Ziele des Workshops

Navigator: Überblick über die Softwareentwicklung aus Sicht des Datenbank-Umfeldes mit Schwerpunkt ORACLE.

Teil 1 - Grundlagen:

- Orientierung zum weitergehenden Studium (Literatur, Fachartikel ...)
- Wiederholung und Übersicht von DB-Basiswissen

Teil 2 - Design, Aktuelle Themen und Werkzeuge:

- Allgemeine Werkzeuge: SQL Plus, Enterprise Manager und TOAD
- Designwerkzeuge -Vergleich: Erwin, Power Designer, Designer/2000

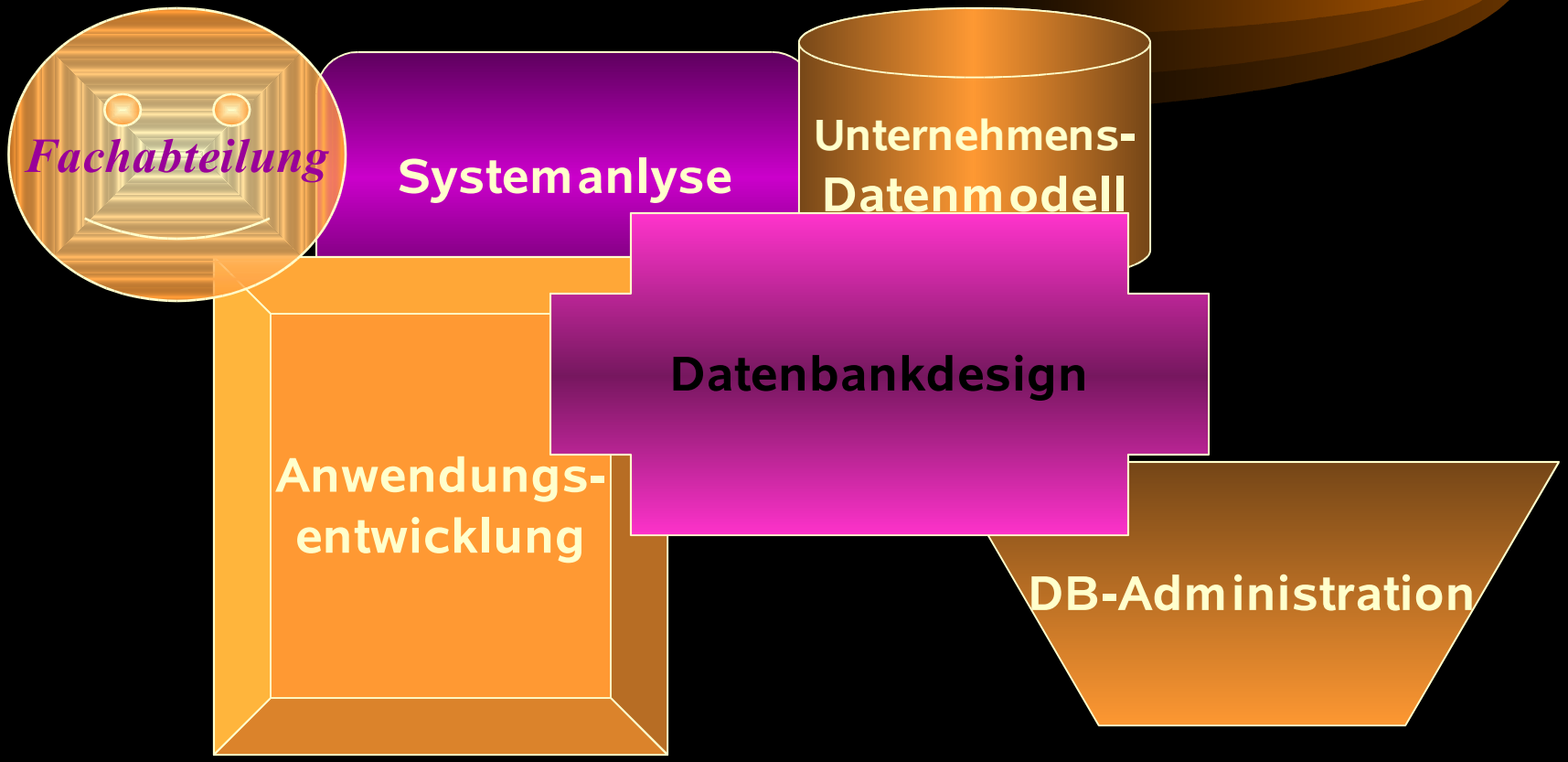
Teil 3 - Design und aktuelle Themen:

- Designprobleme
- Objektorientierung - Mapping / Persistenzschicht
- Installation und Software-Lieferung

