Mobile und Verteilte Datenbank Systeme - Zusammenfassung

Egemen Kaba

Inhaltsverzeichnis

1 Kapitel 0 - Introduction **3**

2 Kapitel 1 - Trigger **4**
2.1 Zweck . . . 4
2.2 Konzepte . . . 4
2.3 Struktur eines Triggers . . . 4
2.4 Beispiel . . . 6
2.5 Databaselinks . . . 6

3 Kapitel 2 - Distributed Design I **7**
3.1 Arten der Fragmentierung . . . 7
3.1.1 PHF . . . 8
3.2 Korrektheit der Fragmentierung . . . 9

4 Kapitel 3 - Distributed Design II **9**

5 Kapitel 4 - Distributed Query Processing **9**

6 Kapitel 5 - Distributed Transactions I **9**

7 Kapitel 6 - Distributed Transactions II **9**

8 Kapitel 7 - Replication I **9**

9 Kapitel 8 - Replication II **9**

10 Kapitel 9 - NoSQL **9**

11 Kapitel 10 - Cassandra **9**

12 Kapitel 11 - MapReduce **9**

13 Kapitel 12 - mongoDB **9**

14 Kapitel 13 - Neo4j **9**

15 Kapitel 14 - Semantic Web **9**

1 Kapitel 0 - Introduction

Verteilte Datenbank (DDB)	Eine verteilte Datenbank ist eine Sammlung mehrerer, untereinander logisch zusammengehöriger Datenbanken, die über ein Computernetzwerk verteilt sind.
Verteiltes Datenbankverwaltungssystem (D-DBMS)	Ein verteiltes Datenbankverwaltungssystem ist die Software, die die verteilte Datenbank verwaltet und gegenüber den Nutzern einen transparenten Zugang erbringt.
Verteiltes Datenbank System (DDBS)	DDBS = DBS + D-DBMS
Nebenläufigkeit	• Synchronisation konkurrierender Transaktionen • Konsistenz und Isolation • Deadlock Erkennung
Zuverlässigkeit	• Robustheit gegenüber Fehler • Atomarität und Dauerhaftigkeit
Architekturen	• Shared Memory Architecture • Shared Disk Architecture • Shared Nothing Architecture
Mobile Datenbank Systeme	Verteilte Datenbank System mit zusätzlichen Eigenschaften und Einschränkungen • beschränkte Ressource • häufig nicht verbunden • verlangt andere Transaktions Modelle • verlangt andere Replikationsstrategien • Ortsabhängigkeit

Date's 12 Regeln

- Lokale Autonomie
- Unabhängigkeit von zentralen Systemfunktionen
- Hohe Verfügbarkeit
- Ortstransparenz
- Fragmentierungstransparenz

- Replikationstransparenz
- Verteilte Anfragebearbeitung
- Verteilte Transaktionsverarbeitung
- Hardware Unabhängigkeit
- Betriebssystem Unabhängigkeit
- Netzwerkunabhängigkeit
- Datenbanksystem Unabhängigkeit

2 Kapitel 1 - Trigger

2.1 Zweck

- Realisieren aktive Datenbanksysteme
- Berechnung abgeleiteter Attribute
- Überprüfen komplexer Integritätsbedingungen
- Implementierung von Geschäftsregeln
- Protokollierung, Statistiken
- Überprüfen von Integritätsbedingungen in verteilten Datenbanken
- Synchronisation von Replikaten

2.2 Konzepte

ECA Prinzip	Event: Ereignis tritt ein Condition : Bedingung ist erfüllt Action: Aktion wird ausgeführt
Ereignis	DML: UPDATE, DELETE, INSERT DDL: CREATE, ALTER, DROP, ... Datenbank: SERVERERROR, LOGON, LOGOFF, STARTUP, SHUTDOWN (DML-Trigger können auf Tabellen oder Views definiert werden)
Timing	BEFORE, AFTER, INSTEAD OF Trigger Der INSTEAD OF Trigger ersetzt den triggernden Befehl und wird nur bei Views eingesetzt.
Granulat	STATEMENT, ROW Trigger

