Integritäts- und Zugriffskontrolle

Inhalt

Semantische Integritätsbedingungen
Trigger
Zugriffskontrolle in SQL

Semantische IBs (1)

- **Unterschied Konsistenz – Integrität**
  - Konsistenz beschreibt die Korrektheit der DB-internen Speicherungsstrukturen, Zugriffspfade und sonstigen Verwaltungsinformation.
  - *Constraints* (Wertebereiche, Check-Klauseln etc.) sind Sprachkonzepte, die eine Überprüfung der Konsistenz durch das DBS gestatten.
  - Integrität beschreibt die Korrektheit der Abbildung der Miniwelt in die in der DB gespeicherten Daten.
    - Die Integrität kann verletzt sein, obwohl die Konsistenz der DB gewahrt bleibt.
    - Ein DBS kann nur die Konsistenz der Daten sichern!
- Trotzdem spricht man in der DB-Welt **von Integritätssicherung** (z. B. Referentielle Integrität, nicht Referentielle Konsistenz).
  - Integritätsbedingungen *(Constraints)* spezifizieren akzeptable DB-Zustände (und nicht aktuelle Zustände der Miniwelt).
  - Änderungen werden nur zurückgewiesen, wenn sie entsprechend der Integritätsbedingungen als falsch erkannt werden.

Semantische IBs (3)

- Klassifikation: Unterscheidung nach
  - Ebenen der Abbildungshierarchie eines DBS (Blöcke, Seiten, Tupel, ...)
  - Reichweite (Attribut, Relation, mehrere Relationen)
  - Zeitpunkt der Überprüfbarkeit (sofort, erst nach mehreren Operationen)
  - Art der Überprüfbarkeit (Zustand, Übergang)
  - Anlass für Überprüfung (Datenänderung, Zeitpunkt)
- Konsistenz der Transaktionsverarbeitung
  - Bei COMMIT müssen alle semantischen Integritätsbedingungen erfüllt sein.
  - Zentrale Spezifikation/Überwachung im DBS: *„system enforced integrity“*

Semantische IBs (4)

- Ebenen der Abbildungshierarchie

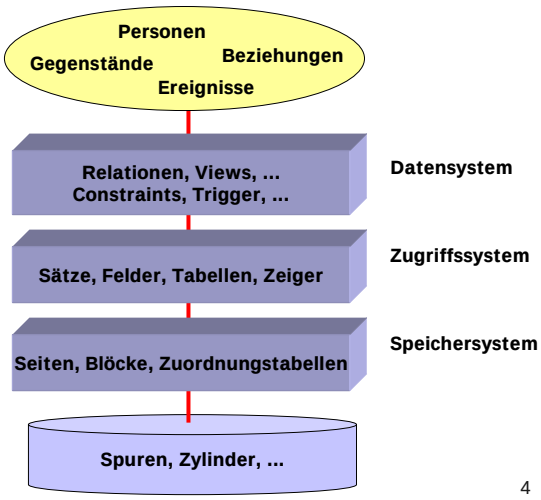
Anwendungsbereich („Miniwelt“): Semantische Integritätsbedingungen

Datenmodell zur formalisierten Abbildung der Miniwelt: Einhaltung der Relationalen Invarianten, benutzerdef. Integritätsbedingungen

Repräsentation in einem linearen Adressraum: Konsistenzbedingungen von Zugriffspf., Tabellen, Zeigern, usw.

Repräsentation in Dateien: Konsistenz-bedingungen und Abbildungsvorschriften bei Blöcken, Seiten, usw.

Physische Repräsentation auf Extern-Speichermedien: Paritäts-, Längen-bedingungen, usw.



