4

Tuning Considerations for Different Applications

Lesson 4

ORACLE®

Objectives

After completing this lesson, you should be able to do the following:

- Use the available data access methods to tune the logical design of the database.
- Identify the demands of online transaction processing systems (OLTP).
- Identify the demands of decision support systems (DSS).
- Reconfigure systems on a temporary basis for particular needs.

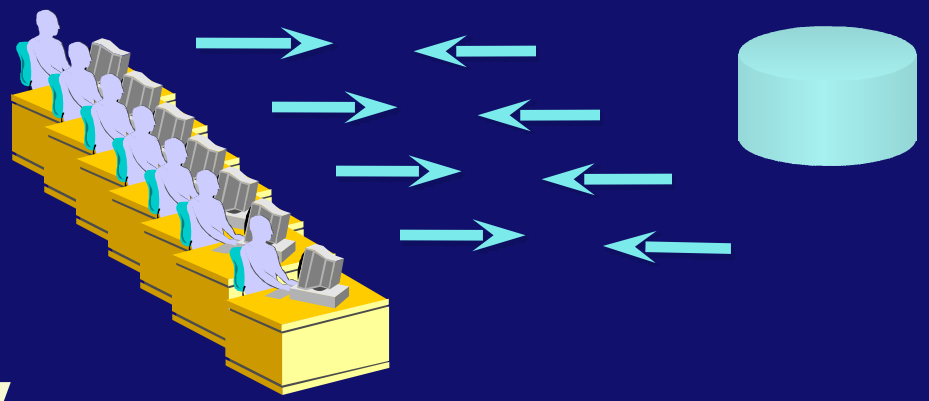
Overview

Data design and logical database structure must cater for the application type:

- online transaction processing (OLTP).
- Decision support systems (DSS).
- Multipurpose applications.

Online Transaction Processing

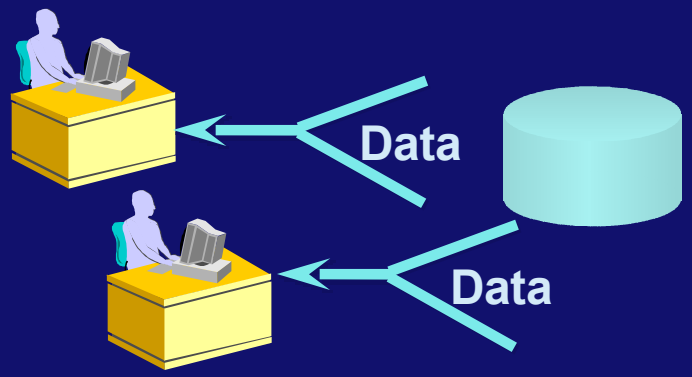
- Are high-throughput, Insert/Update intensive systems
- Contain large volumes of data that
  - Grow continuously
  - Are accessed concurrently by hundreds of users
- The tuning goals are
  - Availability
  - Speed
  - Concurrency
  - Recoverability



Decision Support Systems (DSS)

- Perform queries on large amounts of data
- Make heavy use of full table scans

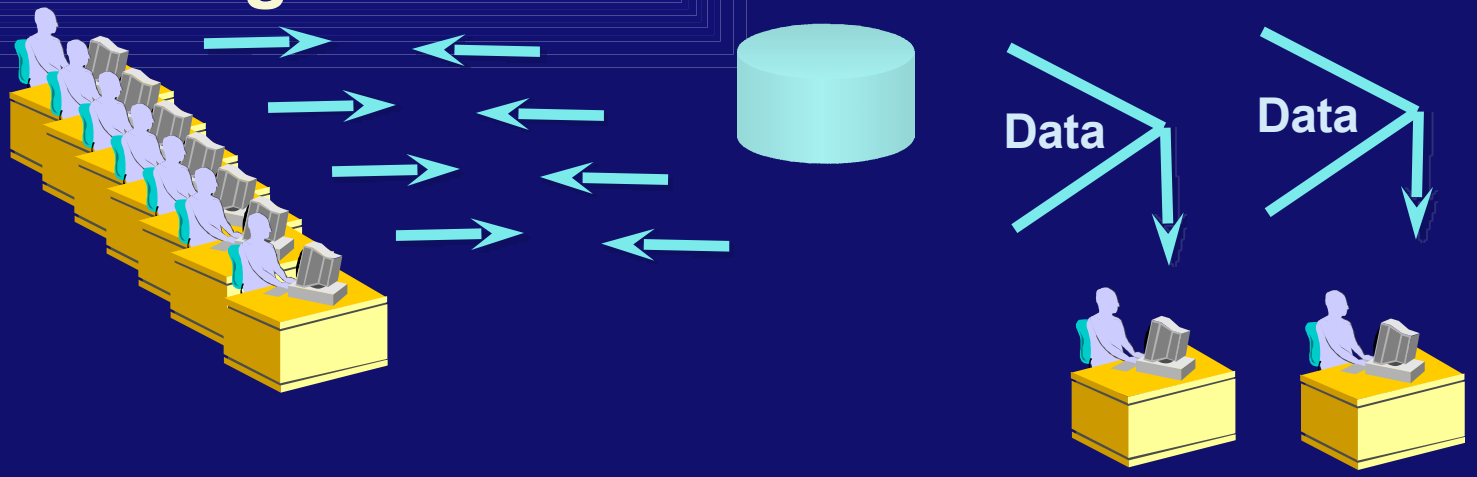
- The tuning goals are
  - Response time
  - Accuracy
  - Availability



