MySQL-Joins

Colletta Hagert

Überblick

- Einführung
- Arten von Joins
- Beispiel

Einführung

- Bei relationalen Datenbanken werden die Daten in der Regel auf mehrere, logisch zusammengehörige Tabellen verteilt.

- Eine wichtige Rolle spielen dabei Primär- und Fremdschlüssel, die die Verbindung von Datensätzen in mehreren Tabellen vereinfachen.

- Eine Tabellenreferenzierung heißt auch Join-Ausdruck.

Arten von Joins

1. **Verknüpfung von Tabellen über Mengen-Operationen**
   - Vereinigungsmenge (UNION)
   - Schnittmenge (INTERSECT)
   - Differenzmenge (MINUS)

Voraussetzung für die Ausführung einer Mengenoperation ist, dass beide Tabellen die gleiche Struktur besitzen, Feldnamen und Wertebereiche müssen übereinstimmen.

2. Verbund von Tabellen:

Der Verbund mehrerer Tabellen wird als Join bezeichnet. Über Joins werden die Datensätze aus zwei oder mehreren Tabellen kombiniert.

- **Cross-Join**:

Das kartesische Produkt beider Tabellen wird gebildet.

=> Jeder Datensatz der einen Tabelle wird mit jedem Datensatz der anderen Tabelle kombini.ert

SELECT datenfelder

FROM tabelle1

CROSS JOIN tabelle2

[**CROSS JOIN** tabelle3 ...]

- **Theta-Join**:

Bestimmte Datensätze werden aus dem kartesischen Produkt zweier Tabellen durch eine Bedingung ausgewählt.

In dieser Bedingung wird eine Spalte aus der einen und eine Spalte aus der anderen Tabelle über eine logische Operation verglichen.

SELECT datenfelder

FROM tabelle1

CROSS JOIN tabelle2

ON tabelle1.datenfeld Op tabelle2.datenfeld

Op: <, >, >=, <=, <>

- **Equi-Join (Inner Join)**

Wird auf Gleichheit geprüft (=), so entsteht eine spezielle Form des Theta-Joins, den Equi-Join

SELECT datenfelder

FROM tabelle1

INNER JOIN tabelle2 **join_bedingung**

[WHERE ..]

join_bedingung:

ON Bedingung | **USING** (Spaltenliste)

- **Equi-Join (Inner Join)**

Variante 1

SELECT datenfelder

FROM tabelle1

INNER JOIN tabelle2 **join_bedingung**

INNER JOIN tabelle3 **join_bedingung**

Variante 2

SELECT datenfelder

FROM tabelle1, tabelle2, ..

WHERE tabelle1.datenfeld = tabelle2.datenfeld

[AND ..]

Beispiel: Equi-Join

SELECT datenfelder

FROM tabelle1

INNER JOIN tabelle2 **join_bedingung**

INNER JOIN tabelle3 **join_bedingung**

- **Natural-Join**

- Arbeitet wie der Inner-Join – mit dem Unterschied, dass in der Ergebnistabelle keine identische Spalten enthalten sind.
- Durch das Hinzufügen einer Projektion wird aus dem Inner-Join ein Natural-Join. Durch Hinzufügen des Schlüsselwortes DISTINCT werden doppelte Datensätze entfernt.

- **Left-Outer-Join** (Linke Inklusionsverknüpfung)

- **Right-Outer-Join** (Rechte Inklusionsverknüpfung)

- **Full-Join (Full-Outer-Join**: Kombination aus dem Left-Outer-Join und dem Right-Outer-Join)

Syntax

SELECT datenfelder

FROM tabelle1

{LEFT|RIGHT|FULL} [OUTER] JOIN tabelle2

join_bedingung

Beispiel: LEFT JOIN

SELECT datenfelder

FROM tabelle1

LEFT JOIN tabelle2

join_bedingung

- **Semi-Join**: (Natural-Join plus Projektion auf die Spalten der erste Tabelle.

- **Self-Join**: Für die Tabelle müssen zwei verschiedene Ersatznamen angegeben werden. Die Tabelle wird über den Befehl Inner-Join mit sich selbst verknüpft.

- **Straight-Join**:

ist bis auf die Tatsache, dass die linke Tabelle immer vor der rechten gelesen wird, identisch mit JOIN. Dies kann für die (wenigen) Fälle genutzt werden, in denen der Join-Optimierer die Tabellen in der falschen Reihenfolge anordnet.

