XML und relationaler Datenbanken

- Es gibt keinen Standard für die Integration von XML und RDBMS.
- Thomas Erl schlägt vor, bei einer bereits vorhandenen Anwendung mit XML oder eines Entwurfes, sich folgende sechs Fragen zu beantworten.

1. Beschreibung der Rolle, die XML in der Anwendung gegenwärtig spielt und sich vergewissern, dass man ein klares Verständniss hat, bezüglich wie und warum XML gewählt wurde.
2. Auswählen eines primären Business-Prozesses und abbilden der Verarbeitungsprozesse zwischen den Anwendungskomponenten, um zu zeigen, wo XML Daten manipuliert und transportiert werden.
3. An der Datenbankebene identifizieren, wie die formatierten XML-Daten abgeleitet sind, wo Einfügungen und Aktualisierungen erforderlich sind, und wie oft die selben Daten wieder verwendet werden.
4. Prüfen der folgenden Integrationsarchitekturen und Identifizierung des aktuellen oder geplanten Entwurfes.
5. Untersuchung der Pro und Contra und Brauchbarkeit, um sicher zustellen, dass es die Beste Integrationslösung ist. Ansonsten eine Alternative berücksichtigen.
6. Wenn keine Architektur zu den Anforderungen passt, ist die Beste auszuwählen und entsprechend zu modifizieren.

Speicherung von XML Dokumenten als Datenbankrekord

- XML Dokumente werden in eigene, separate Tabellen abgespeichert und sind somit vom gängigen Datenmodel abgetrennt.
- Keine Beeinflussung der existierenden, relationalen Daten, da sie vom aktuellen Datenbankmodell getrennt sind.
- Die Gültigkeitsprüfung der Daten ist Aufgabe der Anwendung. Die entsprechenden Schemen werden entweder an der Anwendung selbst angebunden oder zusammen mit dem XML-Dokument abgespeichert.



fhwedel
UNIVERSITY OF APPLIED SCIENCES

Speicherung von XML Dokumenten als Datenbankrekord

Geeignet für:

- Redundante Speicherung von vorhandenen Daten, vorformatiert in XML. Dies erfordert eine Methode für die Synchronisation.
- Zum Speichern von XML Dokumenten, dessen Daten keinen Zusammenhang mit den vorhandenen Datenbankdaten haben.
- Sitzungsinformationen oder andere Statusinformationen, welche temporär oder permanent gespeichert werden sollen, erfordert nur eine weitere Tabelle mit einer Spalte (z.B. CLOB).

Speicherung von XML Dokumenten als Datenbankrekord

Pro:

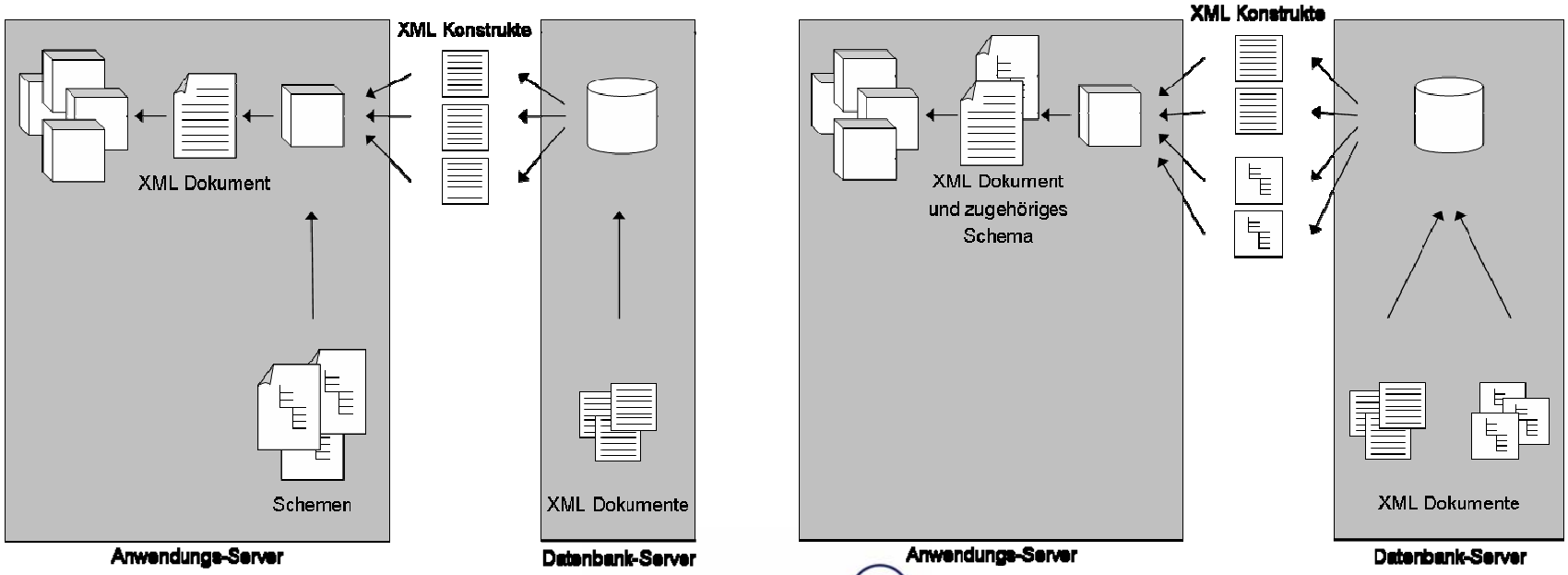
- Das existierende Datenmodell ist nicht betroffen
- Spätere Migration in eine reine XML Datenbank ist einfacher, da diese die XML Dokumente ebenfalls gesamt abspeichern
- XML Daten sind mobil und lose gekoppelt. Sollten die XML Daten zu einem späteren Zeitpunkt in das bestehende Datenmodell integriert werden, kann man die neuen Tabellen einfach entfernen

Contra:

- Abfragen mit diesen Tabellen ist auf eine Volltextsuche beschränkt → langsam
- Viele RDBMS können bei einer Volltextsuche nicht zwischen Daten und Tags unterscheiden → schlechtes Ergebnis
- Dieser Entwurf führt eine Ungleichheit der Daten ins Datenmodell ein
- Bei der Speicherung von XML Dokumenten mit Schemen wird mehr Laufzeitverarbeitung benötigt um ein Dokument zu erhalten (durch Caching-Strategien kann man diese verbessern – periodisches Übertragen des Schemas)

Speicherung von XML Fragmentteile als Datenbankrekord

- Ähnlich der vorherigen Architektur mit einem losen gekoppelten Modell, das die existierenden Daten nicht beeinflusst.
- Der Unterschied liegt in Speicherung des XML-Dokumentes. Es wird in mehreren logischen Einzelteilen separat abgespeichert.
- Schemen können ebenfalls in einzelne Module zerlegt und abgespeichert werden.



Speicherung von XML Fragmentteile als Datenbankrekord

Geeignet für:

- Objekt- oder klassenbasiertes Abwendungs-Interface.
- Anwendungen, die Daten in einer flexible Dokumentenstruktur zur Laufzeit benötigen. Dies ist oft bei parametergesteuerten Geschäftsprozessregeln der Fall.

Pro:

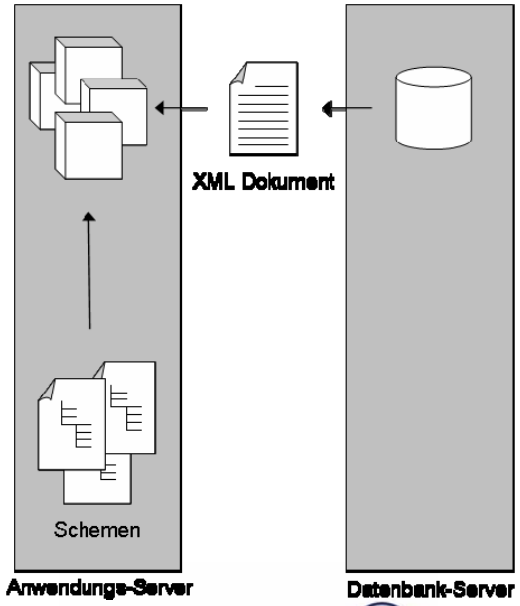
- Führt eine hoch wieder verwendbare Datenplattform ein.
- Das aktuelle Datenmodell ist nicht betroffen.

Contra:

- Führt eine komplexe Datenbankmodellerweiterung ein, die auf Kosten der Anwendung für den Dokumentenzusammenbau geht.
- Kreieren eines Schemas für alle möglichen Kombinationen von XML-Teildokumenten ist eine unhandliche Aufgabe. Erschwert die Korrektheit der Gültigkeitsprüfung.
- Sowie alle Contras der vorherigen Architektur.

XML zum Repräsentieren einer Sicht von Datenbankanfragen

- Erlaubt eine dynamische Erzeugung von Datenbankanfragesichten als XML-Dokument.
- Arbeitet mit der properhitären XML-Erweiterung des entsprechenden Datenbankhersteller.



XML zum Repräsentieren einer Sicht von Datenbankanfragen

Geeignet für:

- Dynamisches generieren von XML Dokumenten mit einer kurzen Lebenszeit. XML wird somit als Transportformat verwendet.
- XML Dokumente werden automatisch durch die properhitäre Datenbankerweiterung generiert, die von den meisten Herstellern angeboten wird.
- Verweistabellen und statische Daten, welche zur Laufzeit von der Anwendung erhalten und im Speicher abgelegt werden.

XML zum Repräsentieren einer Sicht von Datenbankanfragen

Pro:

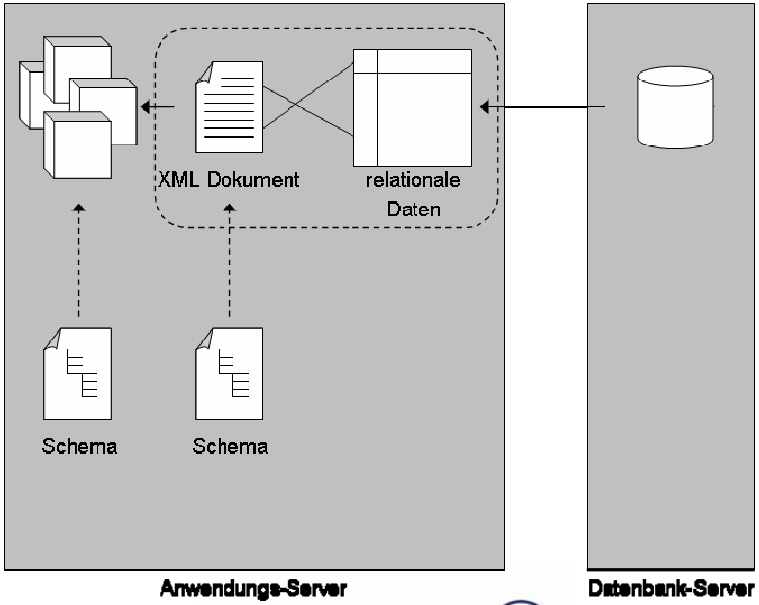
- Einfach zu implementieren bei der Verwendung der mitgelieferten Datenbank-Tools.
- Beeinflusst nicht das aktuelle Modell.