4

Semantische IBs (5)

- Physische Konsistenz der DB ist Voraussetzung für logische Konsistenz
  - Gerätekonsistenz
  - Dateikonsistenz
  - Speicherkonsistenz (Aktionskonsistenz; Speicherungsstrukturen, Zugriffspfade, Zeiger sind konsistent)
- Logische Konsistenz (TA-Konsistenz)
  - modellinhärente Bedingungen (z. B. Relationale Invarianten)
  - benutzerdefinierte Bedingungen aus der Miniwelt

Semantische IBs (6)

- Reichweite
  - **Art und Anzahl** der von einer Integritätsbedingung (genauer: des die Bedingung ausdrückenden Prädikats) betroffenen **Objekte**
    - **ein Attribut** (Bsp.: PNR vierstellige Zahl, NAME nur Buchstaben und Leerzeichen)
    - **mehrere Attribute eines Tupels** (Bsp.: GEHALTS-SUMME einer Abteilung muss kleiner sein als JAHRES-ETAT)
    - **mehrere Tupel derselben Relation** (Bsp.: kein GEHALT mehr als 20 % über dem Gehaltsdurchschnitt aller Angestellten derselben Abteilung, PNR ist Primärschlüssel)
    - **mehrere Tupel aus verschiedenen Relationen** (Bsp.: GEHALTS-SUMME einer Abteilung muss gleich der Summe der Attributwerte in GEHALT der zugeordneten Angestellten sein)
  - **geringere Reichweite = einfachere Überprüfung**

Semantische IBs (7)

- Zeitpunkt der Überprüfbarkeit
  - Unverzögerte Bedingungen
    - müssen immer erfüllt sein, unabhängig davon, was in der DB passiert
    - können sofort nach Auftauchen des Objektes überprüft werden (typisch: solche, die sich auf ein Attribut beziehen)
  - Verzögerte Bedingungen
    - z.B. zyklische Fremdschlüsselbedingungen
    - lassen sich nur durch eine Folge von Änderungen erfüllen (typisch: mehrere Tupel, mehrere Relationen)
    - benötigen Transaktionsschutz (als zusammengehörige Änderungssequenzen)

Semantische IBs (8)

- Art der Überprüfbarkeit
  - Zustandsbedingungen
    - betreffen den zu einem bestimmten Zeitpunkt in der DB abgebildeten Objektzustand
  - Übergangsbedingungen
    - Einschränkungen der Art und Richtung von Wertänderungen einzelner oder mehrerer Attribute
    - Beispiele: GEHALT eines Angestellten darf niemals sinken, FAM-STAND darf nicht von „ledig“ nach „geschieden“ oder von „verheiratet“ nach „ledig“ geändert werden
    - sind am Zustand nicht prüfbar - entweder sofort bei Änderung oder später durch Vergleich von altem und neuem Wert (Versionen)

Semantische IBs (9)

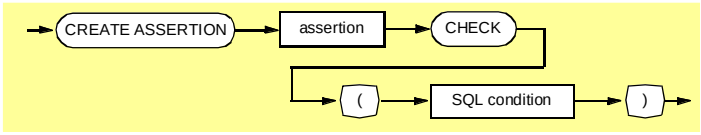
- Anlass für Überprüfung
  - Änderungsvorgang in der DB
    - alle bisherigen Beispiele implizieren Überprüfung innerhalb der TA
  - „Verspätete" Überprüfung: Änderung zunächst nur in (mobiler) Client-DB
  - Ablauf der äußeren Zeit
    - z. B. Daten über produzierte und zugelassene Fahrzeuge – Fahrzeug muss spätestens ein Jahr nach Herstellung angemeldet sein
    - nicht trivial: was ist zu tun bei Verletzung?
      kann an der Realität liegen – abstrakte Konsistenzbedingung erfüllen oder (inkonsistente) Realität getreu abbilden?

