Implementierung eines Data Marts

Gunther Popp
dc soft GmbH

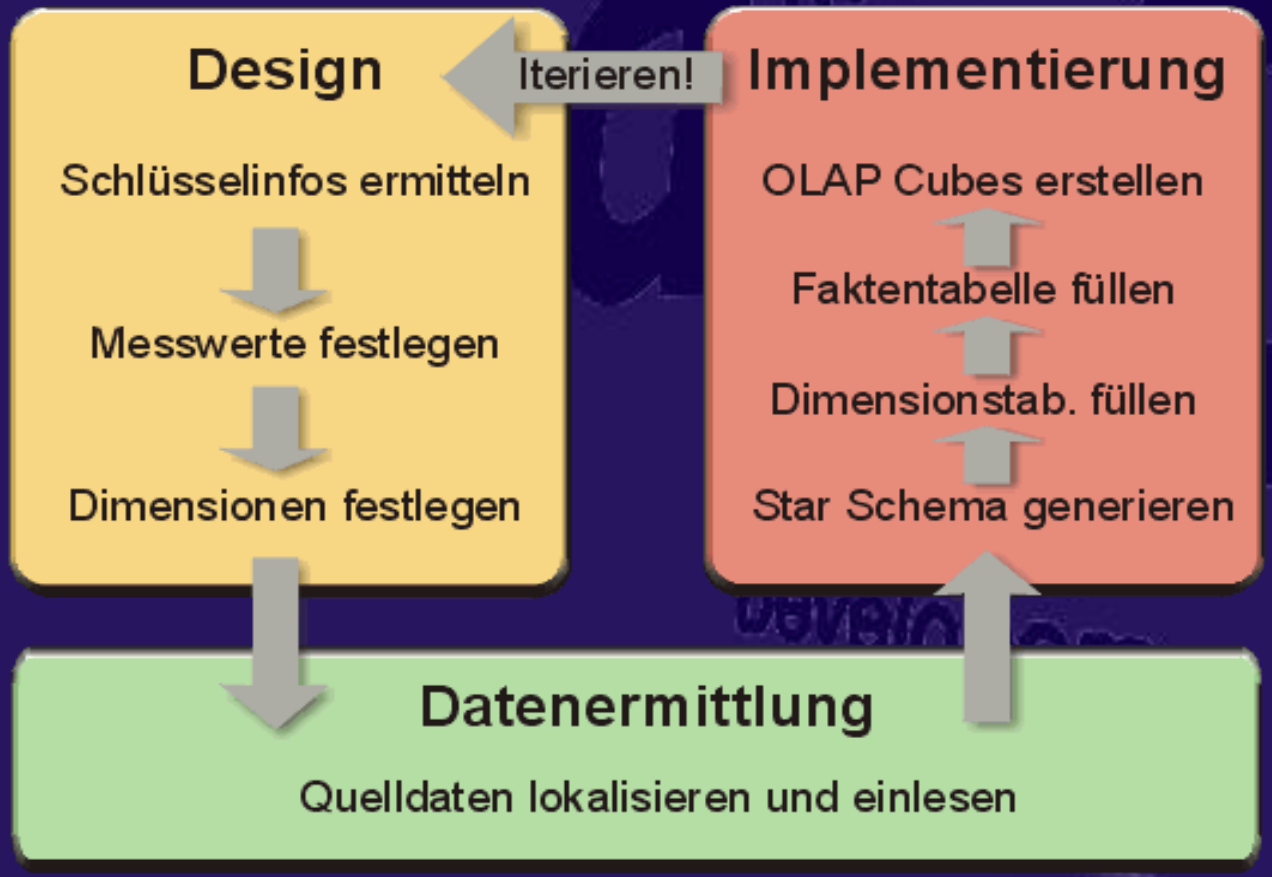
Überblick

- **Vorstellung der Beispielanwendung**
- **Prozeß zur Erstellung eines Data Marts**
  - Design
  - Datenermittlung
  - Implementierung
- **Erläutert am Beispiel-Mart**

Beispielanwendung

- **Einfache Zeiterfassung**
- **Erfassung von Arbeitsstunden**
  - Pro Mitarbeiter
  - Pro Projekt
  - Gegliedert nach Aufwandsarten
- **„Backend“: Flat-File (dBase)**

Prozess zur Erstellung eines Data Marts

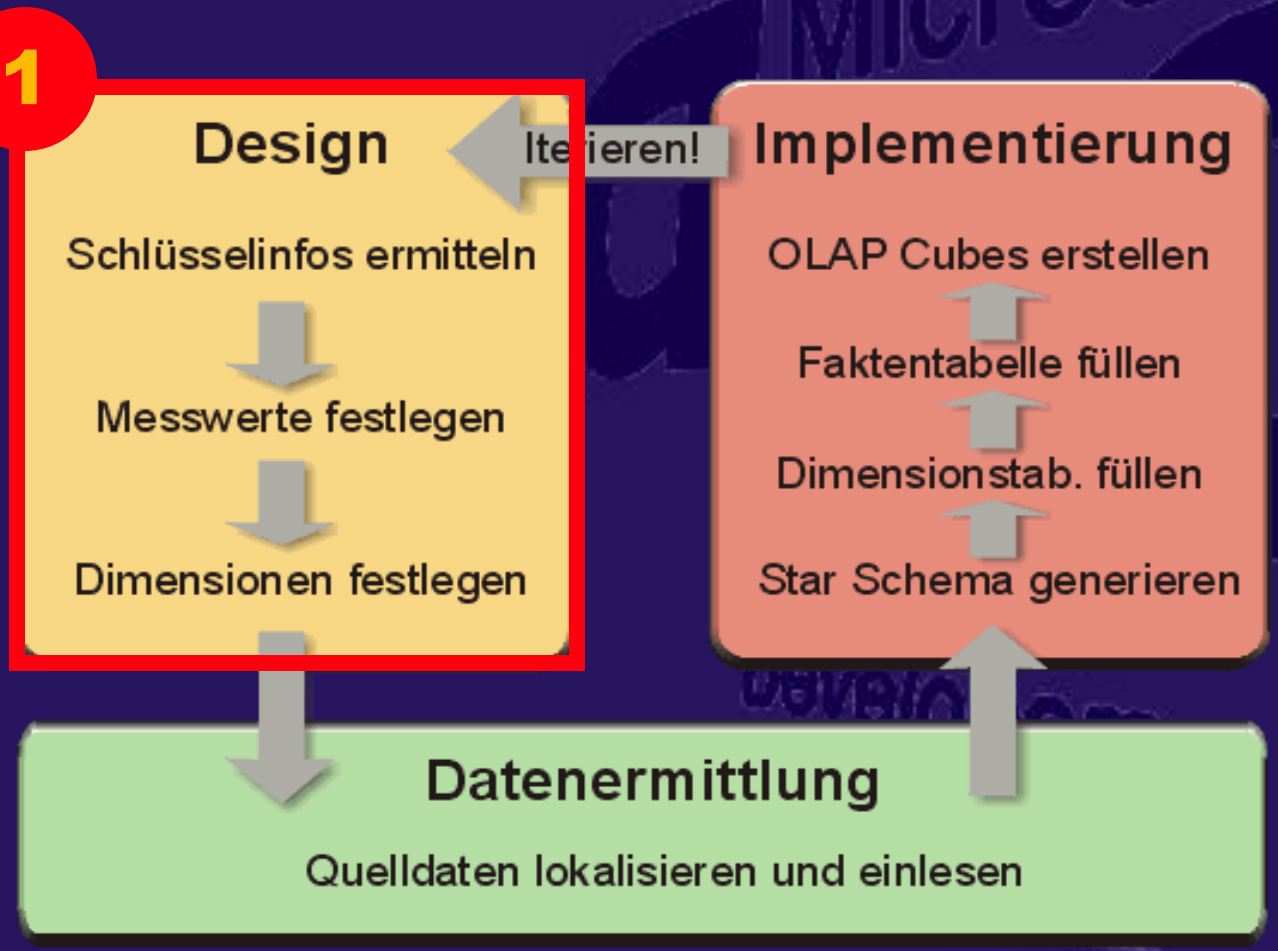


Ziel des Prozesses

- Gesteigerter Unternehmens**erfolg**
- „Wissens-**infrastruktur**“ **herstellen**

Die Anwender brauchen die „richtigen“ Daten!

