SQL

```
CREATE TABLE t (
    id INT PRIMARY KEY,
    c1 TEXT,
    name VARCHAR NOT NULL,
    price INT DEFAULT 0,
    PRIMARY KEY (id),
    FOREIGN KEY (c1) REFERENCES t2(c2),
    UNIQUE(id,name),
    CHECK(id> 0 AND price >= id)
);
DROP TABLE t; //Delete table
ALTER TABLE t ADD column; //Add new column
ALTER TABLE t DROP COLUMN c; //Drop column
ALTER TABLE t ADD constraint; //Add constraint
ALTER TABLE t1 RENAME TO t2; //Rename table
ALTER TABLE t DROP constraint; //Drop constraint
ALTER TABLE t1 RENAME c1 TO c2; //Rename column

INSERT INTO t(column_list) VALUES(value_list);
...SELECT column_list FROM t2;
UPDATE t SET c1= new_value;
UPDATE t SET c1 = new_value WHERE condition;
DELETE FROM t; //Delete all data
DELETE FROM t WHERE condition; //Delete subset

SELECT c1, c2 FROM t //Query data
SELECT c1, c2 FROM t1, t2 //cross join
SELECT * FROM t //Query all rows
SELECT DISTINCT... //Query distinct rows
    ...UNION [ALL] //Combine rows from queries
    ...INTERSECT //Return intersection
    ...MINUS //Subtract result

...WHERE condition //with condition
    ...IS [NOT] NULL
    ...[NOT] LIKE %xy_ //%=beliebig viele; $=ein Zeichen
    ...[NOT] IN value_list
    ...BETWEEN low AND high

...GROUP BY c1 //Group rows
...GROUP BY c1 HAVING condition; //Filter
...ORDER BY c1 ASC [DESC] //Sort result
...HAVING bedingung // !TODO
...LIMIT n OFFSET offset //return n rows

...INNER JOIN t2 ON condition
...LEFT JOIN t2 ON condition
...RIGHT JOIN t2 ON condition
...FULL OUTER JOIN t2 ON condition
...CROSS JOIN t2 //Cartesian product

CREATE VIEW v(c1,c2) AS SELECT c1, c2 FROM t;
CREATE RECURSIVE VIEW...
CREATE TEMPORARY VIEW...
...UNION [ALL]
...WITH [CASCADED | LOCAL] CHECK OPTION;
DROP VIEW view_name;

CREATE INDEX idx_name ON t(c1,c2);
CREATE UNIQUE INDEX idx_name ON t(c3,c4);
DROP INDEX idx_name;

CREATE OR MODIFY TRIGGER trigger_name
    [BEFORE | AFTER] [INSERT | UPDATE | DELETE]
    ON table_name FOR EACH [ROW | STATEMENT]
    EXECUTE stored_procedure;
DROP TRIGGER trigger_name;

CREATE ASSERTION name CHECK (praedikat)
GRANT rechte ON tabelle TO nutzer [with grant option]
REVOKE rechte ON tabelle TO nutzer [restrict | cascade ]

//AGGREGATE FUNCTIONS
AVG      //returns the average of a list
COUNT    //returns the number of elements of a list
SUM      //returns the total of a list
MAX      //returns the maximum value in a list
MIN      //returns the minimum value in a list
```

NoSQL (MongoDB)

```
db.createUser({
    user: "dbadmin",
    pwd: "12345678",
    roles: [{ role: "dbOwner", db: "admin" }],
})
db.users.insert({
    userid: "123",
    age: 18,
    name: "vikash",
})
db.users
    .distinct("name")   //show distinct value for
    .find(
        { //where clauses
            name: "vikash",
        },
        { //select fields
            name: 1,
            age: 1,
        }
    )
    .limit(2)
    .skip(5)
    .sort({
        age: 1,
    })
    .count() // number of documents in collection
db.users.update({
    name: "vikash",
},{
    $set: { age: 19 }, //update field
    $inc: { age: 5 }, //increase field
},{
    multi: true,
})
db.users.remove({})
db.users.remove({
    name: "vikash",
})
db.users.aggregate([
    { $match: {name: "vikash"} },
    { $group: {_id: "$age", num_usr: {$sum: 1} } }
    { $sort: {age: 1} }
])
db.users.ensureIndex()  //Create an index on field
db.users.dropIndex()    //Drop an index from field

show dbs //show all database
db.getName() //show current database
use usersdb //switch or create to database 'userdb'

db.dropDatabase() //drop current database
db.getCollectionNames() //show all collections
db.createCollection("users") //create collection 'users'
db.users.drop() //drop collection 'users'

//Aggregate Functions
$ne: "contr"     // negate clause
$or: [{a},{b}]   // a or b
$gt: x           // greater than x
$lt: x           // less than
$gte: x          // greater than even
name: /ind/,     // name is like %ind%
```

Datenunabhängigkeit:

- Stabilität der Benutzerschnittstelle gegen Änderungen
- physisch: Änderung der Dateiorganisation und Zugriffspfade haben keinen Einfluss auf das konzeptuelle Schema
- logisch: Änderung am konzeptuellen und gewissen externen Schemata haben keine Auswirkungen auf andere externe Schemata und Anwendungsprogramme

