1 Überblick

Was laut den Klausuren relevant sein könnte

- Nennen sie drei Modelle von Datenbanken mit einer Kurzbeschreibung
- Relationale Algebra
- Definieren Sie die erste, zweite und dritte Normalform
- ER-Modelle
- Transaktion
- Constraint/Tabellenconstaint
- Gründe für dein Einsatz von Datenbanken
- Man muss eine Datenbank entwickeln
- Auflösen von einer indirekten Relation
- Vorgehen bei Abfragenverarbeitung, Fragen wie "Was passiert bei SELECT * FROM EMP"
- Was ist ein Pubic Synonym?
- Was ist ein Schema? (Datenbankschema: Tabellen, Views etc.)
- Unterscheid von PL/SQL(Performance, sicher - ANwender bekommt Schnittstelle - ohne dem Nutzer Rechte auf den Tabellen zu geben)
- Was ist eine Transaktion? Eine Folge von SELECTS, INSERTS, DELETES, welche ACID-eigenschaften besitzt
- Beschreiben sie die ACID eigenschaften
- 2 Probleme, die bei Mehrbenutzerbetrieb auftreten können

Auf der Cloud steht auch nochmal was Klausurrelevant sein könnte

2 Zusammenfassung

2.1 Klassifizierung der Datenbanksysteme

- Flache Datein (vor DB-Zeitalter/ohne DB)
- Hierarchische Datenbanken

- Netzwerkdatenbanken
- Relationale und objektrelationlae Datenbanken
- Objektorientierte Datenbanken

Hierarchische Datenbanken

Haben Datenmodell in der Art eines **Baumes**. Die Adressverknüpfungen/Adressverweise werden mit den Daten gespeichert. Ein Datensatz kann nur mit einem übergeordenten und mehreren untergeordneten Datensätzen in Beziehung stehen.

Netzwerkdatenbanken

Die **Adressverweise** sind in der Datenbank gespeichert genau wie in der hierarchische Datenmodel. Datensätze können miteinander über mehrere Verknüpfungen in Beziehung stehen. Muss keine Baumstruktur sein.

Relationale Datenbanken

Die Daten und ihre Beziehungen werden in Tabellen(Relationen) abgebildet. Eine Tabelle ist horizontal in Zeilen und vertikal in Spalten aufgeteilt. Jeder Datensatz wird als Ziele dargestellt. Die Zeile heißen **Tupel** und die Spaltenüberschreifen **Attribute**.

Objektorientierte Datenbanken

Die Daten als Objekte, mit ihren Methoden und Attribute in der Datenbank gespeichert. Die Defintion eigener Objekte ist sehr eng an eine objektorientierte Programmiersprache angelehnt.

Objektrelationale Datenbanken

- Bauen auf relationalen Datenbanken auf
- Erlauben die Definition eigener Datentypen
- Typisirte Tabellen und Tupeltabellen
- Vererbung und Methoden
- In vielen großen Datenbanken heute
- Haben im Gegensatz zu objektorientierten Datenbanken eine große Verbreitung gefunden

Deduktive Datenbank

Eine deduktive Datenbank ist eine Art von Datenbank, die auf der Logikprogrammierung basiert. Sie ermöglicht das Speichern von Fakten und Regeln sowie das Ableiten neuer Informationen durch logische Schlussfolgerungen auf Basis dieser Fakten und Regeln.

DB-Architektur

3-Schichten

- externe Ebene: Schnittstelle für Benutzer und Anwendungen
- konzeptionelle Ebene: beinhaltet Daten und Beziehungen (Ziel $\longrightarrow$ Redundanzfrei)
- interne/physische: abspeicherung im PC

Analyse/Entwurf

1. Analyse
2. Desgin/Entwurf
3. Implementierung
4. Abnahme/Einführung
5. Wartung

2.2 ER-Modelle

Ist ein Bild welches eine Datenbank darstellt

- enthält alle Informationseinheiten, die im zu implementierenden DBS enthalten sein sollen
- hilft beim Desgin von DB

Eine **Entität**

- Objekt(eindeutig identifizierbar)
- z.B. Tabelle

Schlüssel: Min. Menge an Attributen um Entität eindeutig zu bestimmten
Beziehung: Zusammenhang zur Entität siehe andere Zusammenfassung

