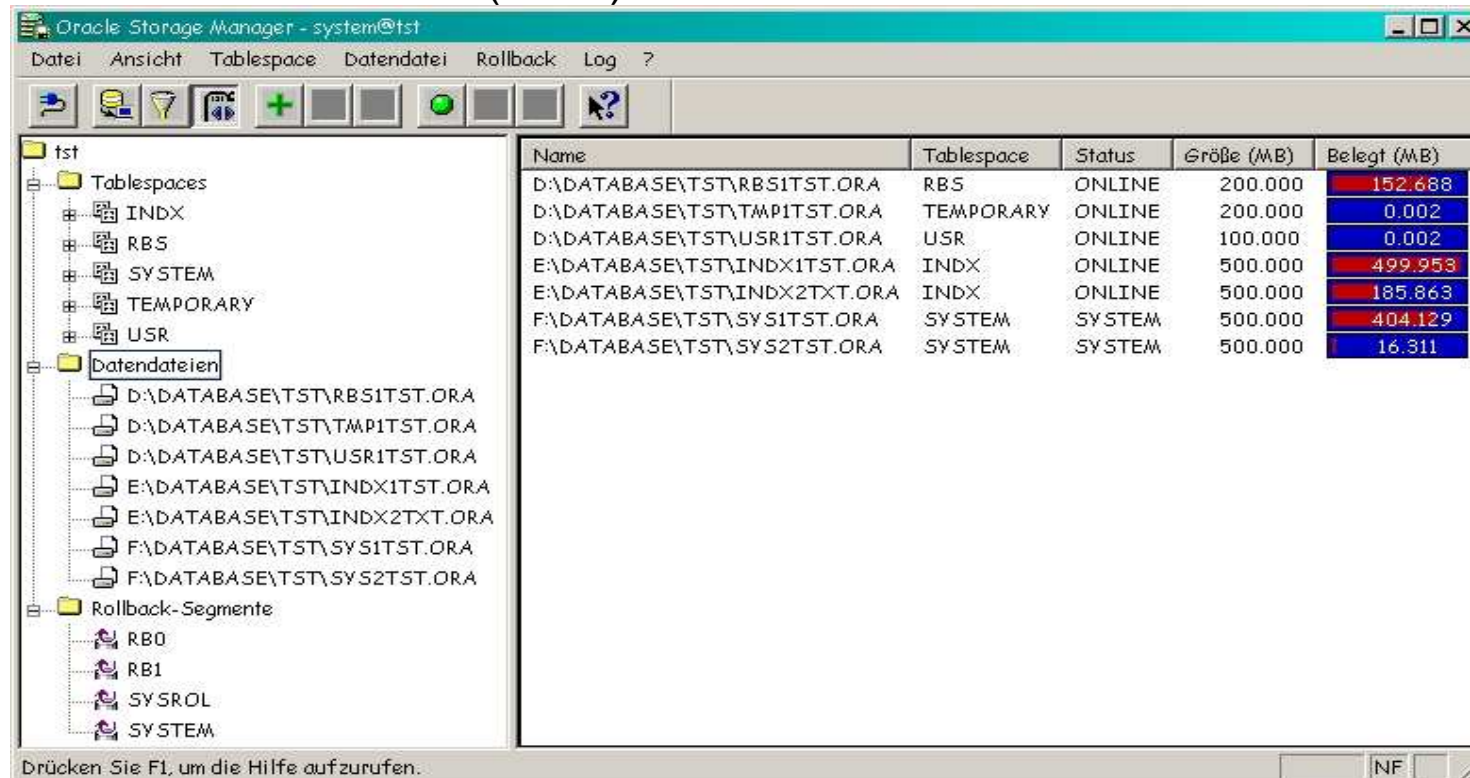


Eine empirische Untersuchung (Studienarbeit)

Windows NT auf Dual-Pentium 200 MMX mit 192 MB RAM, Oracle 8.0.5 für Windows NT

Das System bestand aus 4 physikalischen Festplatten:

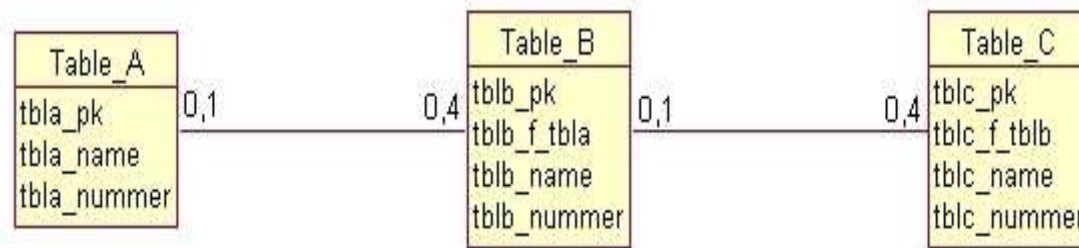
- Lw. C: Betriebssystem und DBMS
- Lw. D: Dateien für Rollback- (200 MB), Temp- (200 MB) und Usersystem (100 MB)
- Lw. E: Dateien für Index (1 GB)
- Lw. F: Dateien für Daten (1 GB)



The screenshot shows the Oracle Storage Manager interface. On the left, a tree view displays the database structure under 'tst', including tablespaces (INDX, RBS, SYSTEM, TEMPORARY, USR), datafiles, and rollback segments. The main pane on the right displays a table of database files with their locations, tablespaces, statuses, sizes, and usage.

Name	Tablespace	Status	Größe (MB)	Belegt (MB)
D:\DATABASE\TST\RBS1TST.ORA	RBS	ONLINE	200.000	152.688
D:\DATABASE\TST\TMP1TST.ORA	TEMPORARY	ONLINE	200.000	0.002
D:\DATABASE\TST\USR1TST.ORA	USR	ONLINE	100.000	0.002
E:\DATABASE\TST\INDX1TST.ORA	INDX	ONLINE	500.000	499.953
E:\DATABASE\TST\INDX2TXT.ORA	INDX	ONLINE	500.000	185.863
F:\DATABASE\TST\SYS1TST.ORA	SYSTEM	SYSTEM	500.000	404.129
F:\DATABASE\TST\SYS2TST.ORA	SYSTEM	SYSTEM	500.000	16.311

Testdaten



Schema TST: Table_A: 1.000.000 Datensätze
 Table_B: 2.000.000 Datensätze
 Table_C: 8.000.000 Datensätze

Schema TST2: Table_A2: 100.000 Datensätze
 Table_B2: 200.000 Datensätze
 Table_C2: 800.000 Datensätze

Schema TST2: Table_A: 10 Datensätze
 Table_B: 20 Datensätze
 Table_C: 80 Datensätze

Die Tabellen Table_B und Table_C enthielten jeweils Fremdschlüssel-Constraints zu ihren abhängigen Tabellen Table_A und Table_B. Weiterhin wurden im Laufe der Untersuchung verschiedenartige Indexe auf die verschiedenen Spalten der Tabellen erzeugt.

Die Reihenfolge in FROM- und WHERE-Klausel

```
SELECT COUNT(*) FROM  TABLE_C2, TABLE_B2, TABLE_A2
WHERE  TBLC_NAME LIKE '%A' AND    TBLC_F_TBLB = TBLB_PK
      AND    TBLB_F_TBLA = TBLA_PK;
```

Statement 1: Dauer ca. 329 sec.

```
SELECT COUNT(*) FROM  TABLE_A2, TABLE_B2, TABLE_C2
WHERE  TBLA_PK = TBLB_F_TBLA  AND    TBLB_PK = TBLC_F_TBLB
      AND    TBLC_NAME LIKE '%A';
```

Statement 2: Dauer ca. 96 sec.

1. Statement

1. Es werden über Table_A und Table_B Datensätze gesucht, die der Anforderung TBLB_F_TBLA = TBLA_PK genügen ((alle Datensätze aus Table_B)
2. Es werden über Table_B und Table_C Datensätze gesucht, die der Anforderung TBLC_F_TBLB = TBLB_PK genügen (alle Datensätze aus Table_C)
3. Es werden in Table_C Datensätze gesucht, die der Anforderung TBLC_NAME LIKE '%A' genügen.

