

MaxDB

Introduction to Query Optimization Release 7.6

Werner Thesing

THE BEST-RUN BUSINESSES RUN SAP™



The task of an optimizer is to find the best search strategy for a given SQL statement.



Optimization

Explain

Strategy Examples

Update Statistics



Goal: Minimizing resource-consumption like

- CPU-time
- I/O-load
- Memory
- Disk space

SQL Commands affected by optimization

- SELECT (mass select, single select)
- Update
- Delete
- Insert



Rule based optimizer

- Access strategy is defined through rules at parsing time.
- Does not depend on values in the WHERE clause
- The rule determines which access type is selected.

Cost based optimizer

- Searching strategy is determined via
 - current column content (values)
 - indexes available
 - estimated count of (page) accesses
- ‚Lowest cost‘ strategy will be used.

There are two types of optimizers for relational database systems: rule-based and cost-based optimizers.

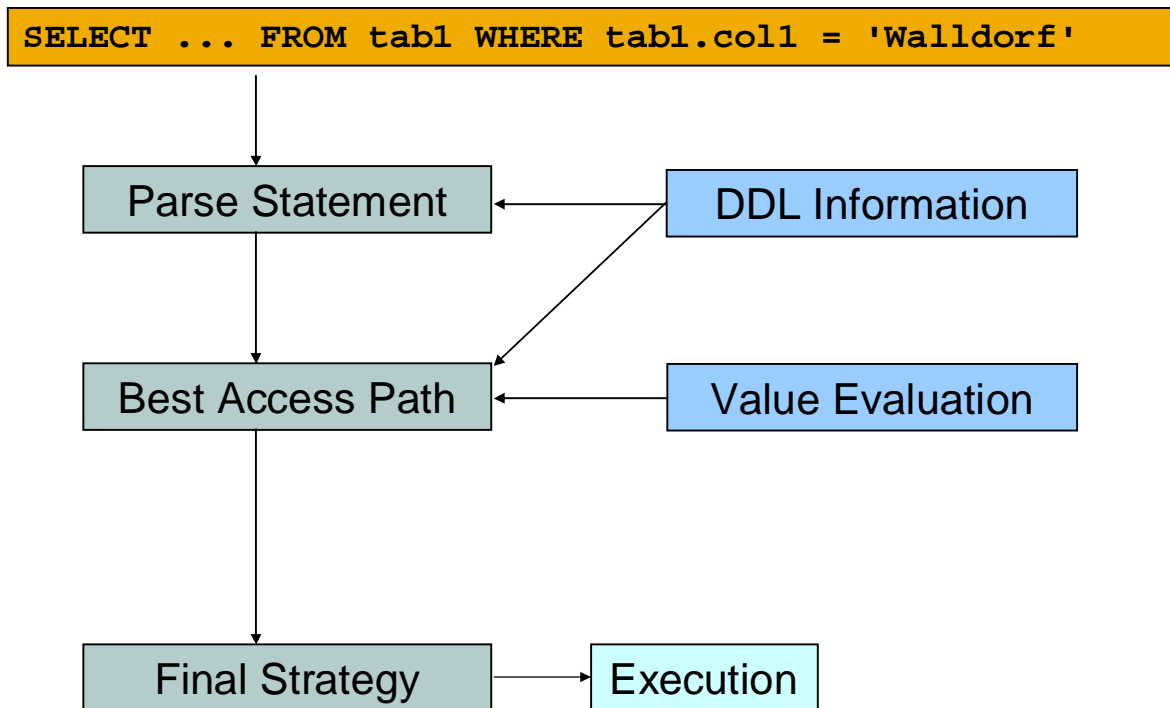
The rule-based optimizer works according to certain rules. For example, if an index is available, this index will be used for access - independent of the values in the WHERE condition. With the rule-based optimizer, the strategy for processing SQL statements is decided at the time of parsing.

Cost-based optimizers determine the best search strategy with the help of statistical information about the size of the table and values within the table columns.

A cost-benefit plan is created for the various access options. The best strategy is chosen to execute the command depending on the values defined in the WHERE condition. Therefore, the eventual search strategy can only be determined at the time of execution.

MaxDB supports cost-based optimizers.

A new implementation of the SQL optimizer was introduced in version 7.4. It makes it easier to maintain the code. New functions can be installed without a great risk of negative side-effects.



First, an SQL statement is processed by the parser. This performs a syntactic and semantic analysis. In the semantic analysis, tables and their column data are checked.

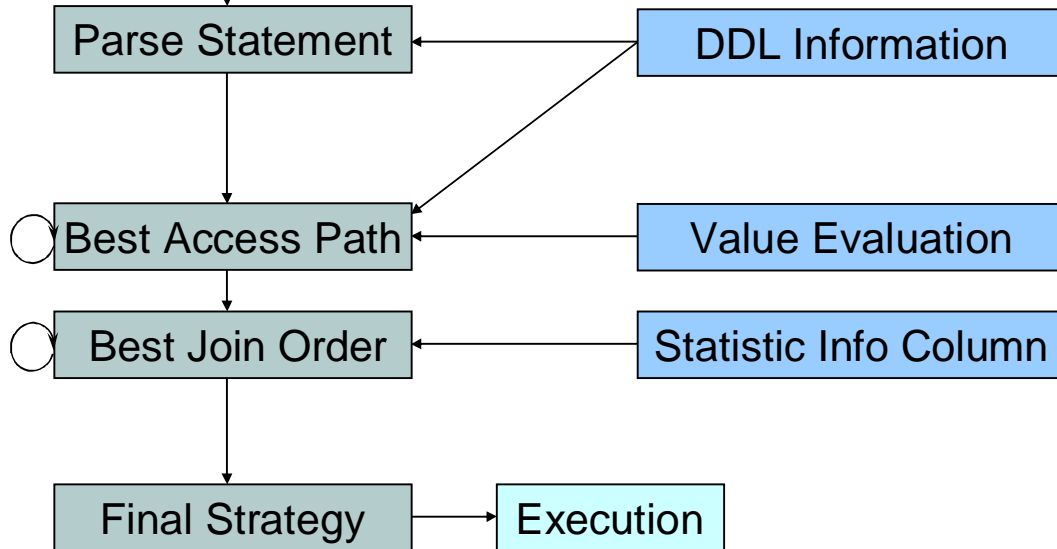
The optimizer determines which primary and secondary keys are available for the table and checks whether a corresponding key can be used to search for values.

For secondary keys, the number of differing values plays an important role. Example: it does not make sense to search using an index if there is only one secondary key value, it is precisely this value that will be searched for, and additional table fields will be queried.

The number of pages that have to be read in the secondary index is determined by generating a start and a stop key. Depending on the number of pages of the table, it is decided whether it is worthwhile to search using the index. The number of pages of the entire table is located in the statistics.

At the end, the strategy with which the SQL statement will be executed is determined.

```
SELECT ... FROM tab1, tab2 WHERE tab1.col1 = 'Walldorf'  
AND tab1.key1 = tab2.key1
```



For a JOIN, the optimizer seeks out the most suitable access path for each table.

Then it has to be decided in which order the tables will be processed and connected with each other. The resulting result sets should be as small as possible. For the join columns, the values are unknown before the execution. Therefore, the joint optimizer can only work with the statistical values for columns.

Which condition will be evaluated ?



