

13

Managing Indexes

Objectives

- **Listing the different types of indexes and their uses**
- **Creating B-tree and Bitmap indexes**
- **Reorganizing indexes**
- **Dropping indexes**
- **Getting index information from the data dictionary**

Classification of Indexes

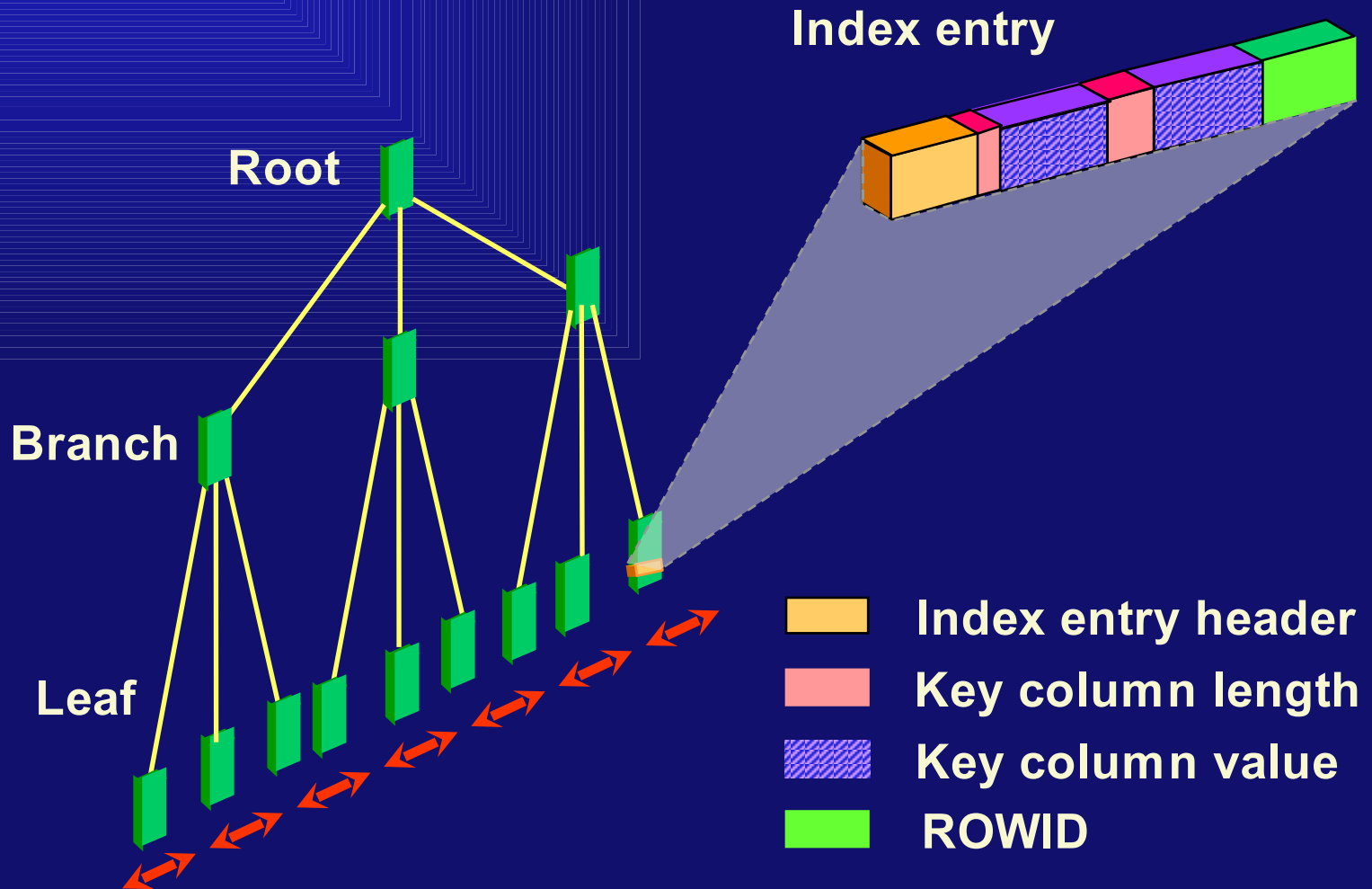
Logical

- **Single column or concatenated**
- **Unique or nonunique**

Physical

- **Partitioned or nonpartitioned**
- **B-tree or bitmap**
 - **Normal or reverse key (B-tree only)**