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	SORT	AGGREGATE		
2	1	1	NESTED LOOPS			
3	2	1	NESTED LOOPS			
4	3	1	TABLE ACCESS	FULL	TABLE_A2	
5	3	2	TABLE ACCESS	BY INDEX ROWID	TABLE_B2	
6	5	1	INDEX	RANGE SCAN	TBLB2_F_TBLA2	NON-UNIQUE
7	2	2	TABLE ACCESS	BY INDEX ROWID	TABLE_C2	
8	7	1	INDEX	RANGE SCAN	TBLC2_F_TBLB2	NON-UNIQUE

2. Statement

1. Es werden in Table_C Datensätze gesucht, die der Anforderung TBLC_NAME LIKE '%A' genügen (eine Einschränkung in Table_C).
2. Es werden über Table_C und Table_B Datensätze gesucht, die der Anforderung TBLB_PK = TBLC_F_TBLB genügen (eine kleine Zahl von Datensätzen in Table_B durch die Einschränkung in Table_C)
3. Es werden über Table_B und Table_A Datensätze gesucht, die der Anforderung TBLA_PK = TBLB_F_TBLA genügen (eine kleine Zahl von Datensätzen durch die Einschränkung in Table_C)

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
---	-----	---	-----	-----	-----	-----
-						
0			SELECT STATEMENT			
1	0	1	SORT	AGGREGATE		
2	1	1	NESTED LOOPS			
3	2	1	NESTED LOOPS			
4	3	1	TABLE ACCESS	FULL	TABLE_C2	
5	3	2	TABLE ACCESS	BY INDEX ROWID	TABLE_B2	
6	5	1	INDEX	UNIQUE SCAN	PK_TABLE_B2	UNIQUE
7	2	2	INDEX	UNIQUE SCAN	PK_TABLE_A2	UNIQUE

Minimieren von Tabellenzugriffen in Sub-Statements

```
SELECT * FROM TABLE_A2 WHERE TBLA_PK IN  
    ( SELECT TBLB_F_TBLA FROM TABLE_B2 WHERE TBLB_NUMMER = 1 )  
      AND TBLA_NUMMER IN  
    (SELECT TBLB_NUMMER FROM TABLE_B2 WHERE TBLB_NAME LIKE '%A' );
```

Statement 3: Dauer ca. 8 sec.

```
SELECT * FROM TABLE_A2 WHERE ( TBLA_PK, TBLA_NUMMER ) IN  
    ( SELECT TBLB_F_TBLA, TBLB_NUMMER FROM TABLE_B2  
      WHERE TBLB_NUMMER = 1 AND TBLB_NAME LIKE '%A' );
```

Statement 4: Dauer ca. 4 sec.

3. Statement

1. Es werden in Table_B Datensätze gesucht, die der Anforderung TBLB_NUMMER = 1 genügen (eine Einschränkung in Table_B).
2. Es werden in Table_A Datensätze gesucht, die der Anforderung TBLA_NUMMER IN (TBLB_NUMMER) genügen.
3. Es werden in Table_B Datensätze gesucht, die der Anforderung TBLB_NAME LIKE '%A' genügen(eine Einschränkung in Table_B).
4. Es werden in Table_A Datensätze gesucht, die der Anforderung TBLA_PK IN (TBLB_F_TBLA) genügen.

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	MERGE JOIN			
2	1	1	SORT	JOIN		
3	2	1	NESTED LOOPS			
4	3	1	VIEW			
5	4	1	SORT	UNIQUE		
6	5	1	TABLE ACCESS	BY INDEX ROWID	TABLE_B2	
7	6	1	INDEX	RANGE SCAN	TBLB2_NUMMER	NON-UNIQUE
8	3	2	TABLE ACCESS	BY INDEX ROWID	TABLE_A2	
9	8	1	INDEX	UNIQUE SCAN	PK_TABLE_A2	UNIQUE
10	1	2	SORT	JOIN		
11	10	1	VIEW			
12	11	1	SORT	UNIQUE		
13	12	1	TABLE ACCESS	FULL	TABLE_B2	

4. Statement

1. Es werden in Table_B Datensätze gesucht, die der Anforderung TBLB_NUMMER = 1 und TBLB_NAME LIKE '%A' genügen (zwei Einschränkungen in Table_B).
2. Es werden in Table_A Datensätze gesucht, die der Anforderung TBLA_PK,TBLA_NUMMER IN (TBLB_F_TBLA,TBLB_NUMMER) genügen.

