1. DBS - Marco Agostini

1.1 Grundlagen

Daten: Maschinell verarbeitbar. **Informationen:** Sind interpretierte Daten. Es gibt für formatierte und umformatierte Daten die passenden Systeme.
Ein **Datenbanksystem (DBS)** besteht aus **Datenbankmanagementsystem (DBMS)** und **1 bis n Datenbasen (DB)**. Die Datenstruktur wird im Datenkatalog(Data Dictionary) beschrieben und wird mit DDL erstellt.
DBMS-Funktionen: Speicherung, Transaktionen, Mehrbenutzerbetrieb, Sicherheit, Backup/Recovery, Datentypen, Abfragesprache, Programmierschnittstellen.
Datenintegrität(Anf. DBMS) *Datenkonsistenz:* Logische Wiederspurchsfreiheit. *Datensicherheit:* Physische Sicherheit *Datenschutz:* Zugriffsschutz

1.2 Datenbankmodelle/Paradigmen

Das Datenbankmodell legt die Datenstruktur fest, Strukturen im DBMS fest (Datentypen, Speicherung).
Hierarchisches DBM: Eine einzige Hierarchie, Beziehungen werden mit den Daten gespeichert. Nachteile: Die Welt ist nicht hierarchisch, Keine Trennung zwischen internen und externem Schema, aufwendige Anpassungsarbeiten.
Netzwerk DBM: Vernetzte Hierarchien, Beziehungen werden in einer Hierarchie gespeichert. Nachteile: Effiziente Anwendung war möglich mit der Kenntnis der komplexen Netzwerk-Datenstruktur.
Relationales DBM: Alle Informationen werde als unstrukturierte Daten (1.NF) gespeichert. Sehr flexibel. Klare und genaue Trennung zwischen Daten und Anwendungen. Nachteile: Independence Missmacht: Zwischen Typensystem der Programmiersprache und den Tabellenstrukturen. Komplexe Abfragen führen zu Performance-Problemen.
Objektrelationales DBM: Erweiterung des rel. Datenmodelles. (benutzerdef: Typen, Tabellen, Vererbung). Verschachteln von Tabellen (1. NF nicht nötig) Neben Daten (Objekte, Attribute) auch Methoden speicherbar.

1.3 Entwurfsprozess

Generalisierung/Spezialisierung Ein Angestellter kann mehrere Rollen haben. is-a Beziehung. *Abstrakte Basisklasse:* Legt Kontrakt für Basisklasse fest, kann nicht installiert werden. *disjoint:* Objekt ist Instanz einer Unterklasse. *overlapping:* Objekt kann Instanz von mehreren überlappenden Unterklassen sein. *complete:* Alle Subklassen sind definiert. *incomplete:* nicht-vollständig: Zusätzliche Subklassen sind erlaubt.
ANSI-3-Ebenen-Modell
Externe Ebene: Sicht einer Benutzerklasse auf Teilmenge. (Externes Schema).
Logische Ebene: Logische Struktur der Daten. (Logisches Schema).
Interne Ebene: Speicherstrukturen (Internes Schema).
Mapping: Zwischen den Ebenen ist eine oder mehrere komplexe Abbildungen notwendig.

1.3.1 Datenmodellierung

Der konzeptuelle Entwurf wird mit UML realisiert. Der logische Entwurf wird mit der relationalen Schreibweise definiert.
Konzeptuelles Datenmodell Definiert die Daten-Objekte und deren Eigenschaften, Zusammenhänge und Beziehungen, legt Konsistenzbedingungen fest. Ist Lösungs- und Technologieabhängig. (UML/ER)
Logischer DB-Entwurf *Schritt 1.* Abbildung des Konzeptuellen Modells (Domain Model) in das relationale Model. *Schritt 2.* Überprüfen der Konsistenz des rel. Modelle. (Normalenformen)

