

Zeitpunkt

- Synchroner Benachrichtigung
 - Quelle propagiert jede Änderung
- Asynchrone Benachrichtigung
 - Periodisch
 - Quellen erzeugen regelmäßig Extrakte
 - DWH fragt regelmäßig Datenbestand ab
 - Ereignisgesteuert
 - DWH erfragt Änderungen vor jedem Jahresabschluss
 - Quelle informiert alle X Änderungen
 - Anfragegesteuert
 - DWH erfragt Änderungen vor jedem tatsächlichen Zugriff

Zeitpunkt der Datenversorgung

Quelle ...	Technik	Aktualität DWH	Belastung DWH	Belastung Quellen
erstellt periodisch Files				
pushed jede Änderung				
erstellt Extrakte auf Anfrage (Poll)	Vor Benutzung	Sehr	Maximal	Medium
	Anwendungsge-steuert	Anwendungsge-steuert	Je nach Frequenz	Je nach Frequenz

Viele Systeme (Mainframe) nicht online zugreifbar: Extraktion im Batch notwendig

Perversion des DWH Gedankens: Mehrbelastung der Quellen

Technisch bisher nicht effizient durchführbar

Differential Snapshot Problem [LGM96]

- Viele Quellen liefern immer den vollen Datenbestand
 - Molekularbiologische Datenbanken
 - Kundenlisten, Angestelltenlisten
 - Produktkataloge
- Problem
 - Ständiges Einspielen aller Daten ineffizient
 - Duplikate müssen erkannt werden
- Algorithmen um DELTA-Files zu berechnen
- Schwierig bei sehr großen Files

Art der Daten

- Snapshots: Quelle liefert immer kompletten Datenbestand
 - Neuer Lieferantenkatalog, neue Preisliste, etc.
 - Änderungen erkennen
 - Historie korrekt abbilden
- Logs: Quelle liefert jede Änderung
 - Transaktionslogs, Anwendungsgesteuertes Logging
 - Änderungen effizient einspielen
- Netto-Logs: Quelle liefert Netto-Änderungen
 - Katalogupdates, Snapshot-Deltas
 - Keine vollständige Historie möglich
 - Änderungen effizient einspielbar

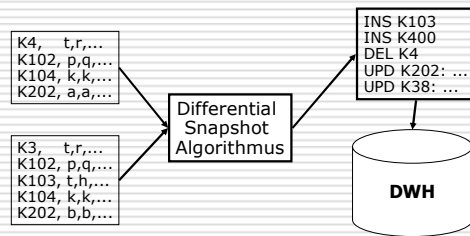
Extraktion aus Legacy Systemen

- Sehr anwendungsabhängig
- Zugriff auf HOST Systeme ohne Online Zugriff
 - Zugriff über BATCH, Reportwriter, Scheduling
- Daten in Non-standard Datenbanken ohne APIs
 - Programmierung in PL-1, COBOL, Natural, IMS, ...
- Unklare Semantik, Doppelbelegung von Feldern, sprechende Schlüssel, fehlende Dokumentation, Herrschaftswissen bei wenigen
- Kommerzielle Tools vorhanden

Szenario

- Quellen liefern Snapshots als File F
 - Ungeordnete Menge von Records (K, A_1, \dots, A_n)
- Gegeben: F_1, F_2 , mit $f_1 = |F_1|, f_2 = |F_2|$
- Berechne kleinste Menge $O = \{INS, DEL, UPD\}^*$ mit $O(F_1) = F_2$
- O nicht eindeutig!
 - $O_1 = \{ (ins(X)), \emptyset, (del(X)) \}$ ■ $O_2 = \{ \emptyset, \emptyset, \emptyset \}$
- ⇒ Differential Snapshot Problem

Szenario



Sattler / Saake

Data-Warehouse-Technologien

13

DS_{naive} – Nested Loop

- Berechnung von O
 - Record R aus F_1 lesen
 - F_2 sequentiell lesen und mit R vergleichen
 - R nicht in $F_2 \rightarrow O=O \cup (\text{DEL R})$
 - R in $F_2 \rightarrow O=O \cup (\text{UPD R})$ / ignorieren
- Problem: INS wird nicht gefunden
 - Hilfsstruktur notwendig
 - Array mit IDs aus F_2 (on-the-fly generieren)
 - R jeweils markieren, abschließender Lauf für INS
- Anzahl IO : $f_1 * f_2 + \text{delta}$
- Verbesserungen ?
 - Suche in F_2 abbrechen, wenn R gefunden
 - Jeweils Partition mit Größe M von F_1 laden – $f_1/M * f_2$

