1 – Analytische Datenbanken („Normal“ transaktional)

|  |  |  |
| --- | --- | --- |
| Anfragen | Transaktional | Analytisch |
| Fokus | Read/Write/Update/Delete | Read, periodic write |
| Transaktionsdauer/Typ | Short read/write operation | Lange Leseoperationen |
| Anfragestruktur | Einfach strukturiert | Komplex |
| Datenvolumen/Anfrage | Wenige Datensätze | Viele Datensätze |
| Datenmodell | Anfragenflexible | auswertungsbezogen |

Datenqualität:Glaubwürdig, Nützlich, Interpretierbar

Datenqualitätsmängel: Dopplung, Unterschiedliche Einheiten, Inkonsistenz

Snowflakeschema(Normal[isiert]) vs Starschema (Faktentabelle + Dimensionstabellen)

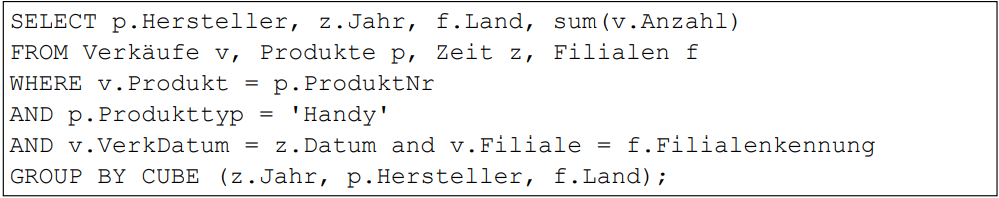
Snowflake: Langsam

Starschema: Schnell, dafür Einfüge/Änderungs/- und Löschanomalien

Roll-Up: Bei SELECT von Daten **wenige** Attribute in GROUP BY -> Daten verdichtet

Drill-Down: Bei SELECT von Daten **viele** Attribute in GROUP BY -> **hoher** detailgrad

Beispiel: Starschema, Faktentabelle Verkäufe wird mit Dimensionstabellen Produkt/Zeit/Filliale verbunden



**create** **Table** Verkauf\_2DC

**AS**

**SELECT** **COALESCE**(p.produkt\_gruppe,'alle') **as** Gruppe, **COALESCE**(TO\_CHAR(k.kunde\_alter), 'alle') **as** "ALTER", **SUM**(v.verkauf\_Anzahl) **as** anzahl

**FROM** Kunde k, Produkt p, Verkauf v

**WHERE** v.verkauf\_datum **>=** TO\_DATE('2020-01-01', 'YYYY-MM-DD')

**AND** v.produkt\_ID **=** p.Produkt\_ID

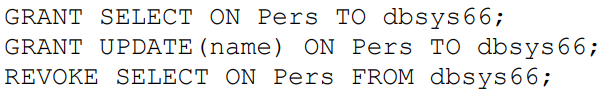
**AND** v.kunde\_ID **=** k.Kunde\_ID

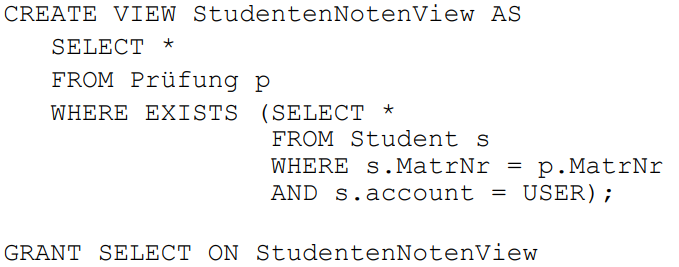
**GROUP** **BY** **CUBE** (p.produkt\_gruppe, k.kunde\_alter);

Decode als Alternative zu Coalesce

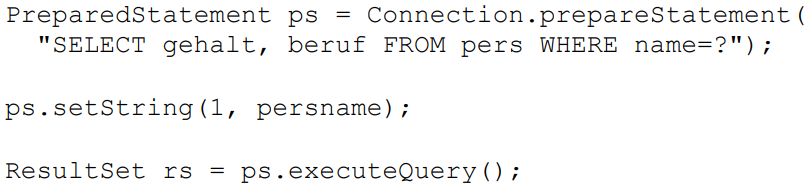
DECODE(GROUPING(region), 1, 'All Regions', 0, region)) AS Region