1.3.2 Relationales Datenmodell

Mathematisches Konzept. Benutzt Tabellen zur Darstellung von Daten und Beziehungen.
Regeln für die Abbildung: Saubere Trennung zwischen der logischen und internen Schicht. Tabellen und Attribute in Gross-und-Kleinschreibweise.
PS werden unterstrichen. **Fremdschlüssel** werden kursiv gschrieben mit REFERENCES TabelleA. **Schlüsselkandidaten-Attribute** sind mit UNIQUE auszuzeichnen. Attribute sind default: NOT UNIQUE, NULL
Schlüssel: Attribut oder Kombination von Attributen, das/die ein Tupel eindeutig identifizieren. Die Kombination muss minimal sein. Eigenschaften: Eindeutig, Laufend Zuteilbar, kurz, invariant.
Sekundärschlüssel: Alternativer Suchschlüssel, nicht unbedingt eindeutig.
Surrogatsschlüssel: Künstlicher Schlüssel, der als Primärschlüssel verwendet wird.

1.3.3 Normalenformen

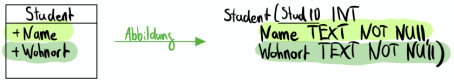
Änderungsanomalie: Änderung muss an mehreren Orten vollzogen werde.
Einfüganomalie: Tupel kann erst hinzugefügt werden, wenn andere Tupel existieren.
Löschanomalie Wenn ein Tupel gelöscht wird, gehen weitere Informationen verloren.
Mutationsanomalie Andere Tupel werden geändert.
1. Normalenform Wertbereiche der Attribute sind atomar.
2. Normalenform Jedes Nichtschlüsselattribut voll funktional abhängig von jedem Schlüsselkandidaten.
3. Normalenform Kein Nichtschlüsselattribut transitiv abhängig ist von einem Schlüsselattribut.

1.4 Abbildungsregeln in Logisches Modell

Das konzeptuelle Model (Domain Model) wird in das logische Modell (relationale Schreibweise) abgebildet.

1.4.1 Regel 1. Abbildung von Klassen

Jede Klasse wird auf eine Tabelle abgebildet.



1.4.2 Regel 2.1 vorausgesetzt (1...n)

Assoziation mit Kardinalität (1...*). Der Primärschlüssel muss in der zweiten Tabelle als Fremdschlüssel vorkommen. Sprich das Objekt braucht eine Beziehung. (FS NOT NULL)

1.4.3 Regel 2.2 I optionale AS mit FS NULL (0...1 zu 0...*)

Wie 2.1 mit optionalen Beziehungsattributen. Mit NULL-Werten für nicht vorhandene Beziehungen. (FS NULL) Für häufig vorkommende Beziehungen.

1.4.4 Regel 2.2 II optionale AS mit Beziehungstabelle (0...1 zu 0...*)

Mit separater, abhängiger Beziehungstabelle (Zwei FS + Attribute) für seltene Beziehungen. (kanonische Lösung)

1.4.5 Regel 2.3 Abbildung von Aggregationen

A) starke Ganzes-Teile-Beziehung (Komposition, abhängige Tabelle PS/FS ref. PS)
B) schwache Ganzes-Teile-Beziehung (Aggregation, Beide haben PS)

1.4.6 Regel 2.4 n:m

Wird mit einer abhängigen Beziehungs-Tabelle mit Zusammengesetzen Primärschlüsseln moderiert.
A) Beziehungsklasse (PS = beide FS) (- Zusammengesetzte Schlüssel können sich fortpflanzen)
B) Assoziative Klasse aus (PS + zwei FS) (- Unique Contstraint)

1.4.7 Regel 3

A) 1:1 Abbildung aller Tabellen Die Superklasse wird auf einte Tabelle abgebildet mit einem Attribut, welches den Typ der Subklasse spezifiziert. (+ Flexibel und Redundantzfrei - Viele Tabellen, Komplexer Zugriff)
B) Eine alleinstehende Tabelle pro Subklasse Jede Subklasse erhält eine eigene Tabelle und enthält alle Attribute. (+ Einfacher Zugriff - Semantikverlusst, Schlüssel-Eindeutigkeit (Kontrolle über mehrere Tabellen), Modulation, keine überlappende Vererbung möglich)
C) Eine Superklasse Die Superklasse erhaltet alle Attribute auch die der Subklassen mit einem diskriminierenden Attribut. (+ Einfache Zugriffe - Viele NULL Werte, 3. oder höhere Normalenform verletzt)