Sattler / Saake

Data-Warehouse-Technologien

15

DS_{sort} – Sort-Merge

- Allgemeiner Fall: $M \ll f_1$ und $M \ll f_2$
- Sortieren auf Sekundärpeicher
 - F in Partitionen P_i mit $|P_i|=M$ lesen
 - P_i in Memory sortieren und schreiben in F^i („Runs“)
 - Alle F^i mischen
- Sortiertes F_2 aufheben für nächstes DS
 - Pro DS muss nur F_2 sortiert werden
- Berechnung von O
 - Sortierte F_1^i und F_2^i öffnen
 - Mischen (paralleles Lesen mit Skipping)
- Anzahl IO: $f_1 + 5 * f_2 + \text{delta}$

Sattler / Saake

Data-Warehouse-Technologien

17

Annahmen

- Berechnung einer konsekutiven Folge von DS
 - Files am 1.1.2000, 1.2.2000, 1.3.2000, ...
- Kostenmodell
 - Alle Operationen im Hauptspeicher sind umsonst
 - IO zählt mit Anz. Records – sequenzielles Lesen
 - Keine Beachtung von Blockgrößen
- Hauptspeichergöße: M (Records)
- Filegrößen $|F_x|=f_x$ (Records)
- Files i.d.R. *wesentlich größer* als Hauptspeicher

Sattler / Saake

Data-Warehouse-Technologien

14

DS_{small} – kleine Files

- Annahme: Hauptspeicher $M > f_1$ (oder f_2)
- Berechnung von O
 - F_1 komplett lesen
 - F_2 sequentiell lesen (S)
 - $S \in F_1$: $O=O \cup (\text{UPD S})$ / ignorieren
 - $S \notin F_1$: $O=O \cup (\text{INS S})$
 - Abschließend: Records $R \in F_1$ ohne Tag: $O=O \cup (\text{DEL R})$
- Anzahl IO: $f_1 + f_2 + \text{delta}$
- Verbesserungen
 - F_1 im Memory sortieren – schnellerer Lookup

Sattler / Saake

Data-Warehouse-Technologien

16

DS_{sort2} – Verschränkung

- Sortiertes F_1 vorhanden
- Berechnung von O
 - F_2 in Partitionen P_i mit $|P_i|=M$ lesen
 - P_i in Memory sortieren und schreiben in F_2^i
 - Alle F_2^i mischen und gleichzeitig mit F_1 vergleichen
- Anzahl IO: $f_1 + 4 * f_2 + \text{delta}$

Sattler / Saake

Data-Warehouse-Technologien

18

DS_{hash} – Partitioned Hash

- Trick: Persistente Sortierung von B_i nicht notwendig
- Berechnung von O
 - F₂ in Partitionen P_i mit |P_i|=M/2 hashen
 - Hashfunktion muss garantieren:
 - $P_i \cap P_j = \emptyset, \forall i \neq j$
 - Partitionen sind „Äquivalenzklassen“ bzgl. der Hashfunktion
 - F₁ liegt noch partitioniert vor
 - F₁ und F₂ wurden mit derselben Hashfunktion partitioniert
 - Jeweils P_{1,i} und P_{2,i} parallel lesen und mischen
- Anzahl IO: $f_1 + 3*f_2 + \text{delta}$

Vergleich - Eigenschaften

	IO	Bemerkungen
DS _{naiv}	$f_1 * f_2$	Außer Konkurrenz Extra Datenstruktur notwendig
DS _{small}	$f_1 + f_2$	Nur für kleine Dateien
DS _{sort2}	$f_1 + 4*f_2$	
DS _{hash}	$f_1 + 3*f_2$	Überlappungsfreie Hashfunktion Partitionsgröße schwierig zu schätzen Verteilungsannahmen (Sampling)

- Erweiterung DS_{hash} für „schlechtere“ Hashfunktionen bekannt

DS mit Zeitstempel

- Annahme: Records sind (K, A₁, ..., A_n, T)
 - T: Zeitstempel der letzten Änderung
 - Erstellen von O
 - Festhalten von T₀: Letztes Update
 - F2 sequentiell lesen
 - Entries mit T > T₀ interessant
 - Aber: INS oder UPD ?
 - Weiteres Problem: DEL wird nicht gefunden
- ⇒ Zeitstempel erspart nur Attributvergleich

Warum nicht einfach ...

- UNIX diff
 - diff erwartet / beachtet Umgebung der Records
 - Hier: Records sind völlig ungeordnet
- In der Datenbank
 - Dreimaliges Lesen jeder Relation notwendig

```

INSERT INTO delta
SELECT 'UPD', ...
FROM F1, F2
WHERE F1.K=F2.K AND K1.W≠F2.W
UNION
SELECT 'INS', ...
FROM F2
WHERE NOT EXISTS (...)
UNION
SELECT 'DEL', ...
FROM F1
WHERE NOT EXISTS (...)
  
