LB-DB 9 - 20.6.2015

Dipl.-Ing. Reinhard Schlager

its
FH Salzburg

2015/ IB-Datenbanksysteme

Gliederung

stored function ueberweisung(. . .)

```
CREATE OR REPLACE FUNCTION ueberweisung(
k_send    Konto.KontoNr%TYPE,
k_empf    Konto.KontoNr%TYPE,
k_betrag    Konto.betrag%TYPE
) RETURN INT
IS
t_send_betr Konto.betrag%TYPE;
```

stored function ueberweisung(. . .)

```
CREATE OR REPLACE FUNCTION ueberweisung(
k_send    Konto.KontoNr%TYPE,
k_empf    Konto.KontoNr%TYPE,
k_betrag    Konto.betrag%TYPE
) RETURN INT
IS
t_send_betr Konto.betrag%TYPE;
```

stored function ueberweisung(...)

```
CREATE OR REPLACE FUNCTION ueberweisung(
k_send    Konto.KontoNr%TYPE,
k_empf    Konto.KontoNr%TYPE,
k_betrag    Konto.betrag%TYPE
) RETURN INT
IS
t_send_betr Konto.betrag%TYPE;
```

stored function ueberweisung(...)

```
CREATE OR REPLACE FUNCTION ueberweisung(
k_send    Konto.KontoNr%TYPE,
k_empf    Konto.KontoNr%TYPE,
k_betrag    Konto.betrag%TYPE
) RETURN INT
IS
t_send_betr Konto.betrag%TYPE;
```

```
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
SELECT betrag INTO t_send_betr FROM Konto
WHERE KontoNr = k_send;
 IF (t_send_betr < k_betrag) THEN
 DBMS_OUTPUT.PUT_LINE ('zu wenig am Konto');

RETURN -1;
END IF;
```

```
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
SELECT betrag INTO t_send_betr FROM Konto
WHERE KontoNr = k_send;
 IF (t_send_betr < k_betrag) THEN
 DBMS_OUTPUT.PUT_LINE ('zu wenig am Konto');

RETURN -1;
END IF;
```

```
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
SELECT betrag INTO t_send_betr FROM Konto
WHERE KontoNr = k_send;
 IF (t_send_betr < k_betrag) THEN
 DBMS_OUTPUT.PUT_LINE ('zu wenig am Konto');

RETURN -1;
END IF;
```

```
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
SELECT betrag INTO t_send_betr FROM Konto
WHERE KontoNr = k_send;
 IF (t_send_betr < k_betrag) THEN
 DBMS_OUTPUT.PUT_LINE ('zu wenig am Konto');

RETURN -1;
END IF;
```

```
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
SELECT betrag INTO t_send_betr FROM Konto
WHERE KontoNr = k_send;
 IF (t_send_betr < k_betrag) THEN
 DBMS_OUTPUT.PUT_LINE ('zu wenig am Konto');

RETURN -1;
END IF;
```

```
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
SELECT betrag INTO t_send_betr FROM Konto
WHERE KontoNr = k_send;
 IF (t_send_betr < k_betrag) THEN
 DBMS_OUTPUT.PUT_LINE ('zu wenig am Konto');

RETURN -1;
END IF;
```

```
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
SELECT betrag INTO t_send_betr FROM Konto
WHERE KontoNr = k_send;
 IF (t_send_betr < k_betrag) THEN
 DBMS_OUTPUT.PUT_LINE ('zu wenig am Konto');

RETURN -1;
END IF;
```

```
UPDATE Konto
SET betrag = betrag + k_betrag
WHERE KontoNr = k_empf;
UPDATE Konto
SET betrag = betrag - k_betrag
WHERE KontoNr = k_send;
COMMIT;
RETURN 0;
END ueberweisung;
```

```
UPDATE Konto
SET betrag = betrag + k_betrag
WHERE KontoNr = k_empf;
UPDATE Konto
SET betrag = betrag - k_betrag
WHERE KontoNr = k_send;
COMMIT;
RETURN 0;
END ueberweisung;
```

```
UPDATE Konto
SET betrag = betrag + k_betrag
WHERE KontoNr = k_empf;
UPDATE Konto
SET betrag = betrag - k_betrag
WHERE KontoNr = k_send;
COMMIT;
RETURN 0;
END ueberweisung;
```

```
UPDATE Konto
SET betrag = betrag + k_betrag
WHERE KontoNr = k_empf;
UPDATE Konto
SET betrag = betrag - k_betrag
WHERE KontoNr = k_send;
COMMIT;
RETURN 0;
END ueberweisung;
```

```
UPDATE Konto
SET betrag = betrag + k_betrag
WHERE KontoNr = k_empf;
UPDATE Konto
SET betrag = betrag - k_betrag
WHERE KontoNr = k_send;
COMMIT;
RETURN 0;
END ueberweisung;
```