Design


1
Design
Iterieren!
Schlüsselinfos ermitteln
Messwerte festlegen
Dimensionen festlegen
Implementierung
OLAP Cubes erstellen
Faktentabelle füllen
Dimensionstab. füllen
Star Schema generieren
Datenermittlung
Quelldaten lokalisieren und einlesen

Schlüsselinfos ermitteln

- Geschäftsprozesse analysieren
- Bestehende Systeme und deren Anwendung untersuchen
- Interviews mit Endanwendern und DBAs
- Ergebnis: Verständnis des Umfeldes

Messwerte festlegen

- Welche Granularität soll der Data Mart haben?
- Welche Messwerte sind gewünscht und realisierbar?
- Beispiel: „Verkaufte Stück“ und „Umsatz“
- Problematisch: Gewinn und Planzahlen

Exkurs: Planzahlen

- Umsätze und Stückzahlen liegen meist mit sehr feiner Granularität vor (z.B. Auftragspositionen)
- Planzahlen sind nur für höhere Ebenen verfügbar
- Beispiel: Geplanter Umsatz pro Region

Exkurs: Planzahlen

- **Lösungsansatz A)**
  - Planzahlen auf kleinste Detailebene herunterrechnen
  - Resultat meistens kompletter Blödsinn
- **Lösungsansatz B)**
  - Separate Fakt-Tabelle mit voraggregierten Ist-Zahlen

Dimensionen festlegen

- **Interviews auswerten**
- **Datenbestände der operationalen Systeme sichten**
- **Vorhandene Stammdaten gute Quelle für Dimensionen**
- **Gewünschte, aber fehlende Daten können ergänzt werden**

Dimensionen festlegen

- **Zusätzliche Entscheidungen in der Design-Phase:**
  - Welche Dimensionen sind dynamisch (Slowly changing dimensions)?
  - Welche zusätzlichen Attribute sind für eine Dimension sinnvoll (member properties → virtual dimensions)?
  - Umfang noch machbar?

Exkurs: Slowly Changing Dimensions

- **Beispiel (siehe Kimball):**
  - Kunden-Dimension
  - Kunde „Mary Jones“
  - Bis 15.01.1999 ledig (MStatus „L“)
  - Ab 15.01.1999 verheiratet (MStatus „V“)
  - Annahme: Name bleibt gleich
- **Was nun?**

Exkurs: Slowly Changing Dimensions

- **Nach Ralph Kimball 3 Typen:**
- **Typ 1: Feld MStatus wird im existierenden Satz in der Dimensionstabelle überschrieben**
- **Vorteil: Einfach zu implementieren**
- **Nachteil: Historie geht verloren**

Exkurs: Slowly Changing Dimensions

- Typ 2: Neuer Satz mit neuem MStatus wird in Dimensionstabelle angelegt
- Vorteil: Historie bleibt erhalten
- Nachteil: Künstliche Primary Keys sind notwendig
- Typ 3: Meist nicht relevant

Design des Beispiel-Marts

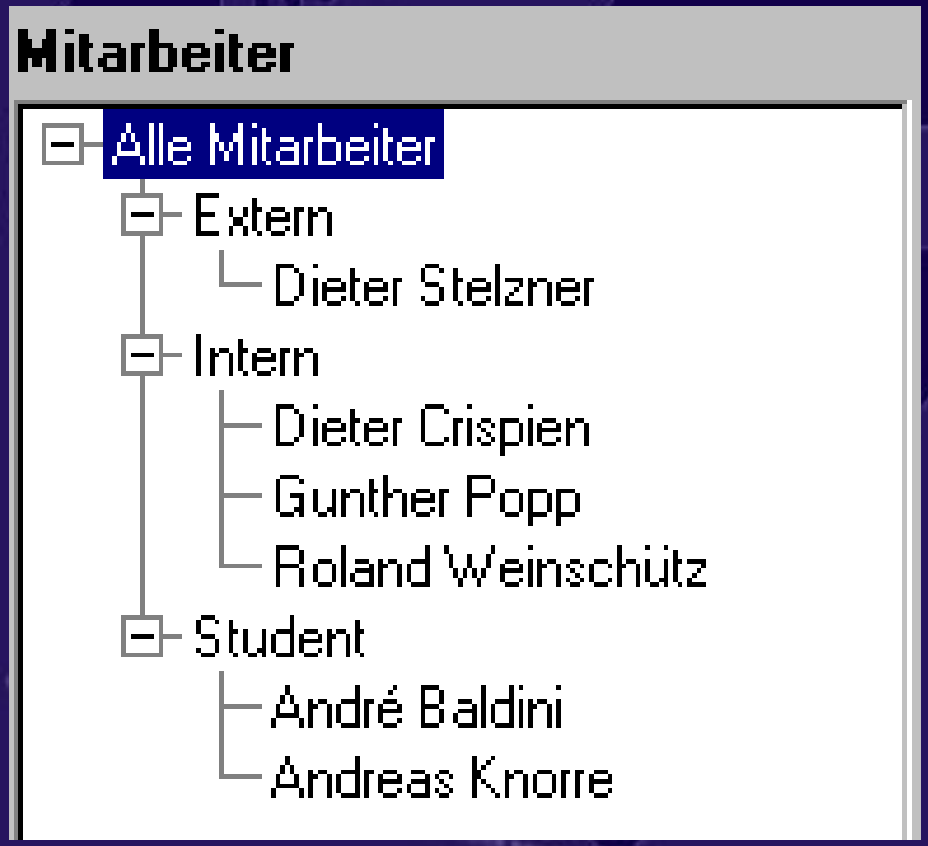
- **Bisherige Analysen über Excel und Pivot-Tabellen**
- **Schlüsselinformationen**
  - Aufwand pro Mitarbeiter und Projekt
  - Aufwand pro Projekt und Monat
  - Aufwand pro Projekt und Aufwandsart

Design des Beispiel-Marts

- **Messwerte festlegen:**
  - Mannstunden
  - Manntag als berechneter Wert (Mannstunden / 8)