```

Weitere DS Verfahren

- Anzahl Partitionen / Runs größer als File-Handles des OS
 - Hierarchische externe Sortierverfahren
- Kompression: Files komprimieren
 - Größere Partitionen / Runs
 - Größere Chance, Vergleich im Hauptspeicher durchzuführen
 - In Realität schneller
- „Windows“ Algorithmus
 - Annahme: Files haben eine „unscharfe“ Ordnung
 - Mischen mit sliding window über beide Files
 - Liefert u.U. redundante INS-DEL Paare
 - Anzahl IO: $f_1 + f_2$

Laden

- Aufgabe
 - Effizientes Einbringen von externen Daten in DWH
- Kritischer Punkt
 - Load-Vorgänge blockieren unter Umständen die komplette DB (*Schreibsperre auf Faktentabelle*)
- Aspekte
 - Trigger
 - Integritäts-Constraints
 - Indexaktualisierung
 - Update oder Insert ?

Satzbasiert

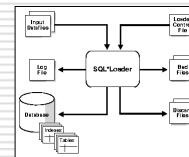
- Benutzung von Standard-Schnittstellen: PRO*SQL, JDBC, ODBC, ...
- Arbeitet im normalen Transaktionskontext
- Trigger, Indexe und Constraints bleiben aktiv
 - Manuelle Deaktivierung möglich
- Keine großräumigen Sperren
- Sperren können durch COMMIT verringert werden
 - Nicht bei Oracle: Leseoperationen werden nie gesperrt (MVCC)
- Benutzung von Prepared Statements
- Teilweise proprietäre Erweiterungen (Arrays) verfügbar

Beispiel: ORACLE sqlldr

Controlfile

```

LOAD DATA
  INFILE 'book.dat'
  REPLACE INTO TABLE book
  (
    book_title POSITION(1) CHAR(35),
    book_price POSITION(37) ZONED(4,2),
    book_pages POSITION(42) INTEGER,
    book_id "book_seq.nextval"
  )
    
```

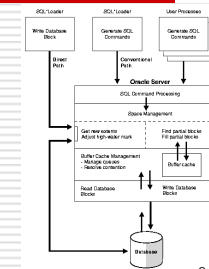


Quelle: Oracle 9.2, Dokumentation

Datenfile

Oracle Essentials	3495	355
SQL*Plus: The Definitive Guide	3995	502
Oracle PL/SQL Programming	4495	987
Oracle8 Design Tips	1495	115

Direct Path Load



Quelle: Oracle 9.2, Dokumentation

BULK Load

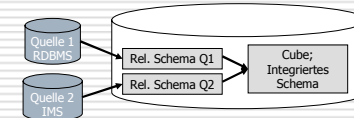
- DB-spezifische Erweiterungen zum Laden großer Datenmengen
- Läuft (meist) in speziellem Context
 - Oracle: DIRECTPATH option
 - Komplette Tabellensperre
 - Keine Beachtung von Triggern oder Constraints
 - Indexe werden erst nach Abschluss aktualisiert
 - Kein transaktionaler Context – kein Logging
 - Checkpoints zum Wiederaufsetzen
- ⇒ Praxis: BULK Uploads

BULK Load Beispiel

- Vielfältige Optionen
 - Behandlung von Ausnahmen (Badfile)
 - Datentransformationen
 - Checkpoints
 - Optionale Felder
 - Konditionales Laden in mehrere Tabellen
 - Konditionales Laden von Records
 - REPLACE oder APPEND
 - Paralleles Laden
 - ...

Technik:

Quelle – Arbeitsbereich – BasisDB



- BULK Load nur für ersten Schritt
- Zweiter Schritt
 - INSERT INTO ... SELECT
 - Logging ausschaltbar
 - Parallelisierbar

Aufgabe:

Quelle-Arbeitsbereich-BasisDB

□ Was macht man wo und wann ?

Keine definierte Aufgabenteilung vorhanden

	Quelle -> Arbeitsbereich	Arbeitsbereich -> Basis-DB
Art des Zugriffs	Satzorientiert	Mengenorientiert
Verfügbare Datenbasen	Eine Quelle (Updatefile)	Viele Quellen
Verfügbare Datensätze	Quellabhängig: Alle, alle Änderungen, Deltas	Zusätzlich Basis-DB verfügbar
Programmiersprache	Skripte: Perl, AWK, ... oder 3GL	SQL, PL/SQL

Sattler / Saake

Data-Warehouse-Technologien

31

Daten- und Schemaheterogenität

□ Hauptdatenquelle: OLTP - Systeme

□ Sekundärquellen:

- Dokumente in firmeninternen Altarchiven
- Dokumente im Internet via WWW, FTP
 - unstrukturiert: Zugriff über Suchmaschinen, Agenten
 - semistrukturiert: Zugriff über Suchmaschinen, Mediatoren, Wrapper als XML-Dokumente o.ä.

