SUB-SELECT's

Sub-Selects (oder Unterabfragen) wurden erstmalig in MySQL 4.1.x implementiert und seither weiterentwickelt (Alle hier beschriebenen Varianten wurden mit MySQL 5.0.x getestet). Meist lassen sich Sub-Selects in JOIN - oder andere Abfragen umformulieren, was aber aus Performance – oder Transparenzgründen nicht zu empfehlen ist.

Var. 1

```
SELECT1 ... FROM1 ...
WHERE col = ( SELECT2 ... FROM2 ... );
```

SELECT$_2$ muss genau einen Wert liefern, der als Vergleichswert in die WHERE-Klausel eingesetzt wird (zulässig sind alle Vergleichsoperatoren).

Variante 2

```
SELECT1 ... FROM1 ...
WHERE col = ANY | ALL ( SELECT2 ... FROM2 ... );
```

In diesem Fall darf SELECT$_2$ mehrere Ergebnisse (value$_1$, value$_2$, . . .) haben.

ANY bewirkt eine logische ODER - Verknüpfung der Form :

```
WHERE col = value1 OR col = value2 ... ;
```

ALL bewirkt eine logische UND - Verknüpfung der Form :

```
WHERE col = value1 AND col = value2 ... ;
```

Achtung ! ALL liefert auch dann TRUE, wenn das SELECT$_2$ keine Datensätze liefert.

Var. 3

```
SELECT1 ... FROM1 ...
WHERE col [NOT] IN ( SELECT2 ... FROM2 ... );
```

SELECT$_2$ liefert eine Liste von Werten, die in der Form von

```
... WHERE col IN ( value1 , value2 , ...) ...
```

ausgewertet warden.

Var. 4

```
SELECT1 ... FROM1 ...
WHERE [NOT] EXISTS ( SELECT2 ... FROM2 ... );
```

Für jeden gefundenen Datensatz aus dem SELECT$_1$ wird das SELECT$_2$ ausgeführt.
Nur wenn das SELECT$_2$ mindestens ein Ergebnis liefert, bleibt der entsprechende Datensatz aus dem SELECT$_1$ im Ergebnis-Dynaset.

Variante 5

```
SELECT1 ROW ( value1 , value2 , ...) =
[ANY] ( SELECT2 col1 , col2 , ... FROM2 ... );
```

Das Ergebnis dieser Abfrage kann nur "1" (wahr) oder "0" (falsch) sein.
Das SELECT$_2$ darf genau einen Datensatz liefern, der mit den (mindestens zwei values !) Werten ROW(value$_1$, value$_2$, ...) verglichen wird.

Bei der optionalen Angabe von ANY, darf das SELECT$_2$ auch mehrere Datensätze liefern
Das Ergebnis wird dann nur 1 (wahr) , wenn mindestens einer dieser Datensätze stimmt.

Var.6

```
SELECT1 ... FROM1 ( SELECT2 ... FROM2 ... ) AS Alias WHERE ... ;
```

Das Ergebnis-Dynaset des SELECT$_2$ dient als Datensatzherkunft in der FROM$_1$ -Klausel des SELECT$_1$ (die Alias-Angabe ist hier zwingend notwendig).