Test der function:

```
declare status INT;
BEGIN
status := ueberweisung (1,2,10);
DBMS_OUTPUT.PUT_LINE
('status=' ||TO_CHAR(status) );
END
```

Test der function:

```
declare status INT;
BEGIN
status := ueberweisung (1,2,10);
DBMS_OUTPUT.PUT_LINE
('status=' ||TO_CHAR(status) );
END
```

Test der function:

```
declare status INT;
BEGIN
status := ueberweisung (1,2,10);
DBMS_OUTPUT.PUT_LINE
('status=' ||TO_CHAR(status) );
END
```

Test der function:

```
declare status INT;
BEGIN
status := ueberweisung (1,2,10);
DBMS_OUTPUT.PUT_LINE
('status=' ||TO_CHAR(status) );
END
```

Was ist ein Data Warehouse?

Subject oriented: Unter *Subjects* versteht man dabei die Objekte der Geschäftsprozesse wie zum Beispiel Kunden, Produktivität, Erzeugung usw.

Integrated: *Integration* hat das Ziel eine einheitliche und konsistente Sicht der Daten im Data Warehouse zu gewährleisten

Nonvolatile: Während sich die Zugriffe bei OLTP Systemen aus Einfügen, Löschen, Ändern und Abfragen zusammensetzen, beschränken sich die Zugriffe auf ein Data Warehouse auf ein (periodisches) Laden und Abfragen der Daten.

Time-variant: damit ist der Zeithorizont gemeint: OLTP Systeme nur *aktuellen* Daten - Data Warehousing im Bereich von 5 bis 10 Jahren oder mehr[Inm96].

Was ist ein Data Warehouse?

Subject oriented: Unter *Subjects* versteht man dabei die Objekte der Geschäftsprozesse wie zum Beispiel Kunden, Produktivität, Erzeugung usw.

Integrated: *Integration* hat das Ziel eine einheitliche und konsistente Sicht der Daten im Data Warehouse zu gewährleisten

Nonvolatile: Während sich die Zugriffe bei OLTP Systemen aus Einfügen, Löschen, Ändern und Abfragen zusammensetzen, beschränken sich die Zugriffe auf ein Data Warehouse auf ein (periodisches) Laden und Abfragen der Daten.

Time-variant: damit ist der Zeithorizont gemeint: OLTP Systeme nur *aktuellen* Daten - Data Warehousing im Bereich von 5 bis 10 Jahren oder mehr[Inm96].

Was ist ein Data Warehouse?

Subject oriented: Unter *Subjects* versteht man dabei die Objekte der Geschäftsprozesse wie zum Beispiel Kunden, Produktivität, Erzeugung usw.

Integrated: *Integration* hat das Ziel eine einheitliche und konsistente Sicht der Daten im Data Warehouse zu gewährleisten

Nonvolatile: Während sich die Zugriffe bei OLTP Systemen aus Einfügen, Löschen, Ändern und Abfragen zusammensetzen, beschränken sich die Zugriffe auf ein Data Warehouse auf ein (periodisches) Laden und Abfragen der Daten.

Time-variant: damit ist der Zeithorizont gemeint: OLTP Systeme nur *aktuellen* Daten - Data Warehousing im Bereich von 5 bis 10 Jahren oder mehr[Inm96].

Was ist ein Data Warehouse?

Subject oriented: Unter *Subjects* versteht man dabei die Objekte der Geschäftsprozesse wie zum Beispiel Kunden, Produktivität, Erzeugung usw.

Integrated: *Integration* hat das Ziel eine einheitliche und konsistente Sicht der Daten im Data Warehouse zu gewährleisten

Nonvolatile: Während sich die Zugriffe bei OLTP Systemen aus Einfügen, Löschen, Ändern und Abfragen zusammensetzen, beschränken sich die Zugriffe auf ein Data Warehouse auf ein (periodisches) Laden und Abfragen der Daten.