□ Grundproblem: Heterogenität der Quellen

Sattler / Saake

Data-Warehouse-Technologien

33

Aspekte der Heterogenität /2

□ unterschiedliche Modellierungen für gleiche Sachverhalte der Realwelt

- bedingt durch Entwurfautonomie
- selbst im gleichen Datenmodell verschiedene Modellierungen möglich, z.B. durch unterschiedliche Modellierungsperspektiven der DB-Designer

Sattler / Saake

Data-Warehouse-Technologien

35

Transformation

□ Problem

- Daten im Arbeitsbereich nicht im Format der Basisdatenbank
- Struktur der Daten unterschiedlich
 - Arbeitsbereich: Quellnahes Schema
 - Basis-DB: Multidimensionales Schema
 - Strukturelle Heterogenität

⇒ Datentransformation

⇒ Schematransformation

Sattler / Saake

Data-Warehouse-Technologien

32

Aspekte der Heterogenität

□ verschiedene Datenmodelle

- bedingt durch autonome Entscheidung über Anschaffung von Systemen in den Unternehmensbereichen
- verschiedene und verschieden mächtige Modellierungskonstrukte, d.h. Anwendungssemantik in unterschiedlichem Ausmaß erfassbar
- Abbildung zwischen Datenmodellen nicht eindeutig

Sattler / Saake

Data-Warehouse-Technologien

34

Aspekte der Heterogenität /3

□ unterschiedliche Repräsentation der Daten

- Unterschiedliche Datentypen möglich
- unterschiedliche Umfang der unterstützten Datentypen
- unterschiedliche interne Darstellung der Daten
- auch unterschiedliche „Werte“ eines Datentyps zur Repräsentation derselben Information

Sattler / Saake

Data-Warehouse-Technologien

36

Klassifikation von Integrationskonflikten

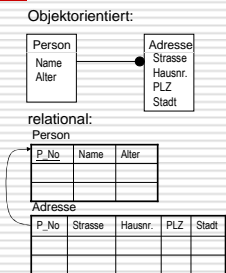
- *hier*: eine oft verwendete Klassifikation mit vier Klassen von Konflikten [Spaccapietra et al. 1992]
 - Semantische Konflikte
 - Beschreibungskonflikte
 - Heterogenitätskonflikte
 - Strukturelle Konflikte
- in der Regel kombiniertes Auftreten dieser Konfliktarten
- *zusätzlich* - für Data Warehouses besonders wichtig:
 - Datenkonflikte

Beschreibungskonflikte

- unterschiedliche Eigenschaften/Attribute derselben Objekte in den lokalen Schemata
- homonyme und synonyme Bezeichnungen
- Datentypkonflikte / Wertebereichskonflikte: unterschiedliche Datentypen / Wertebereiche für die gleiche Eigenschaft
- Skalierungskonflikte: Verwendung unterschiedlicher, aber ineinander umrechenbarer Maßeinheiten
- Konflikte durch zugehörige Integritätsbedingungen oder Manipulationsoperationen

Heterogenitätskonflikte

- unterschiedliche Datenmodelle der zu integrierenden Schemata
 - unterschiedliche Modellierungskonstrukte und Ausdruckskraft
 - impliziert oft auch strukturelle Konflikte
- Auflösung durch Transformation in ein gemeinsames globales Datenmodell



Semantische Konflikte

- semantisch überlappende Weltausschnitte mit einander entsprechenden Klassen
- *Problem*:
 - oft nicht genau die gleiche Menge von Real-Welt-Objekten repräsentiert
 - Unterscheidung zwischen semantisch
 - äquivalenten,
 - sich einschließenden,
 - überlappenden und
 - disjunkten
- Klassenextensionen notwendig
- Inter-Schema-Korrespondenzen: $A=B$, $A \subseteq B$, $A \cap B$, $A \neq B$

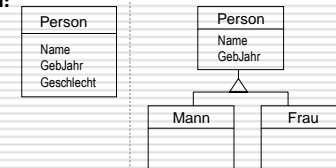
Beschreibungskonflikte /2

- **Beispiel:** Eigenschaften von Waren in verschiedenen Datenbanken
- DB1: *attributes* Preis (USD)
Quantität (integer)
- DB2: *attributes* Wert (PTA)
Anzahl (float)

Strukturelle Konflikte

- selbst bei Verwendung desselben Datenmodells oft unterschiedliche Modellierung eines Sachverhaltes insbesondere bei *semantisch reichen* Datenmodellen (mit vielen Modellierungskonstrukten)

Beispiel:



Beispiel: Datenkonflikte

Personen	Name	Geb.Jahr	Beruf
	Peter Meier	1962	Dipl.-Inform.
	Ingo Schmitt	1928	Dichter

Personen	Name	Geb.Jahr	Beruf
	Meier, Peter	62	Informatiker
	Schmitt, Ingo	28	Lyriker

Sattler / Saake

Data-Warehouse-Technologien

43

Datenkonflikte: Beispiele /2

- verschiedene Ausdrücke:
 - verschiedene Datentypen:
 - z.B. Aufzählungstyp oder numerischer Typ für Notenwerte
 - „sehr gut“, ... , „nicht ausreichend“
 - 1, ... , 5
 - gleicher Datentyp:
 - z.B. bei Strings für Adressen

"Breitestraße"	"Breitestrasse"	"Breitestr."
"Breite Straße"	"Breite Strasse"	"Breite Str."
"Breite-Straße"	"Breite-Strasse"	"Breite-Str."