Contra:

- Beim Gebrauch der properhitären Datenbankerweiterung ist der Grad der Nutzbarkeit herstellerseitig begrenzt. Die Ausgabe kann unbrauchbar sein und bedarf einer späteren Aufarbeitung der Anwendung.
- Die properhitäre Datenbankerweiterung bindet an eine Datenbankplattform und XML verliert an Mobilität/Unabhängigkeit.

XML zum Repräsentieren einer Sicht eines relationalen Datenbankmodells

- XML Dokumente werden modelliert, um genaue Teile der Datenbank zu repräsentieren
- Erfordert den meisten Aufwand, da ein richtiges Mappen der relationalen Datenstruktur mit XML Hierarchie von Nöten ist.



XML zum Repräsentieren einer Sicht eines relationalen Datenbankmodells

Geeignet für:

- Anwendungen, die eine genaue Darstellung von ausgewählten, relationalen Dateneinheiten benötigen.
- Anwendungen, die Updates mit XML Dokumenten durchführen.

Pro:

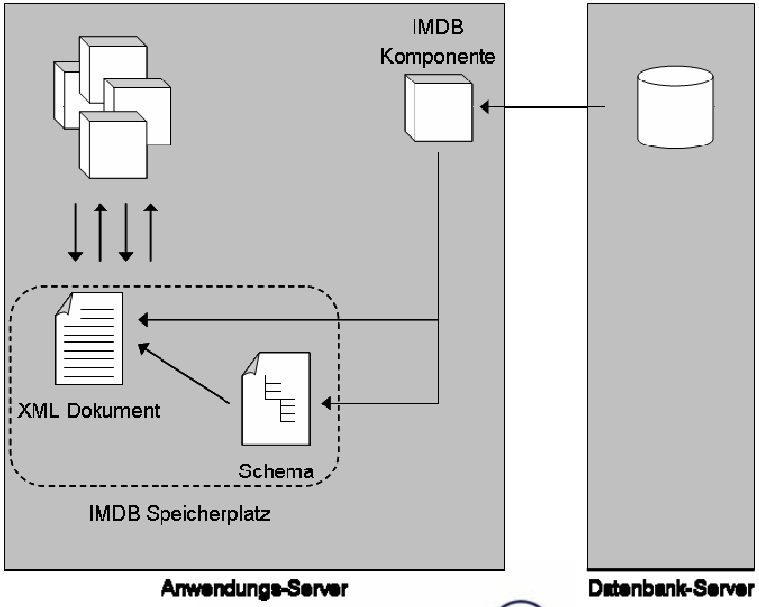
- Führt einen hoch flexiblen und mobilen Datentransportmechanismus ein, der das Datenbankmodell mit beinhaltet
- RDBMS vergleichbare Unterstützung für Einfügungen und Aktualisierungen.
- Der Fokus liegt beim Parsen zum XML, dies hat eine hohe Abstraktion vom Datenzugriff auf RDBMS zufolge.

Contra:

- Komplex zu entwerfen und zu verwalten.
- Hochwertiger Ansatz, der das Abfragen gegen die Datenbank jedoch immer noch nicht entfernt.

XML zum Repräsentieren relationaler Daten in einer IMDB

- XML wird zur Performance-Steigerung der Anwendung verwendet, durch das Zwischenspeichern von relationalen Daten.
- Diese Architektur kann mit anderen kombiniert werden, abhängig von der Repräsentationsart und Art des Mappen.
- Die Daten können einmalig geladen werden oder periodisch.



XML zum Repräsentieren relationaler Daten in einer IMDB

Geeignet für:

- Anwendungen, die große Datenmengen nutzen.
- Relativ statische Daten, wie Verweistabellen .

Pro:

- Ein Zwischenspeichern der Daten auf dem Anwendungs-Server hat eine enorme Performance-Steigerung zur Folge.
- Kostensenkung Methode, da Speicher günstiger ist als Datenbanklizenzen.

Contra:

- Benötigt einen Datenabgleich.
- Performance nimmt zwar zu, aber die Stabilität nimmt ab. (Hauptspeicher ist flüchtiger als Festplattenspeicher).

Vergleich von XML und relationalen Datenbanken

Integrationsarchitektur für XML und relationaler Datenbanken

Integrationsstrategie von XML mit relationalen Datenbanken

„Mapping“ von XML und relationalen Daten

Herstellerunterstützung von XML in Datenbanken

Reine XML-Datenbanken

Nur Daten betrachten - die benötigt werden

Die Betrachtung von nur relevanten Daten eines Geschäftsprozesses ist ein einfaches Entwurfsprinzip, jedoch aus folgenden Gründen sehr wichtig:

Performance

- Das Datenmodell seitens des XML klein halten.
- Vermeiden von autogenerierten XML Dokumenten von Datenbanken mit entsprechenden Erweiterungen.
- Das Neuentwickeln und –implementieren eines Dokumentenmodell nimmt mehr Zeit in Anspruch.

Aufgabenorientierte Darstellung

- Jeder Geschäftsprozess der Anwendung benötigt eine individuelle Teilmenge der Daten.
- Datenbanksichten einführen, die die entsprechenden Informationen darstellen

Beziehungen vermeiden - durch das erstellen von Datenbanksichten

Beim Arbeiten mit immer gleichen Datensätzen empfiehlt es sich, diese in eine Datenbanksicht zusammenzufassen, anstatt die Informationen immer wieder aus mehren Tabellen bilden zu müssen. Diese Sicht lässt sich einfach mit einem XML Dokument mappen.

Pro:

- Datenbanken mit XML Erweiterung können bei selbst beschreibenden Spaltennamen die XML Markierungen „on demand“ optimieren.

Contra:

- Datenbanksichten haben oft nur schreibenden Zugriff und sind für Einfügungen und Aktualisierungen nicht zu gebrauchen.

Entwerfen von XML-freundlichen Datenmodellen

Folgende Punkte sollte man beim Erstellen einer neuen Datenbank für XML Integration beachten.

Vermeiden von zu vielen Beziehungen

Die Integrität des Modells soll nicht darunter leiden, aber das Mappen von einfachen Tabellen ist einfacher als das Mappen bei Tabellen mit Beziehungen.

Unterstützung von vorformatierten XML Sichten

Beim Modellieren daran denken, Tabellen einzuführen, die reine XML Dokumente aufnehmen können. Diese Daten sind zwar redundant, aber vielleicht bietet die Datenbank Speicherfunktionen, „Trigger“ oder andere Erweiterungen für automatisierte Synchronisation.

Beschreibende Spaltennamen

Sollte die Datenbankunterstützung zum Autogenerieren von XML Dokumenten benutzt werden, entstehen dadurch auch selbst beschreibende XML Dokumente.

Nur anwenden, wenn derartige XML Dokumente gebraucht werden (Leistung vs. Lesbarkeit).

Zusammengesetzte Schlüssel vermeiden

Dies gilt bei der Verwendung von DTDs zum Mappen. XSD besitzen hier keine Restrektion.

Schema mit Anmerkungen erweitern

Es empfiehlt sich nicht Anwendungsroutinen zu schreiben, die die Gültigkeitsregeln des Schemas unterstützen. Oft wird dies gemacht, um die Verweis-Integrität sicherzustellen.

Dies sollte jedoch Sache des Schemas bleiben. XSD unterstützt derartige Prozessregeln durch den „appinfo“ Typ. Diese Kommentare werden von der Anwendung zur Laufzeit ausgeführt.

XML Schema ohne XML Datenmodell

Durch das Zusammenfassen aller ausführbaren Kommentare in ein XML Schema ohne Datenmodellierung, erreicht man einen höheren Grad an Mobilität und Erweiterbarkeit des XML Anwendungsmodells.

Entwurf einer „caching“ Strategie

XML Dokumente zur Laufzeit aufzufinden oder zusammenzustellen ist eine prozessorintensive Aufgabe. Die IMDBS Architektur demonstriert den Leistungsnutzen beim Zwischenspeichern von XML Dokumenten auf dem Anwendungs-Server.

Selbst ohne formaler IMDBS Struktur sollten einige XML Dokumente zwischengespeichert werden.

- Statische Berichtsdaten
- Verweistabellen
- Status- und Sitzungsinformationen
- Anwendungskonfigurationsparameter
- Gültigkeits- und Prüfungsregeln

Sowie jegliche Art von statischen Informationen.

Anfragen an das XSD Schema

Mit einem XSD Schema kann genauso umgegangen werden wie mit einem XML Dokument. Somit ist es möglich, mit Hilfe einer „Query“ zur Laufzeit ein dynamisches XSD Schema zu erstellen.

XML Ausgabe steuern mit XSLT

XSLT ermöglicht es ein XML Dokument in unterschiedlichen Ausgabeformaten zu erstellen.

„Query“ über XML Dokumente einschränken

Eine Abfrage in einem XML Dokument ist im Verhältnis zu Datenbankabfragen wesentlich langsamer. Aus diesem Grund sollte das RDBMS möglichst im Vorfeld derartige Arbeit erledigen.

Wenn nicht klar ist, welche Abfragen die Anwendung auf den erhaltenen Daten tätigt, sollte man möglichst kleine XML Dokumente modellieren.