Single table select

- Column = value
- Column <, <=, >=, > value
- Column BETWEEN value AND value
- Column IN (value, value, ...)
- Column LIKE string value (including %,?,...)
- Column = (ANY) <subquery>
- Column IN <subquery>

Join select

- Table1_column = table2_column
- Table1_column <, <=, >=, > table2_column
- (Condition has to be on the 'Top-AND-Level' of the <search condition>)

© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 7

The search conditions that the optimizer can use to determine the optimal search strategy are the following:

- Equality conditions
- Range conditions
- IN conditions
- IN conditions

Here, the search conditions are displayed in the order of their valency. In other words, with the same preconditions an equality condition is evaluated as being better than an IN condition.

The SQL Optimizer also converts conditions under certain circumstances. If a single value is specified in an IN condition multiple times, the condition is converted into an equality condition.

```
Create Table ZZTELE
( NAME          CHAR(40),
  VORNAME       CHAR(20),
  STR           CHAR(40),
  NR            INT,
  PLZ           CHAR(5),
  ORT           CHAR(25),
  CODE          CHAR(31),
  ADDINFO       CHAR(31),
  PRIMARY KEY
  (NAME, VORNAME, STR) )
```

of records: around 115,000

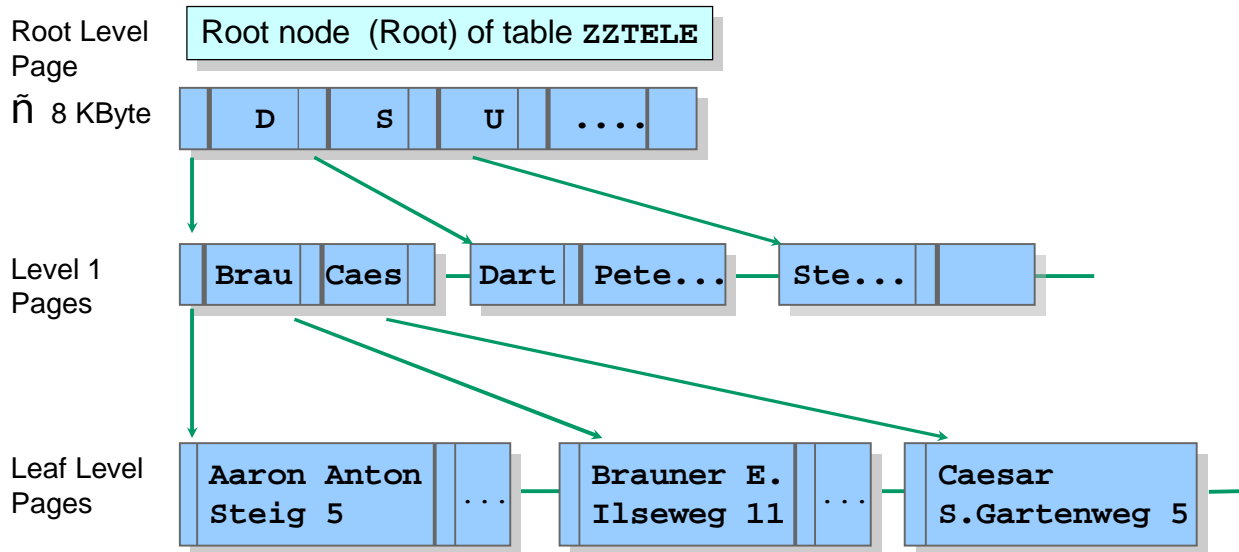
```
Create Table ZZSTADTTEIL
( PLZ           CHAR(5),
  ORT           CHAR(25),
  STADTTEIL     CHAR(40),
  PRIMARY KEY
  (PLZ) )
```

of records : around 20,000

In the following examples, we use the table ZZTELE with approx. 115,000 records.

For joins and subqueries, the examples also refer to the table ZZSTADTEIL with approx. 20000 records.

Storing data in a B*tree



Base data as well as index data (secondary key) is stored in B*tree structures.

© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 9

The data of the base tables and the indexes are stored in B*Tree format.

When creating a table, the root page is created. A root page can contain a maximum of 8 KB.

If data records are entered in the tables, the root page is filled with what are known as separators. A separator is made up of the primary key of the data record. However, due to space limitations, the entire key is not saved as a separator in the root page, but rather only the part of the key up to the first significant digit in the key. The more significant a key is, the smaller the separators are and the more separators can be managed in the root page.

For very small tables, all data records are already stored in the root page. If the root page is filled, entering additional records in the table will automatically generate an additional tree level, or what is known as the Level 1 pages level. The root page will then consist only of separators and pointers to the corresponding lower level containing the information with a distinguishing separator.



Primary key

- The primary key is kept on the data tree (clustered)
- No separate tree for primary key !
- The primary key is used as separator in B*trees
- The records are stored in primary key order

Secondary key (index)

- Create a separate B*tree for the secondary key
- A secondary key does not contain physical addresses pointing to the base data but logical addresses in terms of primary keys



Input : EXPLAIN <Select-Command>
EXPLAIN VIEW < Select-Command >

Output : Description of search strategy

- The explain command cannot be used with UPDATE , DELETE or INSERT commands.
- EXPLAIN is used with Select commands that access base tables
- EXPLAIN VIEW shows names of tables and indexes a (multiple) (JOIN) view reques is based on.
- EXPLAIN does not execute the specified Select command.

In the ABAP-based SAP application server, EXPLAIN VIEW is available in transactions ST05 and DB50 (in the command monitor).

You can display the search strategy for INSERT, DELETE and UPDATE commands by transforming the command into a SELECT. The additional option FOR REUSE ensures that the results table is stored. Example:

Example:

```
UPDATE ZZTELE
SET ADDINFO = 'ledig'
WHERE NAME = 'Mueller'
AND VORNAME = ' Egon'
AND STR = ' Wexstraße'
```

```
SELECT * FROM ZZTELE
WHERE NAME = 'Mueller'
AND VORNAME = ' Egon'
AND STR = ' Wexstraße'
FOR REUSE
```

Explain (2)



OWNER	TABLENAME	COLUMN_OR_INDEX	STRATEGY	PAGECOUNT
User	Table 1	Names of key or index columns	Name of chosen strategy for this table	Number of affected data pages
User	Table 2	Names of key or index columns	Name of chosen strategy for this table	Number of affected data pages
			RESULT IS (NOT) COPIED, COSTVALUE IS	Estimated Costs

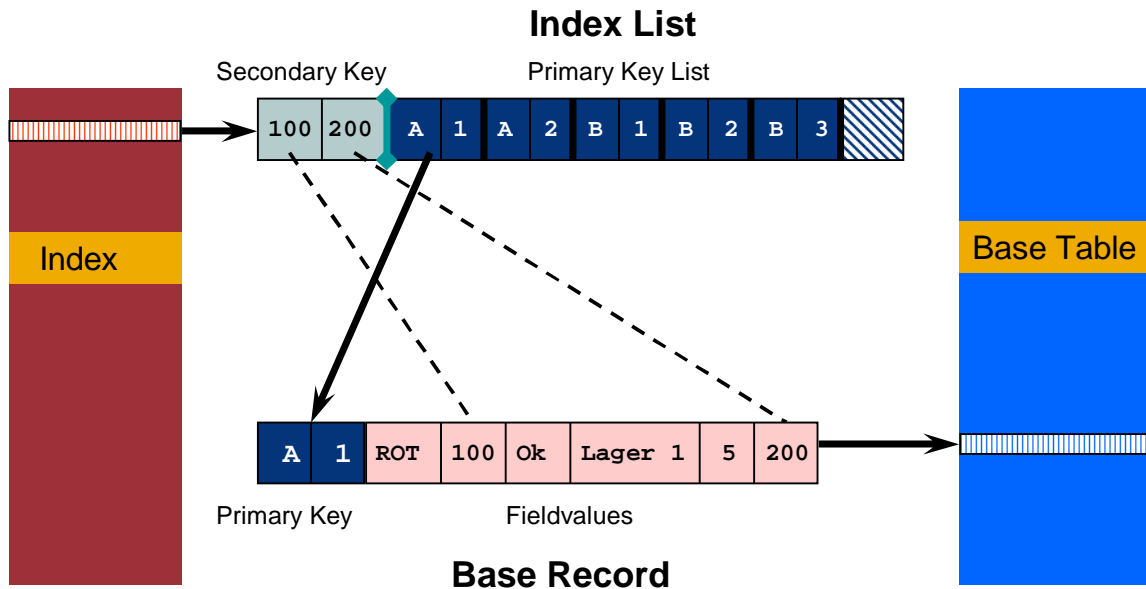
© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 12

EXPLAIN shows:

- one block for each table from the SELECT-FROM list
- the order of the strategies reflects the order of execution
- the order of the strategies reflects the order of execution
- COPIED / NOT COPIED --> Results set is generated/not generated
- "Estimated costs" provides an estimate about the number of disk accesses (logical I/Os).

