Datenorganisation

Shaker-Sort

Verbesserung von Bubble-Sort.

1. Die letzten beiden Elemente werden verglichen – falls nötig ausgetauscht
2. Das drittletzte und das vorletzte Element werden verglichen – falls nötig ausgetauscht, usw. bis zu ersten und zweiten Element

    - Bei jedem Durchlauf abprüfen, ob überhaupt noch eine Vertauschung notwendig war (merken der Position der letzten Vertauschung)
    - Richtung abwechseln

Shaker-Sort hat ein quadratisches Laufzeitverhalten. Es sind nur interne Zugriffszeiten berücksichtigt.

Beispiel:

	↓		↓		↓		↓		↓		↓		
25	2												2
23	25	23	4										4
22	23	22	23	22	7								7
27	22	25	22	23	22	22	9						9
24	27	24	25	24	23	23	22	22	12				12
19	24	19	24	19	24	19	23	19	22	19	13		13
20	19	20	19	20	19	20	19	20	19	20	19	19	16	16
4	20	4	20	7	20	9	20	12	20	13	20	16	19	19
2	4	7	7	9	9	12	12	13	13	16	16	20		20
9	7	9	9	12	12	13	13	16	16	22				22
12	9	12	12	13	13	16	16	23						23
7	12	13	13	16	16	24								24
13	13	16	16	25										25
16	16	27												27
	↑		↑		↑		↑		↑		↑			

Sortieralgorithmen

Wann bezeichnet man einen Sortieralgorithmus für WORST-CASE-OPTIMAL?
Wenn der Algorithmus mit einer beliebigen Anordnung von Elementen mit einem Laufzeitverhalten von O(N*log(N)) auskommt.
Beispiele: Merge-Sort
Welches Laufzeitverhalten kann bei einem nicht optimalen Sortieralgorithmus maximal auftreten?
O(N)2

Wann ist ein Sortieralgorithmus STABIL?
Wenn die relative Ordnung der Elemente mit gleichen Schlüsseln beim Sortieren unverändert bleibt. D.h. Zwei oder mehrere Wert hintereinander bleiben auch nach der Sortierung in der gleichen Reihenfolge.
Wichtig, wenn eine bereits bestehende Ordnung beibehalten werden soll.

Was ist ein IN-SITU-Verfahren?
Verfahren, das während der Ausführung stets den gleichen Speicherplatz benötigen, nennt man In-Situ-Verfahren.
Im Gegensatz dazu wächst der Speicher- und Stackbedarf bei rekursiv arbeitenden Algorithmen mit der Anzahl der zu sortierenden Elemente

B-Bäume

Ein B-Baum ist ein balancierter Baum. Das heißt, die jeweiligen Teilbäume sind ausgeglichen bzw. differieren in der Höhe um max. 1.
Durch die Balancierung ergeben sich stets weitgehend optimale Zugriffszeiten. Diese werden weiter optimiert, in dem die jeweiligen Knoten mehr als zwei Schlüssel (typisch sind 255-1024) zusammengefasst werden. Die Suchzeit innerhalb des Speichers fällt gegenüber den externen Zugriffen nicht ins Gewicht.

Geben Sie ein Beispiel für die Vorteile eines B-Baumes an
Gegeben sind 1.000.000 Knoten. Bei Knotengrößen von 255 Schlüsseln (28-1) ergeben sich $[\log_{256}(1.000.000)]=2,5$ Plattenzugriffe.

Wie wird in einem B-Baum gesucht?
Beginnend bei der Wurzel wird der Wert in einem Knoten gesucht. Man springt evtl. in ein Blatt wobei man in das mit dem nächst größerem Wert verbundene Blatt folgt.

Wie wird in einem B-Baum eingefügt?

Das Einfügen geschieht grundsätzlich in den Blättern.

- Suche des betreffenden Blattes
- Wert einfügen
- Prüfen ob das Blatt übergelaufen ist (enthält mehr Werte als zulässig)

Wenn ja:

- Blatt teilen und mittleren Wert an den Vaterknoten abgeben
- Prüfen ob der Vaterknoten übergelaufen ist

Wenn ja:

- ebenfalls teilen
- den mittleren Wert ebenfalls an den Vaterknoten abgeben
- usw.

evtl. ergibt sich daraus eine neue Wurzel

Wie wird in einem B-Baum gelöscht?