Ziele Relationaler DB / ERM

- Unabhängigkeit von der Anwendung / Unterstützung verschiedener Anwendungen und Werkzeuge
- Unabhängigkeit von der Physik
- Einheitliche Zugriffssprache
- Transparente Struktur
- Freie Kombinierbarkeit von Tabellen
- Redundanzfreiheit durch Normalformen
- Verwendung von (fachlichen) Fremdschlüsseln
- *Neu:* Referenzielle und fachliche Integrität

Datenbankdesign: Positonierung



Gegensatz: OO-Prinzipien

- Objekte sind autonom und enthalten Daten und Methoden.
- Kapselung, Vererbung, Polimorphie
- Objekte können sich gegenseitig über aufrufen
- Eingeschränkter Zugang unterschiedlicher Programmiersprachen (z.B. CORBA)
- Keine freie Kombinierbarkeit

Gemeinsamkeiten OO / ERM

- Implementierbare Abbildung der Wirklichkeit / Problemfeld
- Integrität / Redundanzfreiheit / Konsistenz
- Wartbarkeit und Transparenz
- Performanz

Technische Grundbegriffe RDBMS

- RDBMS: Relationales DatenBank Management System
- SQL Schnittstelle mittels Client-Komponenten: ODBC oder Native Drivers
- Verwaltungsfunktionen für den Administrator (DBA):
  - Verwaltung der Physik, DB-Objekte, User / Rechte
  - Überwachung des Betriebes / Problembhandlung
  - Tuning und Sicherung
- Hauptkonzept: Tabellen (Entity, Relation) mit Zeilen (row, record, Tupel) und Spalten (column, Attribut, Feld) sind über logische Beziehungen verknüpfbar (relationship, join)

Connection / DB Session

- Eine Datenbank ist eine konfigurierte, laufende Instanz eines RDBMS und stellt Dienste als Server bereit.
- Auf der gleichen Maschine oder über das Netzwerk muß sich eine Client-Anwendung an der Instanz anmelden.
- Mit CONNECT User/Password@Instanz wird diese Session eingeleitet, und mit DISCONNECT beendet.
- SQL Net*2, bzw. NET /8 stellt die Kommunikations-Infrastruktur bereit (TNSNAMES.ORA)
- Es können mehrere parallele Sessions von einer Anwendung aus verwendet werden.

Grundbegriffe SQL

- Stuctured Query Language - gesprochen auch Sequel - entwickelt von Codd
- DDL: Data Definition Language: CREATE, ALTER und DROP (DB-Objekt) Statements
- DML: Data Manipulation Language: SELECT, INSERT, UPDATE, DELETE
- DCL: Data Control Language: GRANT und REVOKE
- PL/SQL: Prozedurale Oracle Erweiterungen

Grundbegriffe DDL

- Indirekte Pflege des DB-Catalog = interne Verwaltungstabellen
- CREATE: Erstellt Tabellen, Indexe, Views, Synonyme, Tablespaces, Sequences, User, Trigger, Stored Procedures, Packages, User Defined Functions etc.
- ALTER: Ändert die Struktur der DB-Objekte: Attribute, Constraints, Parameter etc.
- DROP: Entfernen der DB-Objekte
- Sonderstatements: TRUNCATE, ANALYSE ...

Grundbegriffe Tabelle - Beispiel

Primärschlüssel (PK)	*Bezeichnung*	*Fremdschlüssel (FK)*
Kundennummer	Name	Adressverweis
Position	Pos-Text	Rechnungsnr
Rechnungsnr	Datum	Kundennummer

Schlüssel & Attribute

- Eine Entität / Tabelle besteht aus Attributen / Spalten *(Felder, Properties)* und Tupeln / Zeilen *(Sätzen, Instanzen)*
- Eine Zeile wird durch ein Attribut, bzw. mehreren zusammengesetzten Attributen eindeutig identifiziert = Primärschlüssel (PK)
- Die Beziehung der Entität A zu einer weiteren Entität B wird hergestellt, wenn der PK von B zu Attribut(en) in A werden = Fremdschlüssel (FK).
- Eine Entität ist unabhängig, wenn ihr PK sich nicht aus einem FK zusammensetzt.

Wahl der Schlüssel

Schlüssel - vor allem Primärschlüssel (PK) - sollten durch ein (projektspezifisches) Konzept gewählt werden.
Alternativen:

- Fachliche Schlüssel: Eindeutig, konstant / unveränderlich, kompakt.
- Künstliche Schlüssel (Artificial Key AK, Surogate Key ..) durch generierte Nummern (Sequence).
- Zusammengesetzte Schlüssel, z.B. Historsierung ...
- Kind-Tabellen: PK mit Fremdschlüssel (FK) vs. AK.
- Kriterien: Abbildung, Performance und Handhabbarkeit.