Time-variant: damit ist der Zeithorizont gemeint: OLTP Systeme nur *aktuellen* Daten - Data Warehousing im Bereich von 5 bis 10 Jahren oder mehr[Inm96].

OLTP - Data Warehouse

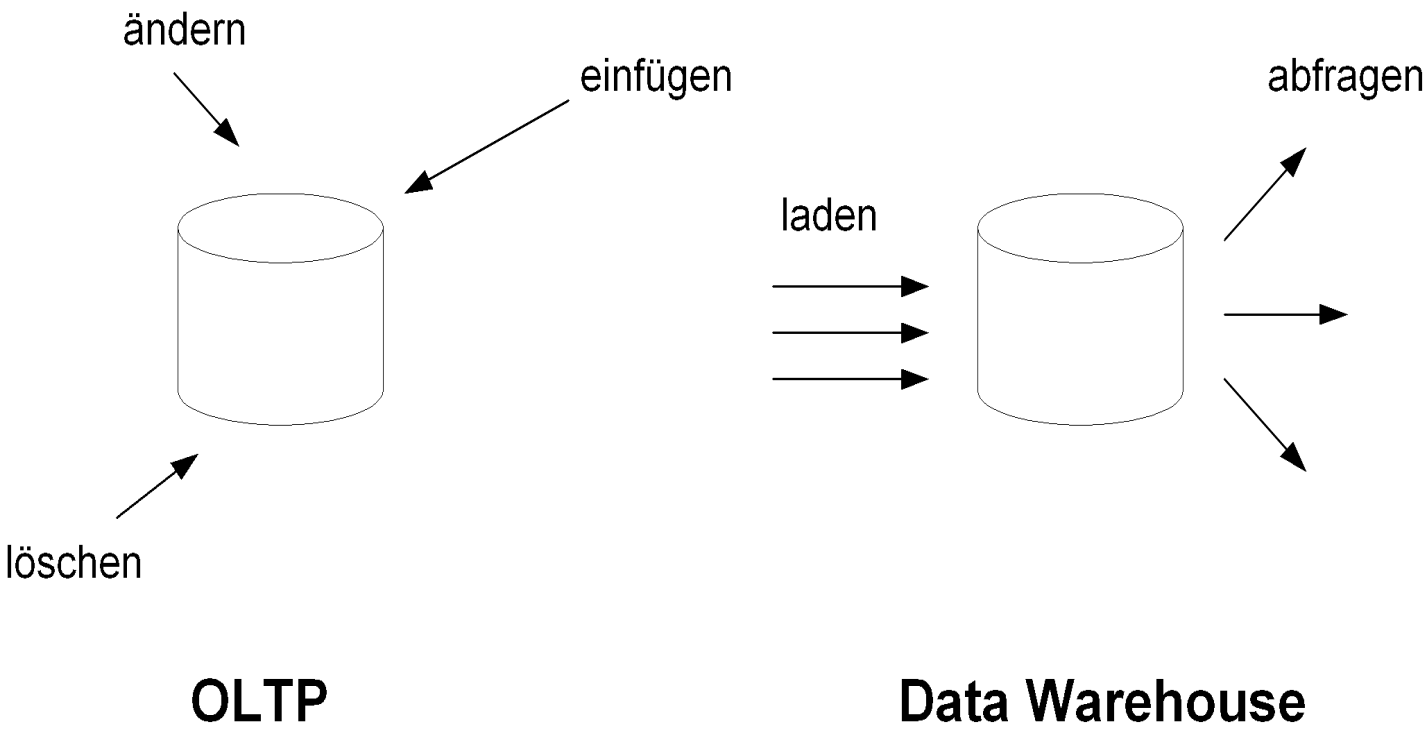

Abbildung: Operationen auf ein OLTP System und ein Data Warehouse [IH94]