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	NESTED LOOPS			
2	1	1	VIEW			
3	2	1	SORT	UNIQUE		
4	3	1	TABLE ACCESS	BY INDEX ROWID	TABLE_B2	
5	4	1	INDEX	RANGE SCAN	TBLB2_NUMMER	NON-UNIQUE
6	1	2	TABLE ACCESS	BY INDEX ROWID	TABLE_A2	
7	6	1	INDEX	UNIQUE SCAN	PK_TABLE_A2	UNIQUE

Benutzung von NOT EXISTS anstatt NOT IN

```
SELECT COUNT(*) FROM TABLE_A WHERE TBLA_PK NOT IN  
    (SELECT TBLB_F_TBLA FROM TABLE_B WHERE TBLB_NAME LIKE '%A' );
```

Statement 5: Dauer ca. 16 sec.

```
SELECT COUNT(*) FROM TABLE_A WHERE NOT EXISTS  
    ( SELECT 1 FROM TABLE_B WHERE TBLB_NAME LIKE '%A'  
      AND TBLB_F_TBLA = TBLA_PK );
```

Statement 6: Dauer ca. 1 sec.

5. Statement

1. Es werde in Table_B Datensätze gesucht, die der Anforderung `TBLB_NAME LIKE '%A'` genügen und die Fremdschlüssel als Liste zurückgegeben.
2. Es werden in Table_A Datensätze gesucht, deren Primärschlüssel nicht in der übergebenen Liste enthalten sind.

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	SORT	AGGREGATE		
2	1	1	FILTER			
3	2	1	TABLE ACCESS	FULL	TABLE_A	
4	2	2	TABLE ACCESS	FULL	TABLE_B	

6. Statement

1. Es wird für jeden Datensatz in Table_A festgestellt, ob es in Table_B einen Datensatz gibt, der der Anforderung TBLB_NAME LIKE '%A' und TBLB_F_TBLA = TBLA_PK nicht genügt.

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	SORT	AGGREGATE		
2	1	1	FILTER			
3	2	1	TABLE ACCESS	FULL	TABLE_A	
4	2	2	TABLE ACCESS	BY INDEX ROWID	TABLE_B	
5	4	1	INDEX	RANGE SCAN	TBLB_F_TBLA	NON-UNIQUE

Benutzung von Joins anstatt von EXISTS

```
SELECT COUNT(A.TBLC_PK) FROM TABLE_C2 A WHERE EXISTS  
    (SELECT 1 FROM TABLE_C2 B WHERE B.TBLC_PK = A.TBLC_PK  
        AND B.TBLC_NAME LIKE '%A' );
```

Statement 7: Dauer ca. 131 sec.

```
SELECT COUNT(A.TBLC_PK) FROM TABLE_C2 A, TABLE_C2 B  
WHERE A.TBLC_PK = B.TBLC_PK AND B.TBLC_NAME LIKE '%A';
```

Statement 8: Dauer ca. 37 sec.

7. Statement

1. Es wird für jeden Datensatz in Table_C2 (A) festgestellt, ob es in Table_C2 (B) einen Datensatz gibt, der der Anforderung TBLC_NAME LIKE '%A' und TBLC_PK (B) = TBLC_PK (A) genügt.

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
---	-----	---	-----	-----	-----	-----
-						
0			SELECT STATEMENT			
1	0	1	SORT	AGGREGATE		
2	1	1	FILTER			
3	2	1	TABLE ACCESS	FULL	TABLE_C2	
4	2	2	TABLE ACCESS	BY INDEX ROWID	TABLE_C2	
5	4	1	INDEX	UNIQUE SCAN	PK_TABLE_C2	UNIQUE

8. Statement

1. Es wird festgestellt, welche Datensätze aus Table_C2 (A) und Table_C2 (B) der Anforderung $TBLC_PK (B) = TBLC_PK (A)$ genügen.
2. Es wird festgestellt, welche Daten der Ergebnismenge der Anforderung $TBLC_NAME LIKE '%A'$ genügen.

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
---	-----	---	-----	-----	-----	-----
-						
0			SELECT STATEMENT			
1	0	1	SORT	AGGREGATE		
2	1	1	NESTED LOOPS			
3	2	1	TABLE ACCESS	FULL	TABLE_C2	
4	2	2	INDEX	UNIQUE SCAN	PK_TABLE_C2	UNIQUE

Benutzung von EXISTS anstatt von DISTINCT

```
SELECT DISTINCT A.TBLB_NAME, A.TBLB_NUMMER FROM TABLE_B A, TABLE_C  
B WHERE A.TBLB_PK = B.TBLC_F_TBLB;
```

Statement 9: Dauer ca. 7 sec.

```
SELECT A.TBLB_NAME, A.TBLB_NUMMER FROM TABLE_B A WHERE EXISTS  
( SELECT 1 FROM TABLE_C B WHERE B.TBLC_F_TBLB = A.TBLB_PK );
```

Statement 10: Dauer ca. 1 sec.