1.5 Relationale Algebra

Definiert Operationen, die sich auf eine Menge von Relationen anwenden lassen.

```
SELECT * FROM mitarbeiter
WHERE gehalt > 50;
SELECT a1, a2 FROM TABLE;
SELECT * FROM t1, t2;
```

Selektion $[\sigma]$
$\sigma_{(gehalt>50)}mitarbeiter$
Projektion $[\Pi]$
Kartesisches Produkt $[x]$

2. SQL

2.1 SQL Data Definition Language

```
CREATE DATABASE foo WITH OWNER = 'bar';
CREATE INDEX indexname ON tbl_name (attribute);
```

```
CREATE TABLE person (
    persnr  INTEGER PRIMARY KEY, -- Collumn Constraint
    vorname VARCHAR(20) NOT NULL,
    name VARCHAR(20) NOT NULL,
    geburtsdatum  DATE,
    -- Table  Constraint
    PRIMARY KEY (persnr)
);
```

2.1.1 Referentielle Integrität

```
ON DELETE CASCADE --Tupel mit FS werden geloescht
ON DELETE RESTRICT --Tupel mit Ref. werden blockiert
ON DELETE SET NULL --Tupel mit FS werden NULL
ON DELETE SET DEFAULT --Tupel mit FS werden Default
```

2.2 SQL Data Manipulation Language

```
INSERT INTO tablename VALUES (23, 'Volkswagen');
INSERT INTO tablename(name, abtnr)
    VALUES('Entwicklung', 20);
```

```
UPDATE abteilung SET name='Verkauf' WHERE abtnr=3;
DELETE FROM abteilung WHERE abtnr=21;
```

Selektion
Mit dem Like kann nach String-Mustern gesucht werden:
(% 0 bis n Zeichen,_ ein Zeichen)

```
SELECT DISTINCT wohnort, name FROM angestellter
WHERE abtnr=3 AND name LIKE '%marc%'
ORDER BY wohnort ASC;
```

Aggregatsfunktionen Null-Werte werden bei Aggregatsfunktionen übersprungen. Wird oft mit GROUP BY kombiniert. Die Having Klausel darf nur nach GROUP BY Verwendet werden.

```
[MIN | MAX | AVG | SUM | COUNT | FIRST]
SELECT AVG(salaer) FROM angestellter
GROUP BY abntnr;
```

2.3 Joins ⋈

Inner Join: Filtern der Zeilen

```
SELECT abt.name, ang.name FROM
 abteilung  INNER JOIN angestellter
ON abt.abtnr=ang.abtnr ;
```

Left Outer Join: Lässt Werte von Links stehen

```
SELECT a.attr,  b. attr  FROM tablename t1
LEFT OUTER JOIN tablename2 t2
ON t1. attr  = t2 . attr ;
```

Equi Join: Tabelle joint sich selbest verknüpft **Natural Join:** Join nach gleichen Attributen.
Theta Join: Kartesische Produkt + JOIN-Bedingung

2.4 Mengenoperatoren

Anhahl und Typ der Felder muss übereinstimmen.

```
[ UNION | UNION ALL | INTERSECT | MINUS | EXCEPT]
SELECT persnr, name
FROM angestellter
UNION
SELECT projnr, bezeichnung
FROM projekt
ORDER BY 1;
```

2.5 Unterabfragen

Für jeden Tupel wird die Unterabfrage ausgewertet. EXISTS = Boolean!

```
[NOT IN | EXISTS | NOT EXISTS | ANY | ALL]
SELECT ang.name, ang.salaer FROM angestellter  ang
INNER JOIN abteilung abt  ON ang.abtnr=abt. abtnr
WHERE abt.name='Marketing'
AND ang.salaer < ALL
(    SELECT salaer FROM angestellter ang1
     INNER JOIN abteilung abt1
     ON ang1.abtnr=abt1. abtnr
     WHERE abt1.name='Entwicklung'
);
```

2.6 Modern SQL

Window Functions Funktionen, die auf ein Datenfenster angewendet werden. Mächtiger als GROUP BY. Tupel werden beibehalten, aber extra Spalte wird angehängt.

```
SELECT persnr, abtnr,  max(salaer)  OVER (PARTITION BY abtnr)
FROM angestellter;
```