Sattler / Saake

Data-Warehouse-Technologien

45

Behebung von Datenkonflikten /2

- Verwendung von Hintergrundwissen
 - über Konventionen z.B. bzgl. Schreibweisen
 - über Homonyme und Synonyme (Wörterbücher, Thesauri)
 - über Zusammenhänge von Begriffen und Konzepten im Anwendungsgebiet (Ontologien)

→ **Einsatz wissensbasierter Verfahren**

Sattler / Saake

Data-Warehouse-Technologien

47

Datenkonflikte: Beispiele

- inkorrekte Einträge:
 - Tippfehler bei Eingabe von Werten
 - falsche Einträge aufgrund von Programmierfehlern in einzelnen Anwendungsprogrammen

→ *i.d.R. nicht automatisch behebbar !!!*
- veraltete Einträge:
 - durch unterschiedliche Aktualisierungszeitpunkte
 - z.B. weil Aktualität einer Quelle ausreicht
 - z.B. weil Aktualisierung durch eine Quelle verzögert wird (Ausführungsautonomie)
 - „vergessene“ Aktualisierungen in einzelnen Quellen
 - z.B. durch Fehler in Anwendungsprogrammen

Sattler / Saake

Data-Warehouse-Technologien

44

Behebung von Datenkonflikten

- explizite Werteabbildung
 - z.B. bei unterschiedlichen Aufzählungstypen
 - exakt bei gleicher Kardinalität der Wertebereiche
- Einführung von Ähnlichkeitsmaßen
 - bei Tippfehlern
 - bei leicht unterschiedlichen Schreibweisen erkennt ggf. zu viel oder zu wenig als *ähnlich*
- Bevorzugung der Werte aus einer lokalen Quelle
 - bei unterschiedlicher Aktualität der Daten
 - bei unterschiedlicher Vertrauenswürdigkeit

Sattler / Saake

Data-Warehouse-Technologien

46

Datentransformation

- Syntax von Werten
 - Datum: 20. Januar 2003, 20.01.2003, 1/20/03
 - Codierungen: „1: Adr. unbekannt, 2: alte Adresse, 3: gültige Adresse, 4: Adr. bei Ehepartner, ...“
 - Abkürzungen/Schreibweisen: Str., strasse, Straße, ...
- Datentypen, Semantik
 - Datentypen: Real, Integer, String
 - Genauigkeit, Feldlänge, Nachkommastellen, ...
 - Skalen: Noten, Temperatur, Längen, Währungen, ...
- **In SQL gut unterstützt**
 - Vielfältige Funktionen im Sprachstandard
 - Stringfunktionen, Decodierung, Datumsumwandlung, Formeln, Systemvariable, ...
 - Funktionen in PL/SQL erstellen – in SQL verwenden

Sattler / Saake

Data-Warehouse-Technologien

48

Beispiele

□ Daten

"Pause, Lilo" → "Pause", "Lilo"
"Prehn, Leo" → "Prehn", "Leo"

□ SQL

```
INSERT INTO kunden (nachname, vorname)
SELECT SubStr(name, 0, inStr(name,',')-1),
       SubStr(name, inStr(name,',')+1)
FROM rawdata;
```

Beispiele /3

□ Daten

"Müller", 'M', "sehr gut" → "Müller" 0 1
"Meier", 'w', "gut" → "Meier" 1 2

□ SQL

```
INSERT INTO schueler(name, geschlecht,
                    klasse)
SELECT name,
       decode(upper(sex), 'M', 0, 1),
       decode(grade, 'sehr gut', 1, 'gut', 2, ...)
FROM rawdata;
```

Beispiele

Quelle(ID, Name, Strasse, PLZ, Umsatz)

Kunde(ID, Name, Umsatz)
Adresse(ID, Strasse, PLZ)

Erfordert zwei Durchläufe
der Quelltable

```
- INSERT INTO Kunde ... SELECT ...
- INSERT INTO Adresse ... SELECT ...
```

PremK(ID, Name, Umsatz)
NormK(ID, Name, Umsatz)

Erfordert zwei Durchläufe
der Quelltable

```
- INSERT INTO PremK ... SELECT ...
  WHERE Umsatz >= XYZ
- INSERT INTO NormK ... SELECT ...
  WHERE Umsatz < XYZ
```

Beispiele /2

□ Daten

"Arved", 11/22/2002 → "Arved", 22.11.2002
"Bennett", 2/18/1999 → "Bennett", 18.02.1999

□ SQL

```
INSERT INTO kunden( name, geburtsdatum)
SELECT name,
       to_date(birthday, 'MM/DD/YYYY')
FROM rawdata;
```

Schematransformationen

□ Unterschiedliche Modellierung

- Unterschiedliche Normalisierung
- Was ist Relation, was Attribut, was Wert ?
- Aufteilung von Daten in Tabellen
- Redundanzen aus Quellsystemen
- Schlüssel

□ In SQL nicht gut unterstützt

- INSERT hat nur eine Zieltabelle
- SQL greift auf Daten zu, nicht auf Schemaelemente
- Erfordert meist Programmierung

Schematransformation

□ Schema

- P₁(Id, Name, Geschlecht)
- P₂(Id, Name, M, W)
- P₃₁(Id, Name), P₃₂(Id, Name)

□ P₁ → P₂

- INSERT INTO P₂(id, name, 'T', 'F') ... SELECT ...
- INSERT INTO P₂(id, name, 'T', 'F') ... SELECT ...