Semantische IBs (10)

- Integritätsbedingungen in SQL
  - Bereits eingeführt (siehe Datendefinition)
    - CHECK-Bedingungen bei CREATE DOMAIN, CREATE TABLE, Attributdefinition
    - UNIQUE, PRIMARY KEY, Verbot von Nullwerten
    - Fremdschlüsselbedingungen (FOREIGN-KEY-Klausel)
  - Die vorgenannten Integritätsbedingungen sind an DB-Schemaelemente gebunden
  - **Allgemeine Integritätsbedingungen**
    - beziehen sich typischerweise auf mehrere Relationen
    - lassen sich als eigenständige DB-Objekte definieren
    - erlauben die Verschiebung ihres Überprüfungszeitpunktes
    - **Assertion**

Semantische IBs (11)

- Integritätsbedingungen in SQL (Forts.)
  - Assertion-Anweisung

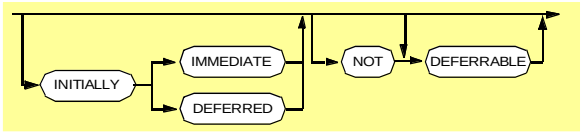


  - Beispiel: Die Relation Abt enthält ein Attribut, in dem (redundant) die Anzahl der Angestellten einer Abteilung geführt wird. Es gilt folgende Zusicherung:

```
CREATE ASSERTION A1
        CHECK (NOT EXISTS
                        (SELECT * FROM Abt A
                         WHERE A.Anzahl_Angest <>
                                (SELECT  COUNT (*) FROM  Pers P
                                         WHERE P.Anr = A.Anr)));
```


Semantische IBs (12)

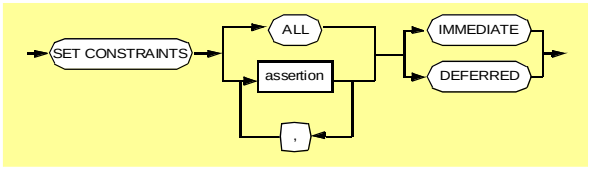
- Integritätsbedingungen in SQL (Forts.)
  - Festlegung des Überprüfungszeitpunktes



  - IMMEDIATE: am Ende der Änderungsoperation (Default)
  - DEFERRED: am Transaktionsende (COMMIT)

Semantische IBs (13)

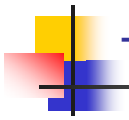
- Integritätsbedingungen in SQL (Forts.)
  - **Überprüfung** kann durch **Constraint-Modus gesteuert** werden



    - Zuordnung gilt für die aktuelle Transaktion
    - Bei benamten Constraints ist eine selektive Steuerung der Überprüfung möglich; so können ‚gezielt' Zeitpunkte vor COMMIT ausgewählt werden.

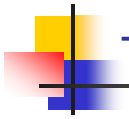
Aktives Verhalten (2)

- Bisher
  - Integritätsbedingungen beschreiben, was innerhalb der DB gültig und zulässig ist.
- Neue Idee
  - Spezifikation und Durchführung von Reaktionen bestimmte Situationen oder Ereignisse in der DB
  - „Zusammenhangsregel" (kausale, logische oder „beliebige" Verknüpfung) statt statischem Prädikat
  - Je mehr Semantik des modellierten Systems explizit repräsentiert ist, umso mehr kann das DBS „aktiv" werden!
- Oft synonyme Nutzung der Begriffe *Produktionsregel, Regel, Aktive Regel, Trigger, Alerter*



Trigger (5)

- Einsatz und Standardisierung
  - Trigger werden schon seit ~1985 in relationalen DBS eingesetzt
  - Ihre Standardisierung wurde jedoch erst in SQL:1999 vorgenommen
- Konzept nach SQL:1999
  - Wann soll ein Trigger ausgelöst werden?
    - Zeitpunkte: BEFORE / AFTER
    - auslösende Operation: INSERT / DELETE / UPDATE
  - Wie spezifiziert man (bei Übergangsbedingungen) Aktionen?
    - Bezug auf verschiedene DB-Zustände erforderlich
    - OLD/NEW erlaubt Referenz von alten/neuen Werten