9. Statement

1. Es wird festgestellt, welche Datensätze aus Table_C (A) und Table_C (B) der Anforderung $TBLC_PK (B) = TBLC_PK (A)$ genügen.
2. Es wird nach doppelten Daten gesucht und sortiert.

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	SORT	UNIQUE		
2	1	1	NESTED LOOPS			
3	2	1	TABLE ACCESS	FULL	TABLE_C	
4	2	2	TABLE ACCESS	BY INDEX ROWID	TABLE_B	
5	4	1	INDEX	UNIQUE SCAN	PK_TABLE_B	UNIQUE

10. Statement

1. Es wird für jeden Datensatz in Table_B festgestellt, ob es in Table_C einen Datensatz gibt, der der Anforderung `TBLC_F_TBLB = TBLB_PK` genügt.

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
---	-----	---	-----	-----	-----	-----
-						
0			SELECT STATEMENT			
1	0	1	FILTER			
2	1	1	TABLE ACCESS	FULL	TABLE_B	
3	1	2	INDEX	RANGE SCAN	TBLC_F_TBLB	NON-UNIQUE

SQL-Statement ohne/mit Benutzung eines Indexes

```
SELECT COUNT(*) FROM TABLE_A2 WHERE TBLA_NUMMER = 1;
```

Statement 11: Dauer ohne Index ca. 2 sec.
 Dauer mit Index ca. 1 sec.

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	SORT	AGGREGATE		
2	1	1	TABLE ACCESS	FULL	TABLE_A2	

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	SORT	AGGREGATE		
2	1	1	INDEX	RANGE SCAN	TBLA2_NUMMER	NON-UNIQUE

Berechnungen mit Indizes

```
SELECT * FROM TABLE_A2 WHERE TBLA_NUMMER + 1 = 2;
```

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	TABLE ACCESS	FULL	TABLE_A2	

```
SELECT * FROM TABLE_A2 WHERE TBLA_NUMMER = 2 - 1;
```

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	TABLE ACCESS	BY INDEX ROWID	TABLE_A2	
2	1	1	INDEX	RANGE SCAN	TBLA2_NUMMER	NON-UNIQUE

Benutzung mehrerer Indizes und deren Abschaltung

```
SELECT * FROM TABLE_A2 WHERE TBLA_NAME = 'ABC'
                        AND  TBLA_NUMMER = 1;
```

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	TABLE ACCESS	BY INDEX ROWID	TABLE_A2	
2	1	1	AND-EQUAL			
3	2	1	INDEX	RANGE SCAN	TBLA2_NAME	NON-UNIQUE
4	2	2	INDEX	RANGE SCAN	TBLA2_NUMMER	NON-UNIQUE

```
SELECT * FROM TABLE_A2 WHERE TBLA_NAME = 'ABC'
                        AND  TBLA_NUMMER + 0 = 1;
```

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	TABLE ACCESS	BY INDEX ROWID	TABLE_A2	
2	1	1	INDEX	RANGE SCAN	TBLA2_NAME	NON-UNIQUE

Benutzung eines Unique-Index

```
// Unique-Index auf TBLA_NAME
```

```
SELECT * FROM TABLE_A2 WHERE TBLA_NAME = 'ABC'  
AND TBLA_NUMMER = 1;
```

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0			SELECT STATEMENT			
1	0	1	TABLE ACCESS	BY INDEX ROWID	TABLE_C2	
2	1	1	INDEX	UNIQUE SCAN	TBLC2_NAME	UNIQUE

Zusätzliche Einschränkungen für Indizes

```
// Index auf TBLB_NUMMER  
SELECT COUNT(*) FROM TST.TABLE_B WHERE TBLB_NUMMER IN  
  (SELECT TBLA_NUMMER FROM TST.TABLE_A );
```

Statement 17: Dauer ca. 49 sec.

ID	PARENT_ID	POS	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE
0		23791	SELECT STATEMENT			
1	0	1	SORT	AGGREGATE		
2	1	1	MERGE JOIN			
3	2	1	INDEX	FULL SCAN	TBLB_NUMMER	NON-UNIQUE
4	2	2	SORT	JOIN		
5	4	1	VIEW			
6	5	1	SORT	UNIQUE		
7	6	1	TABLE ACCESS	FULL	TABLE_A	