Common Table Expressions Ermöglichen Hilf-Queries in bzw. von einer grösseren Query. Sind temporäre Tabellen.

```
WITH sometable1 AS (
    SELECT * from sometable2;
),
sometable2  AS (
    SELECT * FROM sometable1
),
SELECT * FROM sometable2;
```

Rekursive Variante

```
WITH RECURSIVE unter ( name, persnr, chef) AS
    (SELECT A.name, A.persnr, A.chef FROM angestellter A
    WHERE A.chef = 1010
    UNION ALL
    SELECT A.name, A.persnr, A.chef FROM angestellter A
    INNER JOIN unter B ON B.persnr = A.chef
) SELECT * from unter;
```

Lateral Joins Query mit dem Inhalt einer anderen Tabelle ergänzen. Tabellen haben ohne Lateral keinen Zugriff auf Attribute anderer Tabellen. Die

```
SELECT abt.abtnr, ang. persnr , ang.name, ang. salaer
FROM abteilung AS abt
JOIN LATERAL (
    SELECT *
    FROM angestellter
    WHERE abt.abtnr =  angestellter . abtnr
    ORDER BY salaer DESC
    LIMIT 3
) ang ON true;
```

2.7 Views

Datenkapselung auf interner Ebene. Vereinfachung der Abfrage. Sicherheit: irrelevante Daten werden verborgen. Virtuelle Tabelle basierend auf Tabellen oder Views.
Materialized View: existieren in Tabellenform und werden nicht automatisch aktualisiert.

```
CREATE VIEW name (persnr, name, tel) AS
    SELECT persnr,  name,  tel
    FROM angestellter;
```

2.8 Datenbanksicherheit

Systemsicherheit: Identifizierung, Autorisierung von Benutzern. Kontrolle der System-Ressourcen. Auditing. Kommunikationssicherung.
SQL Injection Angreifer erstellt SQL-Kommandos oder verändert existierende um versteckte Daten sichtbar zu machen, Daten zu überschreiben oder auf Systemebene des Hosts zu gelangen. *Massnahmen:* Input prüfen, Datenzugriff abstrahieren, Benutzerverwaltung.
Benutzerverwaltung / SQL Data Control Language Jeder Nutzer erhält System- und Datenprivilegien. Rollen erhalten diese Rechte. (Rollen = Gruppen und Benutzer)

```
CREATE ROLE user_name
WITH LOGIN PASSWORD 'passw0rd';
GRANT INSERT ON TABLE angestellter to user_name;
REVOKE UPDATE on angestellter from user_name;
```

Row-Security Ermöglich Zugriff auf eine Untermenge von Zeilen einer. Tabelle.

2.9 Transaktionen

Höhere Geschwindigkeiten durch Verzahnung/Paralelität mit korrekter Isolation.

2.9.1 ACID

Atomicity: Eine Transaktion wird nur vollständig durchgeführt.
Consistency: Die Daten werden von einem Konsistenten Zustand in den nächsten gebrach. **Isolation:** Eine Transaktion soll so ausgeführt werden, als sei sie isoliert. **Durability:** Alle Änderungen einer Transaktion sind persistent und gehen nicht durch Fehler verloren.

```
BEGIN TRANSACTION;
SAVEPOINT firstsavp;
ROLLBACK TO SAVEPOINT firstsavp;
COMMIT TRANSACTION;
```