2.3 Struktur eines Triggers

Syntax

```
CREATE [OR REPLACE] TRIGGER tname
{BEFORE|AFTER} events
[WHEN(condition)]
pl/sql_block
```

events

```
{DELETE|INSERT|UPDATE
  [OF column [ , column ]...] }
[OR {DELETE|INSERT|UPDATE
  [OF column [ , column ]...]}]...
ON table [FOR EACH ROW]
```

Prinzip

```
DECLARE
  Deklarationsteil
BEGIN
  Programmteil
EXCEPTION
  Ausnahmebehandlung
END;
/
```

Bildschirmausgabe

```
dbms_output.put_line (item IN VARCHAR2);
dbms_output.put_line (item IN NUMBER);
dbms_output.put_line (item IN DATE);
dbms_output.put(item IN VARCHAR2);
dbms_output.put(item IN NUMBER);
dbms_output.put(item IN DATE);
dbms_output.new_line;
-- Ausführung
EXECUTE dbms_output.put_line('Hello world');
-- als Block
BEGIN
  dbms_output.put_line('Hello world');
END;
```

Datentypen

- SQL: VARCHAR2, DATE, NUMBER, ...
- PL/SQL: BOOLEAN, PLS_INTEGER, ...
- Strukturierte Datentypen: TABLE, VARRAY, RECORD
- Datentypen für Spalten und Zeilen aus Tabellen: % ROWTYPE, %TYPE

Zuweisung

```
-- Syntax
variable := expression;
-- Beispiel im Deklarationsteil
name VARCHAR2(30) := 'Kaba';
```

if then else elsif

```
IF condition THEN ... END IF;
IF condition THEN ... ELSIF condition THEN ... ELSE ... END IF;
IF condition THEN ... ELSE ... END IF;
```

if then else elsif


```
IF condition THEN ... END IF;
IF condition THEN ... ELSIF condition THEN ... ELSE ... END IF;
IF condition THEN ... ELSE ... END IF;
```


schleifen

```
WHILE condition LOOP ... END LOOP;
FOR counter IN lower_bound..higher_bound LOOP ... END LOOP;
```

2.4 Beispiel

Trigger

```
CREATE OR REPLACE TRIGGER regdatum_test
  BEFORE INSERT ON registrierungen
  FOR EACH ROW

  DECLARE
    msg VARCHAR2(30) := 'Datum falsch';
  BEGIN
    IF :new.datum > SYSDATE THEN
      RAISE_APPLICATION_ERROR(-20005, msg);
    END IF;
END;
```

2.5 Databaselinks

Datenbanklinks werden benötigt, um Orts- und Namenstransparenz für Tabellen zu erreichen.

Databaselinks

```
-- View
CREATE OR REPLACE VIEW filme AS
SELECT *
FROM filme@ananke.hades.fhnw.ch;
-- Synonyme
CREATE SYNONYM film FOR filme@ananke.hades.fhnw.ch;
```

3 Kapitel 2 - Distributed Design I

Ausgangslage: Anwendungen auf verschiedenen Knoten des Netzwerks greifen auf eine (relationale) Datenbank zu. Nun greifen nicht alle Knoten gleich häufig auf den selben Datensatz zu. Es gilt nun herauszufinden, welche Anwendungen (Queries) auf welchen Knoten welche Daten mit welcher Häufigkeit benötigen. Das Resultat ist eine Menge von Fragmenten, die den verschiedenen Knoten zugeteilt werden.