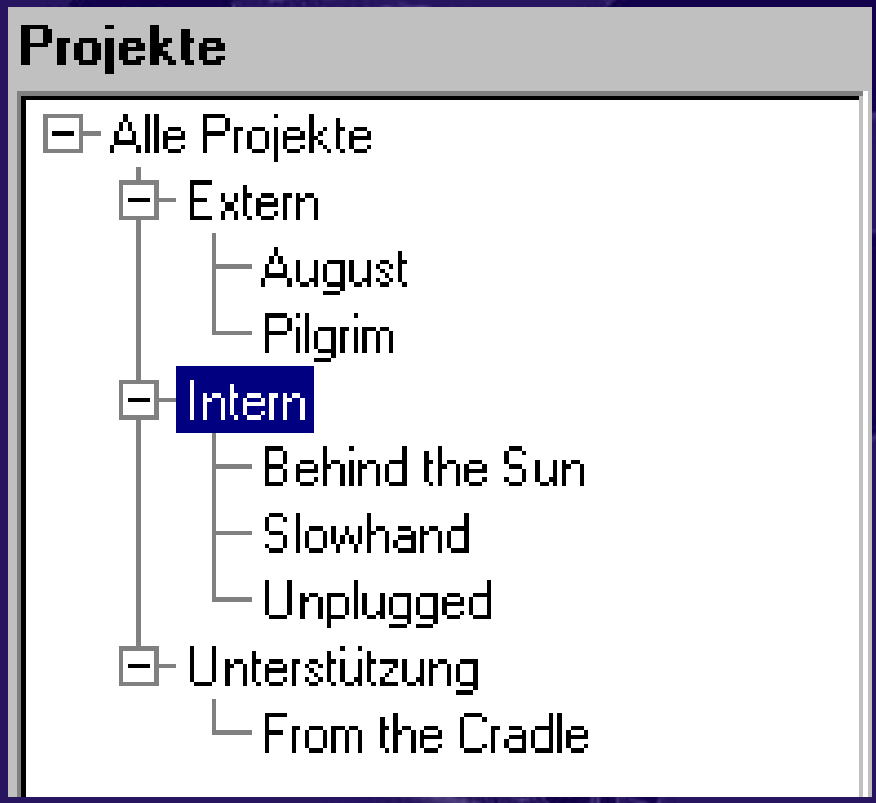
Design des Beispiel-Marts

- **Dimension „Mitarbeiter“**
  - MA-Status
  - MA-Name
- **Slowly Changing Typ 1?**



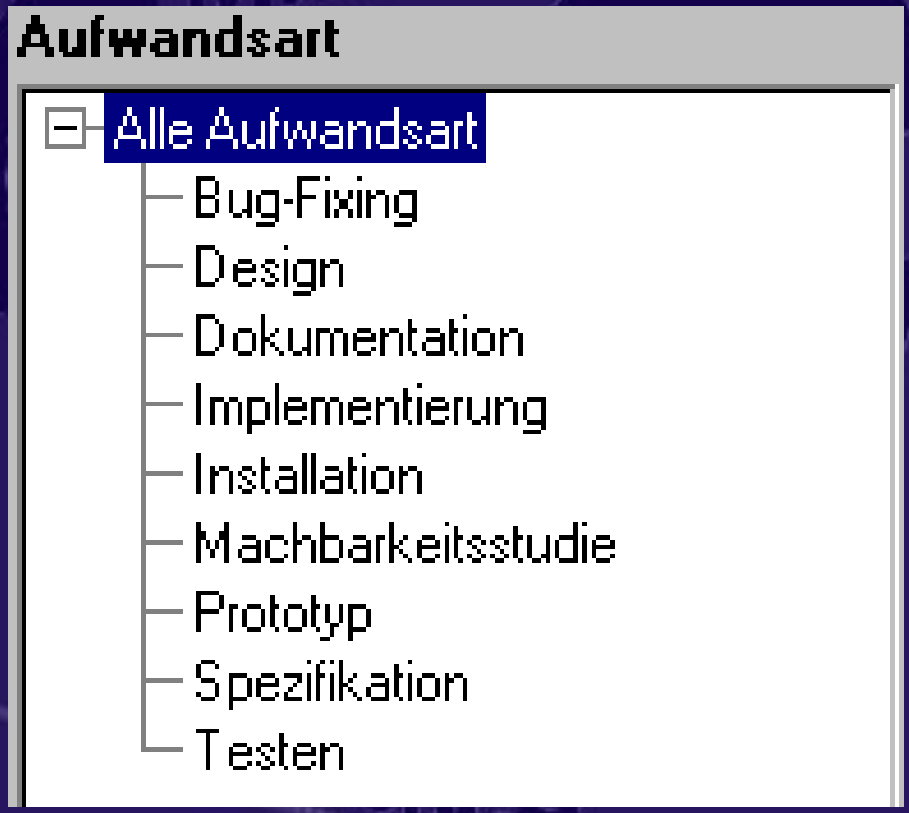
Design des Beispiel-Marts

- **Dimension „Projekte“**
  - Projektstatus
  - Projektname
- **Slowly Changing Typ 1?**