- The Parallel Query is particularly designed for DSS.

Multipurpose Applications

- Combination of OLTP and DSS.
- Hybrid systems rely on several configurations.

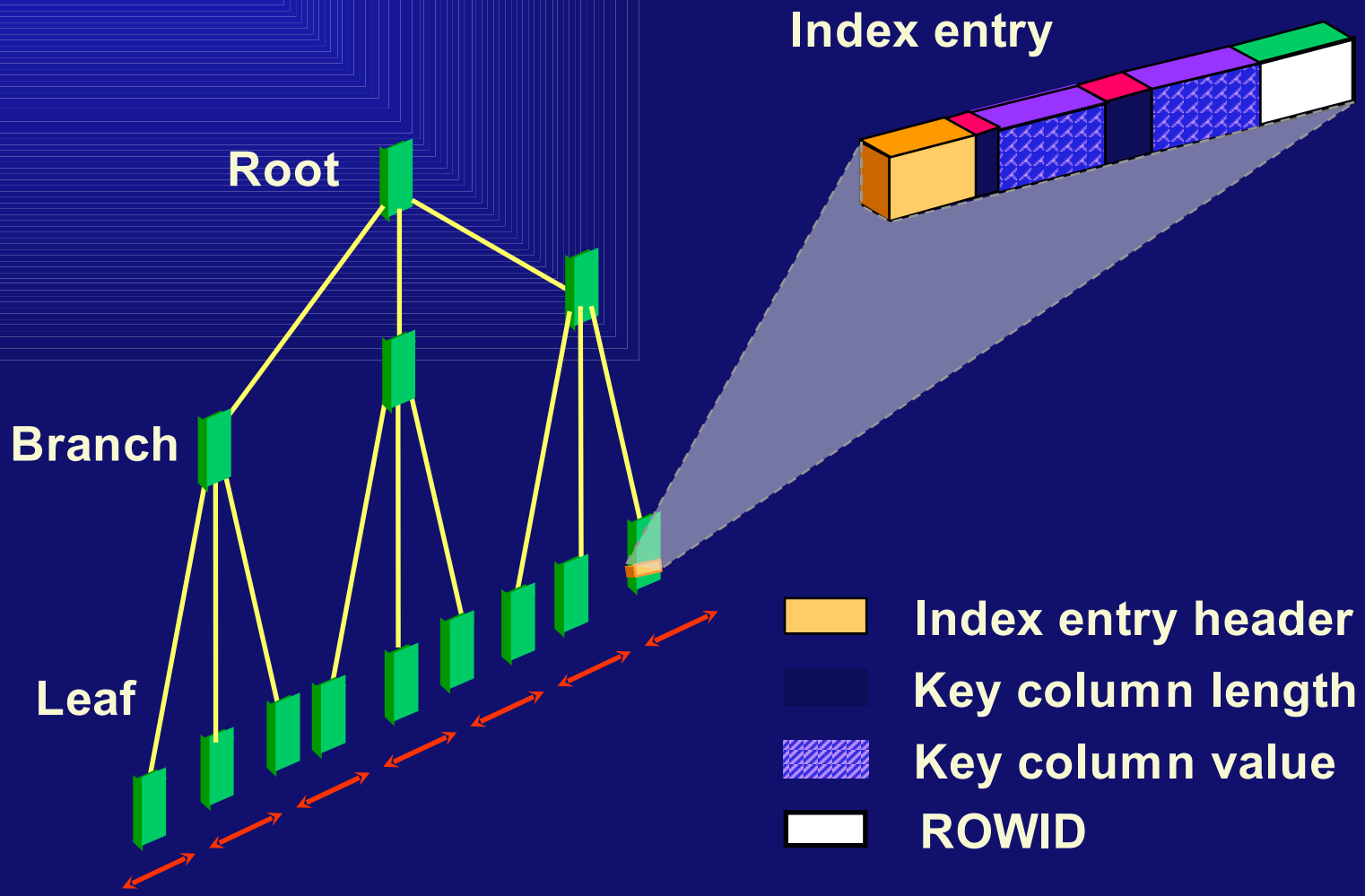


Data Access Methods

To enhance performance, you can use the following data access methods:

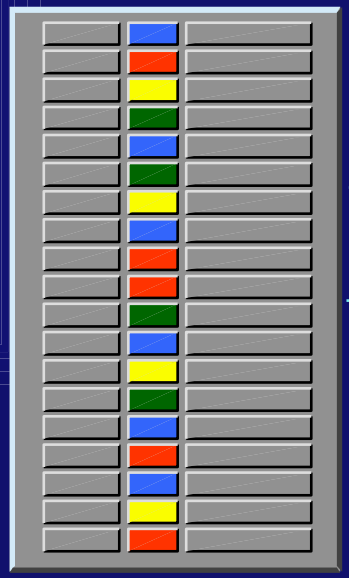
- Indexes:
  - B-tree
  - Bitmap
  - Reverse key
- Index-organized tables
- Clusters
- Histograms

B-Tree Index



Bitmap Index

Table

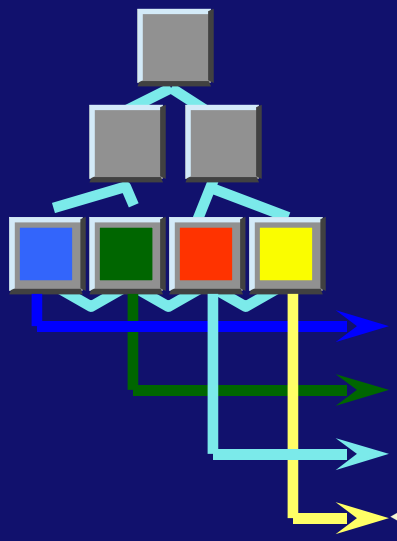


File 3
Block 10

Block 11

Block 12

Index



key	start ROWID	end ROWID	bitmap
<Blue,	10.0.3,	12.8.3,	10001001000010010100>
<Green,	10.0.3,	12.8.3,	00010100001001000000>
<Red,	10.0.3,	12.8.3,	01000000110000001001>
<Yellow,	10.0.3,	12.8.3,	00100010000010000010>

ORACLE®

Bitmap Indexes

- Used for low cardinality columns
- Good for multiple predicates
- Minimal storage space used
- Best for read-only systems
- Good for very large tables

Creating and Maintaining Bitmap Indexes

```
SQL> CREATE BITMAP INDEX scott.ord_region_id_idx
  2  ON scott.ord(region_id)
  3  PCTFREE 30
  4  STORAGE(INITIAL 200K NEXT 200K
  5  PCTINCREASE 0   MAXEXTENTS 50)
  6  TABLESPACE indx01;
```

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
Row-level locking	Bitmap segment level locking
More storage	Less storage
Useful for OLTP	Useful for DSS

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
...	...
...	...