Ist der gesuchte Wert in einem Blatt:

- löschen
- evtl. Unterlauf beseitigen

Ist der gesuchte Wert in einem Knoten:

- man ermittelt den nächst größeren Wert in einem Blatt des Unterbaumes und ersetzt damit den zu löschenden Wert.
- Evtl. Unterlauf im Blatt beseitigen

Unterlauf beseitigen:

- es wandert der kleinste Wert des benachbarten Blattes über den Vaterknoten in das untergelaufene Blatt
- würde in dem Nachbarblatt wieder ein Unterlauf entstehen so muss verschmolzen werden.

Verschmelzung:

- Vaterknoten gibt den kleinsten Wert an die zu verschmelzenden Sohnblätter ab
- Es kann vorkommen, dass nun der Vaterknoten unterläuft

Definieren Sie einen B-Baum

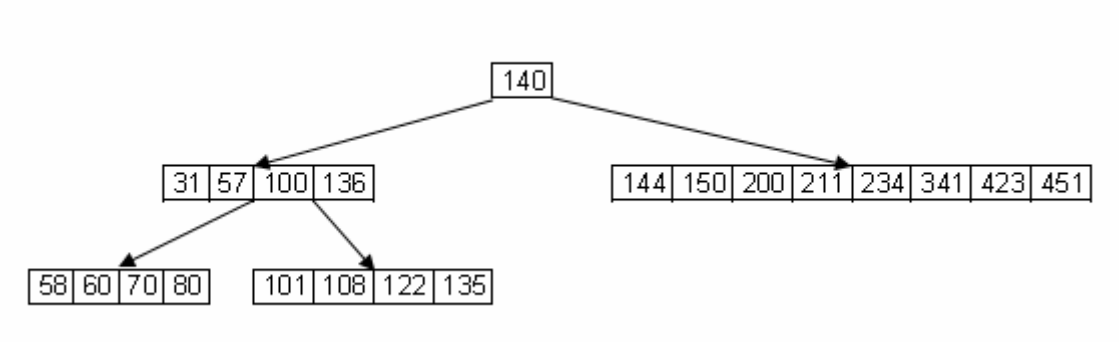
- alle Blätter besitzen das gleiche Niveau
- jeder Knoten (Ausnahme die Wurzel) besitz mindestens m/2 Söhne und höchstens m Söhne
- die Wurzel besitzt mindestens 2 und höchstens m Söhne
- jeder Knoten mit k Söhnen enthält genau k-1 Schlüssel
- Suchbaumeigenschaft wie oben (?)

Welcher Unterschied besteht beim Wachsen von B-Bäumen und anderen Bäumen?

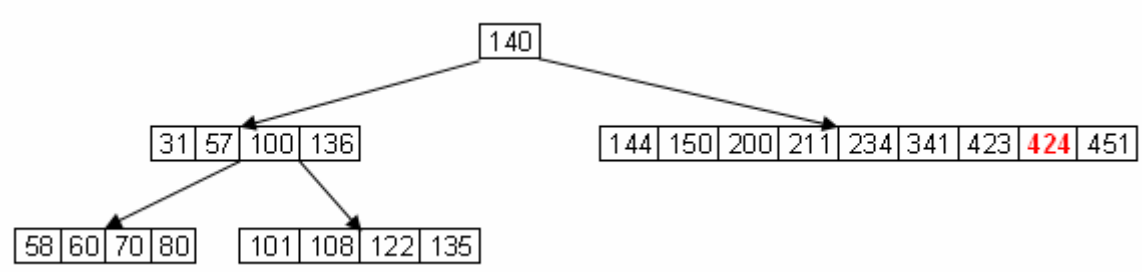
B-Bäume wachsen stets von den Blättern zur Wurzel hin weiter.

Beispiele:

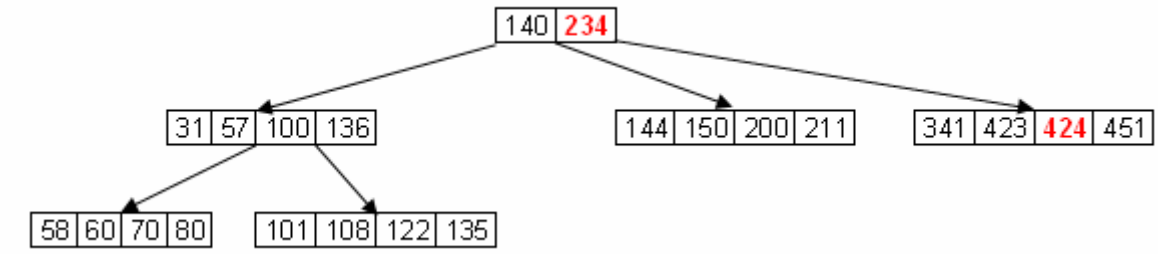
Sehen Sie sich folgenden Ausschnitt eines B-Baumes der Ordnung 4 an:



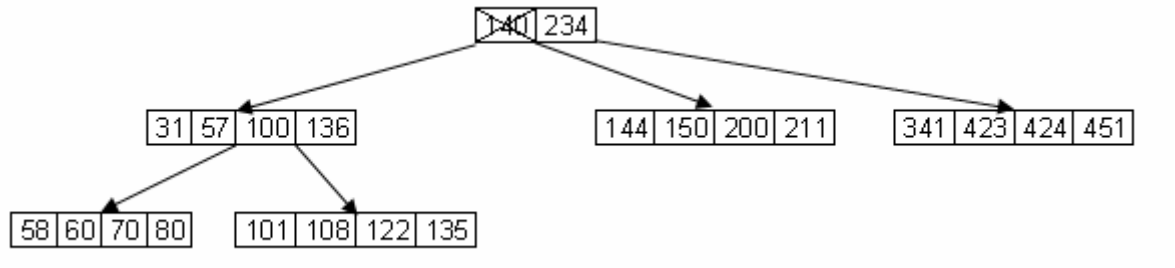
a) Fügen Sie das Element „424“ ein



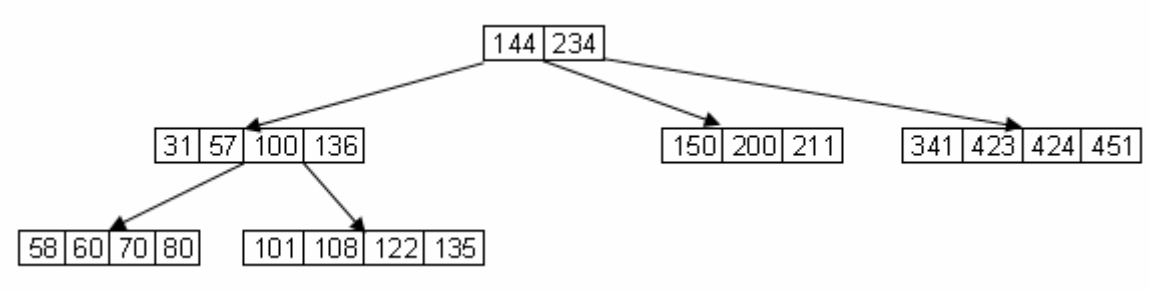
Aufspalten, da mehr als 8 Elemente
Der mittlere Wert „234“ muss in den Vaterknoten



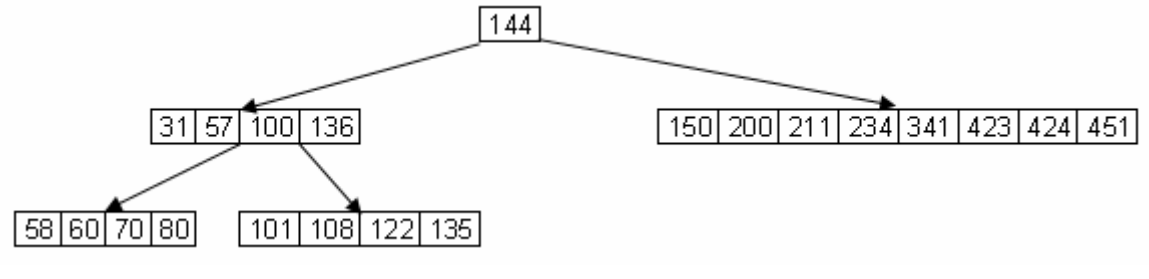
b) Im Baum, der von *a)* verändert wurde, löschen Sie das Wurzelelement „140“.



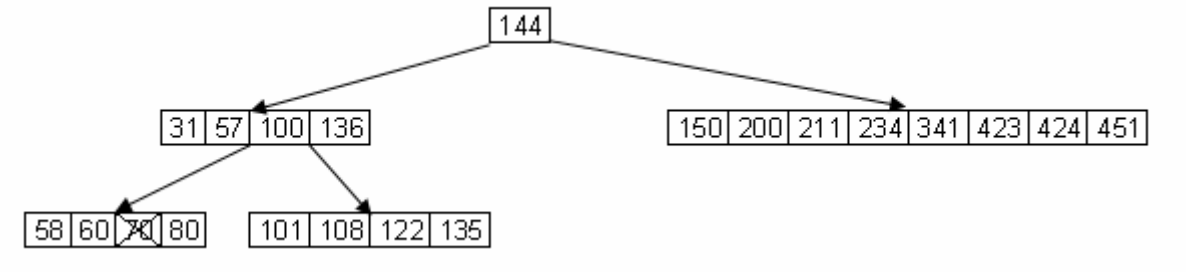
Jetzt muss die „144“ in den Vaterknoten wandern