□ P₃ → P₁

- INSERT INTO P₁(id, name, 'weiblich') ... SELECT ... FROM P₃₁
- INSERT INTO P₁(id, name, 'weiblich') ... SELECT ... FROM P₃₂

⇒ Anzahl Werte muss feststehen

⇒ Neues Geschlecht – Alle Anfragen ändern

Vermeidung redundanter INSERTs

□ Multi-Table INSERT

■ Neues Feature in Oracle 9i

```
INSERT ALL
  WHEN order_total < 100000
  THEN INTO small_orders
  WHEN order_total > 100000 AND order_total < 200000
  THEN INTO medium_orders
  ELSE
  INTO large_orders
SELECT order_id, order_total, sales_rep_id, customer_id
FROM orders;
```

□ Programmierung

- Durchlauf der Quelltable mit Cursor
- Conditionales Insert in Zieltabelle

□ Cursor meistens langsam

Aspekte von Datenqualität

1. Datenintegrität

- Primärschlüssel, Fremdschlüssel, Integritätsbedingungen, ...

2. Fehlende Daten

- fehlende Attributwerte, fehlende Tupel, ...

3. Falsche Daten

- Objektiv falsch: Negative Preise, 32.13.2002, Buchstaben statt Ziffern, unbekannte Codes, ...
- Widersprüchliche Werte aus unterschiedlichen Quellen: Schreibweisen, Adressen, ...

4. Formatfehler

- Verstoß gegen Formatvorgaben (Datum), falsche Genauigkeit, ...

Data Cleaning Operation

□ Ziel: Verbesserung der Datenqualität

- Ergänzung fehlender Werte
- Korrektur durch Lookup, Neuberechnen, Runden, ...
- Erkennen und Löschen „unrettbarer“ Daten
- Optimum kaum erreichbar: 80/20 Regel

□ DQ muss gemessen werden

- DQ Metriken notwendig
- Verbesserung quantifizierbar

□ Data Cleaning

- Immer ein domänenabhängiger Prozess
- Produkte gibt es nur für Adressdaten

Zusammenfassung

□ ETL ist essentieller Bestandteil jedes operativen DWH

□ Häufig vernachlässigt, da viel anwendungsabhängige und wartungsintensive Programmierung

□ Performanzkritische Schritte

- LOAD
- Transformationen

□ Logisch kritische Schritte

- Erkennen von Duplikaten
- Erkennen von Fehlern (Datenfehler, geänderte Formate, Übertragungsfehler, ...)
- Metriken zur Datenqualität

Aspekte /2

5. Unplausible Daten

- Ausreißer, ...

6. Semantik von NULL Werten

- Wert möglich, aber unbekannt:
 - Ist er Professor ?
- Wert möglich, existiert aber nicht:
 - Er ist kein Professor !
- Wert unmöglich:
 - Kinder unter 18 haben keinen Titel

7. Duplikate

- Kunden, Lieferanten, Produkte, etc.

8. Eingabefehler

9. Schlechte / fehlende Dokumentation

Data Cleaning Schritte [KRRT98]

1. Elementizing

- Zerlegung der Werte
- Erste Normalform

2. Standardizing

- Schreibweisen, Vokabulare, Bezeichnungen, Titel

3. Verifying

- Fehlende Werte, Cross-Matching, Plausibilität

4. Matching

- Erkennen gleicher Objekte: Kunden, Lieferanten, ...

5. Household - Matching

- Erkennen von Organisationen: Haushalte, Abteilungen, Firmen

6. Documenting

Nachvollziehbarkeit

- Änderungen durch DC müssen nachvollziehbar sein
- Auch das Fehlen von Daten
- Unprotokolliertes, nicht nachvollziehbares DC
 - Ad-Hoc DC führt aus Sicht der Anwender zu „fehlenden und falschen“ Daten, nicht zu besseren Daten
 - Daten im DWH müssen erklärbar sein
 - „Da fehlt ein Produkt im Report“
 - Analysewerkzeug fehlerhaft ?
 - Report falsch ?
 - Data Mart Definition falsch ?
 - Basisdatenbank unvollständig ?
 - ETL Prozeduren fehlerhaft ?
 - Übertragungsfehler ?
 - ...

