

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
4 Kapitel 3 - Distributed Design II	7
5 Kapitel 4 - Distributed Query Processing	7
6 Kapitel 5 - Distributed Transactions I	7
7 Kapitel 6 - Distributed Transactions II	7
8 Kapitel 7 - Replication I	7
9 Kapitel 8 - Replication II	7
10 Kapitel 9 - NoSQL	7
11 Kapitel 10 - Cassandra	7
12 Kapitel 11 - MapReduce	7
13 Kapitel 12 - mongoDB	7
14 Kapitel 13 - Neo4j	7
15 Kapitel 14 - Semantic Web	7

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	<ul style="list-style-type: none"> • Synchronisation konkurrierender Transaktionen • Konsistenz und Isolation • Deadlock Erkennung
Zuverlässigkeit	<ul style="list-style-type: none"> • Robustheit gegenüber Fehler • Atomarität und Dauerhaftigkeit
Architekturen	<ul style="list-style-type: none"> • Shared Memory Architecture • Shared Disk Architecture • Shared Nothing Architecture
Mobile Datenbank Systeme	<p>Verteilte Datenbank System mit zusätzlichen Eigenschaften und Einschränkungen</p> <ul style="list-style-type: none"> • 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

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

events

```
1 {DELETE|INSERT|UPDATE
2   [OF column [ , column ]... ] }
3 [OR {DELETE|INSERT|UPDATE
4   [OF column [ , column ]... ]}]...
5 ON table [FOR EACH ROW]
```

Prinzip

```
1 DECLARE
2   Deklarationsteil
3 BEGIN
4   Programmteil
5 EXCEPTION
6   Ausnahmebehandlung
7 END;
8 /
```

BildschirmAusgabe

```
1 dbms_output.put_line (item IN VARCHAR2);
2 dbms_output.put_line (item IN NUMBER);
3 dbms_output.put_line (item IN DATE);
4 dbms_output.put(item IN VARCHAR2);
5 dbms_output.put(item IN NUMBER);
6 dbms_output.put(item IN DATE);
7 dbms_output.new_line;
8 -- Ausführung
9 EXECUTE dbms_output.put_line('Hello world');
10 -- als Block
11 BEGIN
12   dbms_output.put_line('Hello world');
13 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

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

if then else elsif

```
1 IF condition THEN ... END IF;
2 IF condition THEN ... ELSIF condition THEN ... ELSE ... END IF;
3 IF condition THEN ... ELSE ... END IF;
```

if then else elsif

```
1 IF condition THEN ... END IF;
2 IF condition THEN ... ELSIF condition THEN ... ELSE ... END IF;
3 IF condition THEN ... ELSE ... END IF;
```

schleifen

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

2.4 Beispiel

Trigger

```
1 CREATE OR REPLACE TRIGGER regdatum_test
2   BEFORE INSERT ON registrierungen
3   FOR EACH ROW
4
5   DECLARE
6     msg VARCHAR2(30) := 'Datum falsch';
7   BEGIN
8     IF :new.datum > SYSDATE THEN
9       RAISE_APPLICATION_ERROR(-20005, msg);
10    END IF;
11 END;
```

2.5 Databaselinks

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

Databaselinks

```
1 -- View
2 CREATE OR REPLACE VIEW filme AS
3 SELECT *
4 FROM filme@ananke.hades.fhnw.ch;
5 -- Synonyme
6 CREATE SYNONYM film FOR filme@ananke.hades.fhnw.ch;
```

- 3 Kapitel 2 - Distributed Design I**
- 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**