B-Tree Index



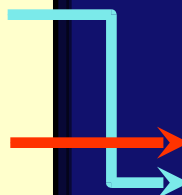
Reverse Key Index

Index on EMP (EMPNO)

KEY	ROWID
EMPNO	(BLOCK# ROW# FILE#)
1257	0000000F.0002.0001
2877	0000000F.0006.0001
4567	0000000F.0004.0001
6657	0000000F.0003.0001
8967	0000000F.0005.0001
9637	0000000F.0001.0001
9947	0000000F.0000.0001
...	...
...	...

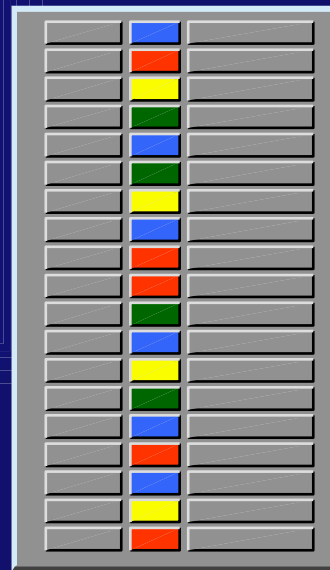
EMP table

EMPNO	ENAME	JOB	...
7499	ALLEN	SALESMAN	
7369	SMITH	CLERK	
7521	WARD	SALESMAN	...
7566	JONES	MANAGER	
7654	MARTIN	SALESMAN	
7698	BLAKE	MANAGER	
7782	CLARK	MANAGER	
...
...



Bitmap Index

Table



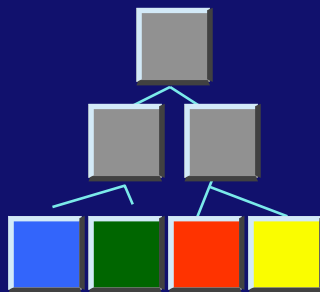
File 3

Block 10

Block 11

Block 12

Index



key	start ROWID	end ROWID	bitmap
<Blue, 10.0.3, 12.8.3, 1000100100010010100>	10.0.3	12.8.3	1000100100010010100
<Green, 10.0.3, 12.8.3, 0001010000100100000>	10.0.3	12.8.3	0001010000100100000
<Red, 10.0.3, 12.8.3, 0100000011000001001>	10.0.3	12.8.3	0100000011000001001
<Yellow, 10.0.3, 12.8.3, 0010001000001000010>	10.0.3	12.8.3	0010001000001000010

Comparing B-Tree and Bitmap Indexes

B-tree	Bitmap
Suitable for high-cardinality columns	Suitable for low-cardinality columns
Updates on keys relatively inexpensive	Updates to key columns very expensive
Inefficient for queries using OR predicates	Efficient for queries using OR predicates
Useful for OLTP	Useful for DSS

Creating Normal B-Tree Indexes

```
CREATE INDEX scott.emp_lname_idx
ON scott.employees(last_name)
PCTFREE 30
STORAGE (INITIAL 200K NEXT 200K
PCTINCREASE 0 MAXEXTENTS 50)
TABLESPACE indx01;
```


Creating Indexes: Guidelines

- **Balance query and DML needs**
- **Place in separate tablespace**
- **Use uniform extent sizes: multiples of five blocks or MINIMUM EXTENT size for tablespace**
- **Consider NOLOGGING for large indexes**
- **Set high PCTFREE if new key values are likely to be within the current range**