Vergleich von XML und relationalen Datenbanken

Integrationsarchitektur für XML und relationaler Datenbanken

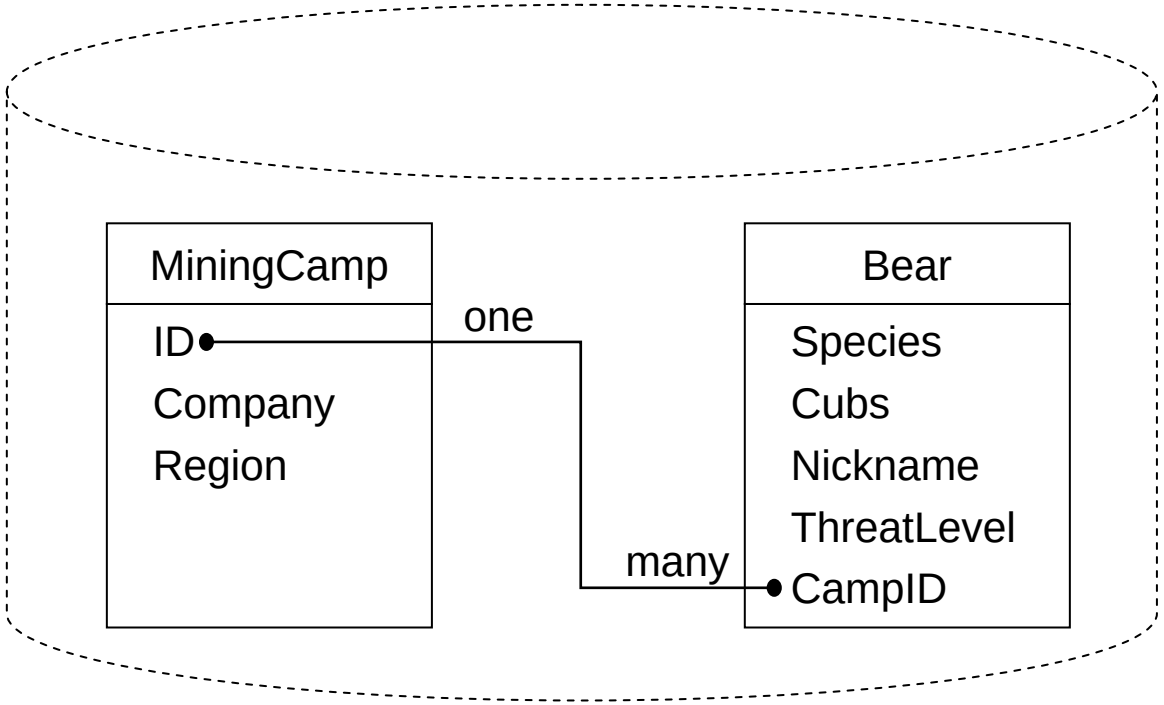
Integrationsstrategie von XML mit relationalen Datenbanken

„Mapping“ von XML und relationalen Daten

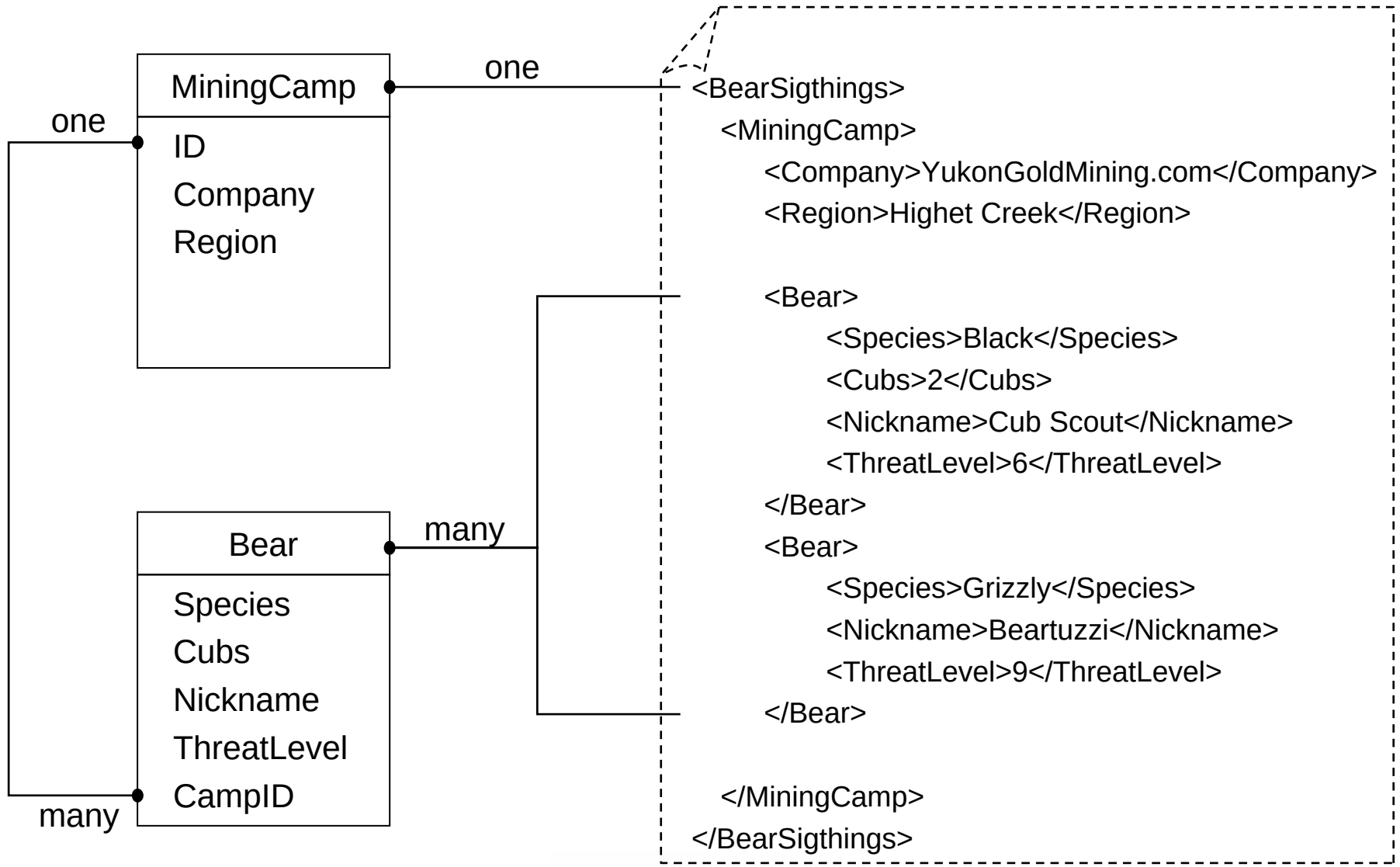
Herstellerunterstützung von XML in Datenbanken

Reine XML-Datenbanken

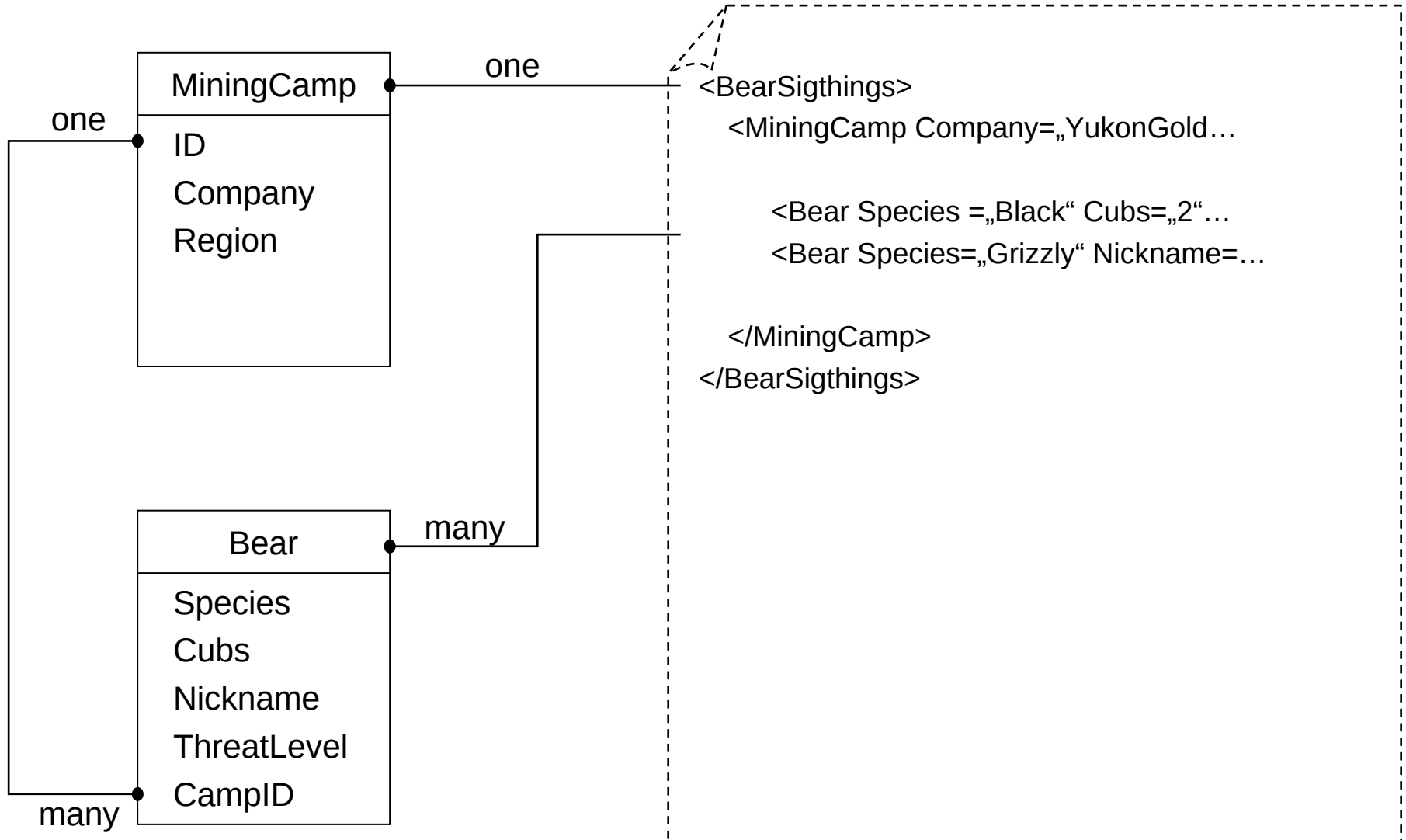
„Mapping“ von XML und relationalen Daten