ETL - Data Warehouse

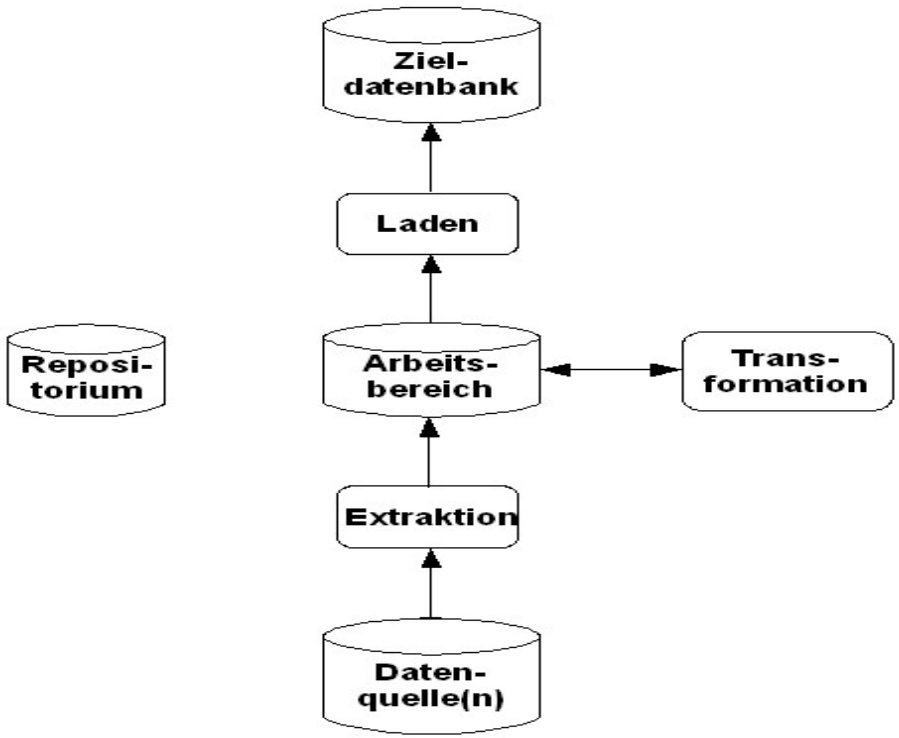

Abbildung: http://de.wikipedia.org/wiki/ETL-Prozess

Star Schema

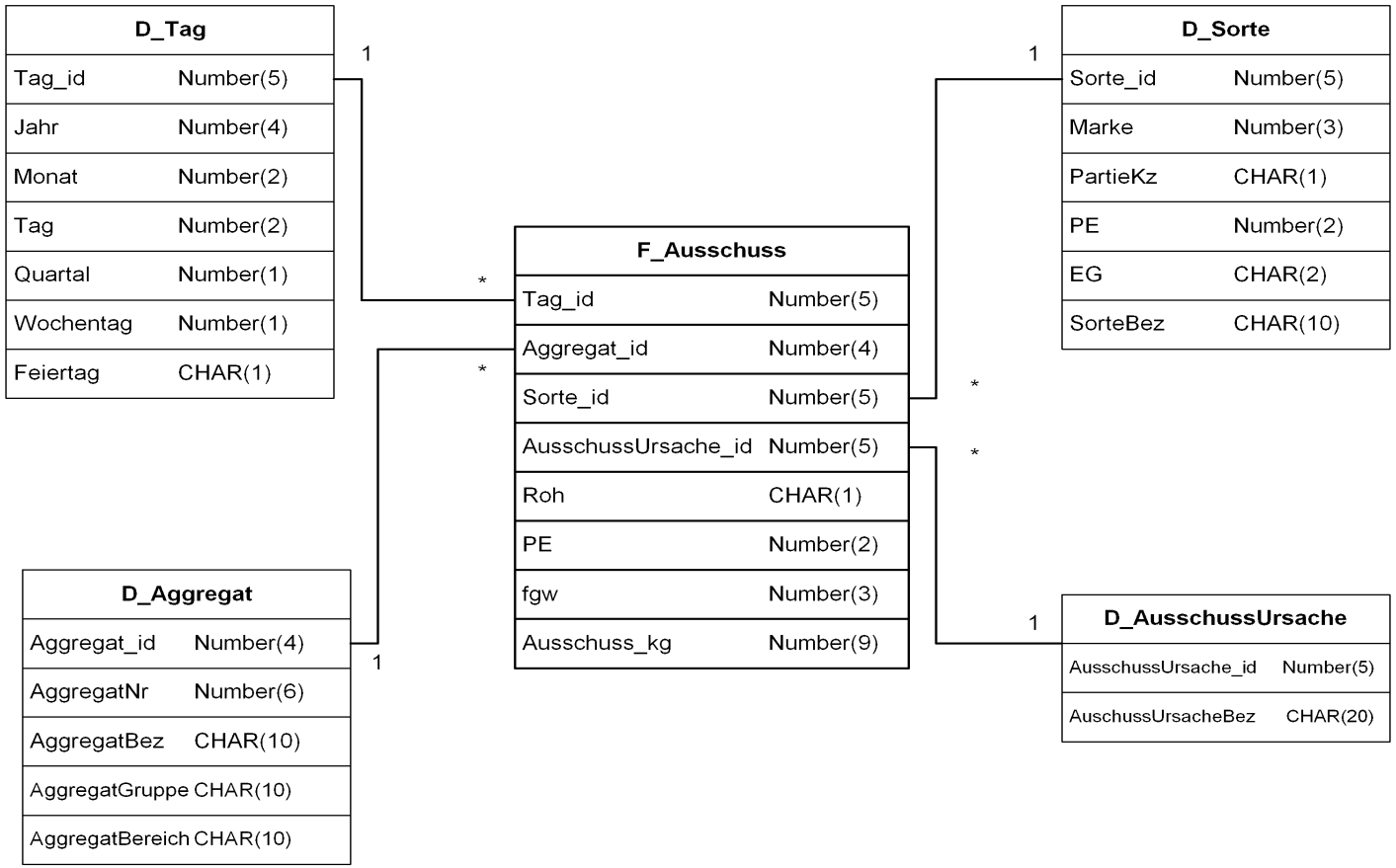

Abbildung: Das Star Schema

Not only SQL

Verzichten meist auf ACID

Kein SQL $\Rightarrow$ Komfortverlust

Aber Fokus auf *Skalierbarkeit*

Not only SQL

Verzichten meist auf ACID

Kein SQL $\Rightarrow$ Komfortverlust

Aber Fokus auf *Skalierbarkeit*

Not only SQL

Verzichten meist auf ACID

Kein SQL $\Rightarrow$ Komfortverlust

Aber Fokus auf *Skalierbarkeit*

Typen von NoSQL Datenbanken

Dokumentenorientierte Datenbanken

Key-Value-Datenbanken

Spaltenorientierte Datenbanken

Graphendatenbanken.

Überblick

Liste der NoSQL DB's

Typen von NoSQL Datenbanken

Dokumentenorientierte Datenbanken

Key-Value-Datenbanken

Spaltenorientierte Datenbanken

Graphendatenbanken.

Überblick

Liste der NoSQL DB's

Typen von NoSQL Datenbanken

Dokumentenorientierte Datenbanken

Key-Value-Datenbanken

Spaltenorientierte Datenbanken

Graphendatenbanken.

Überblick

Liste der NoSQL DB's

Typen von NoSQL Datenbanken

Dokumentenorientierte Datenbanken

Key-Value-Datenbanken

Spaltenorientierte Datenbanken

Graphendatenbanken.

Überblick

Liste der NoSQL DB's