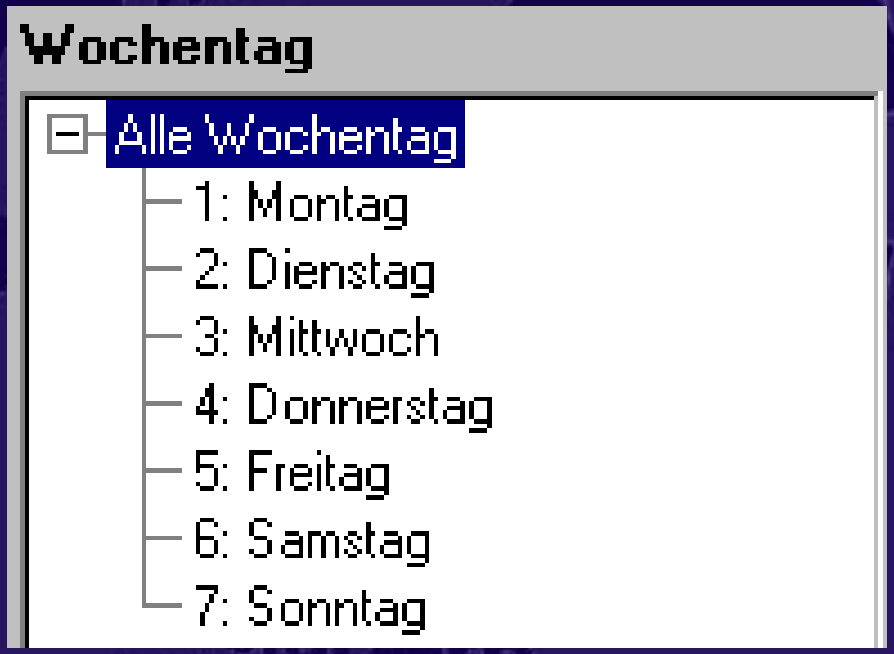
Design des Beispiel-Marts

- **Dimension „Aufwandsart“**
  - Aufwandsart
- **Fixe Dimension**



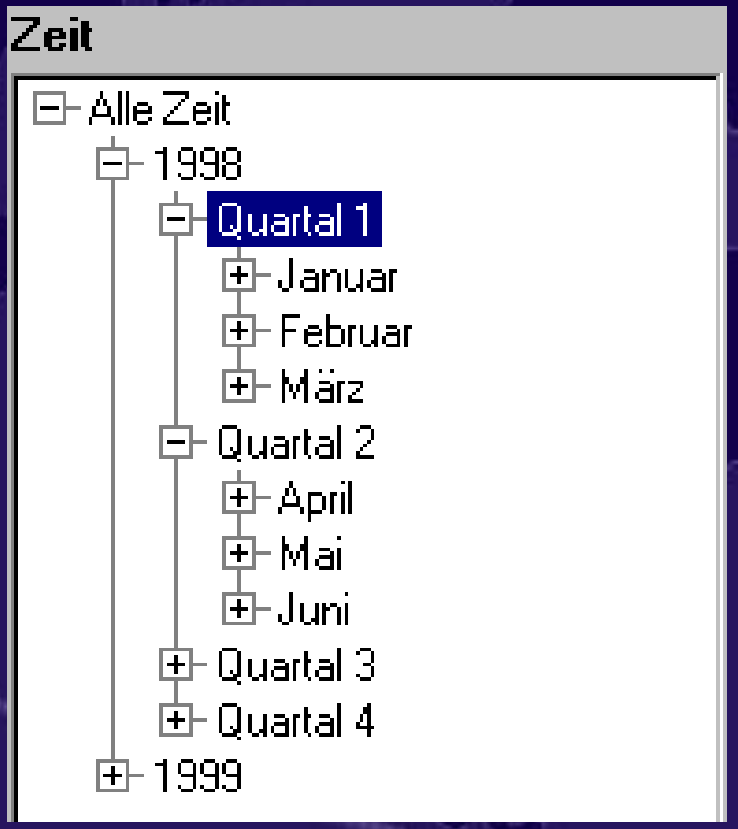
Design des Beispiel-Marts

- **Dimension „Wochentag“**
  - Wochentag
- **Fixe Dimension**

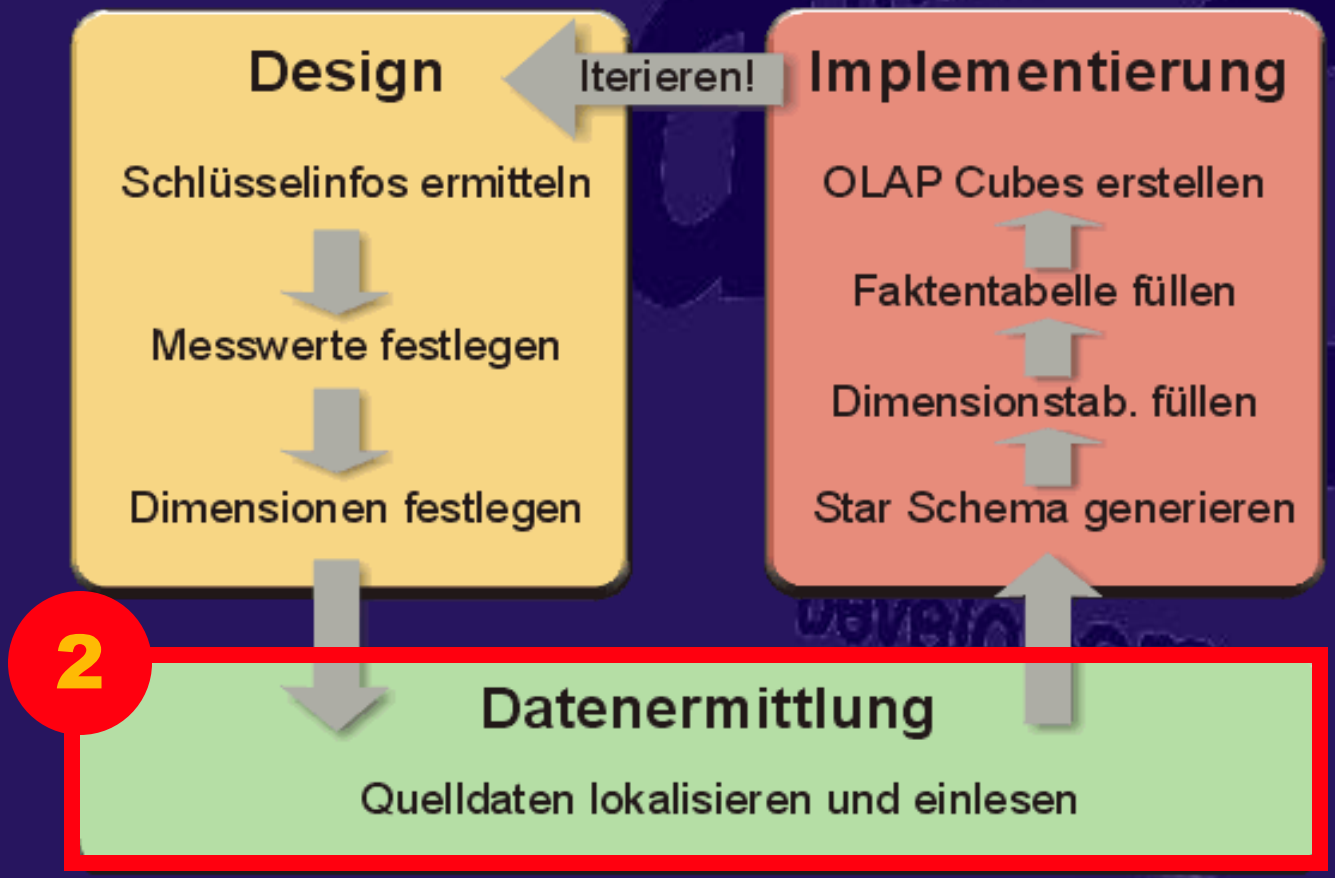


Design des Beispiel-Marts

- **Dimension „Zeit“**
  - Jahr
  - Quartal
  - Monat
  - Tag
- **Fixe Dimension**



Datenermittlung



Quelldaten lokalisieren ...

- **Definitiv komplexer als es klingt ...**
  - PC oder Mainframe ?
  - Administrator finden
  - Datenmodell finden
  - Datenformat identifizieren
  - Aktualität der Daten bestimmen
  - Zugriffsmethode festlegen

... und einlesen

- Zugriff über OLE DB oder ODBC
- Im Notfall: Textdump erstellen (lassen) und diesen einlesen
- Schon an dieser Stelle stichprobenweise die Qualität analysieren (Fehlende Schlüssel, Leere Sätze, etc.)

Quelldaten des Beispiel-Marts

- **Datei: TimeDB.DBF (dBASE-File)**

Nr.	Feldname	Typ	Länge	Dez
1	ID	Charakter	16	0
2	NAME	Charakter	40	0
3	PROJECT	Charakter	40	0
4	ADATE	Datum	8	0
5	ACLASS	Charakter	40	0
6	ADURATION	Numerisch	2	0
7	ADESC	Charakter	40	0

Spaltenbeschreibung

- **ID**
  - Eindeutige, künstlich generierte ID (Primary Key)
  - Zusammengesetzt aus einem Timestamp und den Initialen des Mitarbeiters
- **Name**
  - Mitarbeitername (Vor- und Nachname)