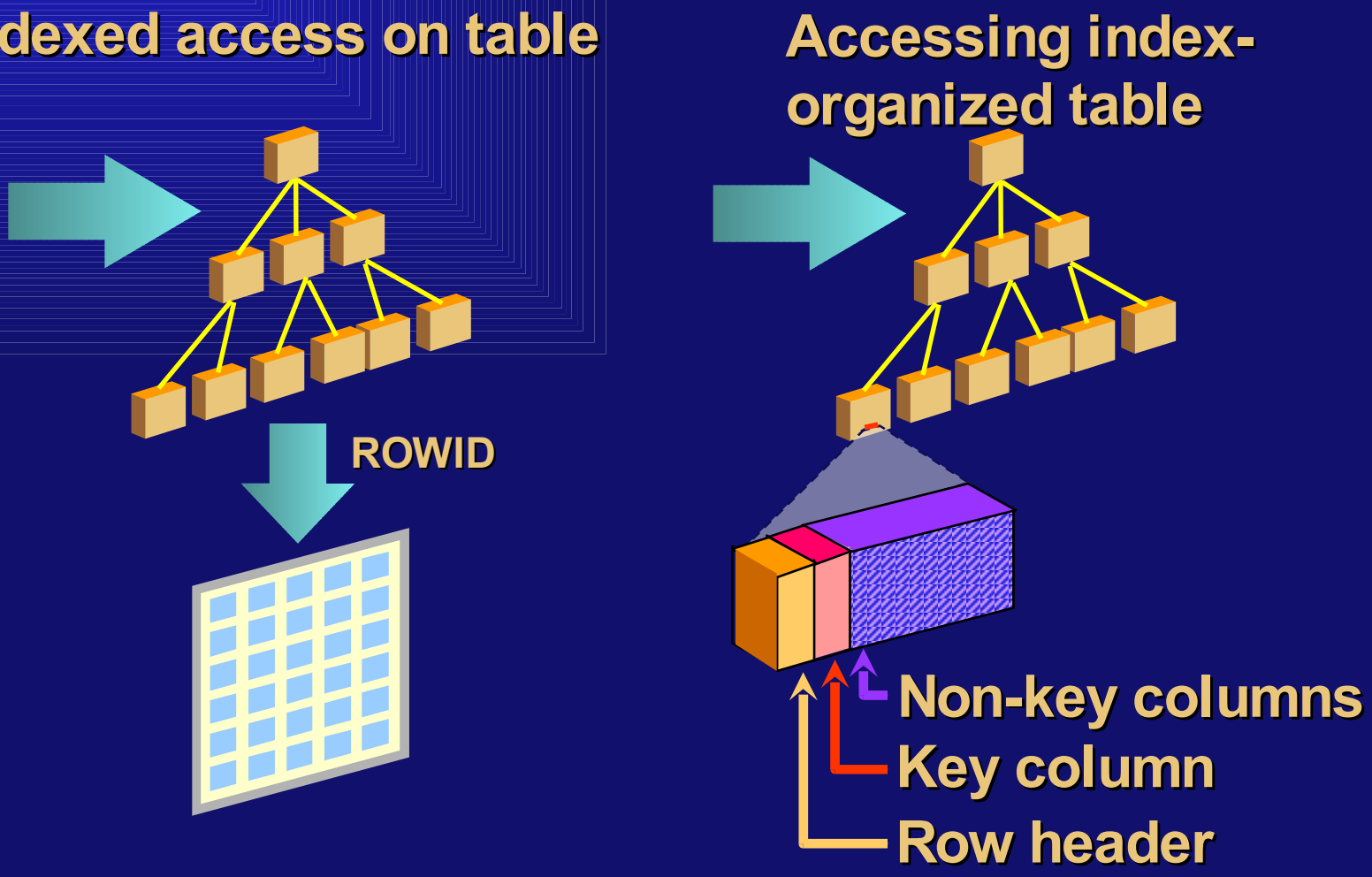
EMP table

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

Creating Reverse Key Indexes

```
SQL> CREATE UNIQUE INDEX
scott.ord_ord_no_idx
    2 ON scott.ord(ord_no) REVERSE
    3 PCTFREE 30
    4 STORAGE(INITIAL 200K NEXT 200K
    5 PCTINCREASE 0   MAXEXTENTS 50)
    6 TABLESPACE indx01;
```