Typen von NoSQL Datenbanken

Dokumentenorientierte Datenbanken

Key-Value-Datenbanken

Spaltenorientierte Datenbanken

Graphendatenbanken.

Überblick

Liste der NoSQL DB's

Typen von NoSQL Datenbanken

Dokumentenorientierte Datenbanken

Key-Value-Datenbanken

Spaltenorientierte Datenbanken

Graphendatenbanken.

Überblick

Liste der NoSQL DB's

Übung 9 IB-DB 20.6.2015

- Ein vorhandenes Lagersystem verwaltet Artikel und kann jederzeit Auskunft über den aktuellen Lagerbestand jeder Artikelnummer geben.
- Versuchen Sie das Schema eines Data Warehouse Systems für dieses Lagersystem zu entwerfen

Übung 9 IB-DB 20.6.2015

- Ein vorhandenes Lagersystem verwaltet Artikel und kann jederzeit Auskunft über den aktuellen Lagerbestand jeder Artikelnummer geben.
- Versuchen Sie das Schema eines Data Warehouse Systems für dieses Lagersystem zu entwerfen

Quellen


W.H. Inmon and Richard D. Hackathorn.
Using the Data Warehouse.
John Wiley & Sons, Inc., 1994.


W.H. Inmon.
Building the Data Warehouse.
John Wiley & Sons, Inc., 1996.

Quellen


W.H. Inmon and Richard D. Hackathorn.
Using the Data Warehouse.
John Wiley & Sons, Inc., 1994.


W.H. Inmon.
Building the Data Warehouse.
John Wiley & Sons, Inc., 1996.