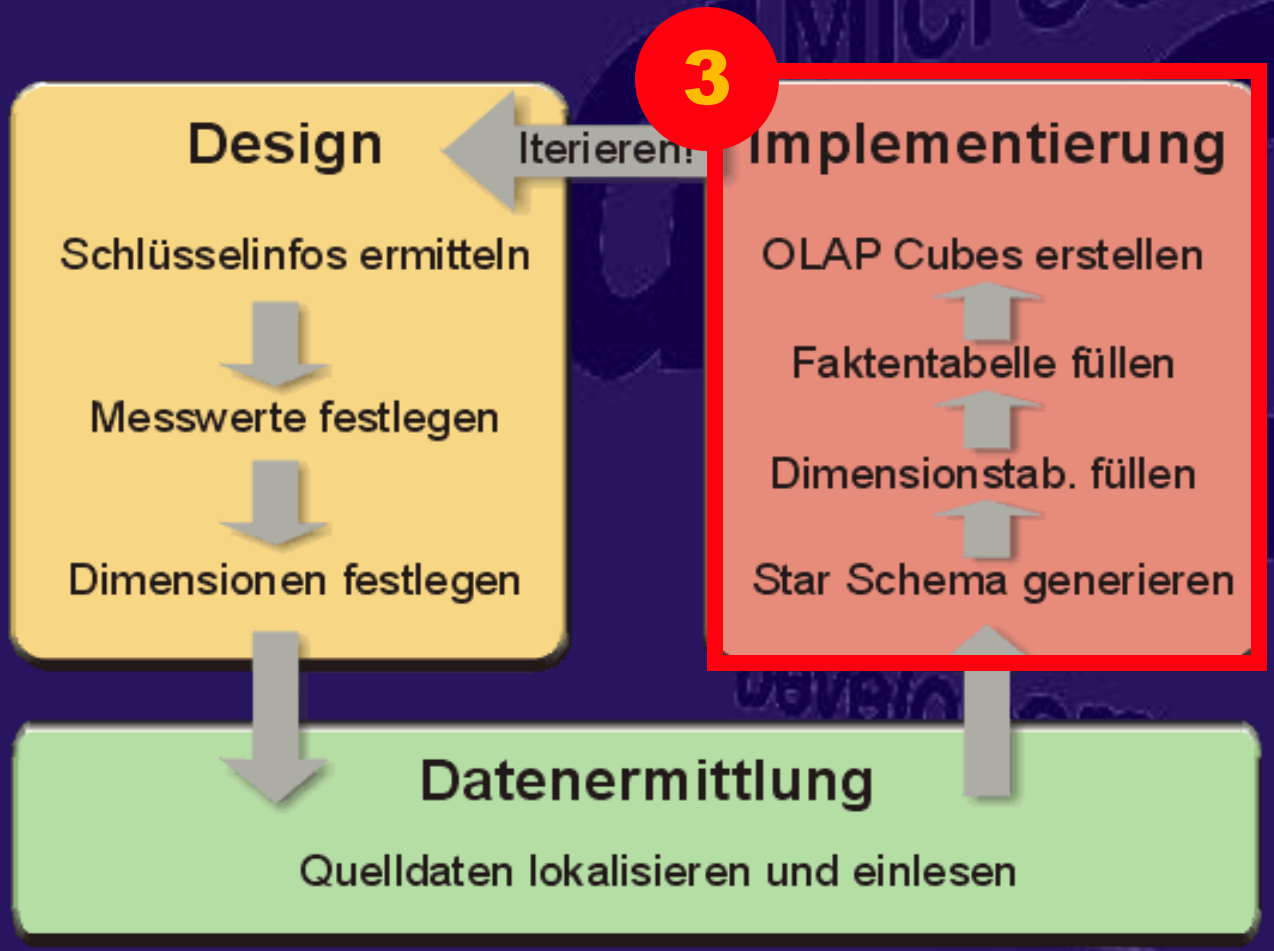
Spaltenbeschreibung

- **Project**
  - Projektname
- **Adate**
  - Datum, an dem eine Leistung erbracht wurde
- **Aclass**
  - Aufwandsart
  - Beispiel: Design, Implementierung, ...

Spaltenbeschreibung

- **Aduration**
  - Erbrachte Leistung in Arbeitsstunden
  - Nur ganzzahlige Werte erlaubt!
- **Adesc**
  - Optional: Kurze Beschreibung der Leistung

Implementierung



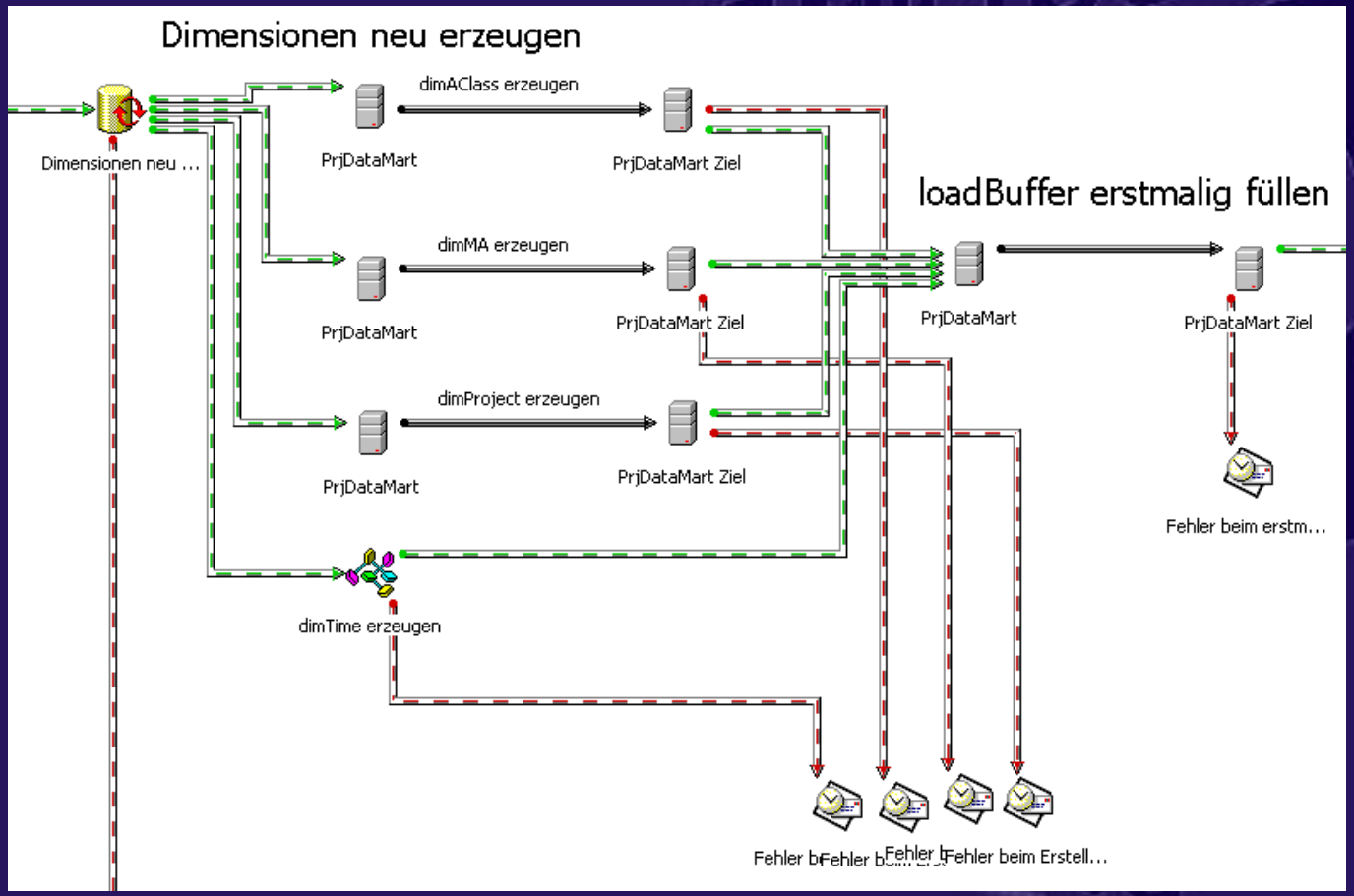
Implementierung: ETL

- **ETL-Prozess**
  - Extraktion
  - Transformation
  - Laden
- **Optimal geeignet: DTS (Data Transformation Services)**
- **Bestandteil von MS SQL 7.0**

DTS: Überblick

- **Package: Sammlung von Connections, Tasks und Steps**
- **Im Prinzip Ersatz für Batch-Files mit BCP Aufrufen**
- **Graphische Editoren:**
  - Import/Export Wizard
  - DTS Designer