Index-Organized Tables



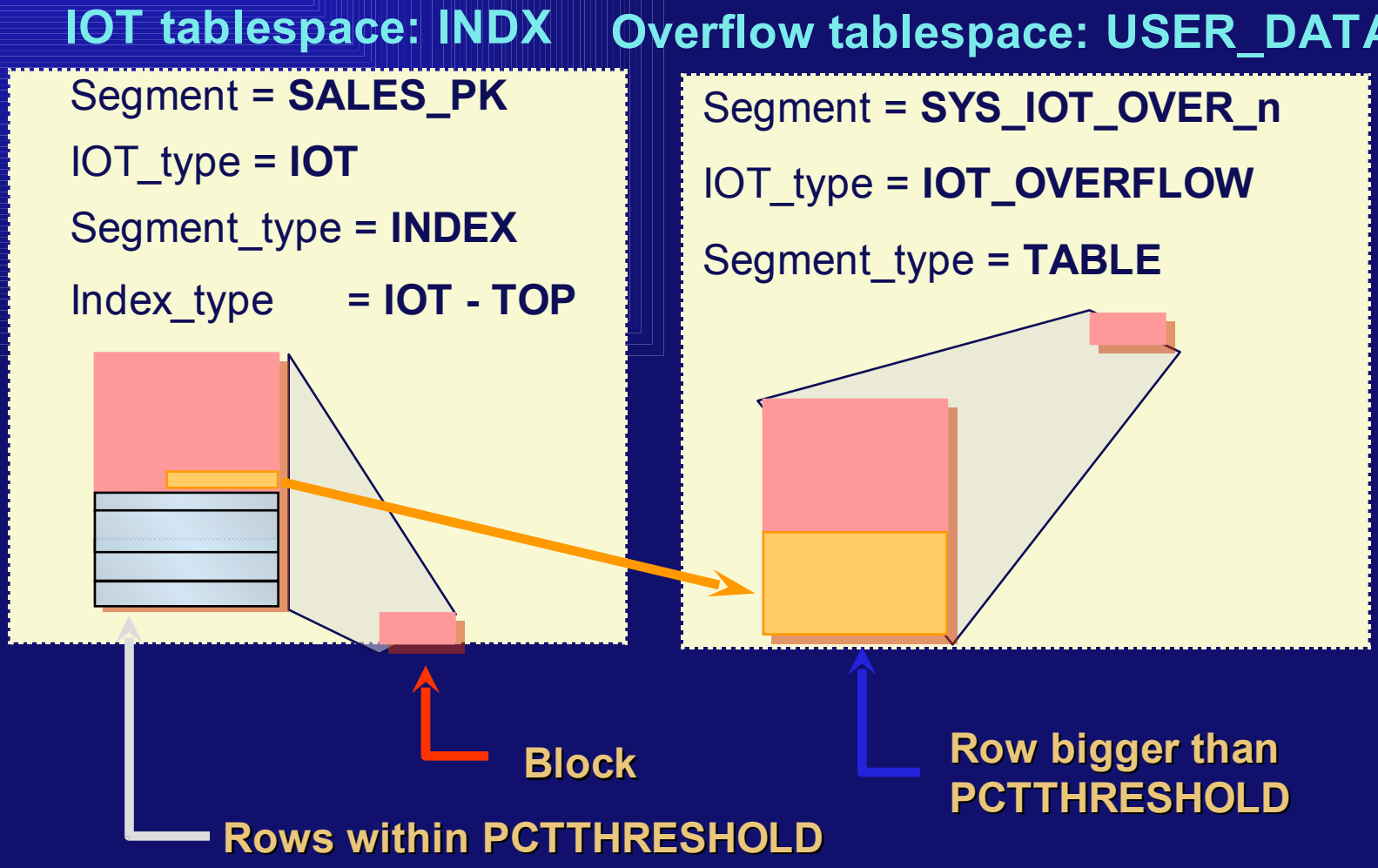
Index-Organized Tables Compared with Regular Tables

Regular Table	Index-Organized Table
Unique identifier—ROWID	Identified by PK
ROWID implicit	No ROWID
Supports several indexes	No secondary indexes
Full Table Scan returns rows in no specific order	Full Table Scan returns rows in PK order
Unique constraints allowed	No unique constraints allowed
More storage	Less storage
Distribution, replication, and partitioning supported	Distribution, replication, and partitioning not supported

Creating Index-Organized Tables

```
SQL> CREATE TABLE scott.sales
  2  ( office_cd     NUMBER(3),
  3  qtr_end         DATE,
  4  revenue         NUMBER(10,2),
  5  review          VARCHAR2(1000),
  6  CONSTRAINT sales_pk
  7  PRIMARY KEY(office_cd, qtr_end))
  8  ORGANIZATION INDEX  TABLESPACE indx
  9  PCTTHRESHOLD 20
 10  INCLUDING review
 11  OVERFLOW TABLESPACE user_data;
```