The columns O, D, T and M, which were output with older database versions, are no longer displayed. These columns had specified additional strategies for search conditions. As of version 7.5, the EXPLAIN statement shows the additional strategies as text in the STRATEGY column.

Index and base table



© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 13

An index contains the data of the secondary key as well as the respective primary key. Using the primary key, the data can be found in the base table. For each index, a B* tree is created, which is sorted according to the values of the secondary key.

There is no record ID or anything similar. The unique ID of a record is the primary key (or for multiple keys, the combination of primary key fields).

If no primary key was specified when the table was generated, the database generates the internal field `SYSKEY` of the type `CHAR(8) BYTE`. This field is filled with unique values.

Searching via an index is relatively costly. The access is only worthwhile if less than approx. 30% of the records can be determined from the index and no results set is generated.

On the following page you will find examples of search strategies. The list of strategies is not complete. A complete list of search strategies can be found in the documentation.

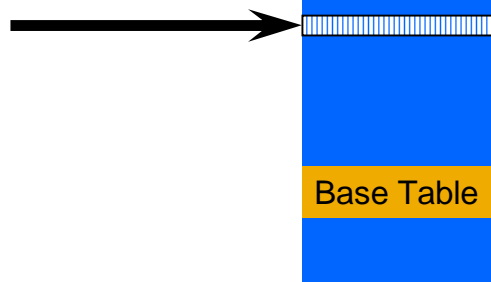
Basic Information -> Background Knowledge -> SQL Optimizer -> Search Strategy -> List of all search strategies

EQUAL CONDITION FOR KEY



```
SELECT * FROM zztele
WHERE Name      = 'Aaron'
AND  Vorname   = 'Anton'
AND  Str        = 'Alt Moabit'
```

Key Value =



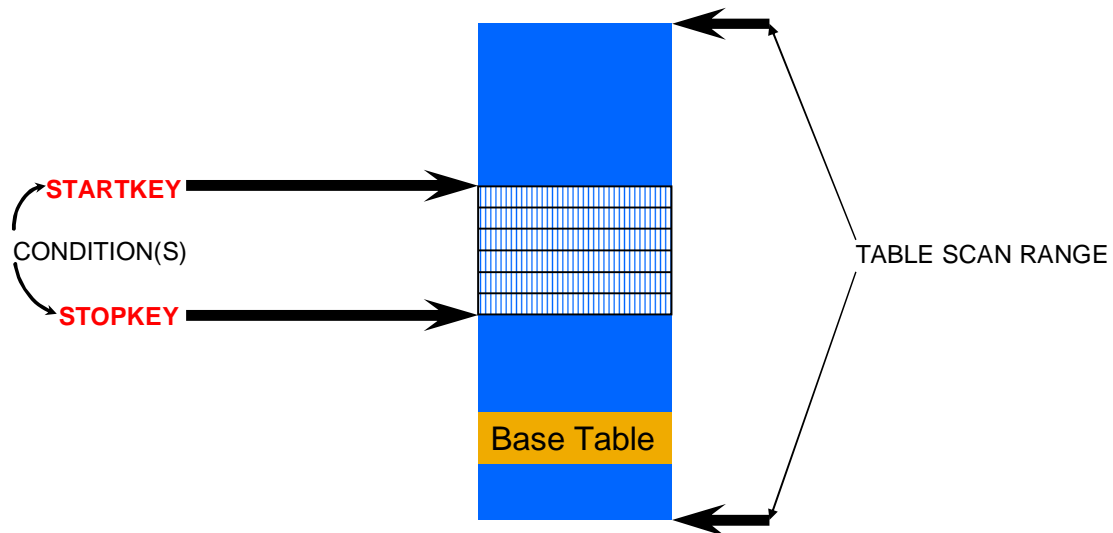
EQUAL CONDITION FOR KEY provides an efficient access path through "direct access" to the base table.

The decision in favor of this strategy will already have been made at the time of parsing because, independent of the data in the search conditions, no better search strategy is possible.

RANGE CONDITION FOR KEY



```
SELECT * FROM zztele WHERE Name = 'Schmidt'  
                        AND  Vorname like 'A%'  
SELECT * FROM zztele
```



© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 15

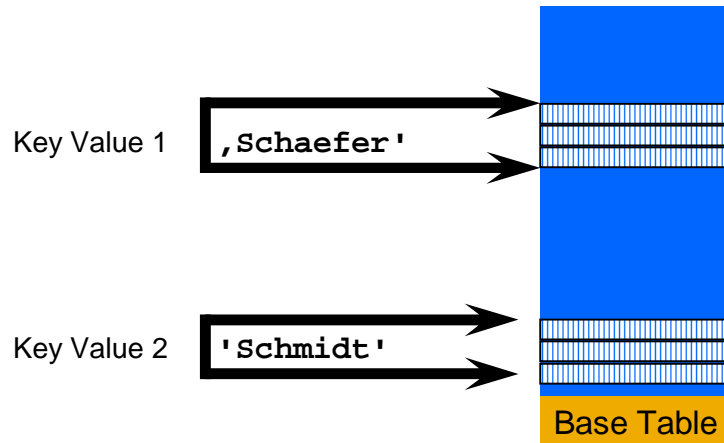
If a portion of the start of the primary key is specified in the WHERE condition, the strategy RANGE CONDITION FOR KEY will be executed.

If the index and primary key cannot be used, the base table will be searched completely (TABLE SCAN).

An intermediate result set is not generated.



```
SELECT * FROM zztele
WHERE Name IN ('Schaefer', 'Schmidt')
```



© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 16

The IN condition can be placed on each field of a primary key.

Only one IN condition is taken into account.

The primary key fields that precede the field with the IN condition may only be specified in an EQUAL condition.

An intermediate result set is generated. The result set is sorted according to the primary key.

As of version 7.4, the optimizer checks whether the RANGE CONDITION FOR KEY is advantageous. This happens if the values in the IN condition are close to each other.