Attribute, Domains und Constraints

- Ein Attribut gehört immer zu einer Domain = Zulässiger Wertebereich.
- Die meist verwendeten Domains bestehen aus den Grunddatentypen:
  - Zeichen / (variable) Zeichenkette (Länge) / Char / Varchar
  - Zahlen: Ganzzahlen, Dezimalzahlen (Länge, Nachkomma)
  - Datum - Zeit
- Domains werden näher spezifiziert durch Constraints:
  - NULL / Not NULL - Constraint
  - CHECK Constraints (statische Einschränkung des Bereiches)
  - Primary Key / PK (UNIQUE) Constraint
  - Foreign Key / FK Constraint -> Deklarative Referenzielle Integrität (DRI)

NULL

- NULL ist ein spezieller Attribut-Zustand, der einen fehlenden Wert kennzeichnet.
- Fehlende Werte deuten an, das der Wert des Attributes unbekannt ist oder im Kontext nicht relevant ist (z.B. Schwanger bei Männlich)
- NULL ist ungleich 0 (Zero) und ungleich ‚' (Leerstring).
- NULL wird in SQL besonders behandelt (z.B. IS NULL)
- NULL-Sachverhalte können in Anwendungen z.T. auch alternativ kodiert werden. Z.B. 1 = Männl. 2 = Weibl. 0 = Unbekannt.
- NULL werden in Oracle-Indexen nicht geführt --> Optimierungsmöglichkeiten für Statusinformationen
- NULL / NOT NULL - Constraint erlaubt oder verbietet NULL
- DEFAULTS (Standardwert) erleichten den Einsatz von NOT NULL

Referenzielle Integrität (RI)

- Sichert die Konsistenz in der Beziehung zwischen Entitäten
- Deklarative RI (DRI) über Foreign Key Constraints nach ANSI-92:
  - Grundsätzlich: Keine abhängige Zeile ohne Parent.
  - Restrict - Läßt keine Löschung oder Änderung des Parent-Keys zu, solange noch abhängige Zeilen vorhanden sind.
  - Cascade - Zieht ggf. Änderungen in den FK der abhängige Zeilen automatisch nach, bzw. löscht abhängige Zeilen
  - Set Null - setzt den FK der abhängigen Zeilen ggf. auf NULL
- Alternative 1: Implementierung als Trigger - z.B. durch Generator - Flexiblere Regeln, erweiterte Business Rules ...
- Implementierung als Batch-Prozess: Ablauf zu unkritischen Zeiten, jedoch häufig logische und zeitliche Probleme.
- Implementierung in der Anwendung: Konsistenz nicht immer sichergestellt.

Business Rules

- Regeln mit dynamischen (flußorientiertem) oder statischen Charakter
- Komplexe Regeln müssen konsistent implementierbar und leicht pflegbar sein: In der Datenbank !?
- Trigger: Implementierbare Prozeduren bei Änderungseignissen (Neu, Update, Löschung) - Vergleich: Event-orientierte Implementierung
- Stored Procedures: Kapselung der Datenhaltung in zentrale Prozeduraufrufe. Einschränkung der Nutzung durch SQL
- Batch-Prozesse zur Prüfung (Fehlerprotokolle) und ggf. automatischer Bereinigung
- Anwendungsnahe Programmierung = Integritäts-Leck bei mehreren schreibenden Anwendungen / Pflege der Regeln
- Datenzugriffsdienste im Sinne von CORBA: Funktional mit Stored Procedures vergleichbar

Ziele des Workshops

Navigator: Überblick über die Softwareentwicklung aus Sicht des Datenbank-Umfeldes mit Schwerpunkt ORACLE.

Teil 1 - Grundlagen:

- Orientierung zum weitergehenden Studium (Literatur, Fachartikel ...)
- Wiederholung und Übersicht von DB-Basiswissen