Coddßche Regeln

Integration einheitliche, nichtredundante Datenverwaltung
Operationen Speichern, Suchen, Ändern
Katalog Zugriffe auf DBbeschreibungen im Data Dictionary
Benutzersichten
Integritätssicherung Korrektheit des Datenbankinhalts
Datenschutz Ausschluss unautorisierter Zugriffe
Transaktionen mehrere DB-Operationen als Funktionseinheit
Synchronisation parallele Transaktionen koordinieren
Datensicherung Wiederherstellung von Daten nach Systemfehlern

Schemata:

- Konzeptuelles Schema (Ergebnis der Dateidefinition)
- Internes Schema (Festlegung Dateiorganisation/pfade = Index)
- Externes Schema (Ergebnis der Sichtdefinition)
- Anwendungsprogramm (Ergebnis der Programmierung)
  - Schema: Metadaten, Datenbeschreibung
  - Instanz: Anwenderdaten, Datenbankzustand

3 Schichten Architektur Klassifizierung

- Definitionskomponenten: Datendefinition, Dateiorganisation, Sichtdefinition
- Programmierkomponenten: DB-Programmierung mit eingebetteten DB-Operationen
- Benutzerkomponenten: Anwendungsprogramme, Anfrage und Update interaktiv
- Transformationskomponenten: Optimierer, Auswertung, Plattenzugriffssteuerung
- Data Dictionary (Datenwörterbuch): Aufnahme der Daten aus Definitionskomponenten, Versorgung der anderen Komponenten

5 Schichten Architektur Verfeinerung

- Datensystem: Übersetzung, Zugriffspfadwahl
- Zugriffssystem: Logische Zugriffspfade, Schemakatalog, Sortierung, Transaktionsverwaltung
- Speichersystem Speicherungsstrukturen, Zugriffspfadverwaltung, Sperrverwaltung, Logging, Recovery
- Pufferverwaltung: Systempufferverwaltung, Seitenersetzung, Seitenzuordnung
- Betriebssystem: Externspeicherverwaltung, Speicherzuordnung

Relationenalgebra

Selektion $\sigma_{Bedingung}(Relation)$: Auswahl von Zeilen (WHERE)
Projektion $\pi_{Attributmenge}(Relation)$: Auswahl von Spalten; entfernt doppelte Tupel; (SELECT DISTINCT)
Verbund $R_1 \bowtie R_2$: verknüpft Tabellen über gleichbenannte Spalten, verschmilzt jew Tupel gleicher Werte; Tupel ohne Partner eliminiert (JOIN/ NATURAL JOIN)
Umbenennung $\beta_{neu \leftarrow alt}(R)$: Ändern von Attributnamen (AS)
Vereinigung $r_1 \cup r_2$ von zwei Relationen r_1 und r_2 (UNION)

- Gesamtheit der beiden Tupelmengen
- Attributmengen beider Relationen müssen identisch sein

Differenz $r_1 - r_2$ eliminiert die Tupel aus der ersten Relation, die auch in der zweiten Relation vorkommen (EXCEPT)
Durchschnitt $r_1 \cap r_2$: ergibt die Tupel, die in beiden Relationen gemeinsam vorkommen (INTERSECT)
Quantoren/Mengenvergleiche $\Theta = \{all \| any \| some\}$
Assertion Prädikat, das eine Bedingung ausdrückt, die von der Datenbank immer erfüllt sein muss

Trigger Anweisung/Prozedur, die bei Eintreten eines bestimmten Ereignisses automatisch vom DBMS ausgeführt wird
Sicht virtuelle Relationen (bzw virtuelle Datenbankobjekte in anderen Datenmodellen)

Datenbankmodelle im Überblick

- HM: hierarchisches Modell,
- NWM: Netzwerkmodell,
- RM: Relationenmodell
- NF 2: Geschachtelte (Non-First-Normal-Form) Relationen
- eNF 2: erweitertes NF 2 -Modell
- ER: Entity-Relationship-Modell, SDM: semantische Datenmodelle
- OODM/C++: objektorientierte Datenmodelle
  - OEM: objektorientierte Entwurfsmodelle (etwa UML),
  - ORDM: objektrelationale Datenmodelle

ER Modell

Entity Objekt/Informationen
Entity Typ Gruppierung von Entitys mit gleichen Eigenschaften
Relation/Relationship Menge aller Einträge

- beschreibt eine Beziehung zwischen Entities
- Menge von Zeilen einer Tabelle

Attribut
- repräsentiert eine Eigenschaft von Entities/Beziehungen
- Spalte/Spaltenüberschrift einer Tabelle

Tupel Zeile einer Tabelle
Werte primitive Datenelemente; Attributwert
Schlüssel
- identifizierende Eigenschaft von Entitys
- minimale Menge von Attributen, die Tupel eindeutig identifizieren

Schlüsselattribute Teilmenge gesamter Attribute von Entity-Typs

- Auswahl des Primärschlüssels bei mehreren Schlüsselkandidaten
- Schlüssel durch Unterstreichen gekennzeichnet