Beispiel: DTS-Package



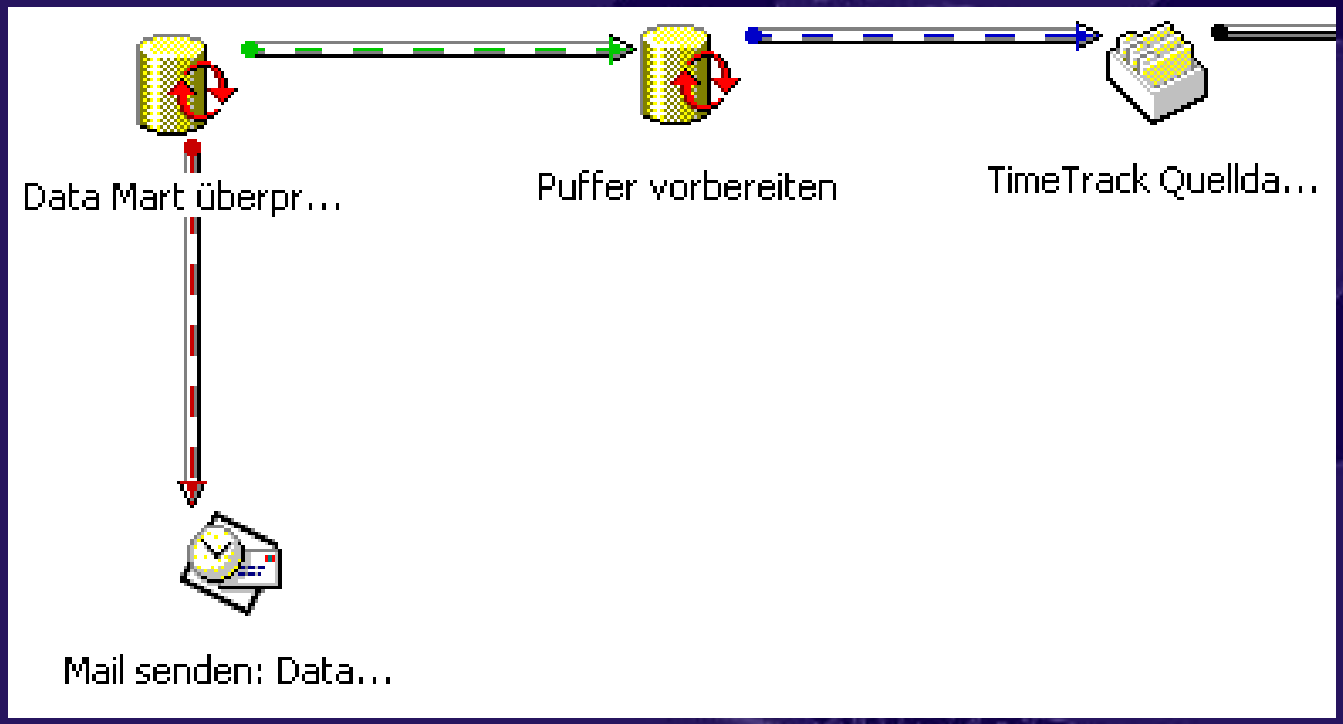
DTS: Task-Typen

- Data Pump Task
- Active Script Task
- Create Process Task
- Execute SQL Task
- Data Driven Query (DDQ) Task
- Transfer Objects Task
- Bulk Insert Task
- Send Mail Task
- Process OLAP Task (separat erhältlich)

DTS: Steps

- **Tasks sind durch Steps verbunden**
- **Rangfolge pro Step**
  - Bei Erfolg (grüner Pfeil)
  - Bei Fehler (roter Pfeil)
  - Bei Beendigung (blauer Pfeil)
- **Legt fest unter welchen Umständen ein Task ausgeführt wird**

DTS: Steps


Data Mart überpr...
Puffer vorbereiten
TimeTrack Quellda...
Mail senden: Data...

DTS: Connections

- Connections stellen die Verbindung zwischen Package und Daten her
- Pro Connection kann zeitgleich nur ein SQL Statement ausgeführt werden

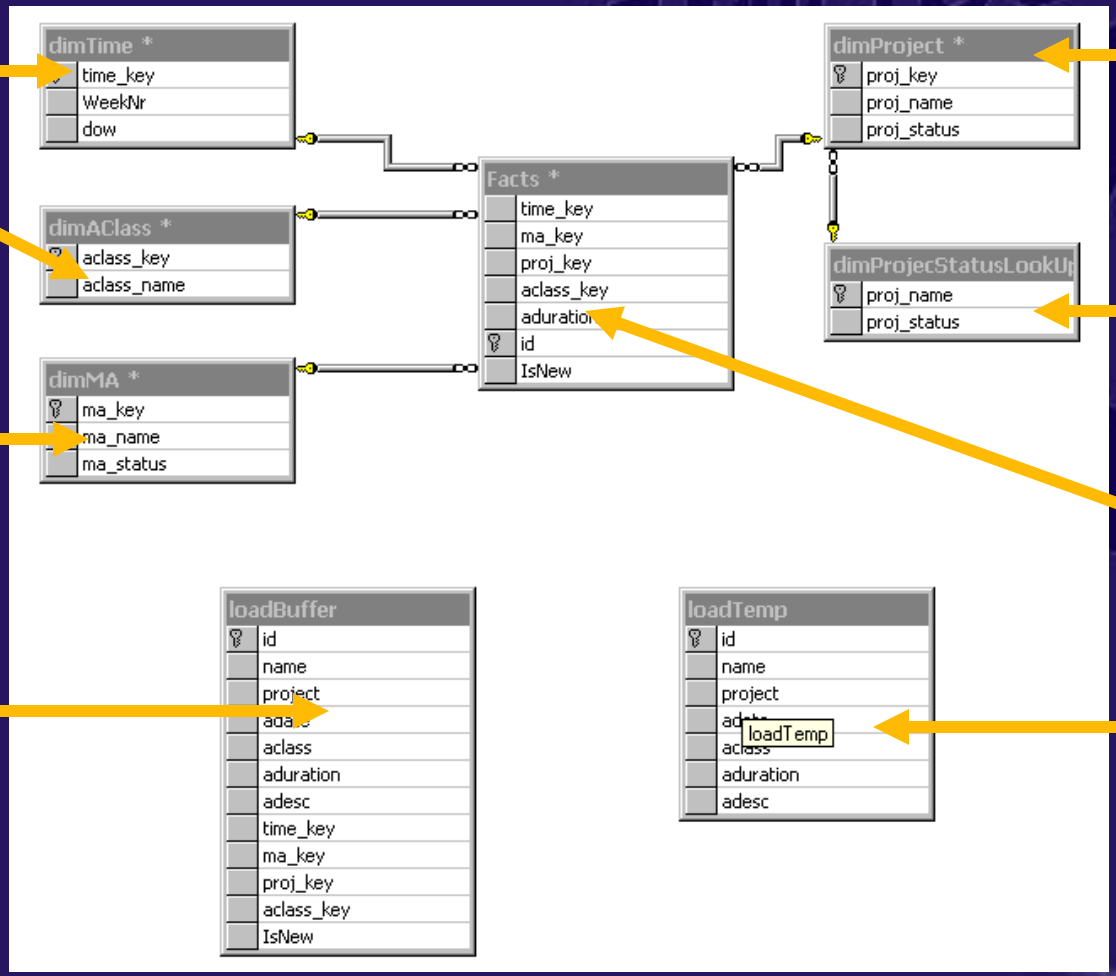
Implementierung des Beispiel-Marts

- **Star-Schema wird mit SQL-Skript angelegt**
- **ETL-Prozeß wird von einem DTS-Package durchgeführt**

Star Schema anlegen: Wichtige Entscheidungen

- **Schlüssel von Fakt- und Dimensions-Tabellen**
- **2 Möglichkeiten:**
  - Vorhandene Schlüssel aus Quelldaten übernehmen
  - Neue Schlüssel generieren
- **Im Beispiel: Neue Schlüssel werden generiert**