2.10 Isolation

Serialisierbarkeit Nebenläufigkeit, soweit das Ergebnis equivalent zur seriellen Ausführung ist. Ziel ist es zyklischen Informationsfluss sicherzustellen.
Formalisierung
$r_a(x)$ Transaktion T_a liest DB-Objekt x
$w_a(x)$ Transaktion T_a schreibt DB-Objekt x
$c_a(x)$ Transaktion T_a ruft Commit auf
Serialisierbarkeitsgraph
Jedes Konfliktpaar symbolisiert eine Verbindung zwischen den Transaktionen. 1. Konfliktpaare bestimmen 2. Graph zeichnen und analysieren.
Zyklenfrei <=> Serialisierbar
Commit nur möglich, falls keine Konfliktpaare einer anderen Transaktion folgen.
Pessimistische Locking Verfahren
Exclusive Lock (x)
Read/Write. Nur eine Transaktion kann den Lock erhalten.
Shared Lock (s)
Read. Mehrere Transaktionen können den Lock erhalten.
Granularität
Locks können Salbe, Table Ranges, Rows, Item erhalten. (Meistens auf der Row)
Two-Phase-Locking 2PL
Zwei Phasen: Growing und Shrinking Phase. Sobald ein Lock freigegeben wird, das sie nichts neues Locken. Probleme: Cascading Rollbacks, Deadlocks. (Behoben durch time out oder Abbruch einzelner Transaktionen.)
Strict Two-Phase-Locking S2PL
Alle gehaltenen Locks werden erst am Ende der Transaktion freigegeben. Probleme: Deadlocks möglich. (Kein Cascading Rollback, kein unklarer Beginn der Shrinking Phase)
Preclaiming Two-Phase-Locking
Transaktion muss alle Locks auf einmal (atomar) am Anfang belegen. Dies garantiert die Deadlock-freiheit. Nicht realistisch umsetzbar!

2.11 Isolation-Levels

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Allowed, but not in PG	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Allowed, but not in PG	Possible
Serializable	Not possible	Not possible	Not possible	Not possible

	Garantiert Serialisierbar	Keine Deadlocks	Keine Cascading Rollbacks	Keine Konflikt-Rollbacks	Hohe Parallelität	Realistisch (ohne Vorsnalyse)
Two-Phase Locking	✓	✗	✗	✓	✗	✗
Strict 2PL	✓	✗	✓	✓	✗	✓
Preclaiming 2PL	✓	✓	✓	✓	✗	✗
Validation-Based	✓	✓	✗	✗	✓	✓
Timestamp-Based	✓	✓	✗	✗	✓	✓
Snapshot Isolation	✗	* ✓	✓	✗	✓	✓
SSI	✓	* ✓	✓	✗	✓	✓

	Read Uncommited	Read Commited	Repeatable Read	Serializable
Dirty Write	möglich	möglich	möglich	unmöglich
Dirty Read	möglich	unmöglich	unmöglich	unmöglich
Lost Update	möglich	möglich	unmöglich	unmöglich
Fuzzy Read	möglich	möglich	unmöglich	unmöglich
Phantom	möglich	möglich	möglich	unmöglich
Read Skew	möglich	möglich	unmöglich	unmöglich
Write Skew	möglich	möglich	möglich	möglich
Deadlock			möglich	unmöglich
Cascading Rollback				unmöglich

Das Isolations-Level Serializable ist die richtige vollständige Isolation nach ACID.
Read Uncomitted: Lesezugriffe werden nicht synchronisiert.
Read Commited: Lesezugriffe werden kurz/temporär synchronisiert.
Repeatable Read: Einzeln zugegriffen Rows werden synchronisiert. (SI/MVCC)
Serializable: Vollständig korrekte Isolation. (SSI)

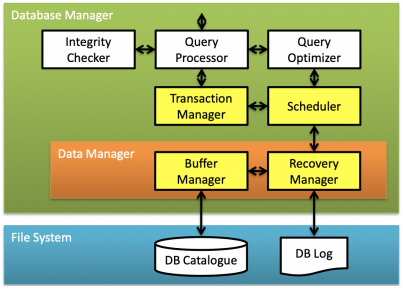
2.11.1 Fehlertypen

Dirty Read: Liest Daten von anderer abgebrochenen Transaktion.
Non-Repeatable Read (Fuzzy Read): Liest Daten mehrmals und sieht andere Werte.
Phantom Read: Suchkriterien treffen während einer Transaktion auf unterschiedliche Datensätze zu, weil eine andere Transaktion Datensätze hinzugefügt oder entfernt hat.