Primärschlüssel ein beim Datenbankentwurf ausgezeichneter Schlüssel
Fremdschlüssel Attributmenge, die Schlüssel einer anderen Relation ist
Beziehungstypen Beziehungen zwischen Entities zu Beziehungstypen
Kardinalitäten/Funktionalität Einschränkung von Beziehungstypen bezüglich mehrfachen Teilnahme von Entitys an Beziehung
Stelligkeit/Grad Anzahl der an einem Beziehungstyp beteiligten Entity Typen

Stelligkeit

0,* legt keine Einschränkung fest (default)
1:1 jedem Entity e_1 ist maximal ein Entity e_2 zugeordnet und umgekehrt
1:N jedem Entity e_1 sind beliebig viele Entities E_2 zugeordnet, aber zu jedem Entity e_2 gibt es maximal ein e_1
N:1 invers zu 1:N, auf funktionale Beziehung
M:N keine Restriktionen
Kardinalitätsangaben
- partielle funktionale Beziehung: $lagertIn(Produkt[0,1], Fach[0,3])$
- totale funktionale Beziehung: $liefert(Lieferant[0,*], Produkt[1,1])$

Normalformen

- legen Eigenschaften von Relationenschemata fest
- verbieten bestimmte Kombinationen von funkt. Abhängigkeiten
- sollen Redundanzen und Anomalien vermeiden

Erste Normalform nur atomare Attribute in den Relationenschemata, d.h. als Attributwerte sind Elemente von Standard-Datentypen wie integer/string erlaubt, aber keine Konstruktoren [array/set]
Zweite Normalform
- Zweite Normalform eliminiert derartige partielle Abhängigkeiten bei Nichtschlüsselattributen
- partielle Abhängigkeit liegt vor, wenn ein Attribut funktional schon von einem Teil des Schlüssels abhängt

Dritte Normalform
- eliminiert auch transitive Abhängigkeiten
- etwa Weingut → Anbaugebiet und Anbaugebiet → Region
- 3 NF betrachtet nur Nicht-Schlüsselattribute als Endpunkt transitiver Abhängigkeiten
- $A \in R$ heißt transitiv abhängig von X bezüglich F genau dann, wenn es ein $Y \subseteq R$ gibt mit $X \to Y, Y \not\to X, Y \to A, A \notin XY$

Boyce-Kodd-Normalform (Verschärfung der 3NF): Eliminierung transitiver Abhängigkeiten auch zwischen Primattributen
$\nexists A \in R$: A transitiv abhängig von einem $K \in \mathbf{K}$ bezüglich F
Minimalität
- Global Redundanzen vermeiden
- andere Kriterien (wie Normalformen) mit möglichst wenig Schemata erreichen
- Beispiel: Attributmenge ABC, FD-Menge $A \to B, B \to C$

Vierte Normalform erweitertes Relationenschema $R = (R, \mathbf{K})$ ist in vierter Normalform (4NF) bezüglich M genau dann, wenn für alle $X \to\to Y \in M^+$ gilt: $X \to\to Y$ ist trivial oder $X \supseteq K$ für ein $K \in \mathbf{K}$

Datenbankentwurf

Anforderungsanalyse

- informale Beschreibung des Fachproblems
- Trennen der Informationen über Daten (Datenanalyse) von den Informationen über Funktionen (Funktionsanalyse)

Konzeptioneller Entwurf

- formale Beschreibung des Fachproblems
  - Sichtentwurf
  - Sichtanalyse
  - Sichtintegration
- Ergebnis: konzeptionelles Gesamtschema
- Integrationskonflikte

  Namenskonflikte Homonyme/Synonyme
  Typkonflikte verschiedene Strukturen für das gleiche Element
  Wertebereichskonflikte verschiedene Wertebereiche für Element
  Bedingungskonflikte verschiedene Schlüssel für ein Element
  Strukturkonflikte gleicher Sachverhalt durch unterschiedliche Konstrukte

Verteilungsentwurf

- sollen Daten auf mehreren Rechnern verteilt vorliegen, muss Art und Weise der verteilten Speicherung festgelegt werden
- horizontale Verteilung z.B. Kunden 1-100 und Kunden 101-200
- vertikale Verteilung z.B. Adresse in DB1, Konto in DB2

Logischer Entwurf

- Datenmodell des ausgewählten DBMS
- Transformation des konzeptionellen Schemas
- Verbesserung des relationalen Schemas anhand von Gütekriterien

Datendefinition

- Umsetzung des logischen Schemas in ein konkretes Schema
  - Datenbankdeklaration in der DDL des DBMS
  - Realisierung der Integritätssicherung
  - Definition der Benutzersichten

Physischer Entwurf

- Ergänzen des physischen Entwurfs um Zugriffsunterstützung
  - Zugriffspfad: Datenstruktur für zusätzlichen schlüsselbasierten Zugriff auf Tupel
  - meist als B*-Baum realisiert
- Sprachmittel: Speicherstruktursprache SSL

Implementierung & Wartung

- Wartung; weitere Optimierung der physischen Ebene
- Anpassung an neue Anforderungen und Systemplattformen
- Portierung auf neue Datenbankmanagementsysteme...