Row Overflow



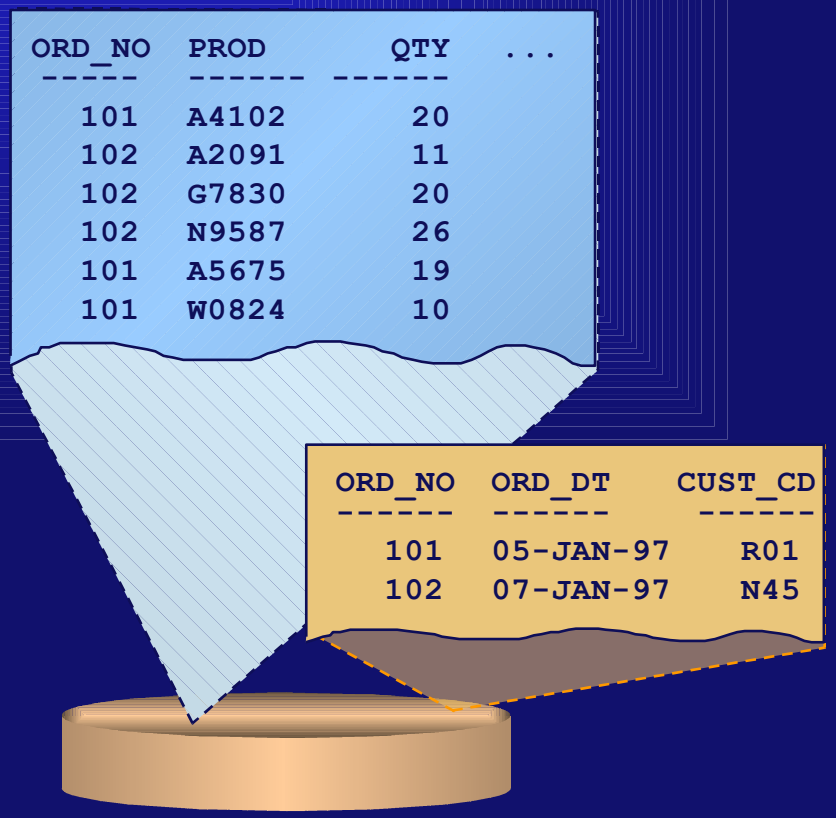
Dictionary Views

```
SQL> select table_name, tablespace_name, iot_name,
iot_type
  2  from dba_tables;
TABLE_NAME          TABLESPACE_NAME      IOT_NAME     IOT_TYPE
------------------  --------------       --------     -------------
SALES                                                    IOT
```

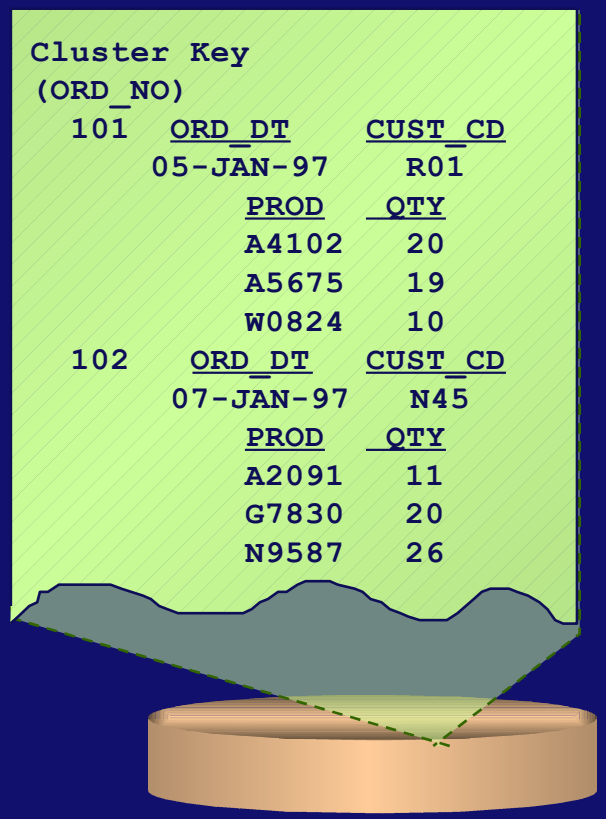
```
SQL> select
index_name,index_type,tablespace_name,table_name
  2  from dba_indexes;
INDEX_NAME                     INDEX_TYPE   TABLESPACE TABLE_NAME
------------------------------ ------------ ---------- ------------
```

```
SQL> select segment_name,tablespace_name,segment_type
  2  from dba_segments;
SEGMENT_NAME          TABLESPACE           SEGMENT_TYPE
------------------    ----------           ---------------
SYS_IOT_OVER_2268     USER_DATA            TABLE
SALES_PK              INDX                 INDEX
```

