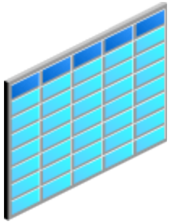
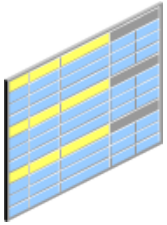
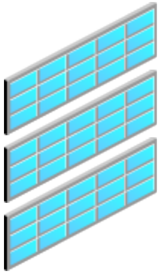
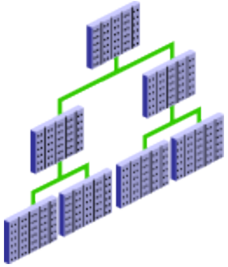
Table Types





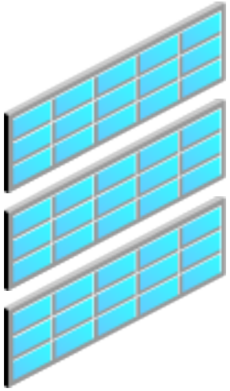
•Type	•Description
•Ordinary (heap-organized) table	•Data is stored as an unordered collection (heap).
•Partitioned table	•Data is divided into smaller, more manageable pieces.
•Index-organized table (IOT)	•Data (including non-key values) is sorted and stored in a B-tree index structure.
•Clustered table	•Related data from more than one table are stored together.





What Is a Partition and Why Use It?

- A partition is:
  - A piece of a “very large” table or index
  - Stored in its own segment
  - Used for improved performance and manageability



RANGE PARTITION

```
CREATE TABLE eladasok ( szla_szam   NUMBER(5),
                        szla_nev    CHAR(30),
                        mennyiseg   NUMBER(6),
                        het         INTEGER )
PARTITION BY RANGE ( het )
  (PARTITION negyedev1  VALUES LESS THAN  ( 13 )
      TABLESPACE users,
   PARTITION negyedev2  VALUES LESS THAN  ( 26 )
      TABLESPACE example,
   PARTITION negyedev3  VALUES LESS THAN  ( 39 )
      TABLESPACE users )
```

DBA_PART_TABLES

DBA_TAB_PARTITIONS

DBA_TAB_SUBPARTITIONS

HASH PARTITION, LIST PARTITION

```
CREATE TABLE eladasok2 (szla_szam   NUMBER(5),
                        szla_nev    CHAR(30),
                        mennyiseg   NUMBER(6),
                        het         INTEGER )
PARTITION BY HASH ( het )
   (PARTITION part1 TABLESPACE users,
    PARTITION part2 TABLESPACE example,
    PARTITION part3 TABLESPACE users );

CREATE TABLE eladasok3 (szla_szam   NUMBER(5),
                        szla_nev    CHAR(30),
                        mennyiseg   NUMBER(6),
                        het         INTEGER )
PARTITION BY LIST ( het )
   (PARTITION part1 VALUES(1,2,3,4,5)   TABLESPACE users,
    PARTITION part2 VALUES(6,7,8,9)     TABLESPACE example,
    PARTITION part3 VALUES(10,11,12,13) TABLESPACE users ) ;
```

SUBPARTITIONS (RANGE-HASH)

```
CREATE TABLE eladasok4 (szla_szam   NUMBER(5),
                        szla_nev    CHAR(30),
                        mennyiseg   NUMBER(6),
                        het         INTEGER )
PARTITION BY RANGE ( het )
SUBPARTITION BY HASH (mennyiseg)
SUBPARTITIONS 3
  (PARTITION negyedev1  VALUES LESS THAN  ( 13 )
     TABLESPACE users,
   PARTITION negyedev2  VALUES LESS THAN  ( 26 )
     TABLESPACE example,
   PARTITION negyedev3  VALUES LESS THAN  ( 39 )
     TABLESPACE users );
```