Kapazitätsändernde Abbildungen

- Kap.**erhöhend**: Abbildung auf R mit genau einem Schlüssel
- Kap.**vermindernd**: Relationsschema mit einem Schlüssel
- Kap.**erhaltend**: mit Schlüssel beider Entity Typen im Relationsschema als neuer Schlüssel

ER-auf-RM Abbildung

- neues Relationenschema mit allen Attributen des Beziehungstyps, zusätzlich Übernahme aller Primärschlüssel der Entity-Typen
- Festlegung der Schlüssel:
  - m:n-Beziehung: beide Primärschlüssel zusammen werden Schlüssel
  - 1:n-Beziehung: Primärschlüssel der n-Seite (Seite ohne Pfeilspitze) wird Schlüssel
  - 1:1-Beziehung: beide Primärschlüssel werden je ein Schlüssel, der Primärschlüssel wird dann aus diesen Schlüsseln gewählt
- optionale Beziehungen $[0,1]$ o. $[0,n]$ werden nicht verschmolzen
- bei Kardinalitäten $[1,1]$ oder $[1,n]$ Verschmelzung möglich
  - $1:n$: Entity-Relationenschema der n-Seite integrieren
  - $1:1$: beide Entity-Relationenschemata integrieren

Transformationseigenschaften

- Abhänggikeitstreue (T1)
  - Menge der Abhängigkeiten äquivalent zu der Menge der Schlüsselbedingungen im resultierenden Datenbankschema
  - S charakterisiert vollständig F genau dann, wenn $F \equiv \{K \to R | (R, \mathbf{K}) \in \mathbf{S}, \mathbf{K} \in \mathbf{K}\}$
- Verbundtreue (T2)
  - Originalrelationen können durch den Verbund der Basisrelationen wiedergewonnen werden
  - nicht verbundtreu $F = \{A \to B, C \to B\}$
  - verbundtreu $F - \{A \to B, B \to C\}$
- Mehrwertige Abhängigkeit (MVD)
  - innerhalb einer Relation r wird einem Attributwert von X eine Menge von Y-Werten zugeordnet, unabhängig von den Werten der restlichen Attribute → Vierte Normalform
  - Beseitigung von Redundanzen: keine zwei MVDs zwischen Attributen
  - Elimination der rechten Seite einer der beiden mehrwertigen Abhängigkeiten,
  - linke Seite mit dieser rechten Seite in neue Relation kopiert
- Verbundabhängigkeit (JD): R kann ohne Informationsverlust in $R_1, ..., R_p$ aufgetrennt werden: $\bowtie [R_1, ..., R_p]$
- Inklusionsabhängigkeit (IND): auf der rechten Seite einer Fremdschlüsselabhängigkeit nicht unbedingt der Primärschlüssel

Ableitungsregel

- F1: Reflexivität $X \supseteq Y \Rightarrow X \to Y$
- F2: Augumentation $\{X \to Y\} \Rightarrow XZ \to YZ$, sowie $XZ \to Y$
- F3: Transitivität $\{X \to Y, Y \to Z\} \Rightarrow X \to Y$
- F4: Dekomposition $\{X \to YZ\} \Rightarrow X \to Y$
- F5: Vereinigung $\{X \to Y, X \to Z\} \Rightarrow X \to YZ$
- F6: Pseudotransitivität $\{X \to Y, WY \to Z\} \Rightarrow WX \to Z$

F1-F3 bekannt als Armstrong-Axiome

- gültig (sound): Regeln leiten keine FDs ab, die logisch nicht impliziert
- vollständig (complete): alle implizierten FDs werden abgeleitet
- unabhängig (independent) oder auch bzgl. $\subseteq$
- minimal: keine Regel kann weggelassen werden

B-Axiome oder RAP-Regeln

- R Reflexivität $\{\} \Rightarrow X \to X$
- A Akkumulation $\{X \to YZ, Z \to AW\} \Rightarrow X \to YZA$
- P Projektivität $\{X \to YZ\} \Rightarrow X \to Y$

Membership Problem Kann eine bestimmte FD $X \to Y$ aus der vorgegebenen Menge F abgeleitet werden, d.h. wird sie von F impliziert? $X \to Y \in F^+$

Reduktionsoperationen Entfernen überflüssiger Attribute auf linker bzw. rechter Seite von FDs.