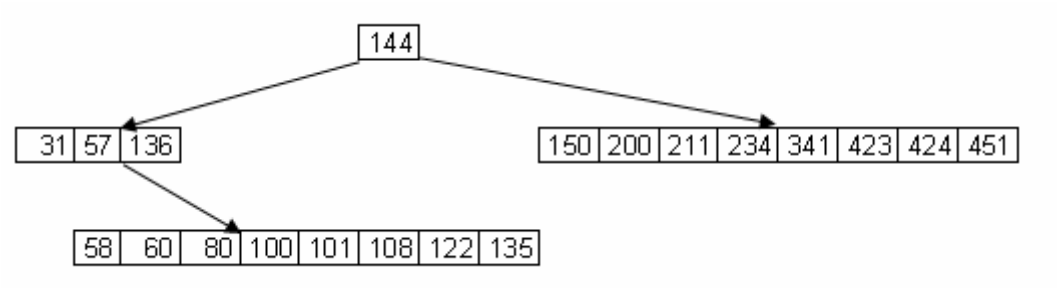
Nun haben wir aber einen Unterlauf (Nur 3 Elemente 150|200|211|). Es müssen ja mindestens 4 und maximal 8 Elemente sein.
Folglich muss verschmolzen werden, indem man die „234“ wieder herunterholt.



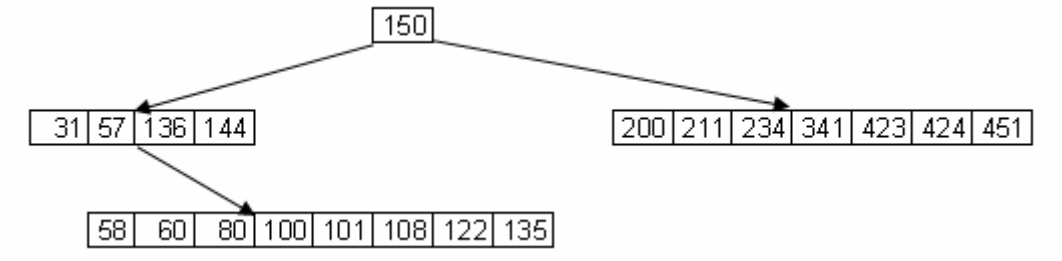
c) Im Baum, der von *b)* verändert wurde, löschen Sie das Element „70“.



Da ein Unterlauf entstanden ist (58|60|80|), wird verschmolzen, in dem man die „100“ nach unten holt.



Da wieder ein Unterlauf entstanden ist (31|57|136|), borgen wir uns die „150“ vom rechten Konten und können somit die 144 runterholen.



Datenbanken

Datenbank-Architektur

Das Ansi-Sparc-Modell beschreibt das grundlegende Modell einer Datenbank-Architektur als interne, konzeptionelle und externe Ebene.

Was ist der Unterschied zwischen interner, konzeptioneller und externer Ebene?

- Die ***interne Ebene*** beschreibt die tatsächliche physische Speicherung.
- Die ***konzeptionelle Ebene*** stellt das zugrunde liegende abstrahierte Datenmodell dar und repräsentiert den Informationsgehalt.
- Die ***externe Ebene*** beschreibt die Sicht des Anwenders, wobei jeder Nutzer, jede Anwendung eine individuelle Sicht auf die Daten hat.

Fremd-, Primär- und Sekundärschlüssel

Felder können Werte enthalten, die in anderen Tabellen Schlüssel bilden. Werden sie zur Referenz auf andere Tabellen verwendet, spricht man von einem **Fremdschlüssel**.

Ein **Primärschlüssel** ist ein beliebig ausgewählter Schlüsselkandidat (Attribut), der zur eindeutigen Identifizierung jeder Zeile (Tupel = Datensatz) benutzt wird.

Die Unterscheidung liegt darin, dass der Primärschlüssel als Referenz in anderen Tabellen dient der **Sekundärschlüssel** hingegen nur als Menge von Attributen, die beim Suchen ein eindeutiges Resultat liefern.

Welche Idee liegt SQL zugrunde?

Das DBMS (Datenbankmanagement Systems) ist vollständig getrennt von der Anwendung, d.h. mit dem standardisierten SQL wird lediglich eine Abfrage definiert, welche ein bestimmtes Ergebnis liefern soll. Wie die zu geschehen hat (incl. möglicher Optimierungen) bleibt Aufgabe des DBMS. Mengenbezogene Denkweise.