Example:

```
SELECT *
FROM zztele
WHERE name IN ('Schaefer', 'Schmidt')
```

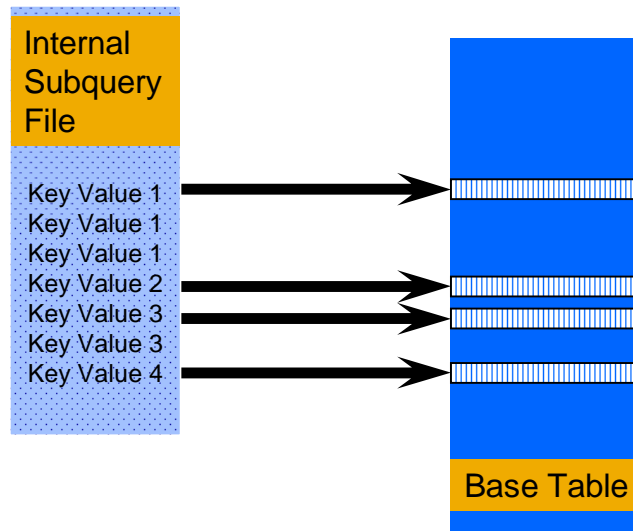
There are additional names in the table that are located between the values 'Schaefer' and 'Schmidt'. There are additional names in the table that are located between the values 'Schaefer' and 'Schmidt'. Thus, using this search condition, records are also included that do not belong to the results set. However, the strategy is more favorable since only one start and stop key have to be determined.

RANGE CONDITION FOR KEY (SUBQUERY)



```
CREATE INDEX "ZZSTADTTEIL~1" ON ZZSTADTTEIL(STADTTEIL)

SELECT * FROM zztele WHERE name IN
(SELECT stadtteil FROM zzstadtteil
WHERE stadtteil = 'Ahlheim' )
```

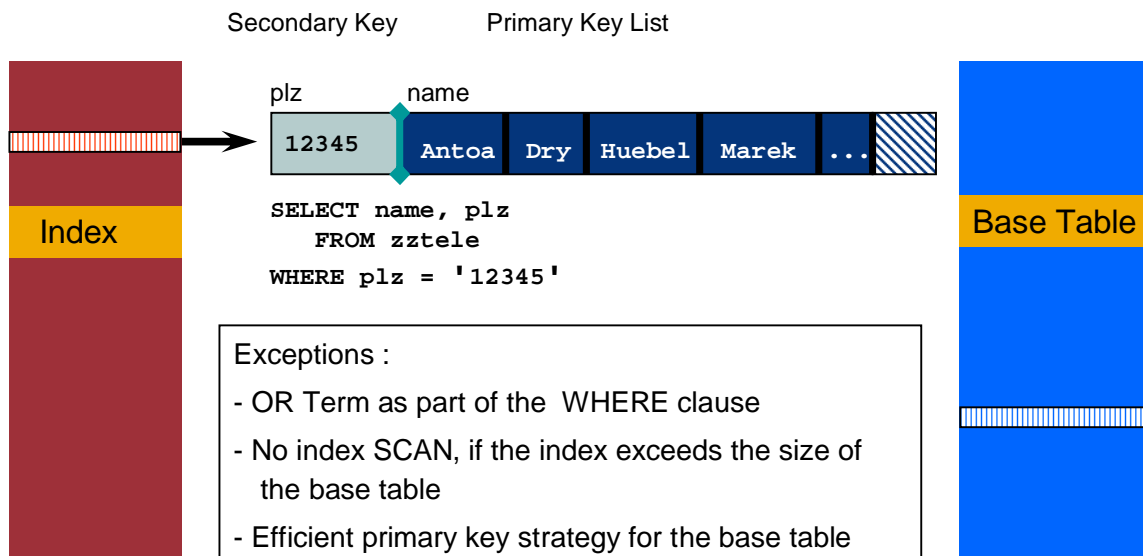


© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 17

If a subquery returns primary key values, EQUAL CONDITION FOR KEY or RANGE CONDITION FOR KEY is used on the base table. The result set is sorted according to primary key values.

An intermediate result set is generated.

ONLY INDEX ACCESSED



Kernel parameter: OPTIM_INV_ONLY

© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 18

If a SELECT statement only addresses columns that are also contained in an index (SELECT list, WHERE clause), then only this index will be accessed for the execution of the command.

Advantage:

- In some cases, significantly fewer pages that have to be searched
- Optimal usage of sorting of secondary and primary keys in the index
- No additional access to the base table
- No determination of access costs (only for the join)

Exceptions:

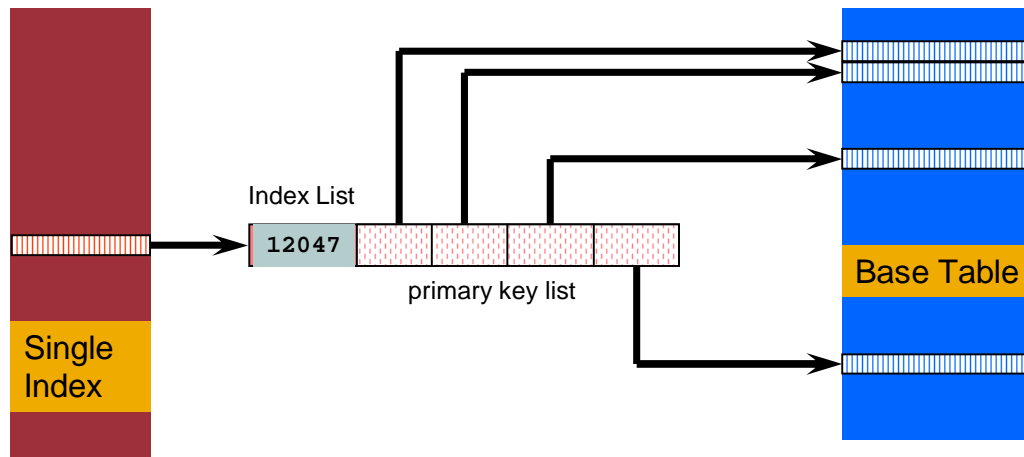
- OR term in the WHERE condition
- No index SCAN if the index is larger than the base table
- Efficient primary key strategy via the base table

EQUAL CONDITION FOR INDEX



```
CREATE INDEX "ZZTELE~3" ON ZZTELE(PLZ)
```

```
SELECT * FROM zztele  
WHERE plz = '10559'
```



© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 19

Efficient access path for fields with greater selectivity

When determining the strategy, additional costs (`index_overhead`) for accessing the base data via the index are taken into account.

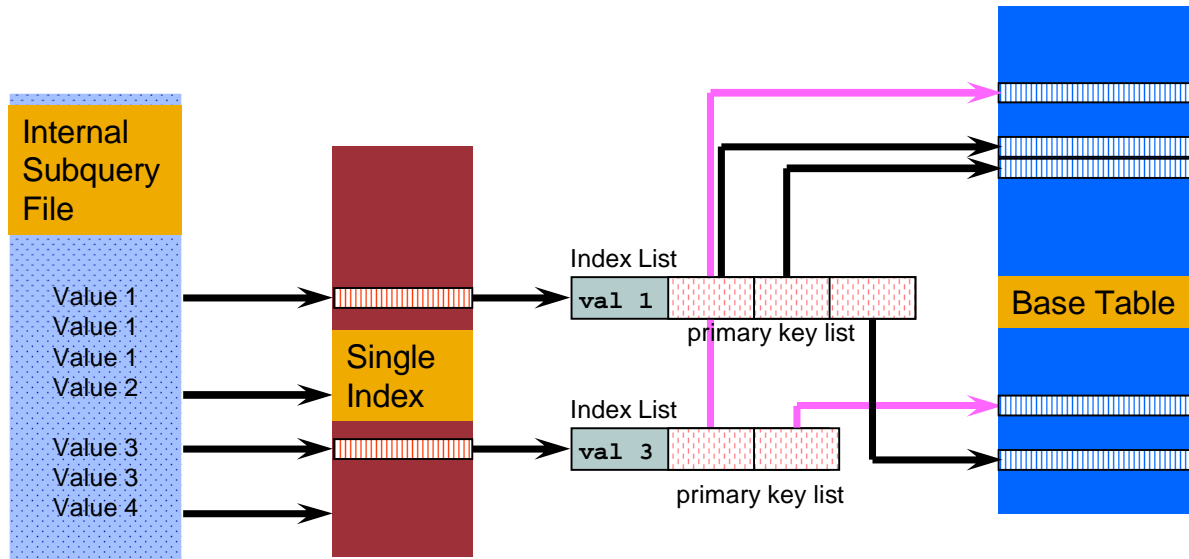
The optimizer also opts for the strategy `EQUAL CONDITION FOR INDEX`, if all fields of a multiple index in the `WHERE` condition are specified with an equality condition.