Unwesentliche Attribute A heißt unwesentlich in $X \to Y$ bzgl. F, wenn

- A kann aus der FD $X \to Y$ entfernt werden, ohne dass sich die Hülle von F ändert
- FD $X \to Y$ heißt linksreduziert, wenn kein Attribut in X unwesentlich ist
- FD $X \to Y$ heißt rechtsreduziert, wenn kein Attribut in Y unwesentlich ist

Minimale Überdeckung Eine minimale Überdeckung ist eine Überdeckung, die eine minimale Anzahl von FDs enthält

Äquivalenzklassen FDs mit äquivalenten linken Seiten werden zu einer Äquivalenzklasse zusammengefasst

Entwurfsverfahren

- T1: S charakterisiert vollständig F
- S1: S ist in 3NF bezüglich F
- T2: Dekomosition von U in $R_1, ..., R_p$ ist verbundtreu bezüglich F
- S2: Minimalität, d.h. „$\nexists S$" : S" erfüllt T1,S1,T2 und $|S < |S|$

Algebra & Kalkül

Anfrage Folge von Operationen, die aus Basisrelationen eine Ergebnisrelation berechnet
Sicht Folge von Operationen, die unter Sichtnamen langfristig abgespeichert wird und unter diesem Namen wieder aufgerufen werden kann; ergibt eine Sichtrelation
Snapshot Ergebnisrelation einer Anfrage, die unter Snapshot-Namen abgelegt wird, aber nie ein zweites Mal berechnet wird

Kriterien für Anfragesprachen

Ad-Hoc-Formulierung Benutzer soll eine Anfrage formulieren können, ohne ein vollständiges Programm schreiben zu müssen
Deskriptivität Benutzer soll formulieren „Was will ich haben?"
Mengenorientiertheit jede Operation soll auf Mengen von Daten gleichzeitig arbeiten
Abgeschlossenheit Ergebnis ist wieder Relation und kann als Eingabe für nächste Anfrage verwendet werden
Adäquatheit alle Konstrukte des zugrundeliegenden Datenmodells werden unterstützt
Orthogonalität Sprachkonstrukte sind in ähnlichen Situationen auch ähnlich anwendbar
Optimierbarkeit Sprache besteht aus wenigen Operationen, für die es Optimierungsregeln gibt
Effizienz jede Operation ist effizient ausführbar (Komplexität max $\leq O(n^2)$)
Sicherheit keine Anfrage, die syntaktisch korrekt ist, darf in eine Endlosschleife geraten oder ein unendliches Ergebnis liefern
Eingeschränktheit Anfragesprache darf keine komplette Programmiersprache sein
Vollständigkeit Sprache muss mindestens die Anfragen einer Standardsprache ausdrücken können

Minimale Relationenalgebra $\Omega = \pi, \sigma, \bowtie, \beta, \cup, -$
unabhängig kein Operator kann weggelassen werden
Relationale Vollständigkeit jede andere Menge von Operationen genauso mächtig wie Ω
strenge relationale Vollständigkeit zu jedem Ausdruck mit Operatoren aus Ω gibt es einen Ausdruck auch mit der anderen Menge von Operationen

Verbundvarianten

- Gleichverbund: Gleichheitsbedingung über explizit angegebene und evtl. verschiedene Attribute $r(R) \bowtie_{C=D} r(S)$
- Theta-Verbund (Θ-join): beliebige Verbundbedingung $r(R) \bowtie_{C>D} r(S)$
- Semi-Verbund: nur Attribute eines Operanden erscheinen im Ergebnis $r(L) \bowtie r(R) = \pi_L(r(L) \bowtie r(R))$
- äußere Verbunde (engl. outer join)
  - voller ä.V. übernimmt alle Tupel beider Operanden
  - linker ä.V. übernimmt alle Tupel des linken Operanden
  - rechter ä.V. übernimmt alle Tupel des rechten Operanden

Anfragekalküle

Kalkül eine formale logische Sprache zur Formulierung von Aussagen
Ziel Kalkül zur Formulierung von Datenbank-Anfragen
allgemeines Kalkül
- Anfrage hat die Form $\{f(\bar{x})|p(\bar{x})\}$
  - x bezeichnet Menge von freien Variablen
  - Funktion f bezeichnet Ergebnisfunktion über $\bar{x}$
  - p Selektionsprädikat über freien Variablen $\bar{x}$
- Bestimme aller Belegungen der freien Variablen in x, für die das Prädikat p wahr wird.

Relationale Kalküle
- Bereichskalkül: Variablen nehmen Werte elementarer Datentypen (Bereiche) an
- Bereichskalkül ist streng relational vollständig, d.h. zu jedem Term τ der Relationenalgebra gibt es einen äquivalenten (sicheren) Ausdruck η des Bereichskalküls.
- Atomare Formeln: $\{x_1, ..., x_n \| \phi(x_1, ..., x_n)\}$

Basiskalkül
- Einschränkung des Bereichskalküls: nur Konstanten, keine Funktionen
- Tupelkalkül: Variablen variieren über Tupelwerte

Semantisch sichere Anfragen Anfragen, die für jeden Datenbankzustand $\sigma(R)$ ein endliches Ergebnis liefern
Syntaktisch sichere Anfragen Anfragen, die syntaktischen Einschränkungen unterliegen, um die semantische Sicherheit zu erzwingen

Transaktion, Integrität & Trigger

Typintegrität Angabe von Wertebereichen zu Attributen; Erlauben/Verbieten von Nullwerten
Schlüsselintegrität Angabe eines Schlüssels für eine Relation
Referentielle Integrität die Angabe von Fremdschlüsseln
Semantische Integrität Korrekter (konsistenter) DB-Zustand nach Ende der Transaktion
Ablaufintegrität Fehler durch "gleichzeitigenSZugriff mehrerer Benutzer auf dieselben Daten vermeiden