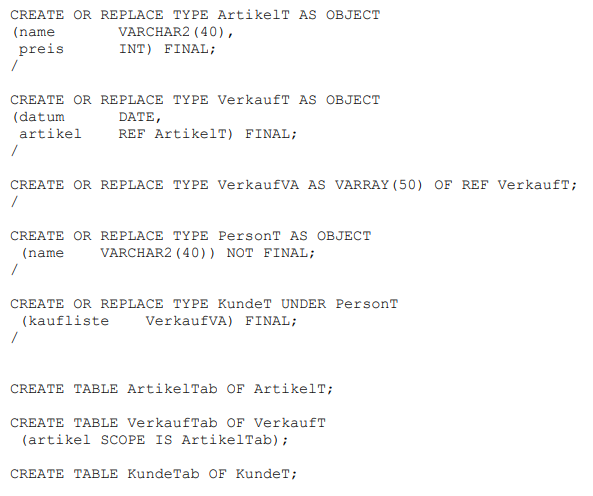
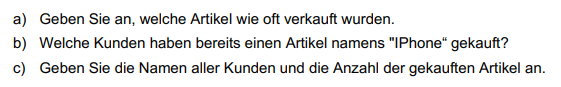
Ersteller von SQL Tabellen hat immer alle Rechte -> Software muss mit anderem Benutzer zugreifen

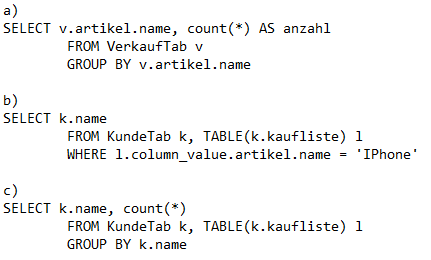


Workaround um Rechte nur auf eine Zeile zu geben:  


Prepared Statements, Input Validierung, Authentifizierung





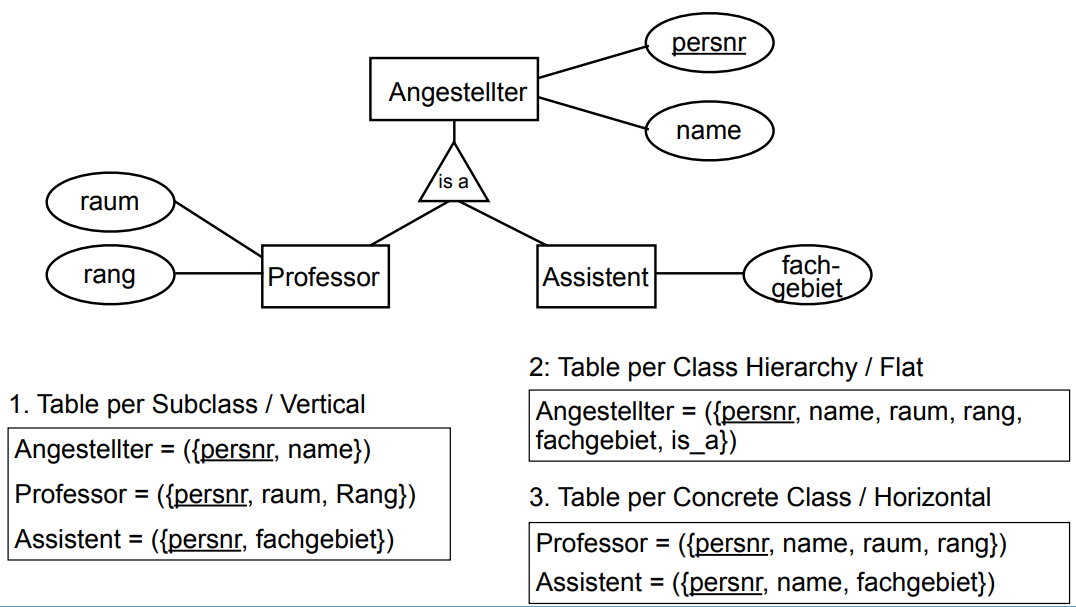


Kapitel 4

OR-Mapper: Seperation of concerns

Factory Method: Genaue Objekt Instanz durch Factory gekapselt, kann ausgetauscht werden

Abbildung der Vererbung, 3 Möglichkeiten:



Identitymap: Keeps track of Objects that have been loaded from db. If object is loaded again the object already in memory is used.