2.3 Relationale Algebra

Unter **Relationalen Algebra** versteht man eine formale Sprache, mit der sich Abfragen über ein Schema formulieren lassen. Sie stellt eine Reihe von Operationen zur Verfügung, um auf Relationen(Tabellen) zuzugreifen, diese zu manipulieren und Informationen abzufragen.
Eine **Relation** ist eine strukturierte Tabelle im relationalen Datenbankmodell, die aus Tupeln (Zeilen) und Attributen (Spalten) besteht. Jedes Tupel repräsentiert eine Instanz einer Entität, während jedes Attribut eine bestimmte Eigenschaft dieser Entität darstellt. **Relationen** ermöglichen die organisierte Speicherung und Abfrage von Daten in einer tabellarischen Form.
Relationen und Attribute

- n-stellige Relationale
- Attribute sind Spaltenüberschriften
- Degree = Anzahl der Spalten
- Domäne = Wertebereich
- Tupel = Zeile

Es gibt grundopertaionen

1. **Auswahl** (Selektion): Ermöglicht dsa Filtern von Tupeln (Zeilen) basierend auf bestimmten Bedingungen.
2. **Projektion:** ermöglicht die Auswahl bestimmter Spalten oder Attribute aus einer Relation.
3. **Union** (=Vereinigung): Kombiniert zwei Relationen und entfernt dabei Duplikate.
4. **Differenz:** Ermittelt die Unterschiede zwischen zwei Relationen.
5. **Kreuzprodukt:** Bildet das kartesische Produkt zweier Relationen.
6. **Verbund** (=Natural Join): Verküpft Tupel aus verschiedenen Relationen basierend auf gemeinsamen Attributen.
7. **Division**

Das **kartesische Produkt** wird verwendet um das Kreuzprodukt zweier Relationen zu erstellen. Jeder Tupel aus den beiden Relationen wird kombiniert, um alle möglichen Kombinationen von Tupeln darzustellen.

$$M_1 = \{A, B, C\}\, M_2 = \{1, 2\}$$
$$M_1 \times M_2 = \{(A,1), (A,2), (B,1), (B,2), (C,1), (C,2)\}$$

Selektion ermöglicht das Filtern von Daten und das Extrahieren von relevanten Inforamtionen aus einer Tabelle. Die Selektionsprädikate enthalten:

SELECT $*$ FROM emp Where...

- Attribute einer Relation und Konstanten
  - $<$ Attribut $>$ <Vergleichsoperator> $<$ Konstante $>$
  - $<$ Attribut $>$ <Vergleichsoperator> $<$ Attribut $>$
- Vergleichsoperator $=, <, <=, >, <=$ und $<>$ und $!=$ (ungleich)
- die logischen Operatoren UND, ODER und NICHT
- eine beliebige Zusammensetzung aus den obengenannten Möglichkeiten

$$B_{a_1,\ldots,a_2} = \text{wahr, falls Kun\_Nr} < 5 \text{ UND Geschlecht} = w$$
$$= \text{wahr, falls Kun\_Nr} > 5\text{ODER Geschlecht} = m$$

Bei der **Projektion** werden nur die spezifizierten Attribute beibehalten, während alle anderen Attribute entfernt werden. Das Ergebnis der **Projektion** ist eine neue Relation, die nur die ausgewählten Attribute enthält. Dadurch kann eine Relation auf bestimmte relevante Informationen reduziert werden. Es werden Attribute/Spalten aus der Tabelle raus gestrichen.

SELECT Kdnr, sa, datum FROM emp

Bei der **Union** (Vereinigung) werden Werte aus beiden Tabellen angezeigt

SELECT $*$ FROM emp UNION SELECT $*$ FROM Kunde

doppelte Zeilen werden nur 1 mal angezeigt

Bei dem **Durchschnitt** werden Werte angezeigt die in beiden Tabellen vorhanden sind

SELECT ... INTERSECT SELECT...

Differenz -zieht zweite von erster abgebildet

SELECT... MINUS SELECT...

alle Zeilen aus Tabelle 1, die nicht in Tabelle 2 sind

Division: Umkehrfunkion des Kartesischem Produkts

Bei einem **Theta-Join** wird zurest ein kartesiches Produkt von 2 Tabellen gebildet. Dannach wird nach einer Bedingung selektiert und übriggeliene Tupel an geziegt.

$$\text{Tabelle1} = \text{Kunden} : \text{KundenID, Name}$$
$$\text{Tabelle2} = \text{Bestellungen} : \text{BestellungsID, KundenID, Produkt}$$
$$\text{Kunden} \bowtie (\text{Kunden.KundenID} = \text{Bestellungen.KundenID})$$
$$\text{Tabelle3} = \text{Bestellungen} : \text{KundenID, Name, BestellungsID, Produkt}$$

Equi-Join ist wie der **Theta-Join** nur das der Vergleichsoperator (=) beschränkt ist.