Transaktionen fordern ACID Eigenschaften

Atomicity Transaktion wird entweder ganz oder gar nicht ausgeführt
Consistency Datenbank ist vor Beginn und nach Beendigung einer Transaktion jeweils in einem konsistenten Zustand
Isolation Nutzer, der mit einer Datenbank arbeitet, sollte den Eindruck haben, dass er mit dieser Datenbank alleine arbeitet
Durability nach erfolgreichem Abschluss einer Transaktion muss das Ergebnis dieser Transaktion „dauerhaft" in der Datenbank gespeichert werden

Kommandos einer Transaktionssprache

- Beginn einer Transaktion: Begin-of-Transaction-Kommando BOT
- commit: die Transaktion soll erfolgreich beendet werden
- abort: die Transaktion soll abgebrochen werden

Probleme im Mehrbenutzerbetrieb

Nonrepeatable Read gleiche Leseanweisung führt zu nicht wiederholbaren Ergebnissen
Dirty read rechnen mit Wert einer anderen Transaktion die abgebrochen wird
Phantom-Problem liest Wert anderer Transaktion ohne zu erkennen, dass andere Transaktion noch nicht abgeschlossen
Lost Update updates gehen verloren, wenn gleiche Variablen gleicheitig beschrieben werden (oder kurz nacheinander)
Deadlock ein oder mehrere Transaktionen warten, einen LOCK auf Datenbankobjekte abzusetzen und behindern sich gegenseitig
Startvation Warteschlange für gesperrte Objekte unfair abgearbeitet. Transaktion wartet endlos

Isolationsebenen

read uncommitted
- schwächste Stufe: Zugriff auf nicht geschriebene Daten, nur für read only Transaktionen
- statistische und ähnliche Transaktionen (ungefährer Überblick, nicht korrekte Werte)
- keine Sperren → effizient ausführbar, keine anderen Transaktionen werden behindert

read committed nur Lesen endgültig geschriebener Werte, aber nonrepeatable read möglich
repeatable read kein nonrepeatable read, aber Phantomproblem kann auftreten
serializable garantierte Serialisierbarkeit

Integritätsbedingungen in SQL

not null Nullwerte verboten
default Angabe von Default-Werten
check (search-condition) festlegung lokaler Integritätsbedingungen innerhalb der zu definierenden Wertebereiche, Attribute und Relationenschemata
primary key Angabe eines Primärschlüssel
foreign key Angabe der referentiellen Integrität
create domain Festlegung eines benutzerdefinierten Wertebereichs

Erhaltung der referentiellen Integrität

on update | **delete** Angabe eines Auslöseereignisses, das die Überprüfung der Bedingung anstößt
cascade | **set null** | **set default** | **no action** Behandlung einiger Integritätsverletzungen pflanzt sich über mehrere Stufen fort
deferred | **immediate** legt Überprüfungszeitpunkt für eine Bedingung fest
deferred Zurückstellen an das Ende der Transaktion
immediate sofortige Prüfung bei jeder relevanten Datenbankänderung

Sichten und Zugriffskontrolle

- Sichten sind externe DB-Schemata folgend der 3-Ebenen-Schemaarchitektur
  - Sichtdefinition
  - Relationenschema (implizit oder explizit)
  - Berechnungsvorschrift für virtuelle Relation, etwa SQL-Anfrage
- Vorteile
  - Vereinfachung von Anfragen für den Benutzer der Datenbank, etwa indem oft benötigte Teilanfragen als Sicht realisiert werden
  - Möglichkeit der Strukturierung der Datenbankbeschreibung, zugeschnitten auf Benutzerklassen
  - logische Datenunabhängigkeit ermöglicht Stabilität der Schnittstelle für Anwendungen gegenüber Änderungen der Datenbankstruktur
  - Beschränkung von Zugriffen auf eine Datenbank im Zusammenhang mit der Zugriffskontrolle

Änderungen auf Sichten

Kriterien

Effektkonformität Benutzer sieht Effekt als wäre die Änderung auf der Sichtrelation direkt ausgeführt worden
Minimalität Basisdatenbank sollte nur minimal geändert werden, um den erwähnten Effekt zu erhalten
Konsistenzerhaltung Änderung einer Sicht darf zu keinen Integritätsverletzungen der Basisdatenbank führen
Datenschutz Wird die Sicht aus Datenschutzgründen eingeführt, darf der bewusst ausgeblendete Teil der Basisdatenbank von Änderungen der Sicht nicht betroffen werden

Klassifikation der Problembereiche

- Verletzung der Schemadefinition
- Datenschutz: Seiteneffekte auf nicht-sichtbaren Teil der Datenbank vermeiden
- nicht immer eindeutige Transformation: Auswahlproblem
- Aggregierungssichten: keine sinnvolle Transformation möglich
- elementare Sichtänderung soll genau einer atomaren Änderung auf Basisrelation entsprechen: 1:1-Beziehung zwischen Sichttupeln und Tupeln der Basisrelation