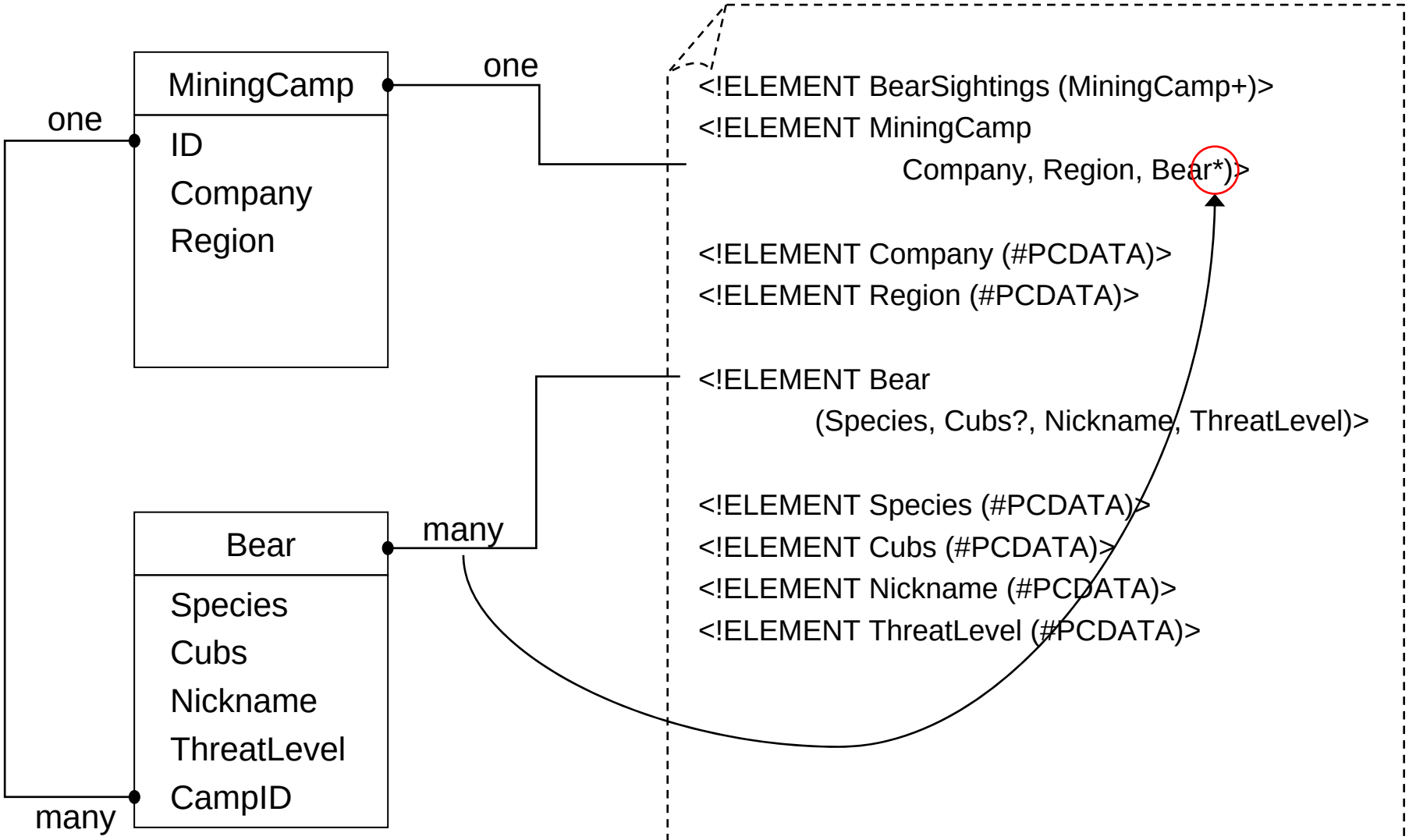
„Mapping“ von XML und relationalen Daten – Spaltennamen sind Elementnamen



„Mapping“ von XML und relationalen Daten – Spaltennamen werden Attributnamen



„Mapping“ von XML und relationalen Daten – mit DTD



„Mapping“ von XML und relationalen Daten – mit DTD

Datentyp

- DTD kennt nur ANY,EMPTY,PCDATA

  Lösung:

  <ThreatLevel DataType=„integer“>9</ThreatLevel>

  Problem:

  Dies ist keine Standardlösung! Außerhalb der Anwendung ist der Zweck des Attributes nicht bekannt.

Null

- DTD hat kein Konzept für den Null-Wert

  Lösung:

  1. Der Null-Wert kann durch ein Schlüsselwort repräsentiert werden <ThreatLevel>NULL</ThreatLevel>
  2. Wenn das Element oder Attribut fehlt, nimmt dieses den Wert Null an

Primärschlüssel

Die XML Spezifikation unterstützt das ID Attribut, das eindeutig ein Element innerhalb des XML Dokumentes identifiziert. Dieses Attribut kann den Primärschlüssel einer Datenbank simulieren.

```
<!ELEMENT MiningCamp (Company, Region, Bear*)>
<!ATTLIST MiningCamp id ID #REQUIRED>
```

Einschränkungen:

- Der Schlüsselwert darf nicht mit einer Zahl beginnen
- Der Schlüsselwert muss im gesamten XML Dokument eindeutig sein. Werden mehrere Tabellen der Datenbank jedoch durch ein XML Dokument repräsentiert, kann es zu Kollisionen kommen.

Lösung:

- Das Voranstellen eines Codes vor dem Schlüssel

```
<MiningCamp ID=„Camp1“>
```

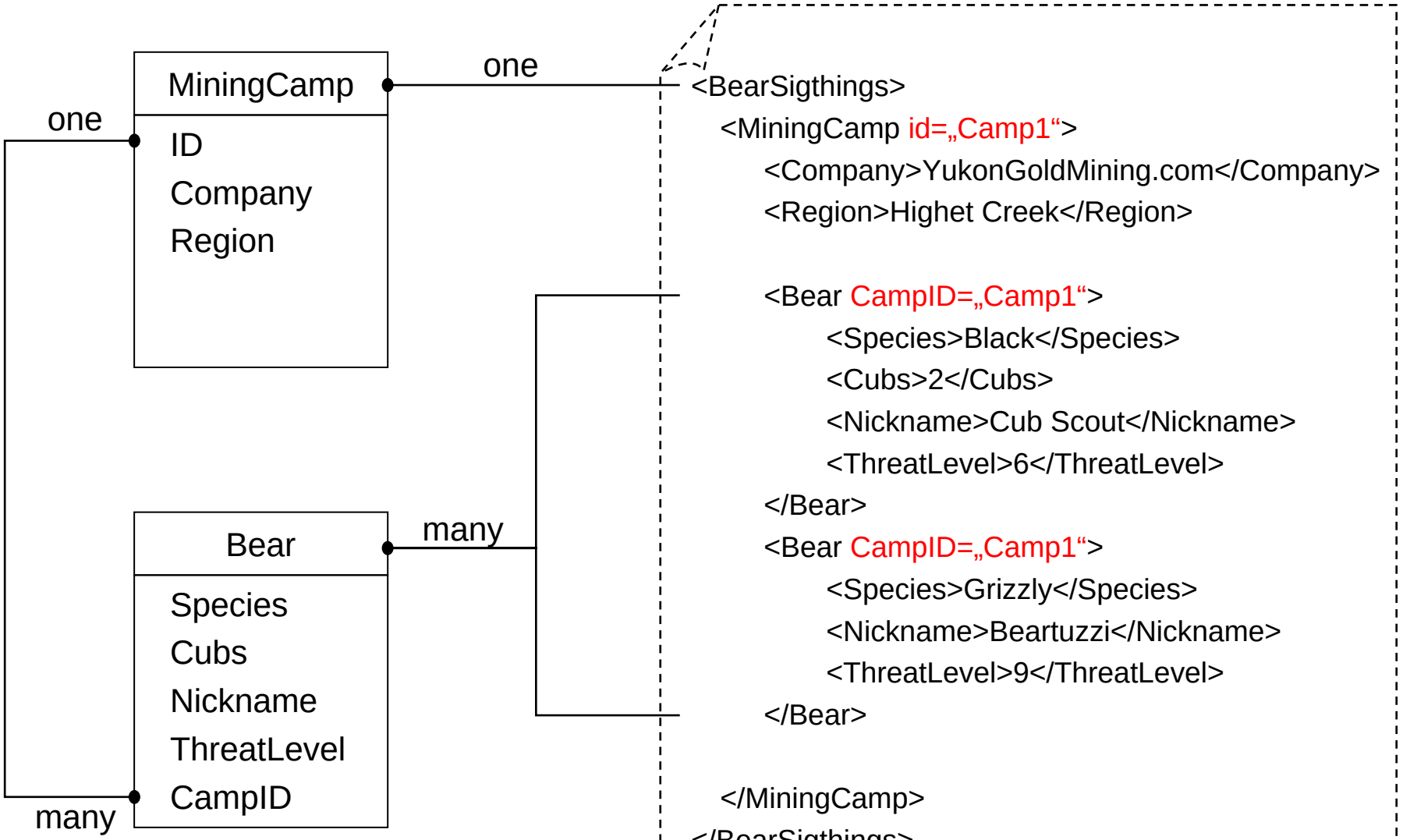
Fremdschlüssel

XML kann Fremdschlüssel durch die Attribute IDREF und IDREFS abbilden. Die Attribute unterscheiden sich nur hinsichtlich der Anzahl von referenzierten ID Attributen.

```
<!ELEMENT Bear (Species, Cubs?, Nickname, ThreatLevel)>
<!ATTLIST Bear CampID IDREF #REQUIRED>
```

IDREFS ermöglicht es sich auf mehre ID Werten zu beziehen, ist jedoch nicht zwingend notwendig. Somit ist bei der Verwendung von IDREFS das Erweitern eines DTD Schema zu einem späteren Zeitpunkt einfacher.

„Mapping“ von XML und relationalen Daten – mit DTD



„Mapping“ von XML und relationalen Daten – mit DTD

```
<BearSigthings>
    <MiningCamp id=„Camp1“>
        <Company>YukonGoldMining.com</Company>
        <Region>Highet Creek</Region>
        <Bear CampID=„Camp1“>
            <Species>Black</Species>
            <Cubs>2</Cubs>
            <Nickname>Cub Scout</Nickname>
            <ThreatLevel>6</ThreatLevel>
        </Bear>
        <Bear CampID=„Camp1“>
            <Species>Grizzly</Species>
            <Nickname>Beartuzzi</Nickname>
            <ThreatLevel>9</ThreatLevel>
        </Bear>
    </MiningCamp>
</BearSigthings>
```

```
<!ELEMENT BearSightings
                (MiningCamp+, Bear*)>
<!ELEMENT MiningCamp
                (Company, Region)>
<ATTLIST MiniCamp id ID #REQUIRED>

<!ELEMENT Company (#PCDATA)>
<!ELEMENT Region (#PCDATA)>

<!ELEMENT Bear
            (Species, Cubs?, Nickname, ThreatLevel)>
<!ATTLIST Bear CampID IDREFS #REQUIRED>

<!ELEMENT Species (#PCDATA)>
<!ELEMENT Cubs (#PCDATA)>
<!ELEMENT Nickname (#PCDATA)>
<!ELEMENT ThreatLevel (#PCDATA)>
```

„Mapping“ von XML und relationalen Daten – mit XSD

XSD beherrscht wesentlich mehr Datentypen als ein DTD und ist außerdem in der Lage eigene Typen zu definieren.
Diese Möglichkeit der Typdefinition nutzt man zum Abbilden einer Tabelle.

```
<element name=„Bear“>
    <complexType>
        <attribut name=„Species“ type=„string“ />
        <attribut name=„Nickname“type=„string“ />
        <attribut name=„ThreatLevel“type=„integer“ />
    </complexType>
</element>
```

Das NULL-Wertproblem besteht jedoch auch bei XSD und muss entsprechend, wie beim DTD, behandelt werden.