Was ist ein ER-Diagramm und welches sind die grundlegenden Elemente?

ER steht für Entity-Relationship und bezeichnet ein relationales Datenbankmodell.
Eine Entität stellt einen Datensatz dar (Raum 007).
Ein Entitätstyp [Räume] besitzt Attribute (qm, Möblierung, DV-Ausstattung, ect.).
Ein Attribut besitzt einen bestimmten Wertebereich (Datum, Zahlen (von x bis y, ect.)
Beziehungstypen (Relationstypen) stellen die Verbindungen zu den Entitätstypen her und werden als Kardinalitäten bezeichnet.

Was ist ein Schlüssel?

Schlüssel können aus zusammengesetzten Feldkombinationen (Attributen) gebildet werden. Ein Schlüssel beschreibt mittels eines Attributes ein eindeutiges Identifikationsmerkmal einer Entität (Datensatz). Ein Schlüssel ist die minimale identifizierende Attributkombination.

Wie werden Primärschlüssel einer Relation (Beziehung) gebildet?

Der Primärschlüssel der Relation wird aus der Zusammenfassung der jeweiligen Primärschlüssel der zugehörigen Entitäten gebildet.
Beispiel:
Entitäten:
Student = @MatrNr
Vorlesung = @Vorlesungsnr
Relation:
Besuch =@<Student>MatrNr+@<Vorlesung>Vorlesungsnr

Kennzeichen Sie kurz das hierarchische Datenmodell

Das hierarchische Datenmodell baut auf einer baumartigen Speicherstruktur auf. Ausgehend von der Wurzel sind die Daten über Suchpfade zu finden. Ist dieser bekannt ergeben sich sehr gute Zugriffsgeschwindigkeiten.
Direkte Beziehungen zwischen den einzelnen Ästen und Blättern sind jedoch nicht abzubilden.
Man muss mit Zeigern durch die Datenbank navigieren, was sehr umständlich und fehlerträchtig ist.

Kennzeichen Sie kurz das relationale Datenmodell

Daten werden grundsätzlich weitgehend redundanzfrei in einzelnen Tabellen gespeichert, wobei die Beziehungen (Relationen) untereinander jederzeit separat und flexibel erstellt bzw. aktualisiert werden können. Die Verknüpfung erfolgt über Primär- und Fremdschlüsselfelder. Durch Indexierung ist der Datenzugriff relativ schnell. Das Datenmodell ist mengenbezogen.
Der größte Vorteil liegt in der Trennung von Abfrage und Datenermittlung. Hierdurch kann über standardisierte Sprachen (SQL) flexibel auf den Datenbestand zugegriffen werden, unabhängig vom verwendeten DBMS.
Vorteile:
Sehr flexibel auch in Bezug auf zukünftige Anforderungen.
Nachteil:
Bei zunehmender Komplexität leidet die Performance.
Hohe Ansprüche an Rechenleistung und Speicherkapazität.

Welche Vorteile haben Netzwerkdatenbanken?

Bei Netzwerkdatenbanken werden die Beziehungen zwischen den Sätzen verschiedener Tabellen mit Pointern hergestellt. Dies muss bereits bei der Anlage beachtet festgelegt werden. Der Zugriff ist sehr schnell.
Einsatz bei statischen Datenbeständen mit häufigen Zugriffen.

Vergleichen Sie hierarchische mit relationalen Datenbanken

Hierarchische DB = sehr schneller Zugriff, aufwendiges Einfügen, keine direkte Verbindung zwischen Datensätzen (Knoten) möglich.
Relationale DB = sehr flexibler Zugriff

Wie wird die Kardinalität n:m im RDBMS umgesetzt?

Es erfolgt stets eine Aufteilung in drei Tabellen.
2 x Entitäten plus 1 x Beziehungen

Wann ist eine Relation in 3. Normalform (NF)?

Eine Relation ist in 3NF, falls sie in 2NF ist und kein Nicht-Schlüsselattribut funktional abhängig von einem andern Nicht-Schlüsselattribut ist.

Beschreiben Sie kurz die Entwicklung von SQL

- ANSI-SQL/86 – erste Veröffentlichung des Standards
- ANSI-SQL/89 – erste Revision
- ANSI-SQL/92 – wesentliche Erweiterung, drei Levels

SQL/92-Entry-Level
SQL/92-Intermediate Level (z. B. incl. div. Joins)
SQL/92-Full-Level (z. B. incl. Client/Server-Anbindung)

Nennen Sie die wichtigsten Erweiterungen von SQL/92

Div. Joins z.B.
Outer-Join
Union-Join
Interselect
Mengenoperationen für Vereinigung, Schnittmenge und Differenz
Unterstützung von Transaktionsverarbeitungen (wichtig für Mehrbenutzerbetrieb zur Aufrechterhaltung der Datenintegrität)

Charakterisieren Sie kurz die SELECT – Anweisung

Die SELECT – Anweisung filtert bestimmte Datensätze aus den angesprochenen Tabellen.