Am 28.8. Fortsetzung

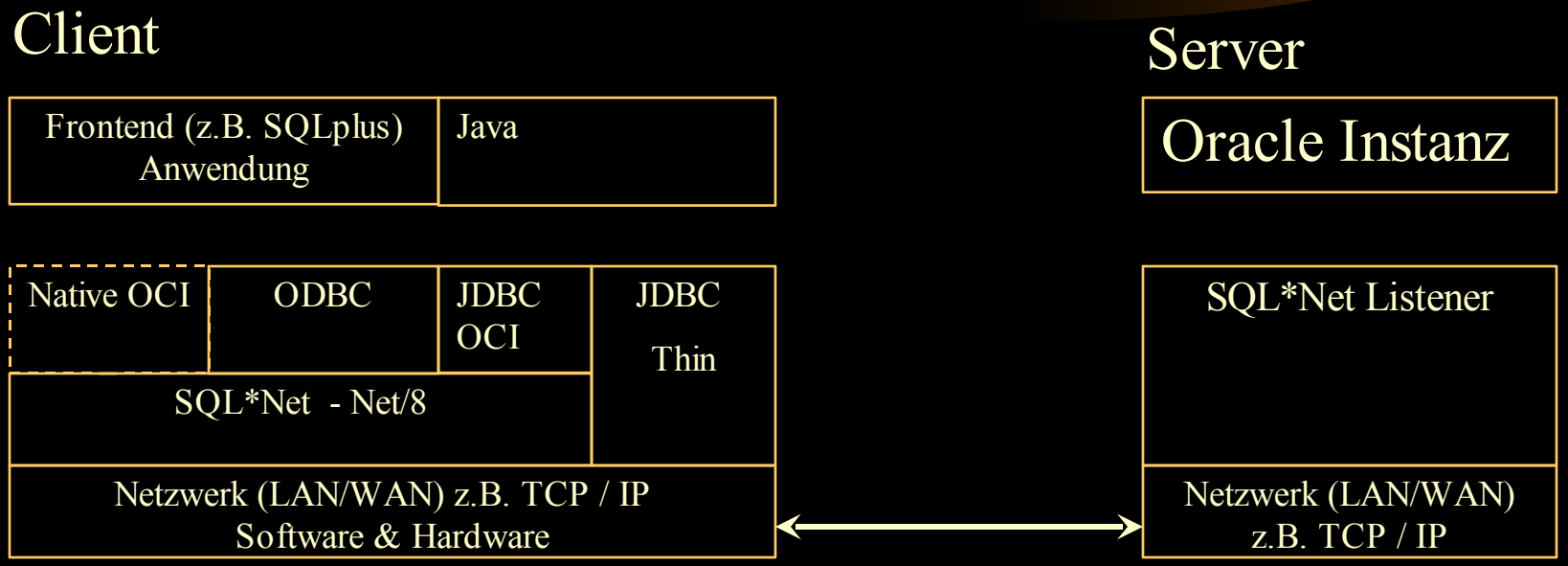
- Allgemeine Werkzeuge: SQL Plus, Enterprise Manager und TOAD

Teil 2 - Design, Aktuelle Themen und Werkzeuge:

- Designwerkzeuge im Vergleich: Erwin, Power Designer, Desgner/2000
- Designprobleme
- Objektorientierung - Mapping / Persistenzschicht
- Installation und Software-Lieferung

Connection über SQL*Net

Konfiguration über TNSNAMES.ORA clientseitig in Oacle-Home\network\admin, (bzw. über \net80\admin für den 8-Client)



Index

- Hilfsobjekt zum beschleunigten direkten Zugriff auf Daten über Schlüssel
- Kann UNIQUEness ermöglichen.
- Indexe werden vom internen Optimizer benutzt und sind für die Anwendung nicht sichtbar (Ausnahme: Hints)
- Indexe werden bei PK automatisch erzeugt und für FK empfohlen
- Indexe lohnen sich für häufig benutzte Zugriffswege auf mittleren bis großen Tabellen.
- Unterschiedliche Arten: Normal, Bitmap (Oracle 8), Index Only, Cluster

Grundbegriffe DML

- SELECT Statement ist das Herz von SQL: Es ermöglicht die Wahl von Spalten und Zeilen einer oder mehrerer Tabellen:
  - Die WHERE -Clause ermöglicht verschachtelte Bedingungen einschließlich JOINS
  - Gruppierungen über Statistik-Funktionen und GROUP BY und HAVING
  - Verschachtelte Sub-SELECTS (correlated & noncorrelated Subqueries)
  - ORDER BY für Sortierungen
  - ***Vertieftes Studium unbedingt empfohlen !***
- INSERT Statement zum Einfügen von Zeilen
- UPDATE Statement zum Ändern vorhandener Zeilen
- DELETE Statement zum Löschen von Zeilen

Cursor

In PL/SQL und verschiedenen Programmiersprachen (embedded SQL) wird neben der Lieferung einer Menge von Tabellenzeilen (Result Set) auf die Ergbnisse eines SELECTS meist über einen DB-Cusor zugegriffen:

- Ein Cursor muß deklariert und vorbereitet werden (DECLARE und PREPARE)
- Die Ergebnisse werden in eine oder mehrere lokale Variablen (Host-Variablen) abgelegt.
- Mit FETCH wird jeweils eine Zeile gelesen.
- Mit CLOSE wird der Cursor geschlossen
- Auch größte Tabellen können sequenziell verarbeitet werden.
- Mit SELECT ... FOR UPDATE werden Locks gesetzt.

Static und Dynamic SQL

- Static SQL:
  - SQL-Statements werden zur Entwicklung / Compile-Time getestet.
  - Auf alle Objekte wird in einem festen Schema zugegriffen.
  - Host-Variablen werden in Where-Clauses verwendet, aber die Struktur des Statements bleibt fest.
  - DB2: Pre-Compiled Statements werden im DB-Katalog abgelegt.
- Dynamic SQL:
  - SQL-Statements werden als Kommando-Strings zur Runtime zusammengebaut.
  - Eingeschränkter Syntax-Check
  - Flexible Abfragen / Variable Schema-Namen.