Index-Organized Tables

- Regular table access

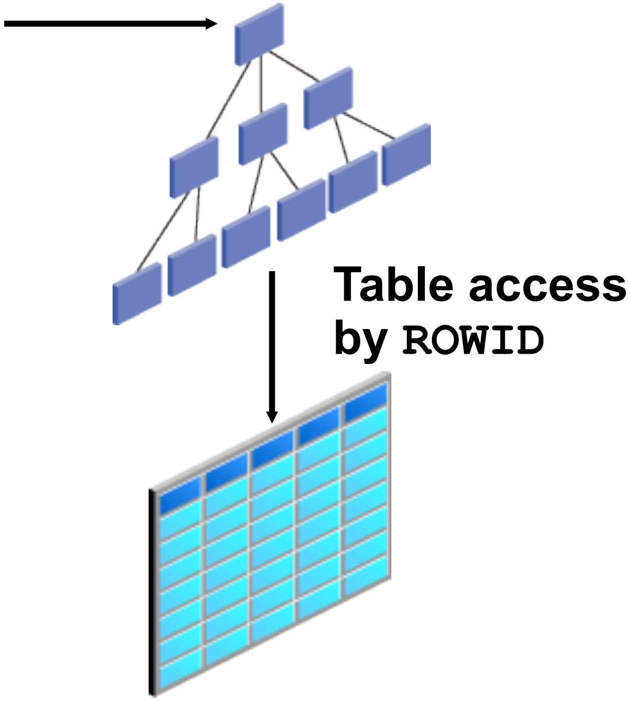
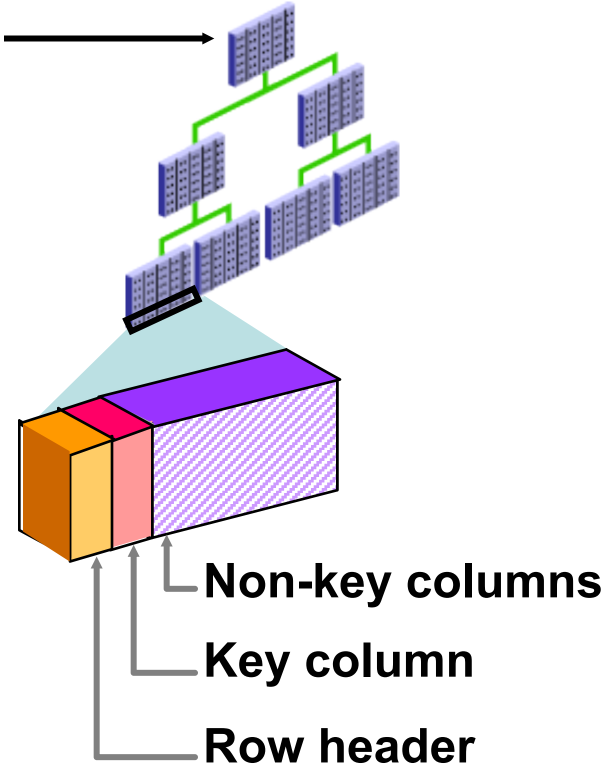


Table access by `ROWID`

IOT access



Non-key columns

Key column

Row header

Index-Organized Tables and Heap Tables

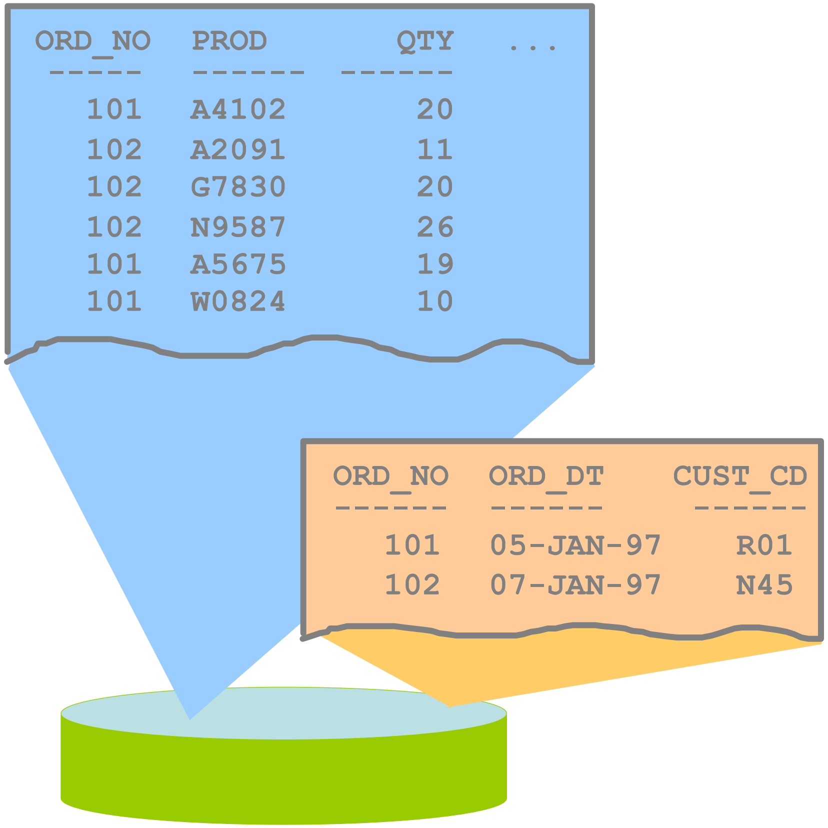
- Compared to heap tables, IOTs:
  - Have faster key-based access to table data
  - Do not duplicate the storage of primary key values
  - Require less storage
  - Use secondary indexes and logical row IDs
  - Have higher availability because table reorganization does not invalidate secondary indexes