Beispiel:

SELECT
jetzt werden die Feldnamen aufgeführt die in der Ergebnismenge erscheinen sollen. Alternativ kann ein * angegeben werden, was zur Folge hat, dass alle Felder der Tabelle/n angezeigt werden (Achtung! Performance)

FROM
Hier wird angegeben aus welcher Tabelle bzw. welchen Tabellen die unter SELECT definierte Ergebnismenge gefiltert werden soll

Optional:

WHERE
Hier werden eine/mehrere Bedingungen definiert, welche die Ergebnismenge einschränken.

Beispiel:

SELECT *
FROM Personal
Ergebnismenge = alle Daten der Tabelle Personal

SELECT *
FROM Personal
WHERE Wohnort = 'Nürnberg'
Ergebnismenge = Führt eine Selektion über diejenigen Pesonen durch, die in Nürnberg wohnen!

Bedingungsart in der WHERE-Klausel:

Bedingung	Beispiel	Erklärung
Vergleichsoperatoren	preis < 100 name = 'Meier'	Vergleicht den Wert eines Datenfelds mit einem vorgegebenen Wert
Bereichsprüfung	preis **BETWEEN** 10 **AND** 100	Prüft, ob der Wert eines Feldes innerhalb eines bestimmten Bereichs liegt
Elementprüfung	abteilung **IN** ('Einkauf', 'Verkauf')	Prüft, ob der Wert eines Feldes sich in der angegebenen Liste befindet
Mustervergleich	name **LIKE** 'M%'	Überprüft einen Feldinhalt auf Übereinstimmung mit einem angegebenen Muster
Nullwertprüfung	preis **IS NULL**	Prüft einen Feldinhalt auf den Wert NULL (Datenfeld enthält keinen Wert)
Logische Operatoren	preis < 100 **AND** preis > 10 name <> 'Meier' **OR** name <> 'Mayer'	Verknüpft mehrere Bedingungen

Vergleichsoperatoren: <, >, <>, =, >=, <=
Logische Operatoren: AND, OR, NOT

Vergleich für Textwerte mit dem LIKE-Operator:

Platzhalter	Erklärung	Beispiel	Mögliches Ergebnis
%	Platzhalter steht für kein, ein oder mehrere beliebige Zeichen	name **LIKE** 'F%' name **LIKE** '%son' name **LIKE** '%ill%'	Funke, Franz, Fahrmann Benson, Jenson, Morrison Miller, Filler, Ofillson

Charakterisieren Sie kurz die GROUP BY - Anweisung und deren Besonderheiten

Die GROUP BY – Anweisung gruppiert die Ergebnismenge nach einem bestimmten Kriterium. Pro Gruppe wird ein Wert ausgegeben.
Zu beachten:
Das Feld nach dem gruppiert werden soll, muss natürlich in der SELECT - Anweisung mit aufgenommen sein. Access verlangt, dass alle Felder der SELECT - Anweisung auch in das GROUP BY mit aufgenommen werden.

Beispiel:

SELECT Wohnort
FROM Personal
Ergebnismenge: Listet alle Wohnorte der Mitarbeiter auf

SELECT Wohnort
FROM Personal
GROUP BY Wohnort
Ergebnismenge: Listet alle Wohnorte der Mitarbeiter auf, wobei jeder Wohnort nur einmal erscheinen darf

Charakterisieren Sie kurz die HAVING - Anweisung und deren Besonderheiten

Soll eine Bedingung in einer mit GROUP BY gruppierten SELECT – Anweisung eingefügt werden, so wird nicht WHERE verwendet sondern HAVING, wobei HAVING nach der GROUP BY Anweisung steht.
HAVING ohne GROUP BY erzeugt einen Fehler!

Beispiel:

SELECT Lieferantennummer, Land, COUNT(Artikelbezeichnung) AS Artikel gelistet
FROM Artikel
GROUP BY Lieferantennummer
HAVING Land = Frankreich
Ergebnismenge: Auflistung aller französischen Lieferanten mit der Anzahl der gelisteten Artikel, Spaltenüberschrift „Artikel gelistet“

Charakterisieren Sie kurz die ORDER BY - Anweisung und deren Besonderheiten

Mit ORDER BY legt man die Sortierreihenfolge der Ergebnismenge fest.
Mit DESC wird in absteigender Reihenfolge sortiert.
Mit dem Schlüsselwort ASC wird aufsteigend sortiert. Dies ist jedoch standardmäßig der Fall und muss somit nicht angegeben werden.
Beachte: DESC|ASC bezieht sich immer nur auf ein Attribut! Die Anweisung steht am Schluss des SQL-Statements.