Konvertierungs- und Normalisierungsfunktionen

- Transformation und Standardisierung heterogener Datenformate
 - Konvertierung unterschiedlicher Formate (z.B. Textdateien in DB-Tabellen über Oracle SQL*Loader)
 - Normalisierung: Abbildung von Datenfeldern in ein gemeinsames Format
 - Zeichenketten in Großschreibung
 - Datumsformat: dd/mm/yyyy
 - Währungen

Domänenunabhängige Bereinigung

- Verschmelzen von Objekten aus unterschiedlichen Quellen, die ein Real-Welt-Objekt repräsentieren
 - Schlüsselvergleich („exact matching“)
 - Attributvergleich bzw. -ähnlichkeit („fuzzy matching“)
 - Beispiel: Felder mit komplexen Zeichenketten
 - Übereinstimmung := Anzahl der übereinstimmenden atomaren Strings / durchschnittl. Anzahl von atomaren Strings

Data Cleaning: Techniken

- Konvertierungs- und Normalisierungsfunktionen
- Domänenspezifische Bereinigung
- Domänenunabhängige Bereinigung
- Regelbasierte Bereinigung

Domänenspezifische Bereinigung

- Nutzung von Domänenwissen zur Bereinigung einzelner Felder
- Einsatz spezielle Werkzeuge möglich (häufig auf Basis von Wörterbüchern)
- Beispiele:
 - Produktbezeichnungen im Pharmabereich
 - Adressen über Adressdatenbanken (Postleitzahlen, Telefonvorwahl)
 - Synonyme und Abkürzungen („Str.“ für „Straße“)

Record Linkage

- Erkennen gleicher Objekte
- Insb. gleicher Personen
 - Statistische Datenbanken: Zensus, Volkszählung, Steueraufkommen
 - Kunden
 - Mitarbeiterlisten bei Großkonzernen
 - ...
- Andere Begriffe
 - Objekt Identification
 - Duplicate Elimination

Merge / Purge Problem

- Gegeben: mehrere Listen mit Objekten
 - **Merge:** Erkennen von Duplikaten
 - **Purge:** Auswahl /Berechnung des „besten“ Vertreters pro Klasse

SSN	Name (First, Initial, Last)	Address
334600443	Lisa Boardman	144 Wars St.
334600443	Lisa Brown	144 Ward St.
525520001	Ramon Bonilla	38 Ward St.
525520001	Raymond Bonilla	38 Ward St.
0	Diana D. Ambrosion	40 BriK Church Av.
0	Diana A. Dambrosion	40 Brick Church Av.
0	Colette Johnen	600 113 th St. apt. 5a5
0	John Colette	600 113 th St. ap. 585

Quelle: David Garner, Merge/Purge Problem

Merge/Purge Algorithmen [HS95]

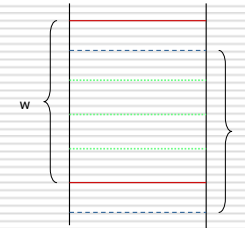
- Typische Vergleichsregeln

```

IF ssn1 = ssn2 THEN match
ELSE IF name1=name2 THEN
  IF firstname=firstname2 THEN
    IF adr1=adr2 THEN match
      ELSE unmatched
    ELSE IF adr1=adr2 THEN match_household
  ELSE IF adr1=adr2 THEN
  ...
    
```

- Naiver Ansatz: „Jeder-gegen-jeden“
 - $O(n^2)$ Vergleiche
 - Maximale Genauigkeit (je nach Regeln)
 - Viel zu teuer

Nearest Neighbour /2



Merge/Purge versus Diff. Snapshot

- Differential Snapshot Problem
 - Eindeutige Keys vorhanden
 - Vergleich zweier Records trivial
 - Eindeutige Lösung möglich und verlangt
 - ⇒ Zahl IO Zugriffe minimieren
- Merge Purge Problem
 - Keine Keys vorhanden
 - Vergleich der Records verursacht dominierende Kosten
 - Keine eindeutige Lösung möglich
 - ⇒ Zahl Vergleich minimieren bei hoher Trefferquote

Nearest Neighbour

1. Berechne einen Schlüssel pro Record
 - Bsp: SSN+„ersten 3 B von Name“ + ...
 - Beachtung typischer Fehler: 0-O, Soundex, Nachbartasten, ...
 2. Sortiere nach Schlüssel
 3. Laufe Liste sequenziell ab
 4. Vergleiche innerhalb eines Windows W , $|W|=w$
 - Mit welchen Tupeln muss wirklich verglichen werden?
- ⇒ Komplexität
- Schlüsselerzeugung $O(n)$
 - Sortieren $O(n \cdot \log(n))$
 - Vergleichen: $O((n/w) \cdot (w^2)) = O(n \cdot w)$
 - Gesamt: $O(n \cdot \log(n))$ oder $O(n \cdot w)$

