C

Glossar

Im Glossar werden einige der in diesem Buch eingeführten Begriffe des Gebiets „Datenbanken“ kurz erläutert. Begriffe, die nur lokal in einem Kapitel eine Rolle spielen oder allgemeine Sachverhalte betreffen, wurden nicht aufgenommen.

Wir haben aber am Ende des Glossars einige wichtige Begriffe aus dem *Informatik-Sprech* aufgenommen, die wir im Laufe dieses Buches ohne genauere Erklärung benutzen. Diese Begriffe werden hier etwas ausführlicher erläutert.

C.1 Datenbankbegriffe

Aggregatfunktionen: Funktionen zur Berechnung von aggregierten Attributwerten: Anzahl von Werten, Minimum, Maximum, Summe, Durchschnitt

Attribut: Name der Spalte einer ↑Relation; Eigenschaft gespeicherter Objekte; Feld eines Datensatzes

Client-Server-Architektur: Systemarchitektur, bei der Dienste von einem entfernten System in Anspruch genommen werden

Data Warehouse: System, das unternehmensübergreifend die Daten aus den operativen Einzelsystemen zusammenführt, integriert und für Analysezwecke aufbereitet

Datenbank (DB): strukturierter Datenbestand, auf den ausschließlich über ein DBMS zugegriffen wird

Datenbank-Management-System (DBMS): Gesamtheit der Software-Module, die die Verwaltung einer ↑Datenbank übernehmen

Datenbankmodell: Formalismus zur Beschreibung und Strukturierung von Datenbeständen und somit zur Definition von ↑Datenbankschemata

Datenbankschema: Festlegung der Struktur einer Datenbank

Datenbanksystem: Datenbank + DBMS

Datendefinition: Umsetzung eines Datenbankentwurfs in eine konkrete Datendefinitionssprache (etwa SQL-DDL)

Datenunabhängigkeit: Abkopplung der logischen Datendarstellung von konkreten Implementierungsdetails und Anwendungsanforderungen

Einbettung: Technik zur Nutzung von Anwendungen einer Datenbanksprache (etwa SQL) in einer anderen Programmiersprache; Technik des Einfügens von Code in HTML-Dokumente, wobei dieser Code vor der Auslieferung der Dokumente ausgeführt und durch das Ergebnis ersetzt wird

Embedded SQL: Standard zur Einbettung von SQL

ER-Modell: Entity-Relationship-Modell; Datenbankmodell zur konzeptionellen Modellierung von Datenbeständen

Fremdschlüssel: Attribut(kombination), das bzw. die Werte aus der Menge der Schlüsselwerte einer (anderen) Relation annimmt, um auf deren Tupel logisch zu verweisen

funktionale Abhängigkeit: spezielle Integritätsbedingung: ein Attributwert ist abhängig von anderen Attributwerten; Grundlage der ↑Normalisierung

gespeicherte Prozedur: Routine, die vom Datenbankserver verwaltet und dort auch ausgeführt wird

Gruppierung: Zusammenfassung von Tupeln nach Gruppierungsattributen; in der Regel zusammen mit ↑Aggregatfunktionen für aggregierte Gruppeneigenschaften genutzt

Index: Datenstruktur zur Beschleunigung des Zugriffs auf eine Relation

Integritätsbedingung: logische Bedingung, die von gespeicherten Daten gewährleistet werden muss

Katalog: enthält die Datenbeschreibungen einer ↑Datenbank

Klasse: Beschreibung von Objekten gleichen Typs in ↑Objektdatenbanken; eingeordnet in Klassenhierarchie mit Vererbung

Log-Buch: Datei zur Protokollierung aller Datenbankaktionen zur Ermöglichung des Recovery im Fehlerfall

logischer Entwurf: Optimierung des Datenbankschemas im Zieldatenmodell, etwa um Redundanzen zu vermeiden

natürlicher Verbund: Verbund, bei dem als Verbundkriterium die Wertegleichheit bei übereinstimmenden Attributnamen genommen wird (Operation der ↑Relationenalgebra)

Normalisierung: Optimierung einer Menge von Relationenschemata zur Vermeidung redundanter Speicherung

Nullwert: undefinierter Wert eines Attributs

Objektdatenbanksystem (ODBS): Datenbanksystem, das ein objektorientiertes Datenbankmodell unterstützt

objektorientiertes Datenbankmodell: Datenbankmodell angelehnt an Konzepte objektorientierter Programmiersprachen (Objekte und Klassen)

objektrelationales Datenbankmodell: Erweiterung des Relationenmodells um Konzepte ↑objektorientierter Datenbankmodelle

OLAP: On-Line Analytical Processing: Systeme, die entscheidungsunterstützend im interaktiven Betrieb eingesetzt werden

Optimierer: Komponente des DBMS, die Anfragen in einer deskriptiven Anfragesprache in einen effizienten Anfrageplan umformt

Partitionierung: Aufteilung einer ↑Relation in Teile zum Zwecke der verteilten Speicherung

Projektion: Auswahl von Spalten einer ↑Relation (Operation der ↑Relationenalgebra)

Recovery (Datensicherung): Wiederherstellung von Daten etwa nach Systemfehlern

Relation: mathematisches Konzept zur Speicherung einer Tabelle

Relationenalgebra: Menge von Operatoren, die ↑Relationen manipulieren, und dabei eine Algebra bilden

Relationenschema: Typ einer ↑Relation (enthält Attributanzahl und jeweilige Attributfestlegungen, etwa Attributnamen und Datentyp)

Replikation: mehrfache Speicherung desselben Datenbestandes in einem ↑verteilten Datenbanksystem