Primärschlüssel

Ein XSD Schema kann Xpath-Anweisungen integrieren, um somit die Integrität des Dokumentes besser zu überprüfen, als ein DTD Schema.

```
<key name=„MiningCampPrimaryKey“>
    <selector xpath“.//MiningCamp“ />
    <field xpath=„PrimaryKey“ />
</key>
```

Das *„key“* Element ist ähnlich dem *„ID“* Attribut von DTD, jedoch wird der zu betrachtende Bereich durch Xpath eingeschränkt.

Fremdschlüssel

Das „keyref“ Element definiert einen Fremdschlüssel und zeigt auf den Primärschlüssel.

```
<keyref name=„MiningCampForeignKey“ refer=„x:MiningCampPrimaryKey“>
    <selector xpath“.//Bear“ />
    <field xpath=„ForeignKey“ />
</keyref>
```

„Mapping“ von XML und relationalen Daten – mit XSD

XSD erlaubt auch das Darstellen von Schlüsseln durch das Zusammensetzen von einzelnen Werten, wie es auch in einigen Datenbanken erlaubt ist.

```
<key name=„BearKey“>
    <selector xpath“.//Bear “ />
    <field xpath=„Nickname“ />
    <field xpath=„Species“ />
</key>
```

Zusätzlich kann man bei XSD auch die Anzahl der Instanzen durch „minOccurs“ und „ maxOccurs“ kontrollieren .