An intermediate result set is not generated.

EQUAL CONDITION FOR INDEX (SUBQUERY)



```
CREATE INDEX "ZZTELE~3" ON ZZTELE(PLZ)
SELECT * FROM zztele WHERE plz IN
( SELECT plz FROM zzstadtteil WHERE plz = '12047' )
```



© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 20

The result set is sorted according to the secondary key sequence. If only values from the index are queried, the Only Index strategy is used.

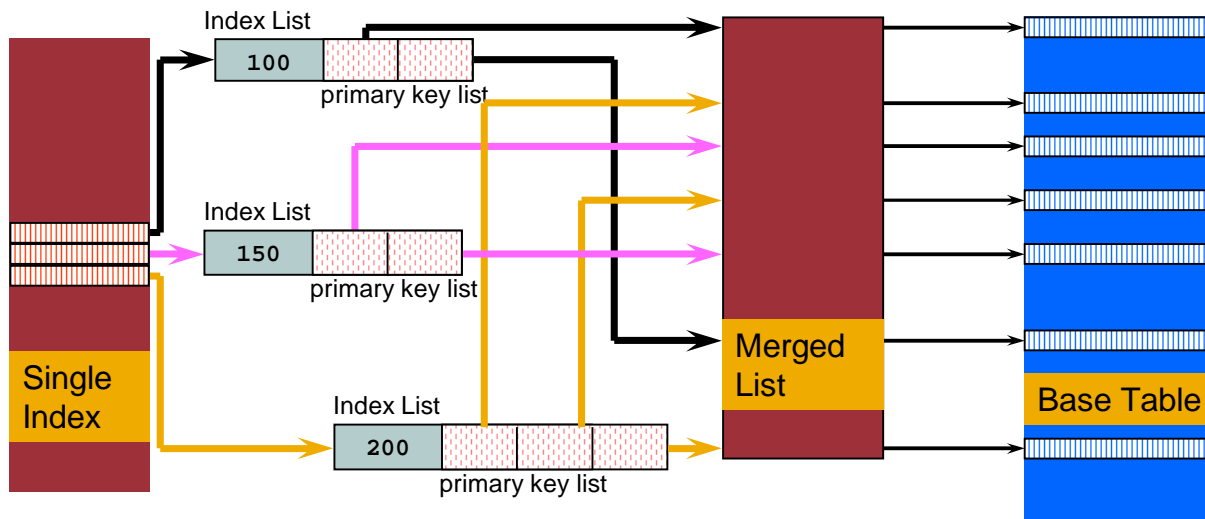
An intermediate result set is generated.

RANGE CONDITION FOR INDEX



```
CREATE INDEX "ZZTELE~3" ON ZZTELE ( STR, NR )
```

```
SELECT * FROM zztele WHERE str = 'Wexstr'  
AND nr BETWEEN 20 AND 23  
ORDER BY name, vorname, str
```



Kernel parameter: OPTIM_MAX_MERGE

© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 21

The result set is sorted according to the primary key.

Using the additional strategy TEMPORARY INDEX CREATED, the primary keys are sorted in a merge list. The optimum cache usage is guaranteed using access to the base data in the order of the primary keys.

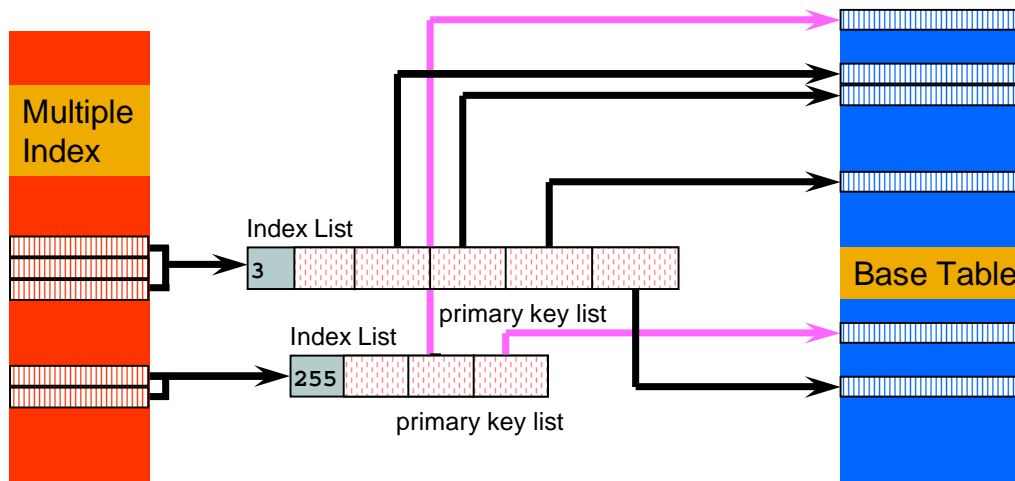
The maximum size of the merge lists that are generated can be configured using the parameter OPTIM_MAX_MERGE.

An intermediate result set is not generated.

IN CONDITION FOR INDEX



```
CREATE INDEX "ZZTELE~3" ON ZZTELE ( STR, NR )
SELECT * FROM zztele
  WHERE str = 'Wexstr' AND nr IN (3, 255)
```



© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 22

A secondary key can be taken into account for an IN condition. Only one IN condition is taken into account.

The secondary key fields that precede the field with the IN condition may only be specified in an EQUAL condition.

The result set is sorted according to the secondary key.

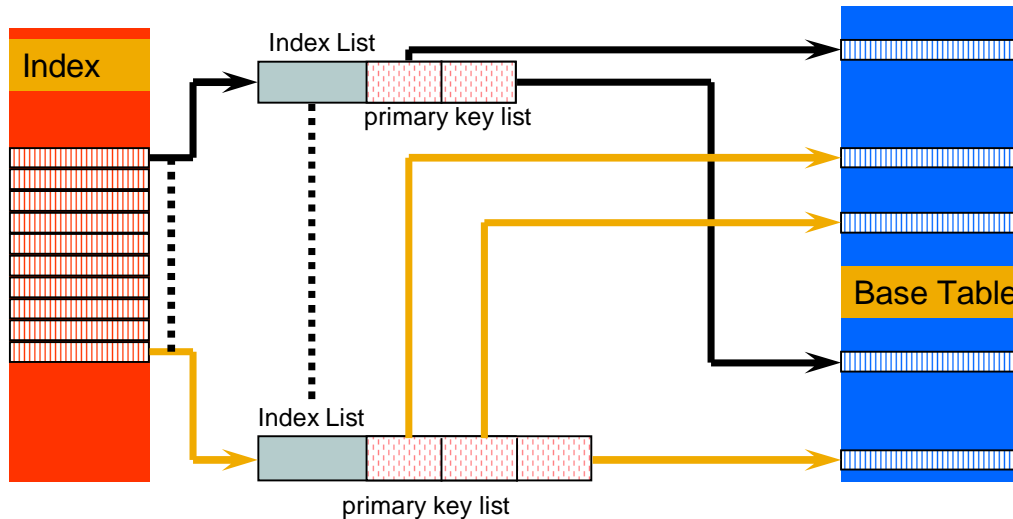
The Only Index strategy can be used.