Transaktionen

- Eine logische Verarbeitungseinheit (Logical Unit of Work - LuW) ist eine Transaktion, die entweder vollständig ausgeführt wird oder garnicht.
- Alle SQL-Statements werden innerhalb einer (impliziten) Transaktion ausgeführt, entweder einzelne Statements oder eine Kette mehrerer Statements.
- Eine Transaktion beginnt mit der DB-Session oder dem Ende der vorherigen Transaktion und wird abgeschlossen durch:
  - COMMIT - Bestätigung der Statements
  - ROLLBACK - Abbruch der Transaktion ohne Änderungen
  - Wenn AUTOCOMMIT eingeschaltet ist mit jedem DML-Statement.
  - Mit jedem DDL Statement wird immer ein COMMIT impliziert

Multi User / Locks / Isolation Level

Arbeiten mehrere Benutzer an den gleichen Datenbeständen, so stellt das RDBMS die Integrität durch verschiedene Mechanismen sicher:

- Locks: Einzelne Zeilen oder ganze Tabellen können für die Veränderung durch eine Session exklusiv gesperrt werden.
- Problem: Leseinkonsistenz durch konkurierende Änderungen Dritter während einer laufenden Transaktionen.
- Isolation Levels in ORACLE:
  - Dirty Read: Schneller Zugriff auf nicht commited Änderungen - in ORACLE nicht möglich.
  - Repeatable Read: Das wiederholte Lesen innerhalb eines Statements ergibt die gleichen Ergebnisse.
  - Serializable: Der Zustand zu Beginn der Transaktion wird über die gesamte Transaktion sichergestellt.
  - Read Only: Wie Serializable, optimierte Transaktionen ohne Änderungen.

Multi User Anwendungsentwicklung

Sicherung der Lesekonsistenz -bei komplexen Objekten- durch Read Only

- Bei Cache: Konsistenzprüfung und Nachladen --> was wird angezeigt?

Pessimistic Locking:

- Alles was gelesen wird, wird auch zum Schreiben gesperrt.
- Blockade anderer User und Batch-Prozesse.
- Langwährende Locks während Benutzerkntrolle

Optimistic Locking: (Kein DB - Lock)

- Annahme: keine Änderung durch Dritte:
- Prüfung der Annahme durch WHERE-Clause in UPDATE und DELETE Statements
- Fehlerbehandlung wenn Annahme nicht bestätigt.

Grundbegriffe DCL

Data Control Language regelt Rechte innerhalb der Datenbank

- Systemrechte: DBA, Resource, Connect etc
- Objektrechte: Select (Spalten), Insert, Update, Delete, Execute
- Rollen - Zusammengefaßte Rechte / Gruppenzugehörigkeit
- Sondergruppe PUBLIC
- CREATE USER / ROLE name IDENTIFIED BY password
- ALTER USER / ROLE - DROP USER / ROLE
- GRANT / REVOKE recht / Rolle TO USER / ROLE

Schema

Alle DB-Objekte gehören einem Owner. Alle Objekte eines Owners bilden ein Schema

- Nur der Owner kann Objektrechte vergeben an User (WITH GRANT OPTION), Rollen oder PUBLIC
- Der DBA und weitere priviligierte User können Systemrechte (> 70) vergeben
- Es empfiehlt sich, für eine Anwendung einen User als Schema-Owner auszuzeichnen, unter dem alle Objekte angelegt werden.
- Der DBA und entsprechend priviligierte User haben das System-Recht SELECT ANY TABLE.

Allgemeine DB-Werkzeuge

- SQL - Plus: Standard, überall verfügbar, unkomfortabel
- Enterprise Manager - mit Teilmodulen:
  - SQL - Worksheet - Ersetzt weitgehend SQL-PLUS
  - Instance Manager - Rahmenparameter
  - Schema Manager - DB-Objekte
  - Security Manager - User und Rollen
  - Storage Manager - Tablespaces und Datafiles
  - Replication Manager - zur Verwaltung von verteilten DB
- TOAD: Generell für Entwicklung und Administration:
  - Obermenge über Enterprise Manager & PL SQL- Debugger
  - Optimierungshilfen und Administrationsfunktionen

DB-Designwerkzeuge Aufgaben

- Unterstützung bei Analyse und Konzeption auch großer Schemata
- Datenbankentwurf und Dokumentation - mit Kommentaren, referenzieller Integratität
- Übersichtliche Darstellung, auch von Teilmodellen
- Foreward Enginieering - Generieren der DB
- Reverse Engineering - Analyse bestehender Schemata
- Vergleich von Versionen
- Pflege der Schemata - Änderungen und Anpassungen
- Arbeiten im Team
- Unterstützung vieler RDBMS-Versionen - Portierung