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="BearSightings">
    <xs:complexType>
        <xs:choice maxOccurs="unbounded">
        <xs:element name="MiningCamp">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="PrimaryKey" type="xs:string" minOccurs="1" />
                    <xs:element name="Company" type="xs:string" />
                    <xs:element name="Region" type="xs:string" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="Bear">
```

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Vergleich von XML und relationalen Datenbanken

Integrationsarchitektur für XML und relationaler Datenbanken

Integrationsstrategie von XML mit relationalen Datenbanken

„Mapping“ von XML und relationalen Daten

Herstellerunterstützung von XML in Datenbanken

Reine XML-Datenbanken

Vergleich Grundfunktionen

	Oracle	SQL Server	IBM DB2
XML aus Relationen lesen	XSU und XSQL Servlet	DAD über SQL- und RDB- Zuordnung. XML-Objektgruppen	FOR XML
XML in Relationen schreiben	XSU und XSQL Servlet, iFS	DAD über RDB-Zuordnung. XML-Objektgruppen	Update Grams
XML Sichten über Relationen		DAD über RDB-Zuordnung. XML-Objektgruppen	XDR-, XMLSchema-Annotationen
Relationale Sicht über XML			OpenXML
Anfragen in XML-Spalten	SQL Constraints, WITHIN	XPath	XPath
Schreiben als Ganzes (CLOB oder XML-Typ)	InterMedia Text, Indizierung möglich, als CLOB, XMLType und extern	DAD, XML-Spalte, Indizierung über Seitentabelle, Update einzelner Elemente möglich, CLOB, XML-Typ	CLOB und extern

Vergleich Eigenschaften

	Oracle	SQL Server	IBM DB2
Standardkonformität (Namensräume, XML Schema)	Ja	Eingeschränkt	Nein
Inhaltsorientierte Zerlegung generisch	Ja	Ja	Nein
Inhaltsorientierte Zerlegung definitorisch	Ja	Ja	Ja
Opake Speicherung	Ja	Nein	Ja
Originaltreue	Ja	Nein	Ja
Speicherung beliebiger XML Dokumente (Speicherung von rekursiven Elementen, gemischtem Inhalt, Kommentaren, Verabeitungsanweis.)	Ja	Nein	Ja
Kodierungen	XML-Generierung: nur UTF-8 und UCS-2	Kann für Ausgabe via HTTP gewählt werden	inkonsistent
Validierung (DTD oder XML Schema)	Ja	Nein	Ja
Schema-Evolution möglich	Nein	Ja	Nein
Offenes Inhaltsmodell möglich	Nein	Ja	Nein

Vergleich Eigenschaften

	Oracle	SQL Server	IBM DB2