An intermediate result set is generated.

INDEX SCAN



```
CREATE INDEX "ZZTELE~2" ON zztele ( str, nr )  
SELECT * FROM zztele WHERE name BETWEEN 'A' and 'D'  
ORDER BY str, nr
```



© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 23

During an INDEX SCAN, all entries are read via the index in the order of the secondary key. An intermediate results set is not generated.

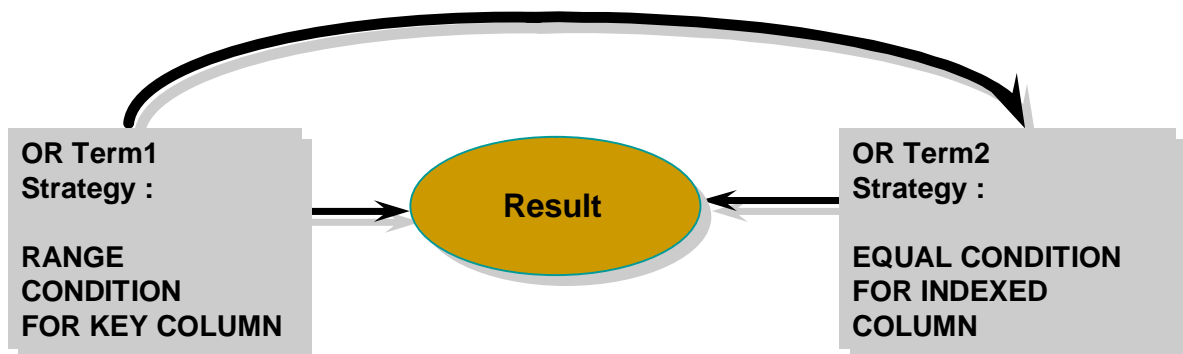
As of version 7.4, NULL values are also included in single indexes. Thus, this strategy can be used on all indexes.

If a Table Scan is to be carried out for an ORDER BY because no index can be used, an intermediate results set is generated.

DIFFERENT STRATEGIES FOR OR-TERMS



```
CREATE INDEX "ZZTELE~3" ON ZZTELE(PLZ)
SELECT * FROM zztele
WHERE name= 'Aaron'
OR plz = '12345'
```



© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 24

Nested OR terms are analyzed down to the third level.

The strategy search is only carried out if there is no adequate strategy on the highest level.

If the costs of the strategy search exceed the costs determined for the highest level, the strategy search is discontinued.

An intermediate result set is generated.

Within the SAP environment, similar statements are also generated by SELECTS with RANGES.

NO STRATEGY NOW (ONLY AT EXECUTION TIME)



Strategy will be determined first during execution of the command

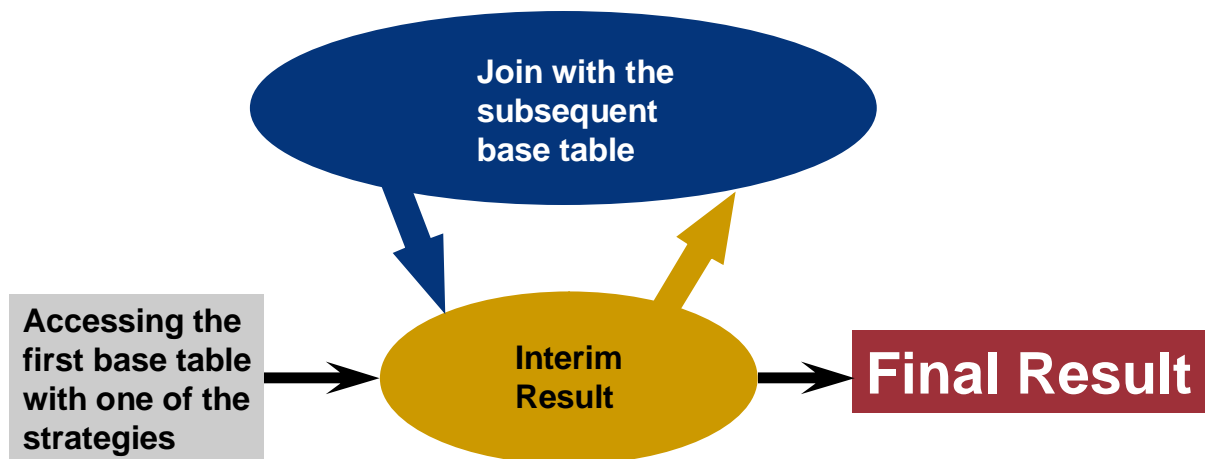
Is displayed for queries if the access path will be determined first when they are executed

Is displayed for queries containing sub-queries or correlated sub-queries: strategy will first be determined when interim results become available.

Join Strategies (1)



```
SELECT * FROM zztele, zzstadtteil
WHERE zztele.Plz = zzstadtteil.Plz
AND zztele.Ort = zzstadtteil.Ort
AND zztele.name = 'Mueller'
```



© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 26

The costs for a join are based on information about the value distribution.

In general, the costs of a join decrease as the number of joined columns increases.

For joins, an intermediate result set is always generated.



```
SELECT * FROM scantab, jointab
WHERE scantab.A = jointab.Col1
AND scantab.B = jointab.Col2
```

Join Strategy	Meaning
JOIN VIA KEY COLUMN	col1 is the sole primary key column col2 is a standard column
JOIN VIA KEY RANGE	col1 is the first primary key column col2 is a standard column
JOIN VIA MULTIPLE KEY COLUMNS	col1 is the first primary key column col2 is the last primary key column
JOIN VIA RANGE OF MULTIPLE KEY COLUMNS	col1 is the first primary key column col2 is the second primary key column



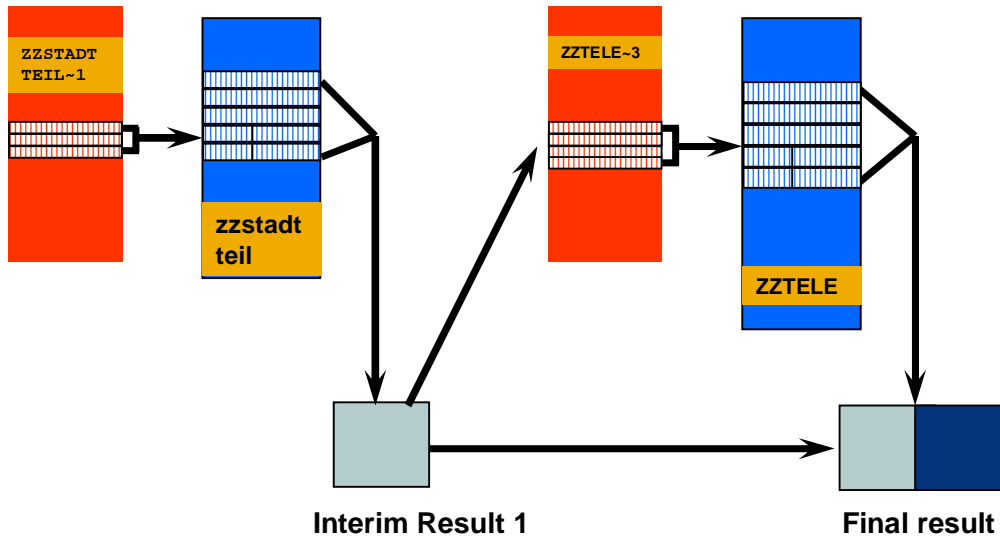
```
SELECT * FROM scantab, jointab
WHERE scantab.A = jointab.Col1
AND scantab.B = jointab.Col2
```