2.11.2 Optimistische Verfahren

Snapshot Isolation SI
Jede Transaktion sieht Snapshot zu seinem Start-Zeitpunkt. Bei Commit wird geprüft, ob Objekte noch unverändert wie bei Start sonst rollback. Erfüllt Serializable nicht. Lässt Write Skew-Anomalie und Read-only transaction Anomalie zu. (Lesen blockiert schreiben, schreiben blockiert lesen nicht)
MVCC Multi-Version Concurrency Control
Realisiert SI. Intern werden verschiedene Versionen eines Objektes gehalten, die durch Zeitstempel oder Transaktionsnummer voneinander unterschieden werden. Bei Update werden Tupel mit x-lock versehen - Deadlocks möglich. Beim Lesen werden keine Locks gesetzt bzw. überprüft. (Schreiber/Leser blockieren keine Leser, Schreiben blockieren Schreiber)

2.12 Architektur



Dateien DBMS Logisch Sammlung von Datenwerten
Abstrakt physisch Folge von Datenblöcken
Physikalisch Sammlung von Sektoren.
Pages in Blöcken abstrahiert.
Transaction-Manger koordiniert die Transaktionen und kommuniziert mit dem Scheduler.
Der Scheduler Ist für die Koordination der parallele laufenden Transaktionen zuständig.
Recovery-Manager Muss bei Abbruch einer Transaktion die Daten wieder in einen konsistenten Zustand zurückführen.
Buffer-Manager Verwaltet den internen DB-Puffer. Stellt mit Storage-Manger den Transfer zwischen Haupt- und Sekundärspeicher sicher.
Planer/Optimizer
Erzeugung eines optimalen Ausführungsplanes (Execution Plan). Wählt alle möglichen Ausführungspläne und wählt denjenigen, der am günstigsten ist.

2.13 Fehlerklassifikation

Fehler einer Transaktion Ursachen: Fehler in der Applikation-SW, Abbruch einer Transaktion durch Applikation oder System (Deadlock-Auflösung). Inkonsistenz: Festgeschriebene Änderungen Recovery: Rollback oder lokales undo. (Millisekunden)
Hauptspeicherverlust Ursachen: Fehler in der DB-System-SW, HW-Fehler, Stromausfall. Inkonsistenz: Verlust von Änderungen, die noch nicht auf die Disk geschrieben wurden. Recovery: Gloabes Undo: alle aktiven Transaktionen zurücksetzen.
Hintergrundspeicherverlust Ursachen: Fehler im Disk-Treiber, Feuer, Wasser etc. Inconsistent: Verlust der Daten auf der Disk. Recovery: Rekonstruktion der verlorenen Daten, Backup und Log-File-Techniken.

2.14 Log-Files

Write-Ahead-Log WAL
$[LSN, TransactionID, PageID, Redo, Undo, PrevLSN]$
$[\#2, T_1, P_A, A = 250, A = 200, \#1]$
$[\#3, T_2, P_C, C = 100, C = 150, \#2]$
Gewährleistung (AC). Modifikationen werden vor dem eigentlichen Schreiben protokolliert.
Log Sequence Number: Eindeutig, monoton ansteigend
TransaktionsID: Transaktionskennung der Transaktion
Page: Kennung der Page der Änderung
Redo: Gibt and, wie die Änderung nachvollzogen werden kann. Absoluter Wert nach der Änderung.
Undo: Gibt an, wie die Änderung rückgängig gemacht werden kann. Absoluter Wert nach der Änderung.
PrevLSN: Zeiger auf den vorhergehenden Log-Eintrag.
Wiederanlauf nach Fehler
Winners - REDO
Transkationen, die nach dem letzten Checkpoint duchgeführt wurden.
Loosers - UNDO
Transaktionen, die zum Zeitpunkt des Absturzes noch aktiv Waren.

2.15 Index-Strukturen

Selbständige Datenstruktur, redundant zur Tabelle. Beschleunigt den Lese-Zugriff auf Kosten INSERT/UPDATE/DELETE.
Zwei Strukturen: Data Pages (Heap) - Linked-List der Daten. Suchbaum - Indexeinträges (Nodes,Leaves)
Index-Sequential Access Method ISAM
+ Einfügen, Suchen / - Aktualisieren
Daten werden über die indesxspalte aufsteigend sortiert. Grösster Schlüsselindex im Block wird vermehrt und in den Index gelegt. Sehr dünner Index mit logischen Adressen.
B-Bäume
+ Lesen Schreiben
Balancierte Mehrwege Suchbäume. Geeignet für Hintergrundspeicher.
B+-Bäume
Nur Blätter/Leaves enthalten Daten. Blätter sind verkettet (Linked-List). Die Verkettung erlaubt eine schneller Interation.
Hash Ordnet Key den Einträgen zu. Problem: Overflow.
Zusammengesetzter Index

```
CREATE INDEX mytable_col12_idx
ON mytable (col1, col2);
```