Problem

- Genauigkeit schlecht
 - Sortierkriterium bevorzugt immer Attribute
 - Sind erste Buchstaben wichtiger für Identität als letzte?
 - Ist Nachname wichtiger als Hausnummer ?
- Window vergrößern?
 - Keine Hilfe
 - Dominanz eines Attributes bleibt gleich, aber Laufzeit verschlechtert sich schnell

Multi-Pass Nearest Neighbour

- Sortieren von N nach mehreren Kriterien
 - Sortierkriterien $S_1 - S_n$
 - Sei $W_x(k) = \{\text{Alle Records um } k \text{ in Window bei Run } S_x\}$
- Vergleich innerhalb der transitiven Hülle mit Länge z

$$R_x = W_x(k)$$

$$R_{x+1} = R_x \cup \{W_y(l) | l \in R_x\}$$

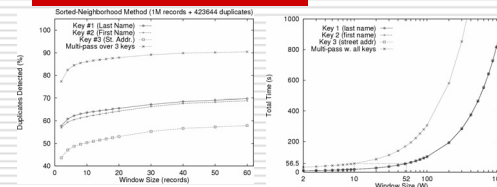
$$R_{x+2} = R_{x+1} \cup \{W_y(l) | l \in R_{x+1}\}$$

...

$$R_{x+z} = R_{x+z-1} \cup \{W_y(l) | l \in R_{x+z-1}\}$$

$$R(k) = \bigcup_{j=1}^n R_j$$

Ergebnisse [HS95]



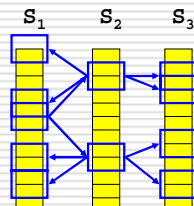
- Gleiche Komplexität, aber höhere Laufzeit
 - Mehrere Sortierungen
 - Abhängig von z und W
- Deutlich höhere Genauigkeit schon bei kleinem |W|

Regelbasierte Bereinigung

- Nutzung von Regeln zur Erkennung von Übereinstimmung zwischen Objekten basierend auf der Kombination mehrerer Felder
- Formen:
 - **benutzer-spezifizierte Regeln:**
 - Kodieren der Regeln durch Benutzer
 - in C oder PL/SQL bzw. im Rahmen spezieller Werkzeuge
 - **Automatisch abgeleitete Regeln:**
 - Ableiten von Regeln aus Analyse der Daten z.B. durch Klassifikationsverfahren (Entscheidungsbäume) oder Assoziationsregeln
 - Beispiel: WizRule

Beispiel

- 3 PASSES
- |W| = 2
- z = Maximale Länge der Pfade



Bewertung

- Hier nur Merge betrachtet
 - Auch Purge nicht trivial!
- Parallelisierung
 - Sortierung parallelisierbar
 - Window-Vergleiche partitionierbar
- Merge/Purge in der Datenbank
 - Sortierungen müssen materialisiert werden
 - Gleichheitsregeln in PL/SQL implementieren
- Nearest Neighbour plus Windowing für jedes Clusteringverfahren verwendbar
 - Abstandsfunktion definieren (hier: Gleichheit)
 - Paare mit geringem Abstand finden
 - ... und dies mit möglichst wenig Vergleichen

Regelbasierte Bereinigung: Beispiele

- Mathematische Regel:
 - A = B * C
 - WHERE
 - A = Total, B = Quantity, C = Unit Price
 - Rule's accuracy level: 0.99
 - rule exists in 1890 records
 - Regelgenauigkeit = Anzahl der Fälle für die Formel gültig ist / Anzahl der relevanten Fälle

Regelbasierte Bereinigung: Beispiele

- IF-THEN-Regel:
IF Customer IS "Summit" AND Item IS
Computer type X
THEN Salesperson = "Dan Wilson"
Rule's probability: 0.98
rule exists in 102 records
error probability < 0.1
- Regelwahrscheinlichkeit := Anzahl der
Datensätze für die Bedingung und
Ergebnis gilt / Anzahl der Datensätze
mit der Bedingung

Literatur

- Baer: „ETL Processing with Oracle 9i“, Oracle Corp. White Paper, 2001
- Rahm, Do: „Data Cleaning: Problems and Current Approaches“, IEEE Data Engineering Bulletin, 2000
- Labio, W. and Garcia-Molina, H. (1996). "Efficient Snapshot Differential Algorithms for Data Warehousing". 22nd VLDB, Bombay, India. pp. 63-74.
- Hernandez, M. A. and Stolfo, S. J. (1995): "The Merge/Purge Problem for Large Databases". SIGMOD
- Pyle, D.: Data Preparation for Data Mining, Morgan Kaufmann, 1999.
- Kimball, R., Reeves, L., Ross, M. and Thornthwaite, W.: "The Data Warehouse Lifecycle Toolkit", John Wiley, 1998.

Zusammenfassung

- Themen der Datenqualität machen
typischerweise 80% des Aufwands
von DWH Projekten aus
- Langsame Anfragen sind ärgerlich
- Falsche Ergebnisse machen der DWH
nutzlos
- Perfekte Daten gibt es nicht
- Metriken festlegen !