Einschränkungen für Sichtänderungen

- änderbar nur Selektions- und Projektionssichten
- 1:1-Zuordnung von Sichttupeln zu Basistupeln: kein distinct in Projektionssichten
- Arithmetik und Aggregatfunktionen im select-Teil sind verboten
- genau eine Referenz auf einen Relationsnamen im from-Teil erlaubt
- keine Unteranfragen mit „Selbstbezug“ im where-Teil erlaubt
- group by und having verboten

Statistische Datenbanken

- Einzeleinträge unterliegen Datenschutz, aber statistische Informationen allen Benutzern zugänglich
  - keine Anfragen, die weniger als n Tupel selektieren
  - statistische Anfragen nicht erlauben, die paarweise einen Durchschnitt von mehr als m vorgegebenen Tupeln betreffen
  - Data Swapping: Vertauschen von Attributwerten einzelner Tupel
  - Generalisierung: Attributwerte durch allgemeinere Werte ersetzen, die einer Generalisierungshierarchie entnommen sind (Alter 30-40, Weglassen von Stellen PLZ)
  - Löschen von Tupeln, welche die k-Anonymität verletzen und damit identifizierbar sind
- k-Anonymität: ein bestimmter Sachverhalt kann nicht zwischen einer vorgegebenen Anzahl k von Tupeln unterschieden werden

Datenmodelle für NoSQL

KV-Stores binäre Relationen, bestehend aus einem Zugriffsschlüssel (dem Key) und den Nutzdaten (dem Value)

- binäre Daten ohne Einschränkung,
- Dateien oder Dokumente, → Document Databases
- oder schwachstrukturierte Tupel → Wide Column Store

Wide Column KV-Store mit schwachstrukturiertem Tupel als Value = Liste von Attributname-Attributwert-Paaren

- schwache Typisierung für Attributwerte (auch Wiederholgruppen)
- nicht alle Einträge haben die selben Attributnamen
- Hinzufügen eines neuen Attributs unproblematisch
- Nullwerte aus SQL ersetzt durch fehlende Einträge

Document Stores KV-Store mit (hierarchisch) strukturiertem Dokument als Value

- JSON-Format: geschachtelte Wide Column-Daten
- XML (eher unüblich auf KV-Stores)

Graph Stores spezielle Form der Datenrepräsentation = Graphen, insb. Beziehungen zwischen Objekten

Tiefensuche (DFS) zunächst rekursiv alle Kindknoten besuchen bevor alle Geschwisterknoten besucht werden (Bestimmung der Zusammenhangskomponente)
Breitensuche (BFS) zunächst alle Geschwisterknoten besuchen bevor die Kindknoten besucht werden (Bestimmung des kürzesten Weges)

Subjekt-Prädikat-Objekt-Modell: RDF

- Sprache zur Repräsentation von Informationen über (Web)-Ressourcen
- zentraler Bestandteil von Semantic Web, Linked (Open) Data
- Repräsentation von Daten, aber auch Wissensrepräsentation (z.B. Ontologie)

Property-Graph-Modell Knoten und (gerichtete) Kanten mit Eigenschaften (Properties)

- Elemente: Nodes, Relationships, Properties, Labels
- Properties = Key-Value-Paare: Key (=String), Value (=Java-Datentypen + Felder)
- Nodes mit Labels (≈ Klassenname)
- Relationships: sind gerichtet, mit Namen und ggf. Properties
- Anfragen $(e : ERZEUGER) - [: LiegtIn] - > (a : ANBAUGEBIET\{gebiet : "NapaValley"\})$
- match: Beispielmuster für Matching
- return: Festlegung der Rückgabedaten (Projektion)
- where: Filterbedingung für „gematchte“ Daten
- create: Erzeugen von Knoten oder Beziehungen
- set: Ändern von Property-Werten

Ontologie = formale Spezifikation einer Konzeptualisierung, d.h. einer Repräsentation von Begriffen (Konzepten) und deren Beziehungen
Vokabular: vordefinierte Klassen und Eigenschaften

Anwendungsprogrammierung

Programmiersprachenanbindung

- prozedurale oder CALL-Schnittstellen (call level interface)
- Einbettung einer DB-Sprache in Programmiersprachen
- Spracherweiterungen und neue Sprachentwicklungen

Datenbankzugriffsschnittstelle

Java JDBC

Embedded SQL für Java SQLJ

LINQ Language Integrated Query; Einbettung einer DB-Sprache in eine Programmiersprache (C#)

Hibernate Java-Framework für objekt-relationales Mapping

- DriverManager: Einstiegspunkt, Laden von Treibern
- Connection: Datenbankverbindung
- Statement: Ausführung von Anweisungen über eine Verbindung
- ResultSet: verwaltet Ergebnisse einer Anfrage, Zugriff auf einzelne Spalten

Transaktionssteuerung

- Methoden von Connection
  - commit ()
  - rollback ()
- Auto-Commit-Modus
  - implizites Commit nach jeder Anweisung
  - Transaktion besteht nur aus einer Anweisung
  - Umschalten mittels setAutoCommit (boolean)