ERwin ERX

- Führendes Werkzeug - Größter Marktanteil - teuer
- Unterstützung aller RDBMS-Hersteller
- Einfache Bedienung
- Großer Funktionsumfang
- Enge Kopplung von konzeptionellem und physischem Modell
- Sehr gutes Reverse-Engineering und Versionsvergleich!
- Von Platinum, früher Logic Works
- Eingeschränkte Gruppentauglichkeit

Power Designer

- Weit verbreitetes Werkzeug - Großer Marktanteil
- Unterstützung aller RDBMS-Hersteller
- Gute Bedienung - etwas komplexer als ERwin
- Sehr großer Funktionsumfang - u.a. Testdatengenerierung - Mehrere Module: Data Archetict, App Modeller ...
- Lose Kopplung von konzeptionellem und physischem Modell - mit akzeptablen Abgleichsmöglichkeiten
- Gutes Reverse-Engineering und Versionsvergleich!
- Von Sybase, früher SDT (Star Designor), dann Powersoft (S-Designor)
- Eingeschränkte Gruppentauglichkeit

Designer /2000

- Werkzeug in ORACLE Umgebungen
- Unterstützung ORACLE-RDBMS
- Teilweise Gute Bedienung - sehr komplex
- Großer Funktionsumfang / Repository - u.a. für ORACLE Anwendungsentwicklung
- Schwache Kopplung von konzeptionellem und physischem Modell - mit unzureichenden Abgleichsmöglichkeiten
- Schwaches Reverse-Engineering ohne Versionsvergleich!
- Von Oracle - nicht mehr aktuelle Version 2.1 - Neu ORACLE DESIGNER Version 6
- Gute Gruppentauglichkeit

ORACLE Database Designer

- Einsteigerwerkzeug in ORACLE Umgebungen
- Unterstützung ORACLE-RDBMS
- Gute Bedienung - sehr einfach
- Geringer Funktionsumfang ohne Teilmodelle - nur kleine Entwürfe
- Nur physisches Modell
- Schwaches Reverse-Engineering ohne Versionsvergleich!
- Von Oracle
- Eingeschränkte Gruppentauglichkeit

Designtools im Vergleich

Schulnoten für	*ERwin*	*Power Designer*	*Designer/2000*
Bedienung / Praxis	**2 +**	**2 -**	**3 - 5**
Geschwindigkeit	**1 -**	**2 +**	**5**
Forward Engineering	**1 -**	**1 -**	**3**
Konzept Design	**3**	**1**	**2**
Reverse Engineering	**1**	**2**	**4**
Modellpflege	**1**	**2**	**5**
Gruppeneigenschaften	**2**	**3**	**1**
RDBMS-Support	**1**	**2**	**4 - 5**
Sonderfunktionen	**2**	**1**	**3**

Bedarfsgerechte Analyse

- Manifester / aktueller Bedarf: Nur nicht zuviel unnötiges.
- Künftiger Bedarf: Flexibles und solides Fundament
- Kompromiß zwischen
  - schlanker und direkter, aber kurzsichtigem Design
  - breitem und tragfähigem, aber überladenem Ansatz

- *Mit viel Weitsicht möglichst stark Vereinfachen / Generalisieren.*
- *Benötigte Basisanforderungen (z.B. Historisierung) möglichst zu Beginn berücksichtigen*
- *Zuviel ‚Zukunft' kann die benötigte Flexibilität einschränken.*

Datenbankdesign: Ziele und Aufgaben

- Bedarfsgerechte und konsistente Abbildung der Realität
- Transparenz / Dokumentation für die betroffenen Parteien
  - Systemanalyse / (Unternehmensdatenmodell)
  - Anwendungsentwicklung
  - *DB Administration*
  - *Fachabteilung*
- Effiziente Zugriffswege für Entwicklung, Speicherung und Abfrage
- Darstellung als statische und semistatische Speicherobjekte
- Strukturierte Analyse - datenorientiert - James Martin

Bedarfsgerechte und konsistente Abbildung der Realität: Konzept

- ***Logisches Modell / Conceptual Schema***
- Analyse der Informationsbedarfe - Ziele
- Noch nicht geforderte Entwicklung-Perspektiven
- Identifikation von zusammengehörigen Objekten (Entitäten) in Form von ‚vererbbaren' Tabellen (1:1 Beziehung zwischen physischen Tabellen zu Entitäten nicht immer gegeben)
- Beziehungen zwischen den Entitäten / Relationship
- Identifikation von Datentypen / Domains
- Redundanzfreies Modell nach Normalisierung
- Entity / Relationship Diagramme (ERD)
- Notation Methoden- / Tool-spezifisch