Metadaten für Mapping als xml Datei oder Annotation im Java Code

Vorgehensweise Hibernate: Topw-down/Bottom-up/Middle-out/Meet in the middle

Kapitel 5

Data Dictionary: Speicherung aller Informationen, die zur Verwaltung der Objekte in der Datenbank benötigt werden

– Schema-Informationen, also Tabellen, Attribute, etc.

– Statistiken

– Verwendeter Speicherplatzbedarf

– Name von Datenbank Benutzern, Rollen, Zugriffsrechte

TAB – SELECT \* FROM Tab; – Alle eigenen Tabellen, Views

USER\_TABLES – Tabellen eines Benutzers

USER\_TAB\_COLUMNS – Attributes einer Tabellen eines Benutzers

USER\_TRIGGERS – Trigger eines Benutzers

USER\_TAB\_PRIVS – Zugriffsrechte auf eine Tabelle eines Benutzers

USER\_VIEWS – Views eines Benutzers

Kapitel 6

**Dirty Read/Write**: Transaktion 1 liest/schreibt Daten, die von Transaktion 2 verändert aber noch nicht committet wurden => Problem bei Rollback

**Lost Update**: Lesen von Daten, die eine andere Transaktion bereits gelesen hat und im Anschluss bearbeiten möchte.

**Incorrect summary**: Erstellen einer Statistik (z.B. Gesamtsumme aller Konten) während der Daten geändert werden.

**Phantom-Problem**: Erstellen einer Statistik, während neue Datensätze eingefügt werden

ACID-Prinzip bei Transaktionen: Atomicity, Consistency, Isolation, Durability

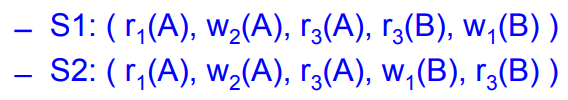
Bei Systemausfall Recovery: Wiederherstellen des jüngsten transaktionskonsistenten Zustands

**Physisches** Logging: Zustand merken (Before und After-Image)

**Logisches** Logging: Protokollierung der Änderungsoperationen mit ihren Parametern

**Seriell**: Transaktionen werden vollständig nacheinander ausgeführt.

**Serialisierbar**: Schedule liefert dasselbe Ergebnis wie mindestens ein **serieller** Schedule



Einfügen von Kante T1→T2, falls eine Operation einer Transaktion T1 vor einer Operation einer Transaktion T2 ausgeführt werden muss (muss vorher ausgeführt werden, wenn ansonsten ein anderes Ergebnis entsteht)

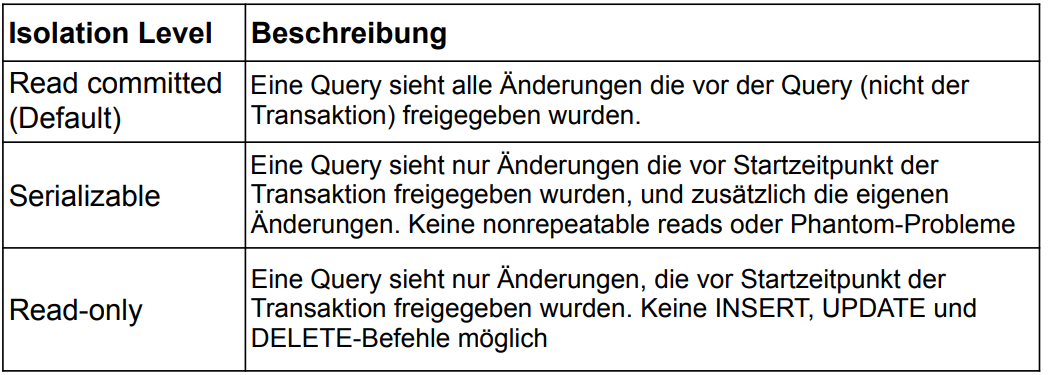
Physische Sperre: Sperre von DB Objekt, keine Lösung zu Phantom Problem, da neu eingefügte Objekte nicht gesperrt sind

Logischer Sperre (Prädikatssperre): Sperren einer Menge von Tupeln(z.B. P1=(Gehalt>50000), P2=(Jahrg<1960))