Join Strategy	Meaning
JOIN VIA INDEXED COLUMN	col1 is a single index column col2 is a standard column
JOIN VIA MULTIPLE INDEXED COLUMNS	col1 is the first column of a multiple index col2 is the last column of a multiple index
JOIN VIA RANGE OF MULTIPLE INDEXED COLUMNS	col1 is the first column of a multiple index col2 is the second column of a multiple index

Join Across two Tables (Sorted Merge)



```
CREATE INDEX "ZZTELE~3" ON ZZTELE(PLZ)
CREATE INDEX "ZZSTADTTEIL~1" ON ZZSTADTTEIL(STADTTEIL)
SELECT * FROM zztele, zzstadtteil
WHERE zztele.plz = zzstadtteil.plz
      AND zzstadtteil.stadtteil = 'Moabit'
```



© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 29

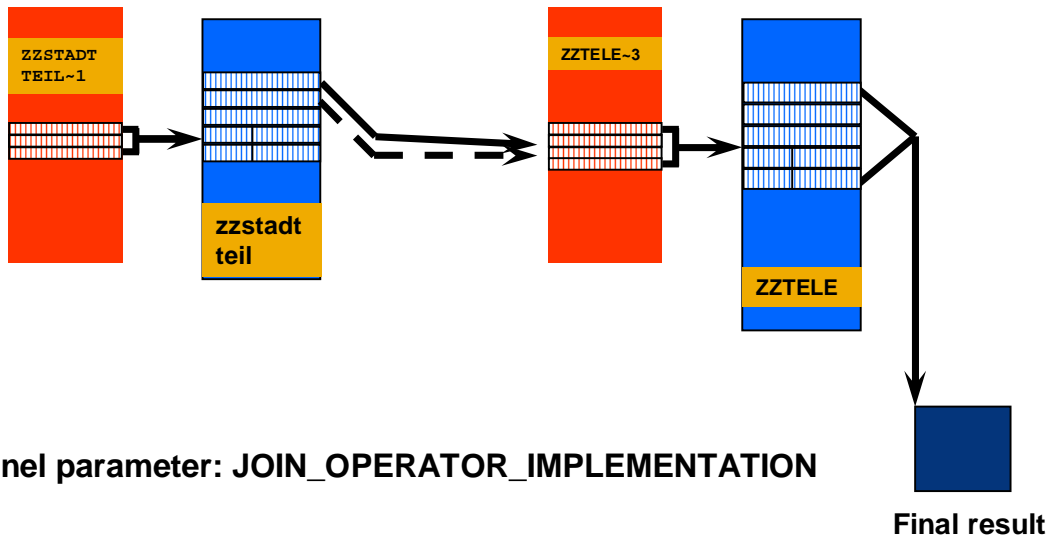
For a join across multiple tables (Sorted Merge), a result set is generated from the first table that is to be processed. Using this result, the join transition is performed on the second table.

A sorted merge is characterized by low CPU consumption. If the intermediate results do not fit in the cache, the subsequent reading of these results into the cache can be costly.

Join Across two Tables (Nested Loop)



```
CREATE INDEX "ZZTELE~3" ON ZZTELE(PLZ)
CREATE INDEX "ZZSTADTTEIL~1" ON ZZSTADTTEIL(STADTTEIL)
SELECT * FROM zztele, zzstadtteil
WHERE zztele.plz = zzstadtteil.plz
      AND zzstadtteil.stadtteil = 'Moabit'
```



© SAP 2007 /MaxDB 7.6 Internals – Optimizer Introduction/Page 30

The nested loops procedure is particularly advantageous when generating aggregates.

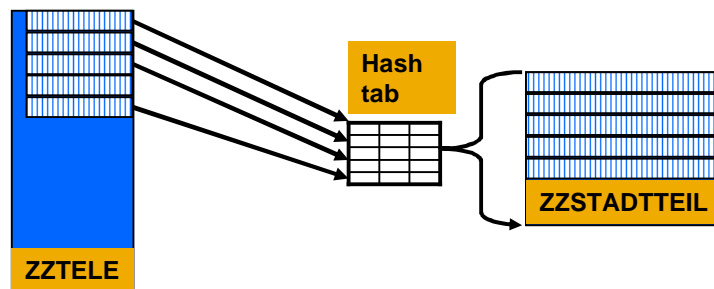
A nested loop forgoes the generation of intermediate results sets. The CPU load is, however, greater than with a sorted merge.

The strategy "NO TEMPORARY RESULTS CREATED" indicates that nested loop was executed.

Hash Join



```
CREATE INDEX "ZZTELE~3" ON ZZTELE(PLZ)
SELECT zztele.plz FROM zztele, zzstadtteil
WHERE zztele.plz = zzstadtteil.plz
AND zzstadtteil.stadtteil = 'Moabit'
```



Kernel parameter: MAX_HASHTABLE_MEMORY + MAX_SINGLE_HASHTABLE_SIZE

The hash join strategy is employed when a join transition to a small table is done and it is probable that a large number of records needs to be read from the small table.

In this case it would be faster to import the small table once and generate a temporary hash table. Searching for the keys in a hash table is faster than searching via the B* tree of the table.

The strategy "TABLE HASHED" identifies the join via a hash table.



Hints provide the Optimizer with rules that it can use if necessary.

Example:

```
SELECT /*+ORDERED*/ zztele.plz
  FROM zztele, zzstadtteil
 WHERE zztele.plz = zzstadtteil.plz
       AND zzstadtteil.stadtteil = 'Moabit,
```

Hints are supported as of:

- MaxDB Version 7.5
- WebAS ABAP Version 6.20

MaxDB supports the following hints, the meaning of which can be extracted from SAP note 832544:

KEYACCESS, KEYSRANGE, INDEXACCESS[(**<INDEXNAME>**)] , KEYSRANGE,
INDEXSCAN, INDEXRANGE, BUILDRESULT, FETCHRESULT, DISABLE_INVONLY,
IN_STRATEGY, SUBQ_STRATEGY, TRACE, ORDERED, COORDINATOR_JOIN,
OPERATOR_JOIN, PARALLEL_SERVER(**<unsigned integer>**), NOACCESSPATH,
ACCESS=**<access hint list>**, BUFFERSIZE, QUERYREWRITE_OP ,
QUERYREWRITE_STMT, QUERYREWRITE_NO



```
UPDATE STAT[ISTICS] [<owner>.<table_name>
([ESTIMATE SAMPLE <unsigned_integer> <PERCENT,ROWS>])
```

To determine the best possible access path, in particular for joins, the Optimizer requires statistical information. If such information is not kept current, the system may make erroneous strategic decisions.

UPDATE STATISTICS determines values about the size of a table as well as the size and value distribution of indexes.

UPDATE STATISTICS should be executed following large-scale change transactions (INSERT/LOAD, UPDATE, DELETE).

Start using the DBM command `sql_updatestat` and `sql_updatestat_per_systemtable` or via the CCMS (transactions DB13, DB21).

As of version 7.5, MaxDB requires statistics data only for joins and selects with a restriction of the records in the result, such as „WHERE ROWNUM <= n“.

For the table itself, Update Statistics only determines data if the current size information is not already in the file directory. This does not apply to table created with databases of versions < 7.6 and for which no size information could yet be determined in the file directory.

Update Statistics determines statistics data for all columns that are primary key or index columns. It also determines the statistics data for all columns outside of the primary key and the index, if statistics are available.

When the Optimizer discovers tables with blunderous statistics data, it enters them in the table `SYSUPDSTATWANTED`. The DBM command `sql_updatestat_per_systemtable` executes Update Statistics for all tables listed in `SYSUPDSTATWANTED`.

The DBM command executes Update Statistics for all tables in the database.