Bedarfsgerechte und konsistente Abbildung der Realität: Server

- ***Physisches Schema / Server Modell***
- Auflösung / Generieren von Fremdschlüssel-Spalten / Vererbung
- Umsetzung des logischen Modells nach praktischen Kriterien: Performance, einfache Verwaltung u.a.
- Denormalisierung
- Unternehmens-Namenskonventionen
- RDBMS / Tool-Spezifisch
- Generierung von SQL Data Definition Language (DDL) Scripten:
  - Tabellen, Primary Key (PK) Constraints, Indexe, Check-Constraints
  - Foreign Key (FK) Constraints, Indexe,
  - Synonyme, Views, Sequences, Tablespaces u.a.
  - Trigger und Stored Procedures

Normalisierung

- Regeln zur Vermeidung von Redundanz
- Aufgestellt von E.F. Codd und durch weitere Autoren erweitert.
- Bis zur 5. Normalform (NF) definiert
- Vieles ist akademisch / wichtig vor allem 3NF

- **ONE FACT IN ONE PLACE**

- ***Normalisierung sagt nichts darüber aus, ob das Modell vollständig und korrekt ist!***

1. Normalform

1NF: Alle Zeilen/Spalten-Koordinaten einer Tabelle haben einen eindeutigen, atomaren Wert.

- Wird physisch durch moderne RDBMS sichergestellt.
- Wiederholgruppen sind nicht zulässig
- Kann logisch durchbrochen werden durch Abspeichern von Wertelisten in Strings.

2. Normalform

2NF: 1. NF & Funktionale Abhängigkeit vom PK - Alle nicht PK-Attribute (Spalten) sind nur von dem vollständigen Primärschlüssel (PK) abhängig.

- Impliziert die Definition eines ‚vernünftigen' PK
- Wird verletzt, wenn das Attribut nur von einem Teil des PK abhängig ist.

3. Normalform

3NF: 2. NF & Funktionale Unabhängigkeit von Nicht-PK-Attributen - Alle nicht PK-Attribute (Spalten) sind von keinem anderen Nicht-PK-Attribut abhängig.

- Wird häufig nicht beachtet, bzw. denormalisiert
- ‚Gewissensfrage' bei Schlüsselwerten / einfachen Codetabellen
- Wird logisch auch durchbrochen bei **Anwendungen**, die im **Code** schlüsselabhängige Wertzuordnung durchführen.

- *‚Jedes Attribut ist vom Schlüssel abhängig, dem vollständigen Schlüssel, und nichts als dem Schlüssel, so wahr mir Codd helfe!'*