Beispiel:

```
SELECT *
FROM Personal
ORDER BY Gehalt
```
Ergebnismenge: Sortiert die Mitarbeiterliste nach dem Gehalt

```
SELECT *
FROM Personal
ORDER BY Gehalt, AbtNr
```
Ergebnismenge: Sortiet die Mitarbeiterliste nach dem Gehalt und danach nach der Abteilungsnummer!

```
SELECT *
FROM Personal
ORDER BY Gehalt DESC
```
Ergebnismenge: Sortiert die Mitarbeiterliste absteigend nach dem Gehalt!

```
SELECT *
FROM Personal
ORDER BY Gehalt DESC, GebDatum
```
Ergebnismenge: Sortiert die Mitarbeiterliste absteigend nach dem Gehalt und danach aufsteigend nach dem Geburtsdatum!

```
SELECT *
FROM Personal
ORDER BY Gehalt, GebDatum DESC
```
Ergebnismenge: Sortiert die Mitarbeiterliste aufsteigend nach dem Gehalt und danach absteigend nach dem Geburtsdatum!

Beschreiben Sie kurz das Wesen des JOIN-Befehles

Für relationale Datenstrukturen ist bezeichnend, dass durch die Normalisierung ursprünglich zusammengehörige Informationen auf verschiedene Tabellen verteilt werden. Der Bezug zueinander wird über Primär- und Fremdschlüsselfelder hergestellt. Die Informationen werden mit dem JOIN-Befehl wieder zusammengeführt.
Der JOIN-Befehl wurde erst in SQL/92 eingeführt.

JOIN-Befehle

Cross-Join (Kartesisches Produkt)	Verbindet jede Zeile der ersten Tabelle mit jeder Zeile der zweiten Tabelle
Natural Join	Verknüpft die beiden Tabellen über die Gleichheit aller Gleichlautenden Spalten-Namen
Inner Join	Verbindet Datensätze aus zwei Tabellen, sobald ein gemeinsames Feld dieselben Werte enthält
Left Outer Join	Mit einem Left Join wird eine sogenannte linke Inklusionsverknüpfung erstellt. Linke Inklusionsverknüpfungen schließen alle Datensätze aus der ersten (linken) Tabelle ein, auch wenn keine entsprechenden Werte für Datensätze in der zweiten Tabelle existiert.
Union Join	Ähnlich dem Full Outer Join werden Datensätze beider Tabellen aufgenommen. Sie werden aber nicht über eine Bedingung verknüpft.
Outer Join	Der Outer Join schließt alle Datensätze aus den verknüpften Tabellen ein, auch wenn keine Werte für den Datensatz existieren.

Cross-Join

Beim Cross-Join wird das kartesische Produkt über zwei Relationen gebildet, d. h. jeder Datensatz der ersten Relation wird mit jedem Datensatz der zweiten Relation kombiniert.

Beispiel:

```
SELECT *
FROM Personal, Abt
```

Ergebnismenge: Stellt das kartesische Produkt der Relationen Personal und Abteilungen dar!

Inner-Join

Beim Inner-Join (= Equi-Join; innerer Verbund) wird das kartesische Produkt über zwei Relationen gebildet, wenn ein oder mehrere gemeinsame Attribute den gleichen Wert haben.
Beachte: In der Relationalen Algebra wird mit „Verbund“ der „innere Verbund“ bezeichnet!

Beispiel:

```
SELECT *
FROM Personal
JOIN Abt ON Personal.AbtNr = Abt.AbtNr
```

oder:

```
SELECT *
FROM Personal
INNER JOIN Abt ON Personal.AbtNr = Abt.AbtNr
```

Ergebnismenge: Stellt den inneren Verbund zwischen den Relationen Personal und Abteilungen dar!

Beispiel:

```
SELECT *
FROM Personal, Bearb, Projekte
WHERE (Personal.PersonNr = Bearb.PersonNr)
AND (Bearb.ProjNr = Projekte.ProjNr)
```

oder:

```
SELECT * FROM Personal
JOIN Bearb ON Personal.PersonNr = Bearb.PersonNr
JOIN Projekte ON Bearb.ProjNr = Projekte.ProjNr
```

Ergebnismenge: Stellen Sie den inneren Verbund zwischen den Relationen Personal und Projekte dar!