Ausnahmebehandlung

- Auslösen einer Ausnahme (Condition) ßignal ConditionName;”
- Deklarieren von Ausnahmen

```
$declare fehlendes_weingut condition;
    declare ungueltige_region
    condition for sqlstate value "40123";$
```

```
create function geschmack (rz int)
returns varchar(20)
    begin
        return case
            when rz <= 9 then "Trocken"
            when rz > 9 and rz <= 18 then "Halbtrocken"
            when rz > 18 and rz <= 45 then "Lieblich"
            else "Suess"
        end
    end
//Aufruf innerhalb einer Anfrage
select Name, Weingut, geschmack(Restzucker) from WEINE
    where Farbe = "Rot" and geschmack(Restzucker) = "Trocken"
```

Prozeduren Prozedurdefinition

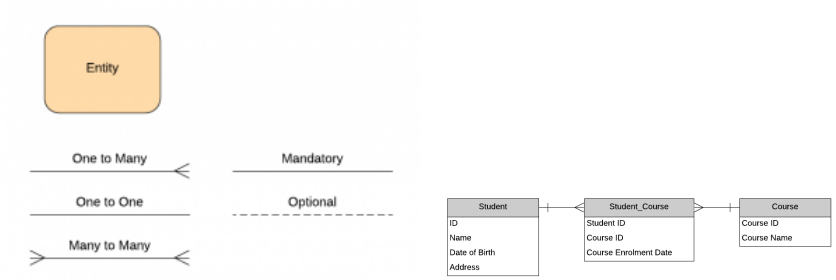
```
create procedure weinliste (in erz varchar(30),
    out wliste varchar(500))
    begin
        declare pos integer default 0;
        for w as WeinCurs cursor for
            select Name from WEINE where Weingut = erz
            do xy
        end for;
    end;
end;
// Nutzung ueber call-Anweisung
declare wliste varchar(500);
call weinliste ("Helena", wliste);
```

Eigenschaften von Prozeduren, die Anfrageausführung und -optimierung beeinflussen

deterministic Routine liefert für gleiche Parameter gleiche Ergebnisse
no sql Routine enthält keine SQL-Anweisungen
contains sql Routine enthält SQL-Anweisungen (Standard für SQL-Routinen)
reads sql data Routine führt SQL-Anfragen (select-Anweisungen) aus
modifies sql data Routine, die DML-Anweisungen (insert, update, delete) enthält

Klausurvorbereitung

Entity-Relationship-Diagramm



relationale Algebra

$\sigma_{\{Datum=14.12.2017\}}(PRUEFUNG) \bowtie (\pi_{\{Matrikel,Name\}}(STUDENT))$

Bereichskalkül

Bereichskalkül: $KUNDE(x, y, z)$ vs. Tupelkalkül: $KUNDE(k)$
Ein Ausdruck hat die Form:$\{x1, x2, ...|\phi(x1, x2, ...)\}$

Beispiel: Kunden mit Bestellung
$KUNDE(kdnr, kname, adresse, ort)$ und $AUFTRAG(auftragsnr, kdnr, warennr, menge)$
Kunden mit Bestellung: $\{x, y, z|KUNDE(x, y, z, \_) \wedge AUFTRAG(\_, x, \_, \_)\}$

SQL Anfrage

```
SELECT Matrikel, Name FROM Student WHERE Matrikel>2010;
```

Integrität

Nutzerdefinierte Domäne eines Attributs, d.h. bestimmte Werte

Primärschlüssel PRIMARY KEY nicht null, existiert und ist unique

Fremdschlüssel FOREIGN KEY wie primary key aus anderer tabelle

CHECK-Bedingungen werte vor insert prüfen

ASSERTIONs

Trigger

CLOSURE-Algorithmus / RAP-Regeln

R	Reflexivität	$\{\} \Rightarrow X \rightarrow X$
A	Akkumulation	$\{X \rightarrow YZ, Z \rightarrow AW\} \Rightarrow X \rightarrow YZA$
P	Projektivität	$\{X \rightarrow YZ\} \Rightarrow X \rightarrow Y$

Normalformen

1.NF Wertebereiche der Merkmale sind atomar (es gibt keine zusammengesetzten Werte).
2.NF 1. NF + Nichtschlüsselmerkmale sind von allen Schlüsselmerkmalen voll funktional abhängig.
3.NF 2. NF + kein Nichtschlüsselmerkmal ist von irgendeinem Schlüssel transitiv abhängig.
Boyce Codd 3.NF + entgernen funktionaler Abhängigkeiten
4.NF BC + Abhängigkeiten von mehrwertigen Attributmengen trivial und Attributmenge der Schlüsselkandidat

Konsultation

Mehrere theoretische Fragen? Nicht multiple choice wie in Moodle sondern zB Definitionen erklären (was ist Transaktion/Sicht?)
ERD-¿Relationenschema bei 1:N Beziehung kommt Schlüssel von N Seite bei M:N Beziehung kommen Schlüssel von beiden Seiten bei 1:1 Beziehung kann Schlüssel von entweder einer oder der anderen seite gewählt werden