Clusters

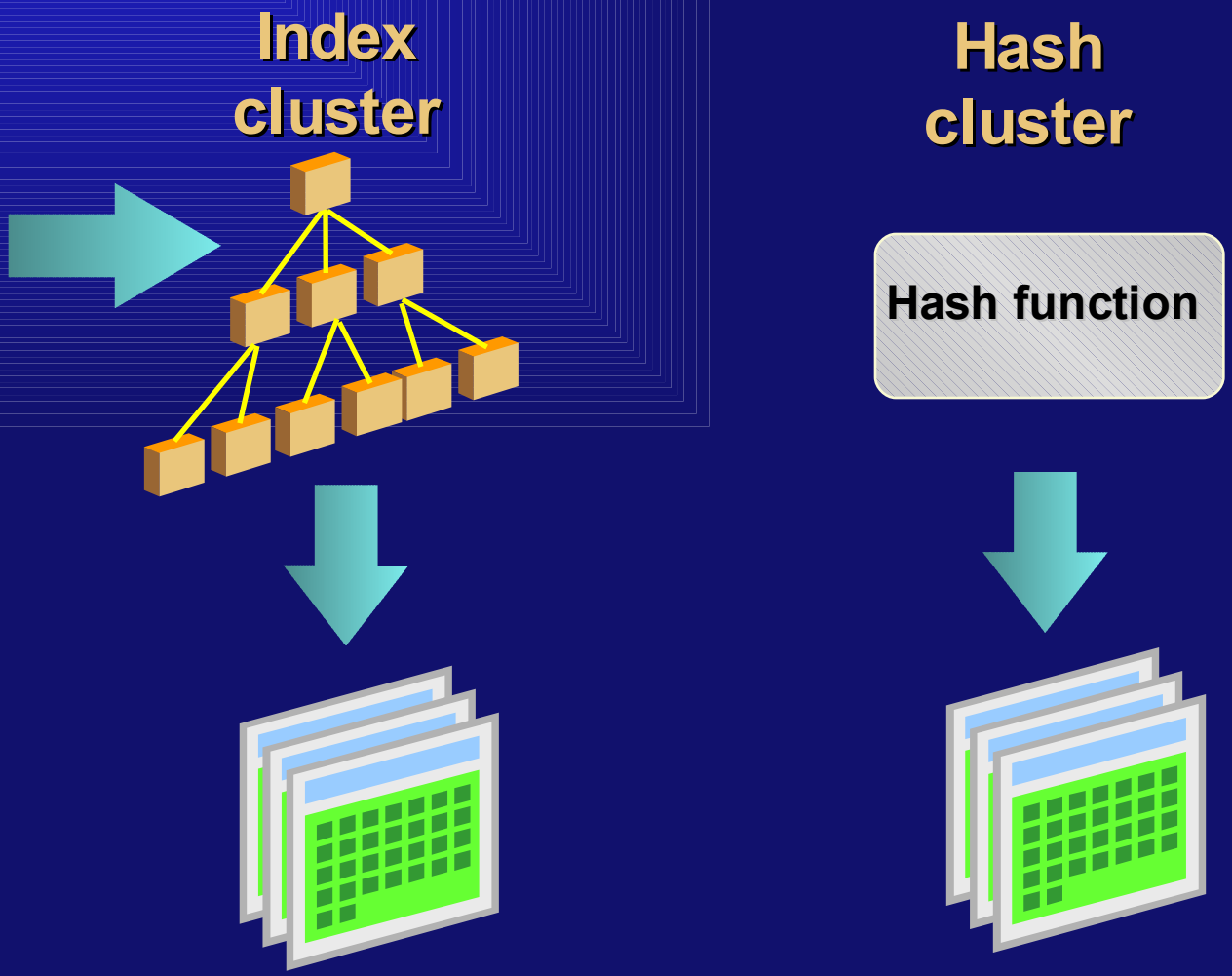


Unclustered ORD and ITEM tables



Clustered ORD and ITEM tables

Cluster Types



Situations Where Clusters Are Useful

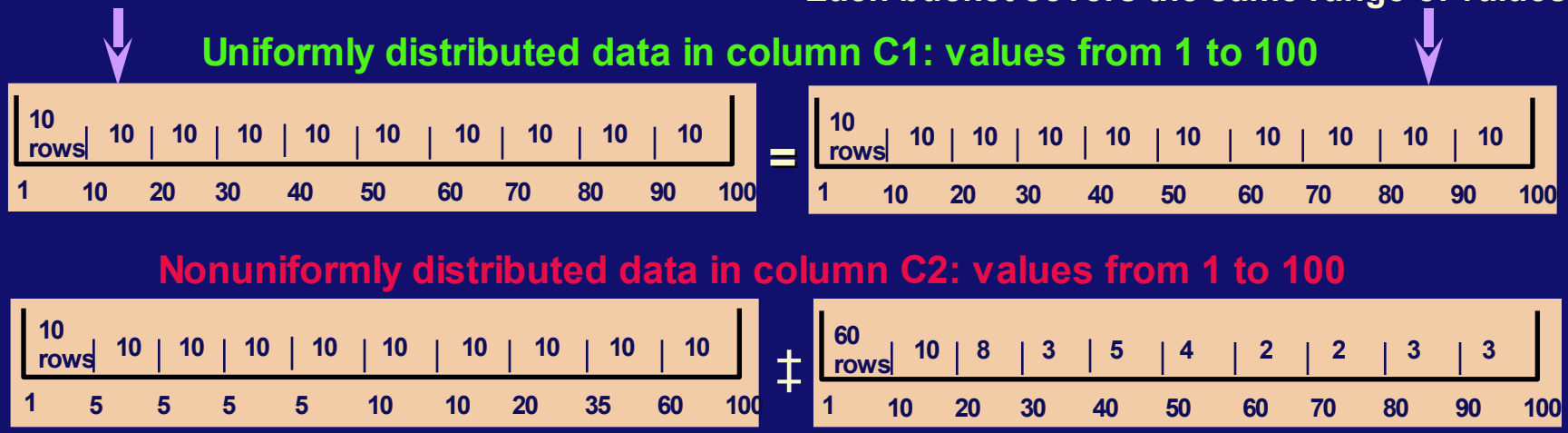
Criterion	Index	Hash
Uniform key distribution	X	X
Evenly spread key values		X
Rarely updated key	X	X
Often joined master-detail tables	X	
Predictable number of key values		X
Queries using equality predicate on key		X