Unterschied zwischen LEFT-, RIGHT-, FULL-OUTER-JOIN.

LEFT	von der ersten Relation werden alle Tupel in die Ergebnismenge aufgenommen. Von der zweiten Relation werden nur die dazugehörigen Tupel übernommen. Die Attributwerte der zweiten Relation bleiben leer, wenn kein entsprechendes Tupel vorhanden ist.
RIGHT	von der zweiten Relation werden alle Tupel in die Ergebnismenge aufgenommen. Von der ersten Relation werden nur die dazugehörigen Tupel übernommen. Die Attributwerte der ersten Relation bleiben leer, wenn kein entsprechendes Tupel vorhanden ist.
FULL	eine Kombination aus dem Left-Outer-Join und dem Right-Outer-Join. Alle Datensätze beider Relationen werden in die Ergebnismenge übernommen. Passen die Tupel aus beiden Relationen laut Vergleichsoperator zusammen, so werden diese verbunden.

Methoden zur Abbildung der Integritätsbedingungen

Semantische Integrität
Die semantische Integrität beschäftigt sich z.B. mit der inhaltlichen Korrektheit der Daten, z.B. Steuerklasse darf nicht 7 sein.

Relationale Integrität
Neben den Integritätsregeln, die die reale Welt vorschreibt, muss eine relationale Datenbank auch die Integritätsregeln einhalten, die sich aus dem relationalen Modell ergeben:
das sind Primär- und Fremdschlüsselintegrität (Entity und Referential Integrity).

Operationale Integrität
Operationale Integrität meint die Aufgabe, den Datenbestand im Mehrbenutzerbetrieb bei konkurrierendem Zugriff vor Inkonsistenzen zu schützen. Die Einhaltung der operationalen Integrität wird in einer Datenbank durch Transaktionen gewährleistet. Eine Transaktion ist dabei eine Folge von SQL-Statements (i.d.R. Änderungsoperationen), die logisch zusammengehören und deshalb als atomar anzusehen sind. Entweder werden alle Statements der Transaktion ausgeführt oder keines.

Einbindung von SQL in 3GL-Sprachen

Um SQL als Datenbankprogrammiersprache einzusetzen, fehlen aber Variablenkonzepte und die elementaren Konstrukte der prozeduralen Programmierung, wie Wiederholungen und Bedingungen. Zweckmäßiger ist es, SQL in Programmiersprachen einzubinden.
Ein Problem bei der Verwendung von SQL in Programmiersprachen der 3. Generation sind die Ergebnistabellen der SQL-Mengenoperationen. Die lassen sich nicht unmittelbar weiterverarbeiten. Zwar bietet SQL mit UPDATE, INSERT und DELETE die Möglichkeit, eine Menge von Datensätzen zu verändern, umgehrt kann man aber auf die Ergebnistabellen von SQL-Mengenoperationen mit Befehlen von 3GL-Sprachen nicht zugreifen. Abhilfe schafft das Cursorkonzept von SQL.

Mit dem Cursorkonzept ist die Möglichkeit einer satzweisen Verarbeitung von Tabellen in 3GL-Sprachen geschaffen.

Call-Schnittstelle, Embedded SQL, Dynamic SQL

Der wesentliche Vorteil der Call-Schnittstelle gegenüber Embedded SQL besteht darin, dass alle SQL-Befehle über C-Funktionsaufrufe realisiert werden. Damit funktioniert der Datenbankzugriff über ODBC sofort in allen Programmiersprachen, die C-Funktionsaufrufe unterstützen. Im Gegensatz dazu ist man bei Embedded und Dynamic SQL auf einen Precompiler angewiesen, der für jede Programmiersprache anders ausfällt. Es ist damit zu rechnen, daß das CLI zunehmend Embedded SQL verdrängen wird.

Weitere Datenmodelle

- Objektorientiert
- Objektrelational
- Multidimensional

Data-Warehouse

Ein Data-Warehouse ist eine zentrale Datensammlung, deren Inhalt sich aus Daten von unterschiedlichen Datenquellen zusammensetzt. Die Daten werden von den Datenquellen in das Data-Warehouse kopiert und dort vor allem für die Datenanalyse und zur betriebswirtschaftlichen Entscheidungshilfe in Unternehmen langfristig gespeichert.

Fazit:
Data-Warehouse gibt es nicht als schlüsselfertige Standardlösung zukaufen. Konzeption und Erstellung eines Datenkaufhauses kann sich – abhängig von der Unternehmensgröße und den eingesetzten Systemen – zu einem langwierigen und teuren Prozeß auswachsen.