Creating Reverse Key Indexes

```
CREATE UNIQUE INDEX scott.ord_ord_no_idx
ON scott.ord(ord_no) REVERSE
PCTFREE 30
STORAGE (INITIAL 200K NEXT 200K
PCTINCREASE 0 MAXEXTENTS 50)
TABLESPACE indx01;
```

Creating Bitmap Indexes

```
CREATE BITMAP INDEX scott.ord_region_id_idx  
ON scott.ord(region_id)  
PCTFREE 30  
STORAGE (INITIAL 200K NEXT 200K  
PCTINCREASE 0 MAXEXTENTS 50)  
TABLESPACE indx01;
```

Changing Storage Parameters for Indexes

```
ALTER INDEX scott.emp_lname_idx  
STORAGE (NEXT 400K  
MAXEXTENTS 100) ;
```

Allocating and Deallocating Index Space

```
ALTER INDEX scott.ord_region_id_idx  
ALLOCATE EXTENT (SIZE 200K  
DATAFILE '/DISK6/indx01.dbf');
```

```
ALTER INDEX scott.ord_ord_no_idx  
DEALLOCATE UNUSED;
```

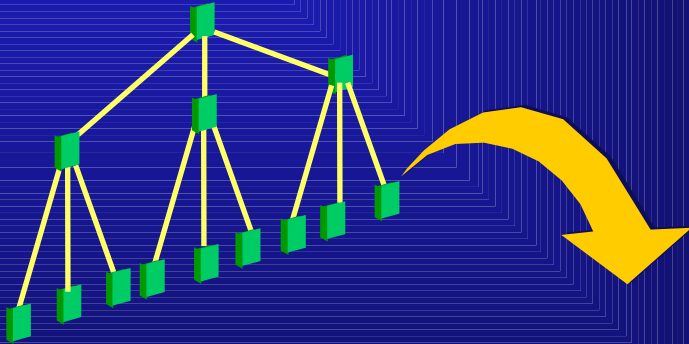
Rebuilding Indexes

Use this command to:

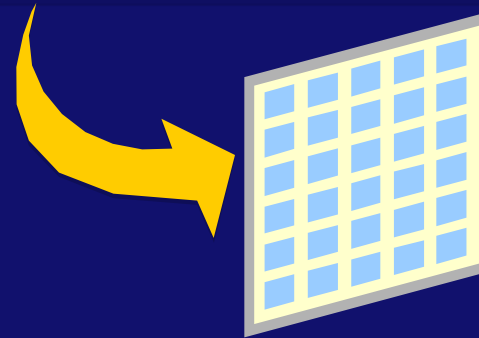
- Move an index to a different tablespace
- Improve space utilization by removing deleted entries
- Change a reverse key index to a normal B-tree index and vice versa

```
ALTER INDEX scott.ord_region_id_idx  
REBUILD  
TABLESPACE indx02;
```

Checking Index Validity



```
ANALYZE INDEX scott.ord_region_id_idx  
VALIDATE STRUCTURE;
```



INDEX_STATS

Dropping Indexes

- **Drop and re-create an index before bulk loads.**
- **Drop indexes that are infrequently needed and build them when necessary.**
- **Drop and recreate invalid indexes.**

```
DROP INDEX scott.dept_dname_idx;
```


Obtaining Index Information

DBA_INDEXES

OWNER
INDEX_NAME
INDEX_TYPE
TABLE_OWNER
TABLE_NAME
UNIQUENESS
TABLESPACE_NAME
LOGGING
STATUS

DBA_IND_COLUMNS

INDEX_OWNER
INDEX_NAME
TABLE_OWNER
TABLE_NAME
COLUMN_NAME
COLUMN_POSITION
COLUMN_LENGTH

Summary

- **Creating different types of indexes**
- **Reorganizing indexes**