Schlüssel: Attributkombination die Tupel in einer ↑Relation eindeutig identifiziert (analog für andere Datenbankmodelle)

Selektion: Auswahl von Tupeln einer Relation anhand einer Selektionsbedingung (Operation der ↑Relationenalgebra)

SFW-Block: Basisstruktur einer SQL-Anfrage: select-from-where

Sicht (View): anwendungsspezifische Darstellung eines Datenbestandes (etwa Umstrukturierung und Auswahl von Daten)

Synchronisation: Überwachung des Mehrbenutzerbetriebs, um jeder Transaktion ein isoliertes Arbeiten zu garantieren

Transaktion: Folge von Datenbankoperationen, die einen konsistenten Datenbankzustand in einen neuen, wiederum konsistenten Zustand überführen und dabei entweder vollständig oder gar nicht ausgeführt wird

Trigger: Regel, die durch eine Datenbankoperation aktiviert werden kann und „feuert“, indem sie weitere Aktionen ausführt

Tuning: Optimierung von Datenbankschema, interner Speicherung und Transaktionsprogrammen zur Erhöhung der Performance

Tupel: Zeile einer ↑Relation

Typ: Strukturbeschreibung eines Objektes in einer Objektdatenbank

Umbenennung: Umbenennung von Spalten einer ↑Relation (Operation der ↑Relationenalgebra)

Verbund (Join): Kombination der Tupel zweier Tabellen, wobei die Auswahl der zu verschmelzenden Tupel durch ein Verbundkriterium erfolgt

Verteiltes Datenbanksystem: Datenbanksystem, bei dem die Datenbank auf mehreren Knotenrechnern verteilt gespeichert wird

View: ↑Sicht

Zugriffskontrolle: Ausschluss unautorisierter Zugriffe auf die gespeicherten Daten

C.2 Informatik-Sprech

Einige wenige Begriffe aus dem Gebiet der Informatik wollen wir hier noch etwas ausführlicher erläutern. Wir verwenden sie in diesem Buch, erläutern sie im Text des Buches aber nur am Rande.

Die drei folgenden Begriffe messen in immer konkreterer Form die Effizienz eines Algorithmus und versuchen damit, Laufzeiten abzuschätzen. Zu einem Algorithmus, etwa der Implementierung eines Relationenalgebra-Operators, kann man die *Komplexität*, den *Aufwand* und die *Kosten* bestimmen.

Komplexität: Mit der *Komplexität* wird ein globales Laufzeitverhalten eines Algorithmus in Abhängigkeit von der Anzahl der zu verarbeitenden Daten bestimmt. Wenn man beispielsweise den Relationenalgebra-Operator *natürlicher Verbund* zwischen den Relationen r_1 und r_2 realisieren möchte, so ist die Anzahl der zu verarbeitenden Daten die Anzahl der Tupel in der Relation r_1 und die Anzahl der Tupel in der Relation r_2, die wir hier vereinfachend beide mit n abschätzen. Der in Unterabschnitt 10.3.4 eingeführte Nested-Loops-Verbund benötigt dann größenordnungsmäßig n^2 Schritte auf den beiden Relationen. Die Komplexität ist also von der *Ordnung* n^2: das notiert man mit der O-Notation $O(n^2)$ (*quadratisch*). Dabei ist es unerheblich, ob in der Realität $4 * n^2 + 8 * n + 15$ Schritte ausgeführt werden müssen: bei diesem Polynom ist nur der Grad, also der höchste Exponent, interessant. In der theoretischen Informatik reduziert man die Angaben häufig sogar nur auf die *Komplexitätsklassen*: da ist es dann nur interessant, ob die Komplexität in einem Polynom mit beliebigen Grad g ausgedrückt werden kann, also $O(n^g)$ (die Komplexitätsklasse ist dann polynomial), oder nur durch eine Exponentialfunktion, bei der die Anzahl der Daten n im Exponenten steht, also etwa $O(2^n)$.

Aufwand: Um den *Aufwand* eines Algorithmus genauer abzuschätzen, werden wir einerseits die Anzahl der zu verarbeitenden Daten näher bestimmen: Im Beispiel mit dem natürlichen Verbund soll die Relation r_1 aus n Tupeln, die Relation r_2 aus m Tupel bestehen. Andererseits können wir dann den Aufwand mit der O-Notation ausdrücken oder auch genauer als konkretes Polynom. Der Aufwand einer Nested-Loops-Implementierung kann dann in einem Polynom mit zwei Unbestimmten n und m ausgedrückt werden, etwa $2*n*m+4*n+3*m+12$, was wieder die Anzahl der durchzuführenden *Schritte* bestimmen soll.

Kosten: Um die *Laufzeit* eines Algorithmus abschätzen zu können, reichen weder die Anzahl der zu verarbeitenden Daten als Parameter noch die Anzahl der durchzuführenden Schritte als Messgröße aus. In Unterabschnitt 10.3.5 haben wir konkrete *Kosten* für die Implementierung

eines Operators bestimmt. Dabei werden als *Kostenparameter* neben der Anzahl der Tupel auch noch die Länge der Tupel, die Anzahl der verschiedenen Attributwerte für jedes Attribut, die Verteilung der Attributwerte und die Größe des Pufferbereichs im Hauptspeicher für die Berechnung der Kosten berücksichtigt. Statt die Anzahl der durchzuführenden Schritte zu messen, werden durch eine *Kostenformel* die Anzahl der *Zugriffe* auf Seiten des Hintergrundspeichers (Festplatte oder SSD, siehe Abschnitt 10.2) abgeschätzt, da diese Zugriffe entscheidend für die Laufzeit des Algorithmus ist.