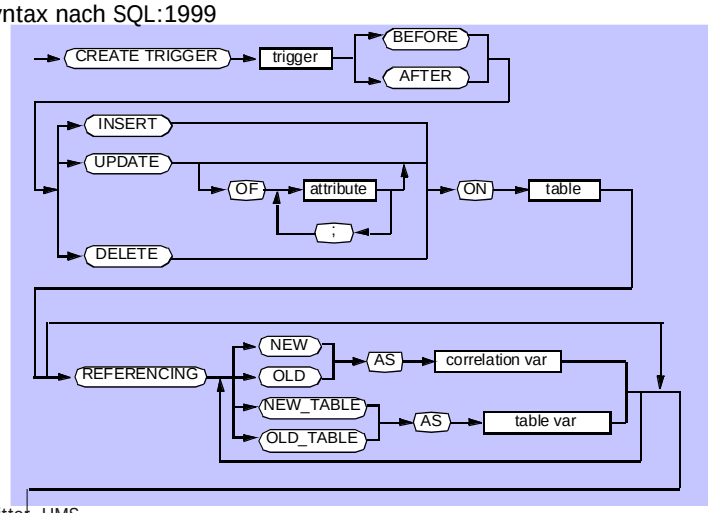


Trigger (6)

- Konzept nach SQL:1999 (Forts.)
  - Ist die Trigger-Ausführung vom DB-Zustand abhängig?
    - WHEN-Bedingung optional
  - Was soll wie verändert werden?
    - pro Tupel oder pro DB-Operation (Trigger-Granulat)
    - mit einer SQL-Anweisung oder mit einer Prozedur aus PSM-Anweisungen (persistent stored module, stored procedure)
  - Existiert das Problem der Terminierung und der Auswertungsreihenfolge?
    - mehrere Trigger-Definitionen pro Relation (Tabelle) sowie
    - mehrere Trigger-Auslösungen pro Ereignis möglich

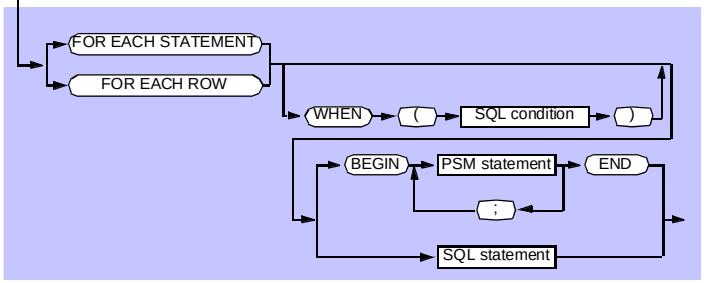
Trigger (7)

- Syntax nach SQL:1999



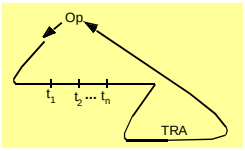
Trigger (8)

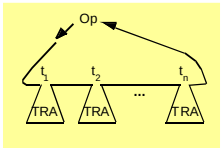
- Syntax nach SQL:1999 (Forts.)



Trigger (9)

- Übergangstabellen und -variablen
  - sie vermerken Einfügungen (bei INSERT), Löschungen (bei DELETE) und die alten und neuen Zustände (bei UPDATE).
  - Übergangstabellen (transition tables) beziehen sich auf mengenorientierte Änderungen
  - Übergangsvariablen (transition variables) beziehen sich auf tupel-weise Änderungen
- Trigger-Granulat
  - FOR EACH STATEMENT: mengenorientiertes Verarbeitungsmodell
  - FOR EACH ROW: tupelorientiertes Verarbeitungsmodell
  - TRA: Trigger-Aktion



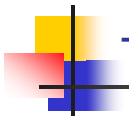


Trigger (10)

- Einsatzbeispiel
  - Gehaltsumme in Abt soll bei Änderungen in Pers, die „Gehälter“ betreffen, automatisch aktualisiert werden
  - es sind Trigger für INSERT/DELETE/UPDATE erforderlich; sie werden bei Auftreten der spezifizierten Änderungsoperationen sofort ausgeführt

Abt	Anr	Aname	Ort	Geh_Summe
	K51	PLANUNG	KAISERSLAUTERN	43500
	K53	EINKAUF	FRANKFURT	45200
	K55	VERTRIEB	FRANKFURT	80000