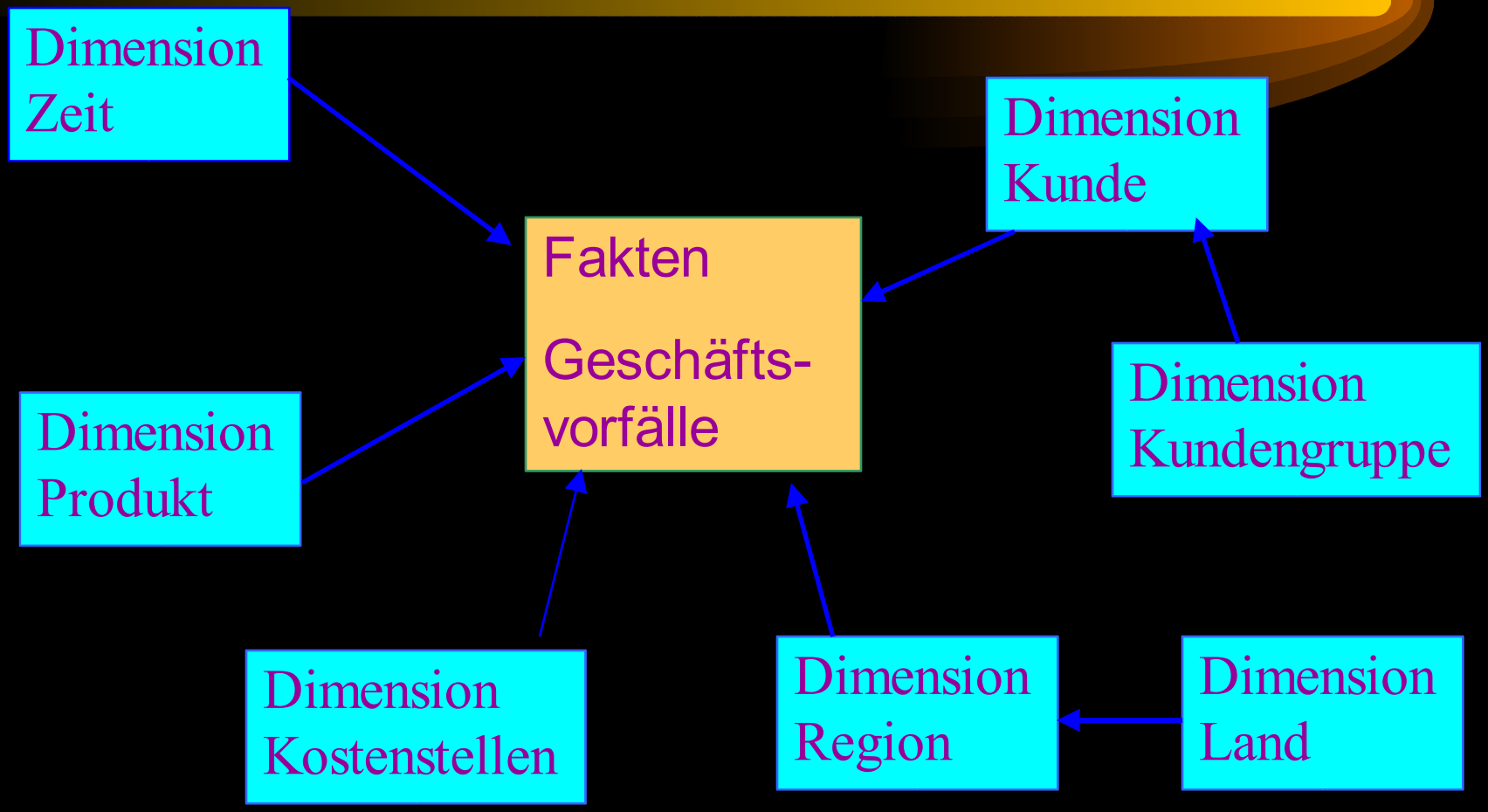
Data Warehouse / Data Mart / OLAP

Flexibele Auswertesysteme, einfach strukturiert für Online Analytical Processing und Enterprise Reporting

Dispositiver Datenbestand /vs.operativer Datenbestand

- Zerschlagung des Prozesses zugunsten von Kreuzauswertungen
- Star Schema: Um die Fakten scharen sich sternförmig Dimensionen - Redundanzen werden zugelassen
- Snowflake Schema: Strukturierte Dimensionen
- Sonderformen: Constallation etc.
- End-user-tools setzen entsprechende Schemata voraus, bzw. arbeiten am besten mit diesen.

Star / Snowflake Schema



Übliche Design Probleme

- Wiederholgruppen: Weitere abhängige Tabellen anlegen
- Mehrfachverwendung des gleichen Attributes: z.B. Start-oder-Endzeit - es ist häufig nicht leicht möglich, beide auseinander zu halten.
- Mehrere Versionen des gleichen Faktums: Verletzung der 2NF
- Widersprüchliche Fakten: unkorrekte Zuordnung / Analyseproblem
- Abgeleitete Attribute: z.B. Geburtsdatum und Alter - Das Alter ist vom Geburtsdatum ableitbar (funktional abhängig) - in Ausnahmefällen kann dies sinnvoll sein (Performance / Denormalisierung)
- Mehrfache Verknüpfungen: z.B. Pers.ID von BEARBEITET und GENEHMIGT im VORGANG ist doppelt verknüpft zu PERSONAL
- Standardattribute in allen Tabellen: z.B. Last_update, Last_User
- Historisierung

‚Neue‘ Möglichkeiten und Chancen

- Integrität durch Trigger auch in denormalisierten Modellen
- Replikation und verteilte Datenbanken
- Objektorientierung / Mapping
- Objektrelationale Erweiterungen

Objektorientierung

- Polimorphie und Vererbung
- Methoden an Entitäten
- Kapselung
- Verwendung von Schlüsseln
- Persistenzschicht / Mapping
- Zuständigkeit: Joins, dynamische Bedingungen und Puffer

Objektrelationale Erweiterungen

- Lobs: Large Binary Objects
- User Defined Datatypes
- Verknüpfte Methoden
- Keine Vererbung und Polimorphie
- Zeitreihen, Spatial Objects, Bild, Ton, Video
- Nested Tables

Objektorientierung Persistenzschicht

- Mapping von Objekten zu Tabellenzeilen und Joins
- Kapselung von Zugriffsmechanismen
- Problem: Flexibilität und Transparenz - Where-Clauses, Joins, Sort
- Chancen: Puffer und Cache (PowerTier)
- Beachten: Locking und Lesekonsistenz

Objektorientierung: Schlüssel

- Fachliche Schlüssel vs. Objekt-ID
- Objekt-ID -> generierte Nummer
- Vorteil: Gleicher Datentyp - einfache Verknüpfung
- Nachteil: Unnötige Abstraktion - zusätzliche Abfragen
- Problem: mehrere Versionen des gleichen Objektes - Historisierung