* Nur bei Lesezugriff möglich, da bei überlappenden Mengen 2x Write Locks angefordert werden

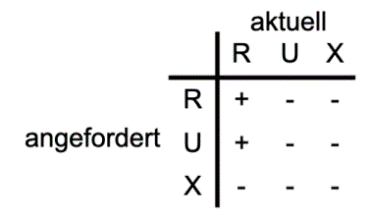
Ganze Tabelle sperren:

LOCK TABLE name IN EXCLUSIVE MODE; LOCK TABLE name IN SHARE MODE;



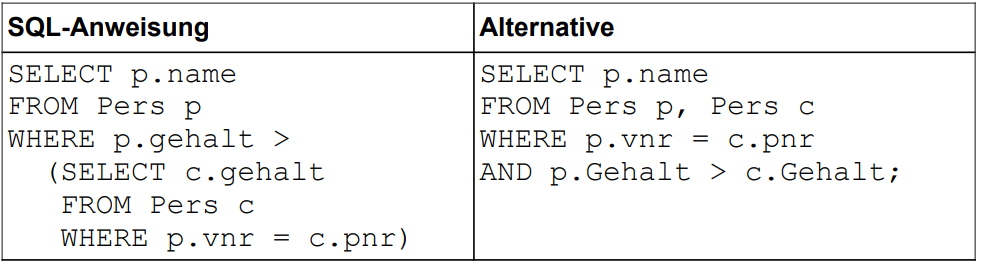
Multiversion Concurrency Control in Oracle:

Beim Lesen Überschriebener Daten wird durch UNDO-Daten der Alte Stand wiederhergestellt/simuliert

U(pdate)-Sperre -> Kann angefordert werden wenn andere R-Locks aktiv sind.

Wenn U-Sperre vorhanden werden keine neuen R-Locks vergeben.

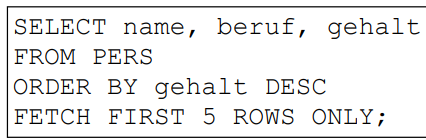
Wenn alle R-locks freigegeben wurden Update zur X-Sperre

Kapitel 7

Korrelierte Unterabfrage

DISTINCT nur wenn notwendig

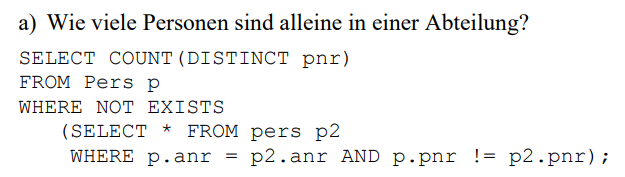
Top-N-Analyse

Ermittlung der Top-Verdiener der Firma

ALternativen:

FETCH FIRST 10 PERCENT ROWS ONLY

FETCH FIRST 3 ROWS WITH TIES

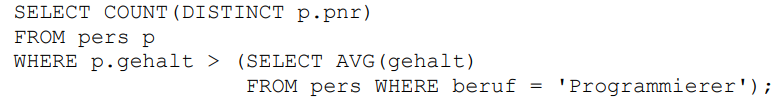
**SELECT** **COUNT**(**DISTINCT** p.pnr) **AS** AnzahlAlleineInAbteilung

**FROM** Pers p

**LEFT** **JOIN** Pers p2 **ON** p.anr **=** p2.anr **AND** p.pnr !**=** p2.pnr

**WHERE** p2.pnr **IS** **NULL**;

**CREATE** MATERIALIZED **VIEW** ProgGehalt **AS**

 **SELECT** **AVG**(gehalt) **AS** durchschnitt

**FROM** pers

**WHERE** beruf **=** 'Programmierer';

**SELECT** **COUNT**(**DISTINCT** p.pnr)

**FROM** pers p, ProgGehalt pg

**WHERE** p.gehalt **>** pg.durchschnitt;

Hints (Auf Index): SELECT /\*+ INDEX(p Pers IndexPersGehalt) \*/ \* FROM Pers p

Clustering: Speicherung von Tabellen, die häufig verknüpft werden als physisch vorberechneter Join

Partitionierung: Aufteilung einer Tabelle in physisch mehrere Teile (z.B. Kunden Partition nach Land)