SELECT **STRAIGHT_JOIN** .. {table_factor [**ON** join_bedingung]}

SELECT … FROM tabelle1 **STARIGTH_JOIN** tabelle2 **STARIGHT_JOIN** tabelle3 ..

table_factor:

tbl_name [[AS] alias] [{USE | IGNORE | FORCE} INDEX (key_list)]
  | (table_references)
  | { OJ table_reference LEFT OUTER JOIN table_reference
        ON conditional_expr }

Unister

Beispiel

Als Beispiel dienen die folgenden Tabellen, die durch Joins verbunden werden.

tabelle1

Nr	Name	Vorname
1	Naumann	Karl
2	Baumann	Beate
3	Hartmann	Heike
4	Naumann	Jens

tabelle2

Nr	Telefon
1	12345
7	98765
10	24680

Theta-Join

```
SELECT tabelle1.*, tabelle2.*
FROM tabelle1
CROSS JOIN tabelle2
ON tabelle1.Nr < tabelle2.Nr
```

Nr	Name	Vorname	Nr	Telefon
1	Naumann	Karl	7	98765
1	Naumann	Karl	10	24680
2	Baumann	Beate	7	98765
2	Baumann	Beate	10	24680
3	Hartmann	Heike	7	98765
3	Hartmann	Heike	10	24680
4	Naumann	Jens	7	98765
4	Naumann	Jens	10	24680

Equi-Join

```
SELECT tabelle1.*, tabelle2.*
FROM tabelle1
INNER JOIN tabelle2
ON tabelle1.Nr = tabelle2.Nr
```

Nr	Name	Vorname	Nr	Telefon
1	Naumann	Karl	1	12345

Natural-Join

```
SELECT DISTICNT tabelle1.*, tabelle2.Telefon
FROM tabelle1
INNER JOIN tabelle2
ON tabelle1.Nr = tabelle2.Nr
```

Nr	Name	Vorname	Telefon
1	Naumann	Karl	12345

Left-Outer-Join

```
SELECT tabelle1.*, tabelle2.*
FROM Tabelle1
LEFT OUTER JOIN tabelle2
ON tabelle1.Nr = tabelle2.Nr
```

Nr	Name	Vorname	Nr	Telefon
1	Naumann	Karl	1	12345
2	Baumann	Beate	NULL	NULL
3	Hartmann	Heike	NULL	NULL
4	Naumann	Jens	NULL	NULL

Right-Outer-Join

```
SELECT tabelle1.*, tabelle2.*
FROM Tabelle1
RIGHT OUTER JOIN tabelle2
ON tabelle1.Nr = tabelle2.Nr
```

Nr	Name	Vorname	Nr	Telefon
1	Naumann	Karl	1	12345
NULL	NULL	NULL	10	98765
NULL	NULL	NULL	7	24680

Full-Outer-Join

```
SELECT tabelle1.*, tabelle2.*
FROM Tabelle1
FULL JOIN tabelle2
ON tabelle1.Nr = tabelle2.Nr
```

Nr	Name	Vorname	Nr	Telefon
1	Naumann	Karl	1	12345
2	Baumann	Beate	NULL	NULL
3	Hartmann	Heike	NULL	NULL
4	Naumann	Jens	NULL	NULL
NULL	NULL	NULL	7	98765
NULL	NULL	NULL	10	24680

Semi-Join

```
SELECT tabelle1.*
FROM tabelle1
INNER JOIN tabelle2
ON tabelle1.Nr = tabelle2.Nr
```

Nr	Name	Vorname
1	Naumann	Karl

Self-Join

```
SELECT t1.Nr, t1.Name, t1.Vorname, t2.Name
FROM tabelle1 AS t1
INNER JOIN tabelle1 AS t2
ON t1.Nr <> t2.Nr
```

Nr	Name	Vorname	Chef
1	Naumann	Karl	NULL
2	Baumann	Beate	1
3	Hartmann	Heike	2
4	Naumann	Jens	2

tabelle1

Nr	Name	Vorname	Chef
1	Naumann	Karl	NULL
2	Baumann	Beate	Naumann
3	Hartmann	Heike	Baumann
4	Naumann	Jens	Baumann

Ergebnis der Abfrage

Quellen

http://dev.mysql.com/doc/refman/5.1/de/join.html