Schema-Evolution möglich	Nein	Ja	Nein
Offenes Inhaltsmodell möglich	Nein	Ja	Nein
Speicherung schemaloser Dokumente möglich	Ja	Nein	Ja
Textsuche	(nur wenn ein Textindex definiert ist)	Nein	(nur wenn einer der Text-Extender vorhanden ist und ein Textindex definiert ist)
Direkte Web-Anbindung (reine URL-Adressierung)	unvollständig	gut	Nein
Erweiterbarkeit	Ja	Nein	Ja
Speicherungsstrukturun-abhängigkeit	Nein	(es gibt nur eine Speicherungsform)	Nein

Vergleich von XML und relationalen Datenbanken

Integrationsarchitektur für XML und relationaler Datenbanken

Integrationsstrategie von XML mit relationalen Datenbanken

„Mapping“ von XML und relationalen Daten

Herstellerunterstützung von XML in Datenbanken

Reine XML-Datenbanken

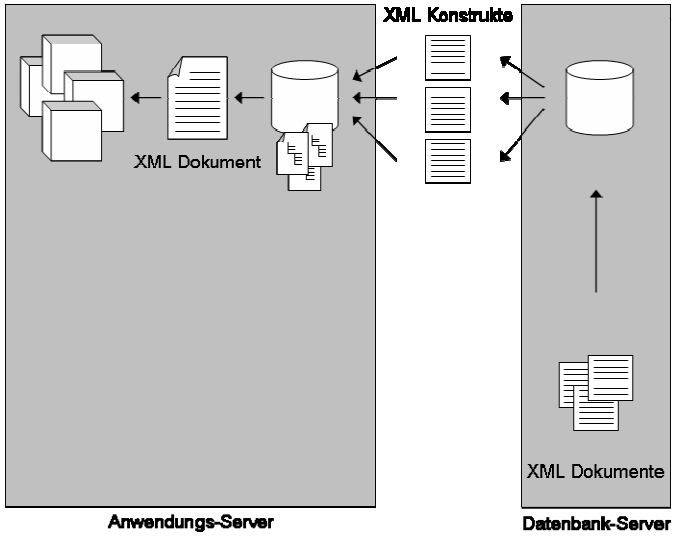
Reine XML-Datenbanken

- Entworfen für die Speicherung von dokumentzentrierten Daten.
- XML Dokumentstruktur werden unabhängig von ihrem Inhalt gespeichert und unterscheidet somit Daten von Markierungen.
- Integration des XML Schemas.
- Volle Unterstützung der entworfenen Anfragesprachen für XML (Xpath, XQuerry, etc.).

Reine XML-Datenbank zur Zwischenspeicherung

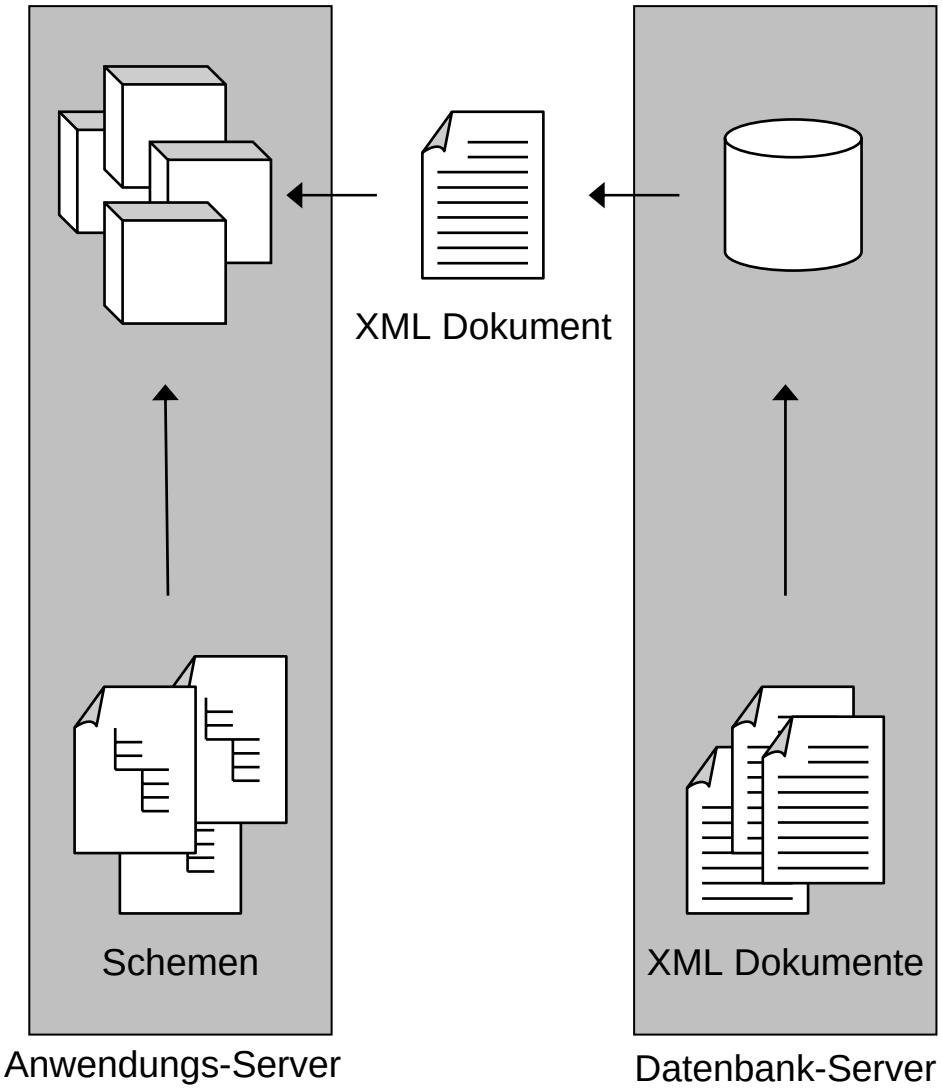
Meistens werden naive XML-Datenbanken unterstützend in vorhandener Umgebung integriert. Sie können als Zwischenspeicher auf der Anwenderseite dienen.

- Leistungsgewinn
- Vorüberprüfung der Daten

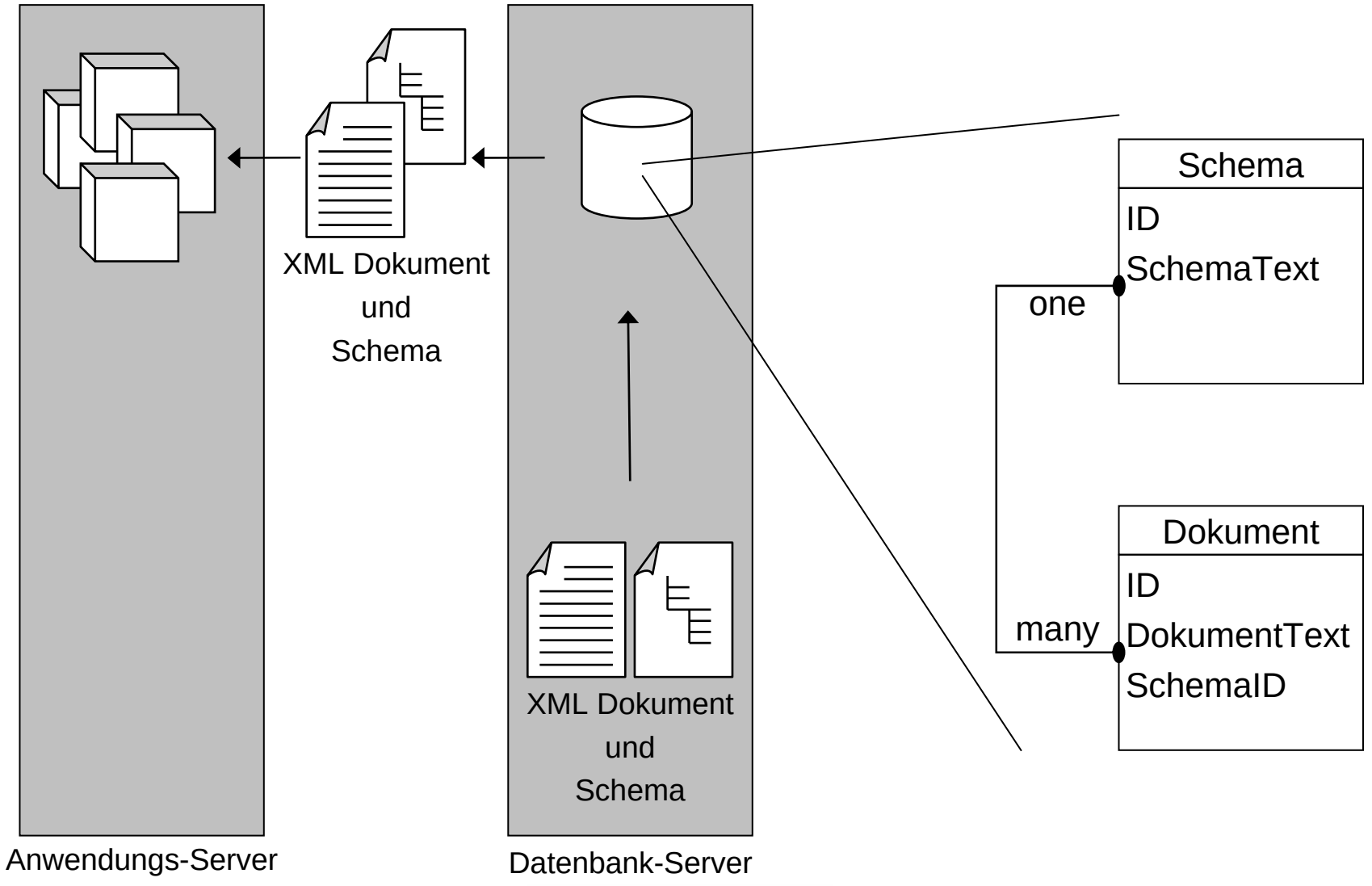


ENDE

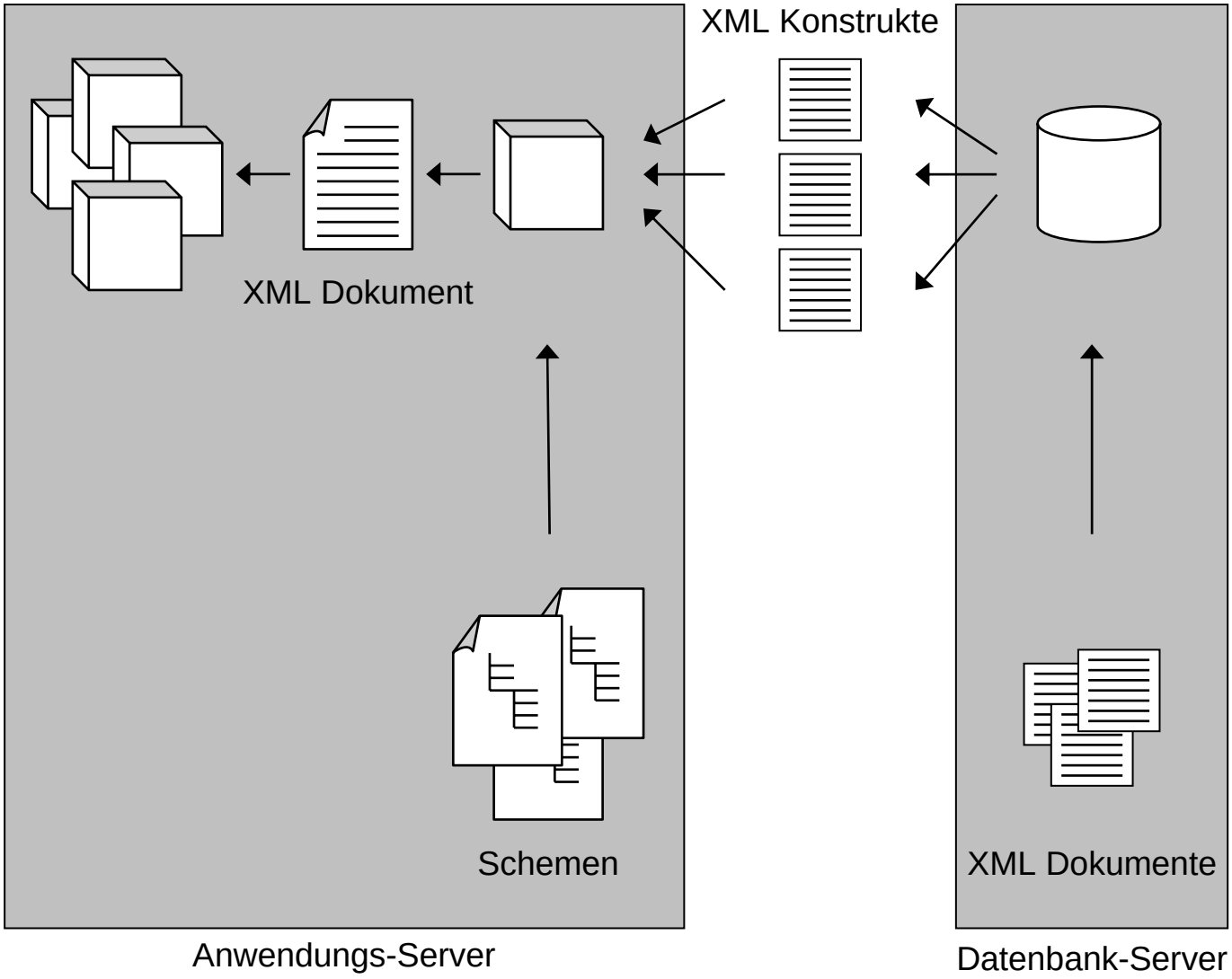
Speicherung von XML Dokumenten als Datenbankrekords



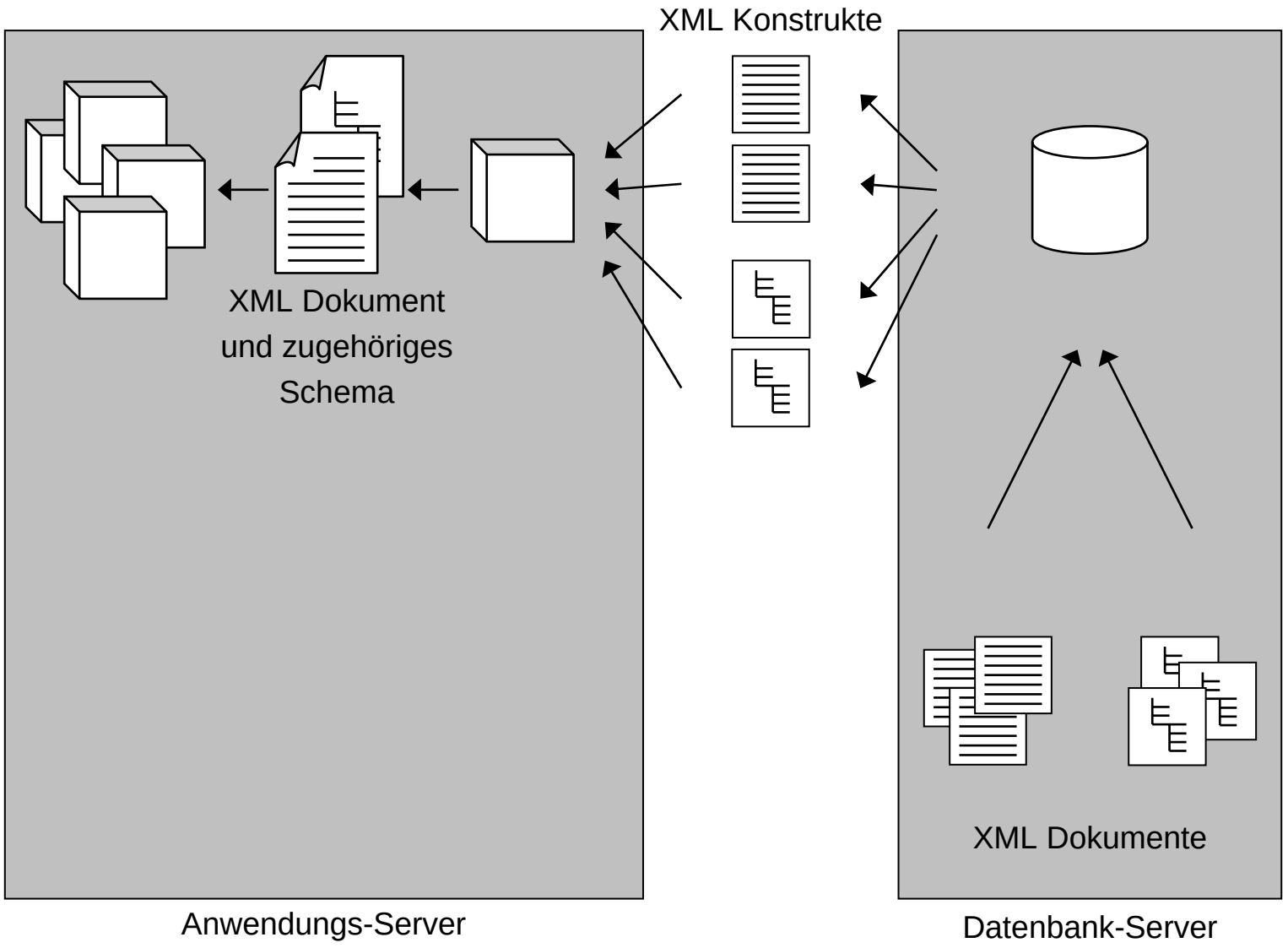
Speicherung von XML Dokumenten als Datenbankrekords



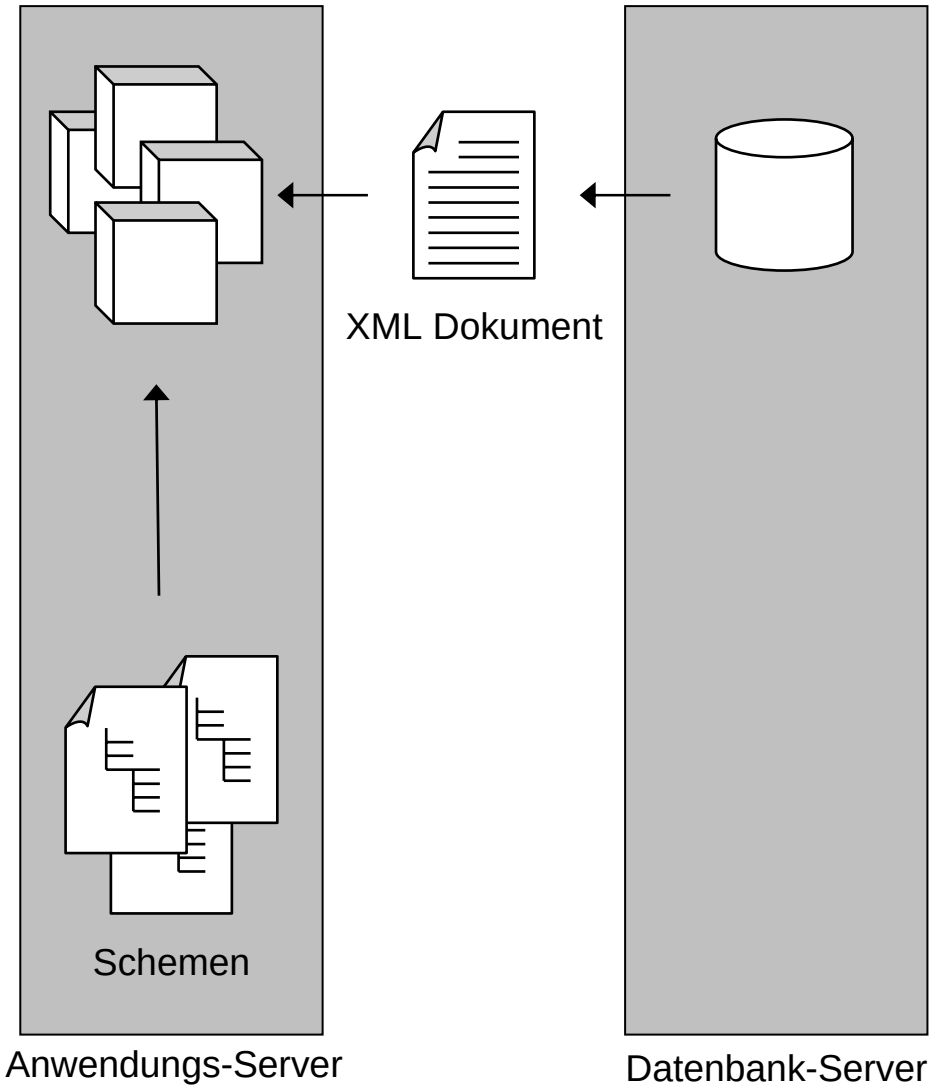
Speicherung von XML Fragmentteile als Datenbankrekords



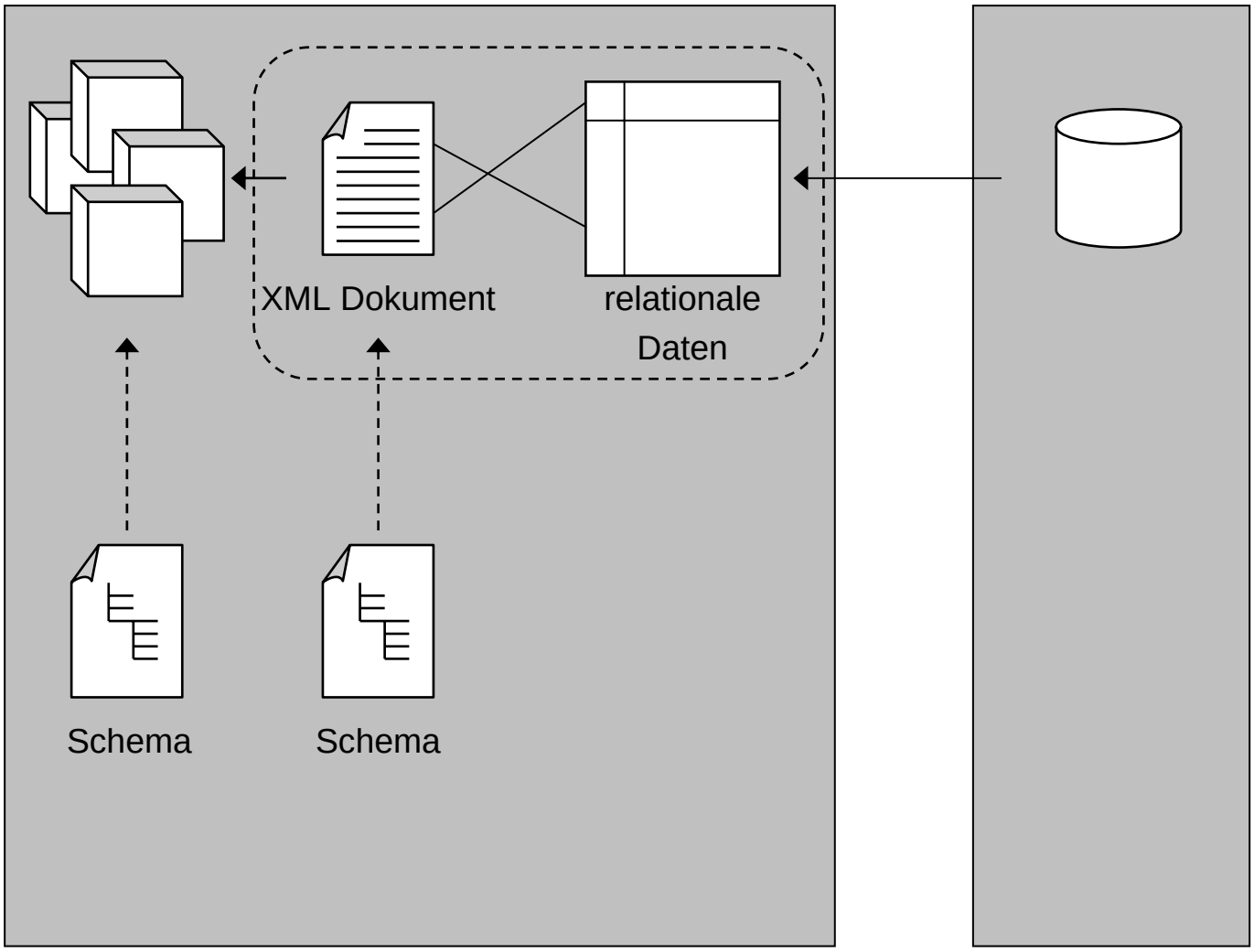
Speicherung von XML Fragmentteile als Datenbankrekords



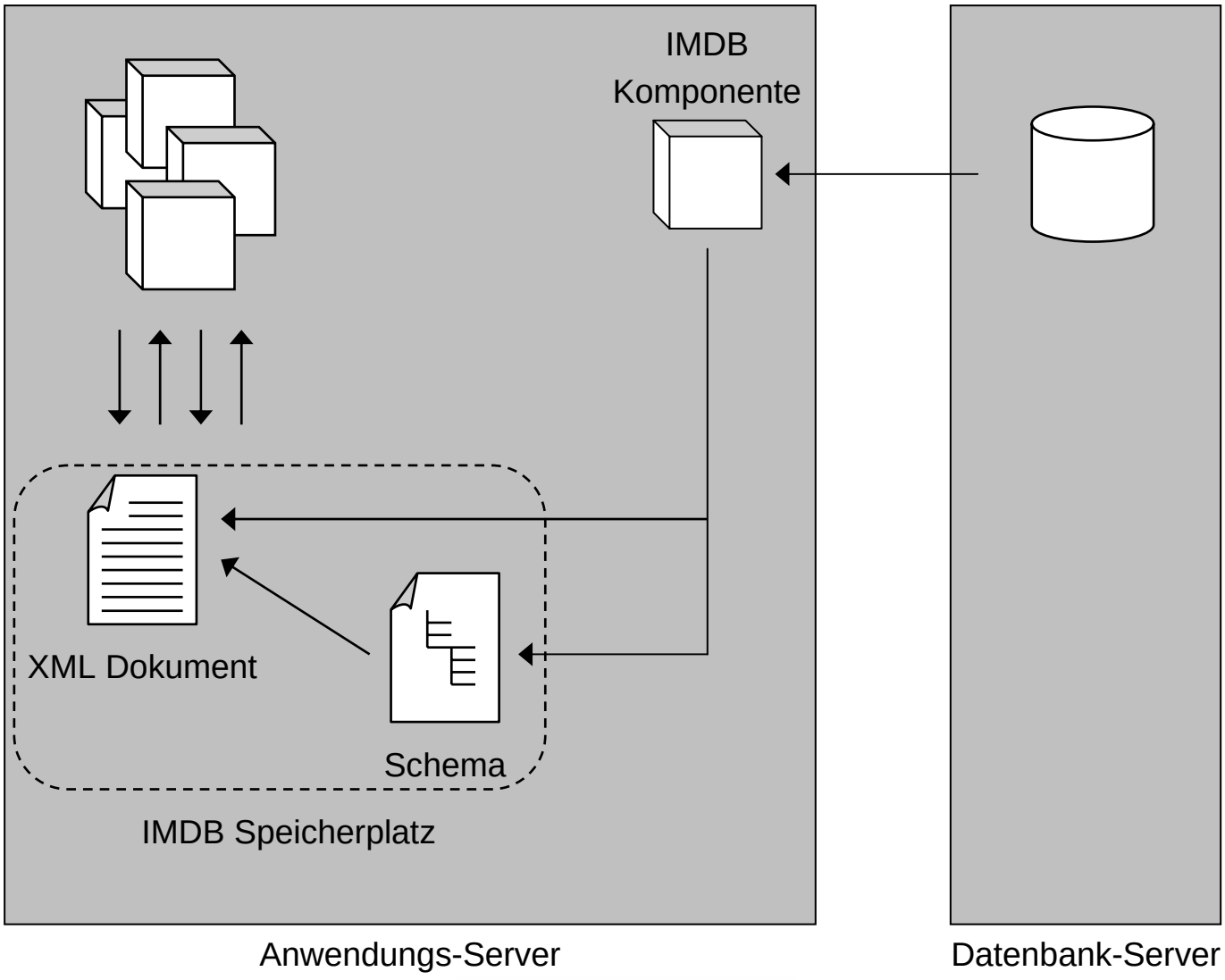
XML zum Repräsentieren einer Sicht einer Datenbankanfragen



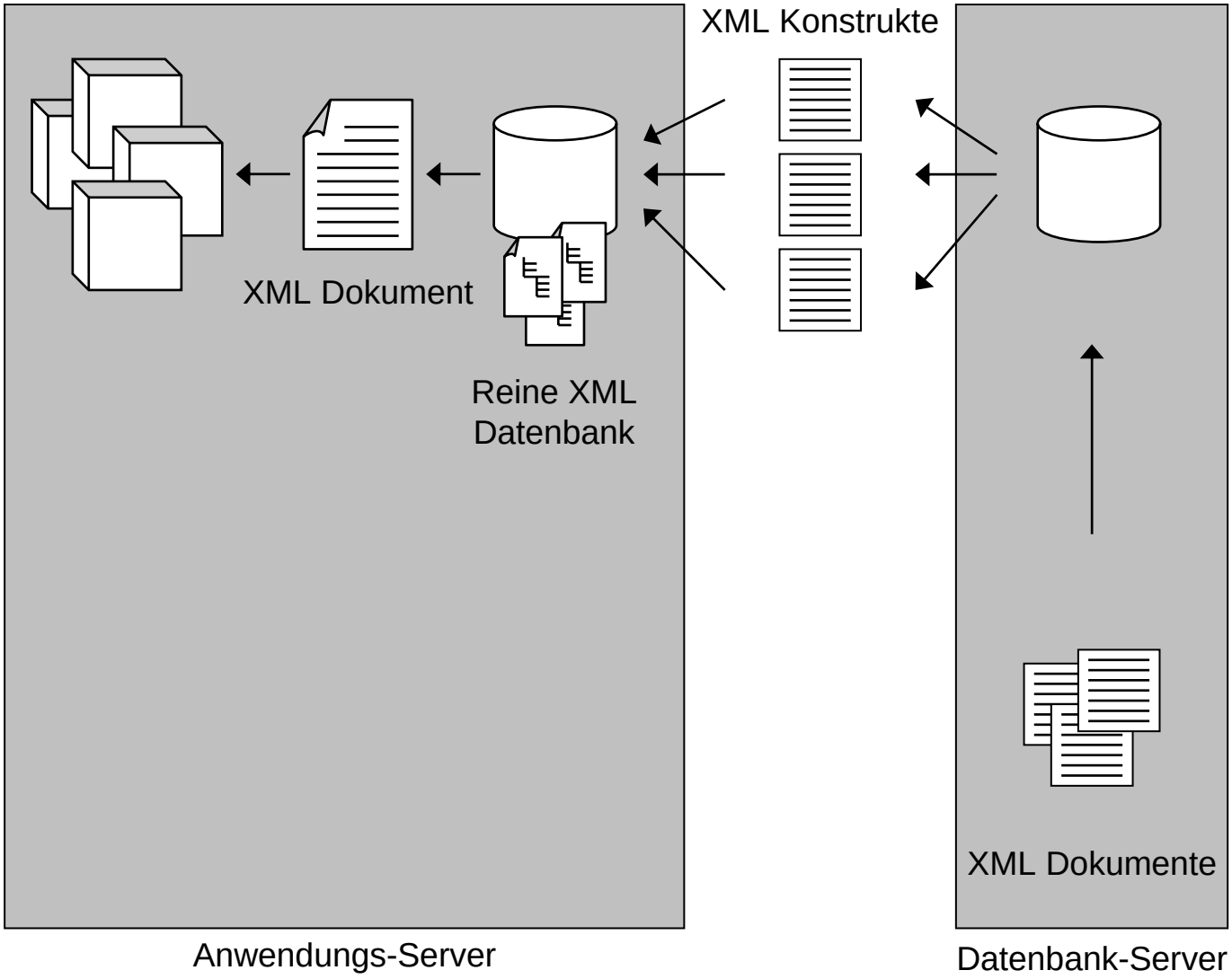
XML zum Repräsentieren einer Sicht eines relationalen Datenbankmodells



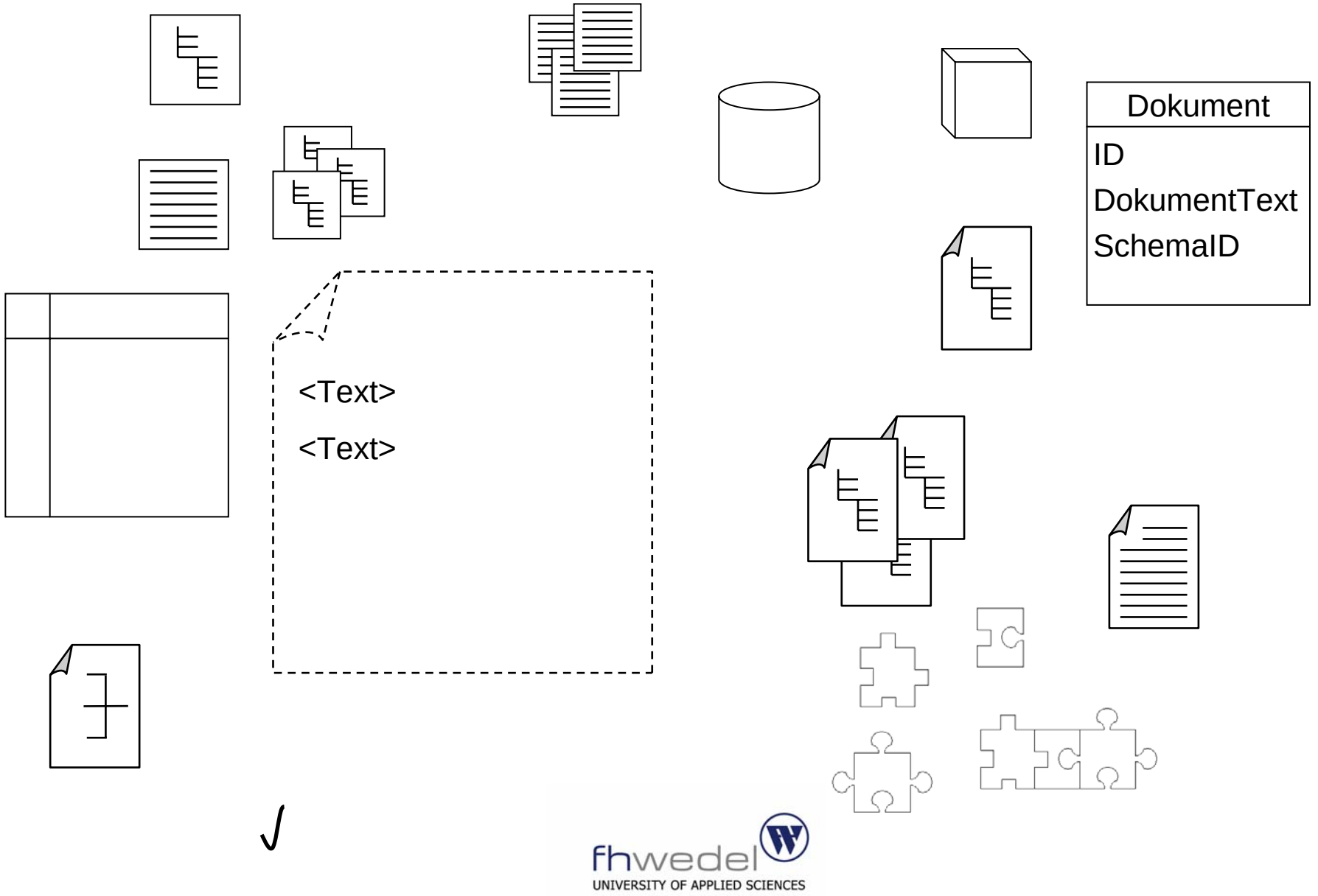
XML zum Repräsentieren relationaler Daten in einer IMDB



Reine XML-Datenbank zur Zwischenspeicherung



Symbole