Natural-Join: Equi-Join bei dem alle Attribute in beiden Tabellen gleich heißen und auf Gleichheit überprüft werden

- Verlustfreie Join Operatoren: Equi-Join und Natural Join sind Verlustfrei, wenn alle Tupel beider Tabellen am Verbund teilnehmen
- Dangling Datensätze: Datensätze, denen bei Join-Operationen die entsprechenden Datensätze zur Verknüpfung mit anderen Tabellen fehlen

Linker-Outer-Join Join mit Bedingung, bei dem Tabelle 1 komplett übernommen wird und fehlende Werte aus Tabelle 2 mit NULL aufgefüllt werden
Rechter-Outer-Join Tabelle 2 wird komplett übernommen fehlende Werte aus Tabelle 1 mit NULL aufgefüllt
Eigenschaften von Join-Operatoren

- Attribute müssen keine Schlüsselattribute sein
- Join-Attribute beider Relationen müssen nicht gleichen Namen haben, außer bei Natural Join wegen Equivalenzbedingung
- Jede Relation kann mit jeder gejoint werden
- Die Domänen (Wertebereich) beider Join-Attributen müssen gleich sein

Rechen-Operationen

- SELECT = Projektion
- FROM = kartesisches Produkt
- Tabelle Join Tabelle ON Bedingung & Join
- WHERE = Selektion

Eigenschaften von Relationen

- hat keine doppelten Tupel
- Tupelreihenfolge nicht definiert
- Informationen werden durch Werte darstellt
- Attribute sind atomar (Was auch immer das heißt?)
- NULL-Werte sind erlaubt

Operatorbäume stellen Schritte einer Abfrage grafisch dar.

$$\text{Projektion} \Longrightarrow \text{Selektion} \Longrightarrow ...$$

Funktionale Abhängigkeit

- Konzept der relationalen Entwurfstheorie
- Grundlage der Normalisierung
- FA, wenn Attribute eindeutig die Werte anderer bestimmten
- $x \longrightarrow y$ "x bestimmt y"
- z.B Kind hat immer gleichen Vater/Mutter $\Longrightarrow$ FA
- z.B Kind hat unterschiedliche/mehrere Opas/Omas $\Longrightarrow$ nicht FA

Schlüssel

- Primärschlüssel -Jede Relatin hat genau einen
- Fremdschlüssel -Primärschlüssel einer anderen Relation
- Zweitschlüssel - dienen zum schnellen Zugriffe auf Spalten, die nicht zum Primärschlüssel gehören

3 Fehlende Punkte in der anderen Zusammenfassung

3.1 Abfrageverarbeitung

3.2 Anfrageoptimierung

3.3 Normalisierung

3.4 1. Normalverteilung

3.5 2. Normalverteilung

3.6 3. Normalverteilung

3.7 Constraints

3.8 Datendefinitionssprache

3.9 Spaltendefinition

- Spaltenname (ohne Umlaufe, Sonderzeichen)
- Datentyp
- Default
- Constraints

Alternative Ansatz: **Integritätsprüfung**

- inden Anwedungsprogrammen programmiert
- statische Integritätsprüfung: müssen beim einfügen geprüft werden
- dynamische Integritätsprüfung: müssen dauerhaft überprüft werden

DB konsistent wenn alle Integritätsbedingungen erfüllt sind. DB zu Beginn immer Konsistent $\longrightarrow$ Inkonsistenz nur durch Veränderung(DML-Befehle) möglichen

3.10 Regeln zum Ändern von Spalten

Jederzeit: Nachkommatellen erhöhen, Anz. Ziffern erhöhen
Wenn Spalte NULL-Werte hat: Nachkommastellen reduzieren Anz. Ziffern erhöhen verringern Datentyp ändern
! Nicht möglich, Spalte nachträglich umzubennen

3.11 Datenbankschema und Modellierung

3.12 Dreiwertige Logik

- TRUE, FALSE, UNKNOWN
- unknown, wenn NULL mit anderem Wert verglichen wird
- False hat Vorrang vor unknown
- $TRUE \cap$ unknown = unknown
- F $\cap$ unknown = False
- T $\cup$ unkown = unknown

3.13 Vorteile von PL/SQL

3.14 Unterschied zwischen Prozedur und Trigger

3.15 Cursor

3.16 Transaktion

3.17 ACID-Eigenschaften

3.18 Commit und Rollback

3.19 Synonyme

3.20 Probleme bei Mehrbenutzerbetrieb

3.21 Lösung dieser Probleme

3.22 Deadlock