Update Statistics imports the data for a table from all data volumes in parallel. This makes it very speedy.

As of version 7.6, the sampling procedure in the standard uses a new algorithm for calculating the statistics data. You can determine the algorithm to be used with the parameter `UPDATESTAT_SAMPLE_ALGO`. The new algorithm generates more accurate statistics with fewer records read.

The programs "xpu" and "updcpl" are no longer available as of version 7.6.



```
ALTER TABLE <table_name>  
  SAMPLE <unsigned_integer> <PERCENT,ROWS>
```

The default value for the number of rows to be included when determining the statistics is stored in the database catalog.

This value can be changed either directly with ALTER TABLE or using transaction DB50 -> Problem Analysis -> Tables/Views

For tables that grow and shrink very quickly, such as spool tables, for example, it is a good idea to set the sampling rate to 0. This prevents Update Statistics from being requested and executed for these tables.

For tables that were created with versions < 7.6, the counters for size data in the file directory after upgrade to version 7.5 are not yet available. You can determine the counters with a CHECK DATA in the ADMIN state or with CHECK TABLE WITH SHARE LOCK. CHECK TABLE sets a share lock for the duration of the check.

After the upgrade from versions < 7.6 to versions >= 7.6, all table names are transferred to the table SYSUPDATECOUNTERWANTED. With every restart, the database attempts to determine the counters for all remaining tables in SYSUPDATECOUNTERWANTED for the file directory. A share lock is set on a table during processing. Determination of the counters is immediately terminated for a table if the share lock causes a lock collision.



```
SELECT * FROM OPTIMIZERSTATISTICS
WHERE tablename = '...'
```

- Shows the current statistic values that will be used by the optimizer to determine the strategy.

TABLENAME	INDEXNAME	COLUMNNAME	DISTINCTVALUES	PAGECOUNT
ZZTELE	?	ADDINFO	1969	?
ZZTELE	?	CODE	2	?
ZZTELE	?	NAME	13363	?
ZZTELE	?	NR	255	?
ZZTELE	?	ORT	2	?
ZZTELE	?	PLZ	20001	?
ZZTELE	?	STR	8	?
ZZTELE	?	VORNAME	5156	?
ZZTELE	CODE	?	?	1155
ZZTELE	ZZTELE~1	?	?	1165
ZZTELE	ZZTELE~3	?	?	1112
ZZTELE	ZZTELE~4	?	?	1334
ZZTELE	ZZTELE~2	?	?	1548
ZZTELE	?	TABLE STATISTICS	114199	1800

The Optimizer only uses the statistics data for tables only if the counters for size data are not in the file directory.



```
SELECT f.type, r.tablename, r.indexname, f.entrycount,
       f.treeindexsize, f.treeleavessize, f.lobsize
FROM   files f, roots r
WHERE  f.fileid = r.tableid
AND    r.tablename IN ('ZZTELE' )
```

- Displays the current counter values in the file directory.

TYPE	TABLERNAME	INDEXNAME	ENTRYCOUNT	TREEINDEXSIZE	TREELEAVESSIZE	LOBSIZE
TABLE	ZZTELE	?	114199	144	14400	0
INDEX	ZZTELE	CODE	2	9240	9240	?
INDEX	ZZTELE	ZZTELE~1	10	9320	9320	?
INDEX	ZZTELE	ZZTELE~3	20001	8896	8896	?
INDEX	ZZTELE	ZZTELE~4	5156	10672	10672	?
INDEX	ZZTELE	ZZTELE~2	513	12384	12384	?

The values for TREENINDEXSIZE, TREELEAVESIZE and LOBSIZE are entered in KB.

For tables, ENTRYCOUNT shows the number of records per table. For indexes, ENTRYCOUNT shows the number of different values for the secondary key.

Optimizer restrictions



Maximum number of JOIN tables in SELECT Commands	256
Maximum number of JOIN connections	32767
Maximum number of ORDER columns	128
Maximum number of strategies	127
Maximum number of primary key columns within a strategy	20
Maximum number of OR terms	50

An overview of general restrictions can be found in the reference handbook in the *Restrictions* chapter.

Thank you!





No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, Duet, Business ByDesign, ByDesign, PartnerEdge and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned and associated logos displayed are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

The information in this document is proprietary to SAP. This document is a preliminary version and not subject to your license agreement or any other agreement with SAP. This document contains only intended strategies, developments, and functionalities of the SAP® product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. SAP assumes no responsibility for errors or omissions in this document. SAP does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. This limitation shall not apply in cases of intent or gross negligence.

The statutory liability for personal injury and defective products is not affected. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third-party Web pages nor provide any warranty whatsoever relating to third-party Web pages

Weitergabe und Vervielfältigung dieser Publikation oder von Teilen daraus sind, zu welchem Zweck und in welcher Form auch immer, ohne die ausdrückliche schriftliche Genehmigung durch SAP AG nicht gestattet. In dieser Publikation enthaltene Informationen können ohne vorherige Ankündigung geändert werden.

Einige von der SAP AG und deren Vertriebspartnern vertriebene Softwareprodukte können Softwarekomponenten umfassen, die Eigentum anderer Softwarehersteller sind.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, Duet, Business ByDesign, ByDesign, PartnerEdge und andere in diesem Dokument erwähnte SAP-Produkte und Services sowie die dazugehörigen Logos sind Marken oder eingetragene Marken der SAP AG in Deutschland und in mehreren anderen Ländern weltweit. Alle anderen in diesem Dokument erwähnten Namen von Produkten und Services sowie die damit verbundenen Firmenlogos sind Marken der jeweiligen Unternehmen. Die Angaben im Text sind unverbindlich und dienen lediglich zu Informationszwecken. Produkte können länderspezifische Unterschiede aufweisen.

Die in diesem Dokument enthaltenen Informationen sind Eigentum von SAP. Dieses Dokument ist eine Vorabversion und unterliegt nicht Ihrer Lizenzvereinbarung oder einer anderen Vereinbarung mit SAP. Dieses Dokument enthält nur vorgesehene Strategien, Entwicklungen und Funktionen des SAP®-Produkts und ist für SAP nicht bindend, einen bestimmten Geschäftsweg, eine Produktstrategie bzw. -entwicklung einzuschlagen. SAP übernimmt keine Verantwortung für Fehler oder Auslassungen in diesen Materialien. SAP garantiert nicht die Richtigkeit oder Vollständigkeit der Informationen, Texte, Grafiken, Links oder anderer in diesen Materialien enthaltenen Elemente. Diese Publikation wird ohne jegliche Gewähr, weder ausdrücklich noch stillschweigend, bereitgestellt. Dies gilt u. a., aber nicht ausschließlich, hinsichtlich der Gewährleistung der Marktgängigkeit und der Eignung für einen bestimmten Zweck sowie für die Gewährleistung der Nichtverletzung geltenden Rechts.

SAP übernimmt keine Haftung für Schäden jeglicher Art, einschließlich und ohne Einschränkung für direkte, spezielle, indirekte oder Folgeschäden im Zusammenhang mit der Verwendung dieser Unterlagen. Diese Einschränkung gilt nicht bei Vorsatz oder grober Fahrlässigkeit.

Die gesetzliche Haftung bei Personenschäden oder die Produkthaftung bleibt unberührt. Die Informationen, auf die Sie möglicherweise über die in diesem Material enthaltenen Hotlinks zugreifen, unterliegen nicht dem Einfluss von SAP, und SAP unterstützt nicht die Nutzung von Internetseiten Dritter durch Sie und gibt keinerlei Gewährleistungen oder Zusagen über Internetseiten Dritter ab.

Alle Rechte vorbehalten.