Index-Organized Tables

```
CREATE TABLE cikk_iot
  ( ckod integer,
    cnev varchar2(20),
    szin varchar2(15),
    suly float,
       CONSTRAINT cikk_iot_pk PRIMARY KEY (ckod)  )
ORGANIZATION INDEX
PCTTHRESHOLD 20 INCLUDING cnev
OVERFLOW TABLESPACE users;
```

DBA_INDEXES index_type → ‘IOT-TOP’ table_name → ‘CIKK_IOT’
DBA_TABLES.IOT_TYPE → ‘IOT’ or ‘IOT_OVERFLOW’
DBA_TABLES.IOT_NAME → ‘CIKK_IOT’ for overflow segment

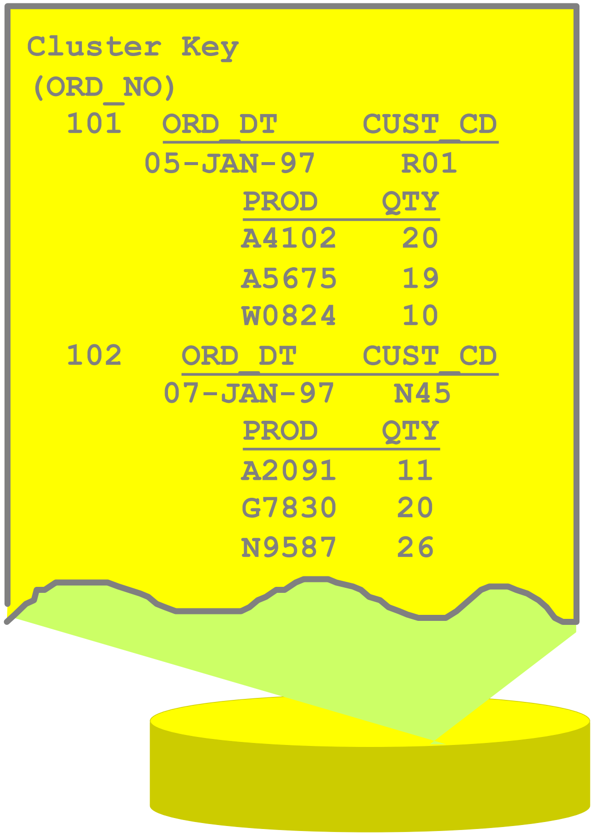
Clusters



```
ORD_NO   PROD        QTY   ...
-----   ------    ------
  101   A4102         20
  102   A2091         11
  102   G7830         20
  102   N9587         26
  101   A5675         19
  101   W0824         10
```

```
ORD_NO   ORD_DT      CUST_CD
------   -------     ------
   101   05-JAN-97      R01
   102   07-JAN-97      N45
```

