Alter

Alter Table Syntax

Correlated Queries

- A correlated subquery is a subquery that contains a reference to a table that also appears in the outer query.

```
SELECT * FROM t1
  WHERE column1 = ANY (SELECT column1 FROM t2
                       WHERE t2.column2 = t1.column2);
```

Scoping rule

- MySQL evaluates from inside to outside.

```
SELECT column1 FROM t1 AS x
  WHERE x.column1 = (SELECT column1 FROM t2 AS x
    WHERE x.column1 = (SELECT column1 FROM t3
      WHERE x.column2 = t3.column1));
```

- In this statement, x.column2 must be a column in table t2 because SELECT column1 FROM t2 AS x ... renames t2. It is not a column in table t1 because SELECT column1 FROM t1 ... is an outer query that is farther out.

Query Implementation

```
SELECT * FROM t1
    WHERE ( SELECT a FROM t2
                WHERE t2.a=t1.a ) > 0;
```

....Transformed

```
SELECT t1.* FROM t1
    LEFT OUTER JOIN
        (SELECT a, COUNT(*) AS ct FROM t2 GROUP BY a) AS derived
    ON  t1.a = derived.a
        AND
        REJECT_IF(
            (ct > 1),
            "ERROR 1242 (21000): Subquery returns more than 1 row"
            )
    WHERE derived.a > 0;
```

Note: Check for the subquery not return more than one row.

Row constructor

- Where (1,2) = (col1,col4)
- Where ROW(1,2) = (co1,col4)

```
SELECT * FROM t1
 WHERE (col1,col2) = (SELECT col3, col4 FROM t2 WHERE id = 10);
SELECT * FROM t1
 WHERE ROW(col1,col2) = (SELECT col3, col4 FROM t2 WHERE id
= 10);
```

Functions and Operators

- MYQL Functions and Operators: Weblink
- XML Functions (image linked)

Name	Description
`ExtractValue()`	Extract a value from an XML string using XPath notation
`UpdateXML()`	Return replaced XML fragment

Miscellaneous

- Natural Language Full-Text Searches

```
SELECT * FROM table_name WHERE MATCH(col1, col2)
AGAINST('search terms' IN NATURAL LANGUAGE MODE)
```

Advanced topic:

Creating a FULLTEXT Index that Uses the ngram Parser

Relevance ordering

When MATCH() is used in a WHERE clause, as in the example shown earlier, the rows returned are automatically sorted with the highest relevance first.

Relevance values are nonnegative floating-point numbers.

- Zero relevance means no similarity.
- Relevance is computed based on -
  - the number of words in the row
  - the number of unique words in that row
  - the total number of words in the collection
  - the number of documents (rows) that contain a particular word.

Relevance Score

The score of relevance can be computed and listed

- Select match(col-list) against (string IN N-L-M) as score ....

Boolean Mode

```
SELECT * FROM table_name WHERE
MATCH(col1, col2)

AGAINST('search terms' IN BOOLEAN MODE)
```

See this page

Boolean Operators

+join +union'	Find rows that contain both words.
+join union'	Search rows that contain the word 'join', but rank rows higher if they also contain 'union'
+join -union'	Find rows that contain the word 'join' but not 'union'.
join -union'	Search rows that contain at least one of the two words.
+join +(>left <right)'	Find rows that contain the words 'join' and 'left' or 'join' and 'right' (in any order), but rank 'join left' higher than 'join right'.
+join ~left'	Find rows that contain the word 'join', but if the row also contains the word 'left', rate it lower than if row does not.
join*'	Find rows that contain words such as 'join', 'joins', 'joining' etc.
"left join"'	Find rows that contain the exact phrase "let join".

User-Defined Variables

User variables are intended to provide data values.

SET @var_name = expr [, @var_name = expr] ...

Permitted Data Types

integer, decimal, floating-point, binary or nonbinary string, or NULL value.

Examples

```
mysql> SET @v1 = X'41';

mysql> SET @v2 = X'41'+0;

mysql> SET @v3 = CAST(X'41'
 AS UNSIGNED);

mysql> SELECT @v1, @v2, @v3;
+------+------+------+
| @v1  | @v2  | @v3  |
+------+------+------+
| A    |   65 |   65 |
+------+------+------+
```

```
mysql> SET @v1 = b'1000001';
mysql> SET @v2 = b'1000001'+0;

mysql> SET @v3 = CAST(b'1000001'
AS UNSIGNED);

mysql> SELECT @v1, @v2, @v3;

+------+------+------+

| @v1  | @v2  | @v3  |

+------+------+------+

| A    |   65 |   65 |

+------+------+------+
```

Guess what happens?

```
mysql> SET @c = "c1";
Query OK, 0 rows affected (0.00 sec)

mysql> SET @s = CONCAT("SELECT ", @c, " FROM t");
Query OK, 0 rows affected (0.00 sec)

mysql> PREPARE stmt FROM @s;
Query OK, 0 rows affected (0.04 sec)
Statement prepared

mysql> EXECUTE stmt;
```

??

```
SELECT
  CONCAT(z.expected, IF(z.got-1>z.expected,
     CONCAT(' thru ',z.got-1), '')) AS missing
FROM (
  SELECT
    @rownum:=@rownum+1 AS expected,
    IF(@rownum=YourCol, 0, @rownum:=YourCol) AS
got
  FROM
    (SELECT @rownum:=0) AS a
    JOIN YourTable
    ORDER BY YourCol
  ) AS z
WHERE z.got!=0;
```

How can we find gaps in sequential numbering in MySQL???

```
SELECT
  CONCAT(z.expected, IF(z.got-1>z.expected,
     CONCAT(' thru ',z.got-1), '')) AS missing
FROM (
  SELECT
    @rownum:=@rownum+1 AS expected,
    IF(@rownum=YourCol, 0, @rownum:=YourCol) AS
got
  FROM
    (SELECT @rownum:=0) AS a
    JOIN YourTable
    ORDER BY YourCol
  ) AS z
WHERE z.got!=0;
```