```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"›
    <xs:element name="BearSightings"›
    <xs:complexType>
        <xs:choice maxOccurs="unbounded"›
        <xs:element name="MiningCamp"›
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="PrimaryKey" type="xs:string" minOccurs="1" />
                    <xs:element name="Company" type="xs:string" />
                    <xs:element name="Region" type="xs:string" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="Bear"›
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="ForeignKey" type="xs:string" minOccurs="0" />
                    <xs:element name="Species" type="xs:string" />
                    <xs:element name="Nickname" type="xs:string" />
                    <xs:element name="Cubs" type="xs:integer" minOccurs="0" />
                    <xs:element name="ThreatLevel" type="xs:integer" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:choice>
</xs:complexType>
<xs:key name="MiningCampPrimaryKey"›
<xs:selector xpath=".//MiningCamp" />
    <xs:field xpath="PrimaryKey" />
</xs:key>
<xs:keyref name="MiningCampForeignKey" refer="MiningCampPrimaryKey"›
    <xs:selector xpath=".//Bear" />
    <xs:field xpath="ForeignKey" />
</xs:keyref>
</xs:element>
</xs:schema>
```

Oracle

Oracle 8i unterstützt erstmalig XML

Folgende Werkzeuge werden angeboten

- Java XML SQL Utility (XSU) und XSQL-Servlet
- *Inter*Media
- Internet File System (iFS)

Java XML SQL Utility

XSU ermöglicht es XML Dokumente in eine Tabelle zu schreiben bzw. zu lesen. Das Tool ermöglicht auch eine Anpassung des Ergebnisdokumentes. Das Werkzeug wurde in Java geschrieben.