Pers	Pnr	Name	Alter	Gehalt	Anr	Mnr
	406	COY	47	50 000	K55	123
	123	MÜLLER	32	43 500	K51	-
	829	SCHMID	36	45 200	K53	777
	574	ABEL	28	30 000	K55	123

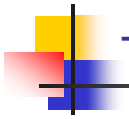


Trigger (11)

- Einsatzbeispiel (Forts.)

```
CREATE TRIGGER T1
AFTER INSERT ON Pers                        (* Ereignis *)
REFERENCING NEW AS NP
FOR EACH ROW
        UPDATE Abt A                        (* Aktion *)
        SET A.Geh_Summe = A.Geh_Summe + NP.Gehalt
        WHERE A.Anr = NP.Anr;
```

```
CREATE TRIGGER T2
AFTER UPDATE OF Gehalt ON Pers              (* Ereignis *)
REFERENCING OLD AS OP NEW AS NP
FOR EACH ROW
        UPDATE Abt A                        (* Aktion *)
        SET A.Geh_Summe = A.Geh_Summe + (NP.Gehalt - OP.Gehalt)
        WHERE A.Anr = NP.Anr;
```

Trigger (12)

- Einsatzbeispiel (Forts.)

```
CREATE TRIGGER T3
AFTER  UPDATE OF Gehalt ON Pers             (* Ereignis *)
REFERENCING OLD_TABLE AS OT NEW_TABLE AS NT
FOR EACH STATEMENT
        UPDATE Abt A                         (* Aktion *)
        SET A.Geh_Summe = A.Geh_Summe +
                (SELECT SUM (Gehalt) FROM NT WHERE Anr = A.Anr) -
                (SELECT SUM (Gehalt) FROM OT WHERE Anr = A.Anr)
        WHERE A.Anr IN (SELECT Anr FROM NT);
```

Zugriffskontrolle - Allgemeines (1)

- Zugriffskontrolle: technische Maßnahme des Datenschutzes
- Kernfrage: Wie kann ich erreichen, dass Benutzer mit unterschiedlichen Rechten gemeinsam auf Daten zugreifen können?
  - Frage nach der Zugriffskontrolle (bei Daten)
- Zugriffskontrolle (Autorisierung)
  - Vergabe von Zugriffsrechten (Lesen, Schreiben, . . .) auf DB-Objekten, Programmen usw.
  - Ziele
    - Verhinderung von zufälligen oder böswilligen Änderungen
    - möglichst weitgehende Isolation von Programmfehlern
    - Verhinderung von unberechtigtem Lesen/Kopieren

Zugriffskontrolle - Allgemeines (2)

- Autorisierungsmodell
  - Explizite Autorisierung:
    - Dieses Modell wird im Englischen als Discretionary Access Control (DAC) bezeichnet. Wegen seiner Einfachheit ist DAC weit verbreitet („discretionary“ bedeutet in etwa „nach dem Ermessen des Subjekts“).
    - Der Zugriff auf ein Objekt o kann nur erfolgen, wenn für den Benutzer (Subjekt s) ein Zugriffsrecht (Privileg p) vorliegt
    - Autorisierungsregel (o, s, p)
  - Schutzinformation als Zugriffsmatrix
    - *Subjekte:* Benutzer, Programme, Terminals
    - *Objekte:* Programme (Anwendungs-, Dienstprogramme), DB-Objekte (Relationen, Sichten, Attribute)
    - *Zugriffsrechte:* Lesen, Ändern, Ausführen, Erzeugen, Weitergabe von Zugriffsrechten usw., ggf. abhängig von Terminal, Uhrzeit usw.