Unclustered `orders` and `order_item` tables

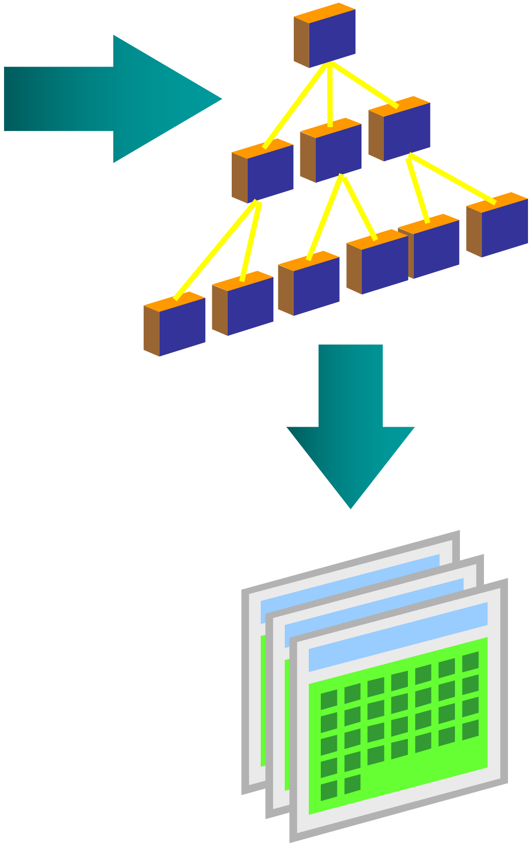


```
Cluster Key
(ORD_NO)
  101   ORD_DT        CUST_CD
       05-JAN-97        R01
              PROD      QTY
              A4102      20
              A5675      19
              W0824      10
  102    ORD_DT       CUST_CD
        07-JAN-97       N45
              PROD      QTY
              A2091     11
              G7830     20
              N9587     26
```

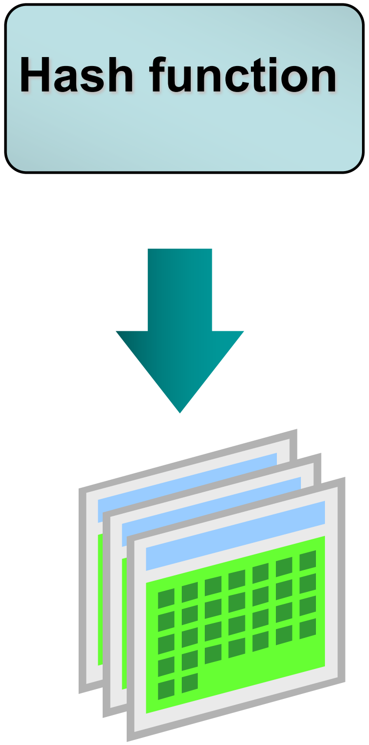
Clustered `orders` and `order_item` tables

Cluster Types

Index cluster



Hash cluster



Hash function

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

INDEX CLUSTER

```
 CREATE CLUSTER personnel
   ( department_number NUMBER(2) )   SIZE 512;
CREATE TABLE emp_cl
 ( empno NUMBER PRIMARY KEY,ename VARCHAR2(30),
   job VARCHAR2(27), mgr NUMBER(4), hiredate DATE,
   sal NUMBER(7,2), comm NUMBER(7,2),
   deptno NUMBER(2) NOT NULL)
 CLUSTER personnel (deptno);
CREATE TABLE dept_cl
   ( deptno NUMBER(2), dname VARCHAR2(9), loc VARCHAR2(9))
 CLUSTER personnel (deptno);
 CREATE INDEX idx_personnel ON CLUSTER personnel;
```

DBA_CLUSTERS

DBA_CLU_COLUMNS

DBA_TABLES.CLUSTER_NAME → ‘PERSONNEL’

HASH CLUSTER

```
CREATE CLUSTER personnel1
( department_number  NUMBER )
  SIZE 512  HASHKEYS 500
  STORAGE (INITIAL 100K  NEXT 50K);

CREATE CLUSTER personnel2
  ( home_area_code  NUMBER,  home_prefix  NUMBER )
    HASHKEYS 20
    HASH IS MOD(home_area_code + home_prefix, 101);

CREATE CLUSTER personnel3
   (deptno NUMBER)
   SIZE 512 SINGLE TABLE HASHKEYS 500;
```

```
DBA_CLUSTERS
DBA_CLU_COLUMNS
DBA_CLUSTER_HASH_EXPRESSIONS
```