Star Schema des Beispiels



Zeit

Aufwands-art

Mitarbei-ter

Archiv

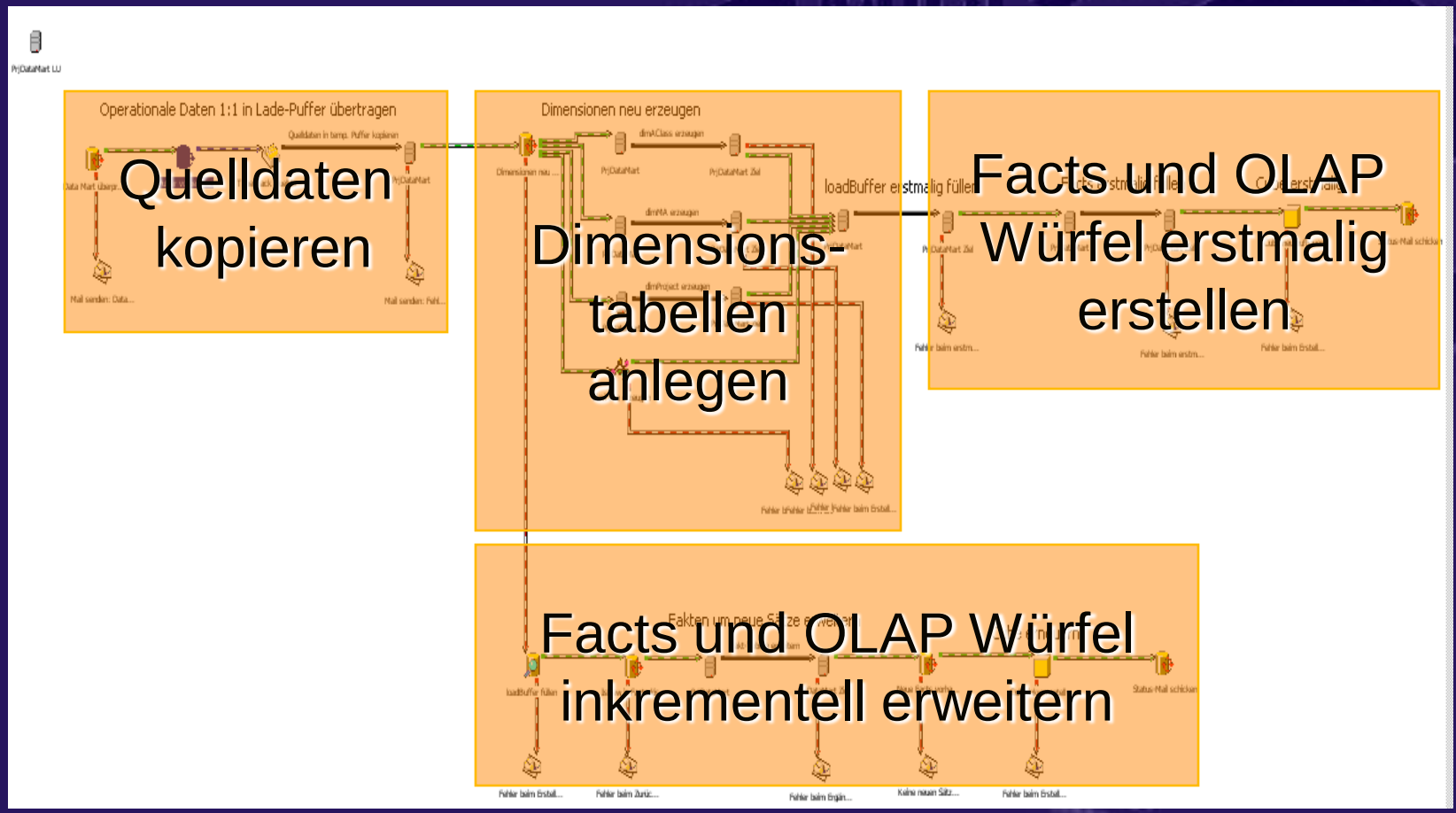
Projekte

Projekt-status

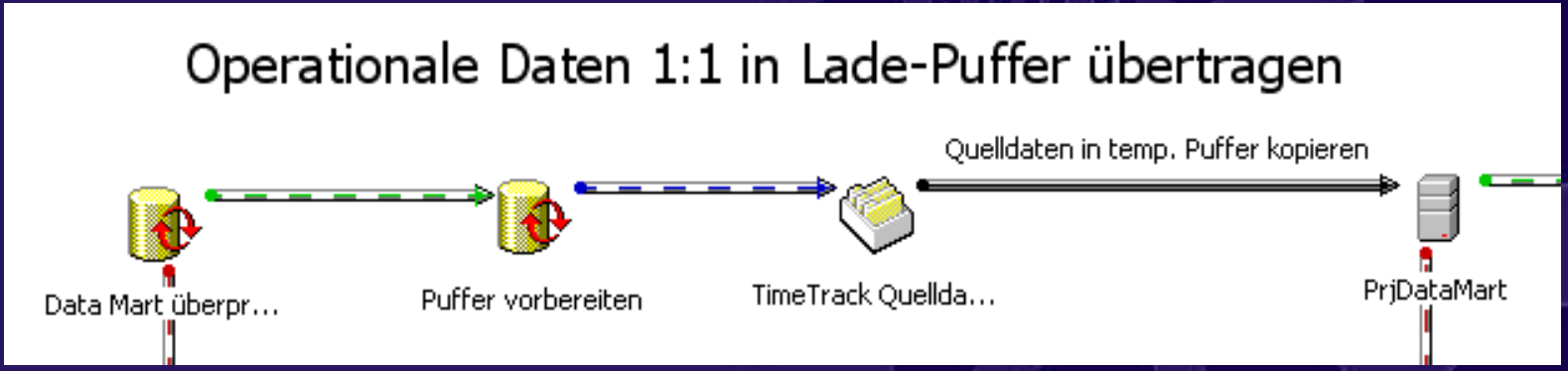
Fakten

Temp. Puffer

DTS-Package: Überblick



Quelldaten kopieren



SQL-Task: Data Mart überprüfen:

```
IF not Exists( Select * from master..sysdatabases
   where name = 'TimeTrack' )
   Select * from xxx

IF not Exists( Select * from TimeTrack..sysobjects
   where name = 'Facts' )
   Select * from xxx
```