3.1 Arten der Fragmentierung

- Horizontale Fragmentierung (HF) (Abbildung 1)
  - Primäre horizontale Fragmentierung (PHF)
  - Abgeleitete horizontale Fragmentierung (DHF)
- Vertikale Fragmentierung (VF) (Abbildung 2)
- Gemischte Fragmentierung (MF)

BIKES

BNr	BName	Preis	Typ	Bestand
B5	MCD03	4490.00	Road	2
B4	Siena	2390.00	Mountain	4
B2	City Cross	2190.00	Trekking	3
B3	Valiant	1090.00	Trekking	7
B1	Luxor	980.00	City	10
B6	Atlanta	890.00	Trekking	8
B7	Striker	890.00	Mountain	7

```
SELECT *
FROM bikes
WHERE preis < 2000
```

BIKES1

BNr	BName	Preis	Typ	Bestand
B3	Valiant	1090.00	Trekking	7
B1	Luxor	980.00	City	10
B6	Atlanta	890.00	Trekking	8
B7	Striker	890.00	Mountain	7

```
SELECT *
FROM bikes
WHERE preis >= 2000
```

BIKES2

BNr	BName	Preis	Typ	Bestand
B5	MCD03	4490.00	Road	2
B4	Siena	2390.00	Mountain	4
B2	City Cross	2190.00	Trekking	3

Abbildung 1: Horizontale Fragmentierung

BIKES

BNr	BName	Preis	Typ	Bestand
B5	MCD03	4490.00	Road	2
B4	Siena	2390.00	Mountain	4
B2	City Cross	2190.00	Trekking	3
B3	Valiant	1090.00	Trekking	7
B1	Luxor	980.00	City	10
B6	Atlanta	890.00	Trekking	8
B7	Striker	890.00	Mountain	7

```
SELECT bnr, bname, preis
FROM bikes
```

BIKES1

BNr	BName	Preis
B5	MCD03	4490.00
B4	Siena	2390.00
B2	City Cross	2190.00
B3	Valiant	1090.00
B1	Luxor	980.00
B6	Atlanta	890.00
B7	Striker	890.00

```
SELECT bnr, typ, bestand
FROM bikes
```

BIKES2

BNr	Typ	Bestand
B5	Road	2
B4	Mountain	4
B2	Trekking	3
B3	Trekking	7
B1	City	10
B6	Trekking	8
B7	Mountain	7

Abbildung 2: Horizontale Fragmentierung

3.1.1 PHF

Predicates

Simple predicates	Vergleich eines Attributs mit einem Wert (WHERE-Klausel)	p_1: Typ = 'Road' p_2: Typ = 'Trekking' p_3: Typ = City p_4: Typ = 'Mountain' p_5: Preis $\leqslant$ 2000 p_6: Preis $>$ 2000
Minterm predicates	Verknüpfung von Simple predicates mit AND und NOT	m_1: Typ = 'Road' AND Preis $\leqslant$ 2000 m_2: NOT(Typ = 'Road') AND Preis $\leqslant$ 2000 m_3: Typ = 'Road' AND NOT(Preis $\leqslant$ 2000) m_4: NOT(Typ = 'Road') AND NOT(Preis $\leqslant$ 2000) ...

3.2 Korrektheit der Fragmentierung

vollständig	Wenn R zerlegt wird in R1, R2, ..., Rn, dann muss jedes Datenelement aus R in einem Ri enthalten sein.
rekonstruierbar	Wenn R zerlegt wird in R1, R2, ..., Rn, dann muss es relationale Operatoren geben, so dass R wiederhergestellt werden kann.
disjunkt	Wenn R horizontal zerlegt wird in R1, R2, ..., Rn, dann müssen die Fragemente paarweise disjunkt sein. Wenn R vertikal zerlegt wird in R1, R2, ..., Rn, dann müssen die Fragmente bezogen auf die nichtprimen Attribute paarweise disjunkt sein.

4 Kapitel 3 - Distributed Design II

5 Kapitel 4 - Distributed Query Processing

6 Kapitel 5 - Distributed Transactions I

7 Kapitel 6 - Distributed Transactions II

8 Kapitel 7 - Replication I

9 Kapitel 8 - Replication II

10 Kapitel 9 - NoSQL

11 Kapitel 10 - Cassandra

12 Kapitel 11 - MapReduce

13 Kapitel 12 - mongoDB

14 Kapitel 13 - Neo4j

15 Kapitel 14 - Semantic Web