Partieller Index

```
CREATE INDEX mytable_col_part_idx
ON mytable (col)
WHERE archived IS NOT NULL;
```

Funktionaler Index

```
CREATE INDEX mytable_col_part_idx
ON mytable (immutable_function (col)); --upper, lower
```

Index mit INCLUDE

```
CREATE INDEX magic_idx
ON test (nr, id) INCLUDE (txt);
```

2.16 Optimierung

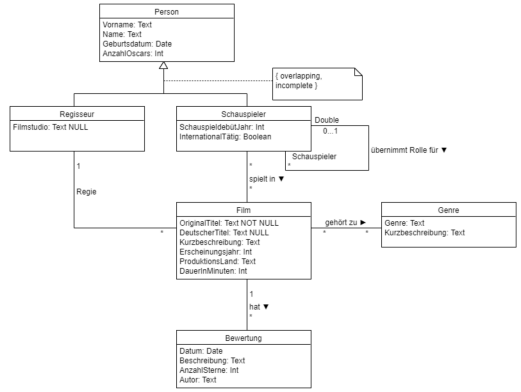
Die syntaktische/lexikalische Reihenfolge der Klauseln entspricht nicht der logischen Reihenfolge der Operationen und auch nicht der effektiven.
Übersetzen einer SQL-Abfrage
1. SQL parsen und in Baumstruktur abbilden
2. Query validieren
3. Views auflösen
4. Query Optimieren
5. Execution Plan erzeugen
6. Code erzeugen bzw. ausführen
7. Resultat in Query ausgeben

Logische Optimierung
Abfrage so umformulieren, dass sie dasselbe Resultat liefert, aber effizienter berechnet werden kann.
Selektion so früh wie möglich, um die Zwischenergebnisse so klein wie möglich zu halten.
Physische Optimierung
Effiziente Algorithmen wählen, um die Operationen der relationale Algebra zu implementieren.Kostenbasierte Auswahl aus einer Menge von Plänen. Kosten sind u.a. von den verfügbaren Plänen abhängig.
Ausführungspläne Kommunikationskosten, Berechnungskosten, I/O-Kosten, Speicherkosten.
Statistiken Informationen über Relationen und Indexe in Systemkatalog verwaltet. (Je nach DBMS nur periodisch aktualisiert.
Der Planner
Der Planner beginnt bei der Generierung bei den Relationen (Tabellen) der Query. *Full Taube Scan / Seq Scan* Scannt den ganzen Table. *Index Scan* Falls es ein Attribut gibt, nach dem gefiltert werden kann. Bei kleineren Datenmengen aus dem selben Disk-Block ist ein Seq Scan Scheller als ein Index-Scan. Lädt einen Tutel-Zeiger nach dem andere aus dem Index und greift sofort auf das entsprechende Element ind er Tabelle zu. *Bitmap Index/Heap scan* Lädt alle Tutel-Zeiger auf einmal aus dem Index, erzeugt ad-hoc einen internen Bitmap-Index, um sie dann im Hauptspeicher zu sortieren und lädt alle Tabellen-Tupeln.

```
select Titel
from Professoren, Vorlesungen
where Name = 'Popper' and PersNr = gelesenVon
```

$\pi_{Titel}(\sigma_{Name = 'Popper' \text{ and } PersNr=gelesenVon}(\text{Professoren} \times \text{Vorlesungen}))$

VACUUM
Bei MVCC entsteht eine Tupel-Kopie pro Transaktion. Diese Tupel werden nach Transaktionsabschluss als gelöscht markiert - jedoch nicht gelöscht. VACUUM ist eine Art Garbage-Collection als Hintergrundprozess und aktualisiert die Statistik. Verwendet keinen Locks und kann im Betrieb ausgeführt werden. (VACUUM FULL verwendet locks)



Logisches Modell

```
bewertung (
id INTEGER,
datum DATE,
beschreibung TEXT,
anzahlSterne SMALLINT,
autor TEXT,
film NOT NULL REFERENCES film );
```