Histograms

- Describe data distribution of a particular column
- Allow the Cost Based Optimizer to estimate the selectivity of a query

Height-balanced histogram:
Each bucket contains the same nb of values.

Width-balanced histogram:
Each bucket covers the same range of values .



ORACLE®

OLTP Requirements

- Space Allocation
  - Explicit space allocation
- Indexes
  - Not too many (prefer B*Tree to bitmap)
  - Reverse key for sequence columns and Parallel Server applications
  - Rebuilt regularly
- Clusters for tables in join queries
  - Index clusters for growing tables
  - Hash clusters for stable tables

OLTP Requirements: Rollback Segments

- Short transactions
  - Are unlikely to run out of rollback space
  - Need enough rollback segments to prevent contention
- Extent size can be relatively small.
- MINEXTENTS to 20 avoids extent allocation

```
SQL > CREATE ROLLBACK SEGMENT RBS01
   2  STORAGE ( INITIAL 10K   NEXT 10K
   3            MINEXTENTS 20  MAXEXTENTS 121
   4            OPTIMAL 300K )
   5  TABLESPACE RBS;
```

OLTP Application Issues

- Use constraints instead of application code.
- Make sure that code is shared.
- Use bind variables rather than literals.

DSS Requirements

Storage Allocation

- Set DB_BLOCK_SIZE to the maximum.
- Set DB_FILE_MULTIBLOCK_READ_COUNT carefully.
- Ensure that extent sizes are multiples of this number.
- Run ANALYZE regularly.

DSS Requirements

- Indexing
  - Evaluate the need for indexes.
  - Use bitmap indexes when possible.
  - Use index-organized tables for large data retrieval by PK.
  - Generate histograms for data indexes that are distributed nonuniformly.
- Clustering
  - Hash clusters for performance access.
- Partitioning

DSS Application Issues

- Parse time is less important.
- Execution plan must be optimal.
  - Use Parallel Query feature.
  - Tune carefully, using hints if appropriate.
  - Test on realistic amounts of data.
  - Consider using PL/SQL functions to code logic into queries.
- Bind variables are problematic.

Hybrid Systems

OLTP	DSS
Needs more indexes	Performs more full table scans
B*Tree indexes	Bitmap indexes
Reverse key indexes	IOT tables.
Needs more, smaller rollback segments	Needs fewer, larger rollback segments
Should not be using parallel query	Employs parallel query for large operations.
Index clusters	Hash Clusters, Partitions
PCTFREE according to expected update activity	PCTFREE can be set to 0, if re-creation of the updated table
Shared code with bind variables	Literal variables and Hints
ANALYZE indexes	Histograms generation

Parameters for Hybrid Systems

- Memory use:
  - SHARED_POOL_SIZE
  - DB_BLOCK_BUFFERS
  - SORT_AREA_SIZE
- Parallel query:
  - Reconfigure parameters for DSS.

Hybrid Systems Configuration

- Rollback segments:
  - More small rollback segments during the day
  - Fewer, large rollback segments at night
- Multi-threaded server:
  - For peak-time use, not for DSS

Summary

OLTP

- Immediate access to small amounts of data (indexing, hashing)
- Immediate concurrent access to transaction tables
- Shared code to cut down parse time
- No space allocation during peak hours

Summary

DSS

- Data tightly packed into large blocks
- Careful tuning of queries
- Histograms generation
- Less concern about parse time
- Well configured parallel query support