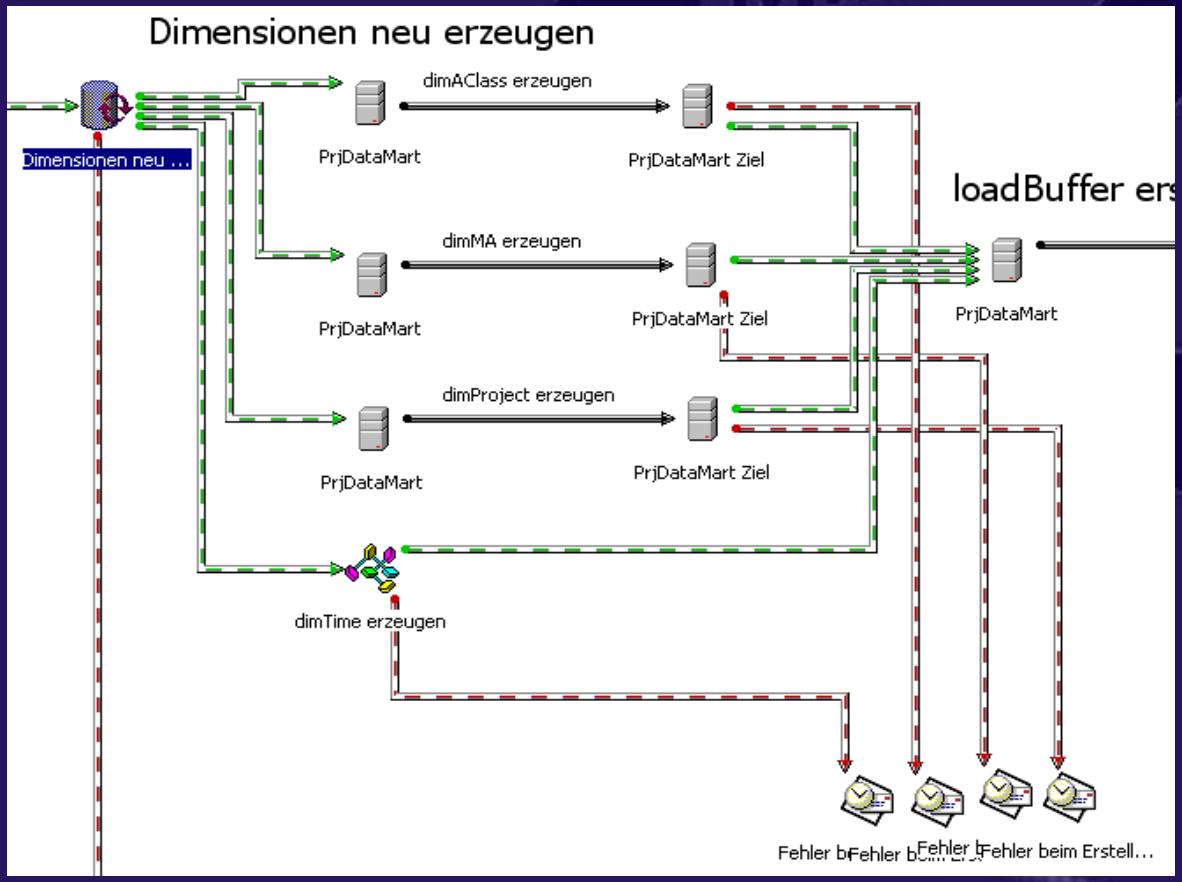
Dimensionstabellen anlegen

- Wird nur einmalig durchgeführt: Slowly Changing Dimension nicht realisiert
- Entscheidung über SQL-Task:

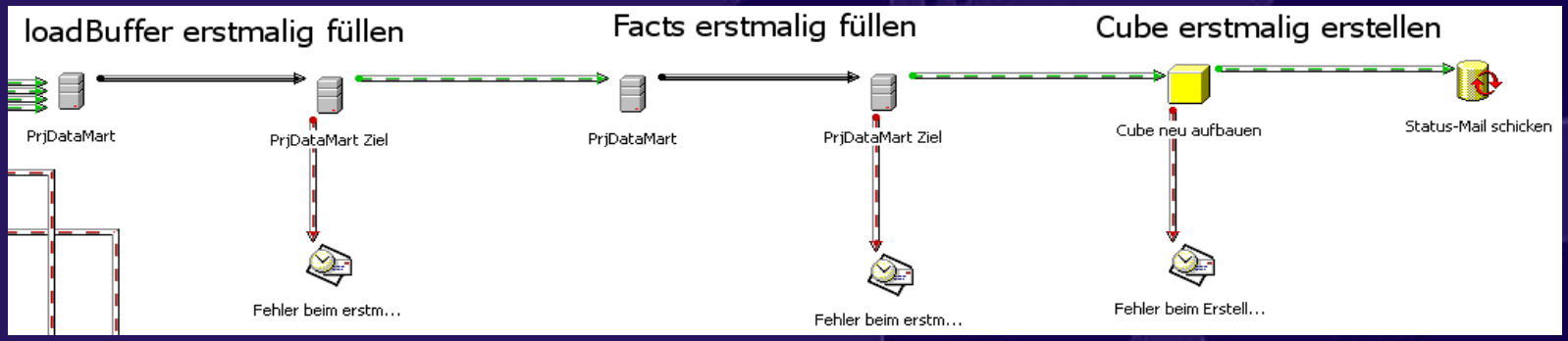
SQL-Task: Dimensionstabellen neu erstellen?

```
/* Wenn KEINE Dimensionssätze vorliegen müssen die
   Dimensionen neu aufgebaut werden */
if (select count(*) from dimProject) > 0
   select * from xxx
```

Dimensionstabellen anlegen



Facts & OLAP Würfel erstmalig erzeugen



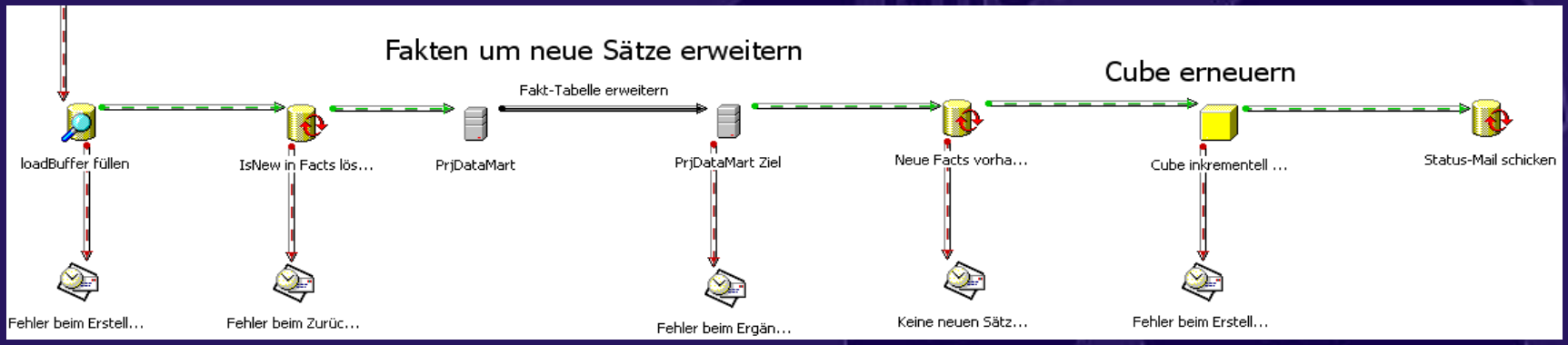
SQL-Task: Status-Mail schicken

```
DECLARE @nNewFacts AS INT
DECLARE @strMsg AS varchar(50)

SET @nNewFacts = ( SELECT COUNT(*) FROM Facts)
SET @strMsg = 'Es wurden ' + RTRIM( CONVERT( char, @nNewFacts ) ) + '
   Sätze geladen'

EXEC master..xp_sendmail @recipients = 'Gunther Popp', @message =
   @strMsg, @subject = 'PrjDataMart: Cube wurde erstellt.'
```

Facts & OLAP Würfel inkrementell erweitern



SQL-Query für Task „loadBuffer füllen“:

```
SELECT loadTemp.*, dimTime.time_key AS TKey, dimAClass.aclass_key
AS ACKey, dimMA.ma_key AS MAKey, dimProject.proj_key AS PKey
FROM loadTemp
LEFT OUTER JOIN dimMA ON loadTemp.name = dimMA.ma_name
LEFT OUTER JOIN dimAClass ON loadTemp.aclass = dimAClass.aclass_name
LEFT OUTER JOIN dimProject ON loadTemp.project = dimProject.proj_name
LEFT OUTER JOIN dimTime ON loadTemp.adate = dimTime.time_key
```

Literatur

- **Ralph Kimball: „The Data Warehouse Toolkit“**
  ISBN 0-471-15337-0
- **Internet:**
  - http://www.olapreport.com
  - http://www.rkimball.com
  - News://msnews.microsoft.com/microsoft.public.sqlserver.olap
  - News://msnews.microsoft.com/microsoft.public.sqlserver.datawarehouse

Building Killer Applications!