Zugriffskontrolle - Allgemeines (5)

- Autorisierungsmodell (Forts.)
  - Autorisierung
    - zentrale Vergabe der Zugriffsrechte (DBA)
    - dezentrale Vergabe der Zugriffsrechte durch Eigentümer der Objekte
  - Objektgranulat
    - wertunabhängige oder
    - wertabhängige Objektfestlegung (Sichtkonzept)
  - Wirksamkeit der Zugriffskontrolle beruht auf drei Annahmen:
    - fehlerfreie Benutzer-Identifikation/-Authentisierung
    - erfolgreiche Abwehr von (unerwarteten) Eindringlingen (vor allem strikte Isolation der Benutzer- und DBS-Prozesse sowie Übermittlungskontrolle)
    - Schutzinformation ist hochgradig geschützt!

Zugriffskontrolle in SQL (1)

- Sicht-Konzept erlaubt wertabhängigen Zugriffsschutz
  - Untermengenbildung, Verknüpfung von Relationen, Verwendung von Aggregat-Funktionen
  - Umsetzung durch Anfragemodifikation möglich
- Vergabe von Rechten

```
GRANT   {privileges-commalist | ALL PRIVILEGES}
        ON accessible-object TO grantee-commalist
        [WITH GRANT OPTION]
```

  - Objekte (accessible-object)
    - Relationen bzw. Sichten
    - aber auch: Domänen, Datentypen, Routinen usw.

Zugriffskontrolle in SQL (2)

- Vergabe von Rechten (Forts.)
  - Zugriffsrechte (privileges)
    - SELECT, INSERT, UPDATE, DELETE, REFERENCES, USAGE, EXECUTE, . . .
    - Attributeinschränkung bei INSERT, UPDATE und REFERENCES möglich
    - Erzeugung einer „abhängigen“ Relation erfordert REFERENCES-Recht auf von Fremdschlüsseln referenzierten Relationen
    - USAGE erlaubt Nutzung spezieller Wertebereiche (character sets)
    - dynamische Weitergabe von Zugriffsrechten: WITH GRANT OPTION (GO: dezentrale Autorisierung)
  - Empfänger (grantee)
    - Liste von Benutzern bzw. PUBLIC
    - Liste von Rollennamen

Zugriffskontrolle in SQL (3)

- Vergabe von Rechten (Forts.)
  - Beispiele
    - **GRANT SELECT ON** Abt **TO PUBLIC**
    - **GRANT INSERT, DELETE ON** Abt **TO** Mueller, Weber **WITH GRANT OPTION**
    - **GRANT UPDATE** (Gehalt) **ON** Pers **TO** Schulz
    - **GRANT REFERENCES** (Pronr) **ON** Projekt **TO PUBLIC**
- Rücknahme von Zugriffsrechten

  REVOKE [GRANT OPTION FOR] privileges-commalist ON accessible-object FROM grantee-commalist {RESTRICT | CASCADE}

  - Beispiel: **REVOKE SELECT ON** Abt **FROM** Weber **CASCADE**

Zusammenfassung (1)

- Semantische Integritätskontrolle
  - Relationale Invarianten, referentielle Integrität und Aktionen
  - Benutzerdefinierte Integritätsbedingungen (*assertions*)
    - zentrale Spezifikation/Überwachung im DBS wird immer wichtiger
- Aktives DB-Verhalten zur
  - Integritätssicherung
  - Wartung abgeleiteter Daten
  - Durchführung allgemeiner Aufgaben (Regeln, Alerter, Trigger)
- Triggerkonzept in SQL99 standardisiert

Zusammenfassung (2)

- Verallgemeinertes Konzept: ECA-Regeln (nicht behandelt)
  - Event: Welche Events werden unterstützt?
  - Condition: Wie komplex sind Conditions?
  - Action: Wie komplex sind Actions?
- Zugriffskontrolle in DBS
  - wertabhängige Festlegung der Objekte (Sichtkonzept)
  - Vielfalt an Rechten erwünscht
  - zentrale vs. dezentrale Rechtevergabe
  - verschiedene Entzugssemantiken bei dezentraler Rechtevergabe
  - Rollenkonzept: vereinfachte Verwaltung komplexer Mengen von Zugriffsrechten