

# Physical Design: Oracle

# Physical Structures

---

- Primary structures
  - Heap
  - Ordered with sparse B+-tree
  - Hash
  - Multitable clusters
- Secondary indexes:
  - dense B+-tree
  - bit-map

# CREATE TABLE

---

```
CREATE TABLE [ schema. ] table  
( column_definition [ table_constraint ] [ ,...n ] )  
[ physical_properties ]  
  [ table_partitioning_clauses ];
```

# column\_definition, table\_constraint

---

- As in SQL Server

# [ physical\_properties ]

---

```
{TABLESPACE tablespace
| ORGANIZATION
  { HEAP
    [TABLESPACE tablespace]
  | INDEX
    [TABLESPACE tablespace]
  }
| CLUSTER cluster (column [, column ]...)
}
```

# ORGANIZATION HEAP

---

- The data rows of table are stored in no particular order. This is the default.

# ORGANIZATION INDEX

---

- Use ORGANIZATION INDEX to create an index-organized table.
- Oracle Database maintains the table rows, both primary key column values and nonkey column values, in an index built on the primary key.
- Index-organized tables are therefore best suited for primary key-based access and manipulation.
- You must specify a primary key for an index-organized table, because the primary key uniquely identifies a row.

# CLUSTER

---

- A cluster is a schema object that contains data from one or more tables, all of which have one or more columns in common.
- Oracle Database stores together all the rows from all the tables that share the same cluster key.
- A cluster must be created beforehand with `CREATE CLUSTER`
- Two types of clusters:
  - Index cluster: an index is used to access the data
  - Hash cluster: a hash function is used to access the data



# CREATE CLUSTER

---

```
CREATE CLUSTER [ schema. ]cluster  
  (column datatype [ SORT ]  
    [, column datatype [ SORT ] ]...  
  ) { INDEX  
      | HASHKEYS integer  
  };
```

# INDEX Clause

---

- Specify INDEX to create an indexed cluster.
- If you specify neither INDEX nor HASHKEYS, then Oracle Database creates an indexed cluster by default.
- After you create an indexed cluster, you must create an index on the cluster key before you can issue any data manipulation language (DML) statements against a table in the cluster. This index is called the cluster index.

# HASHKEYS Clause

---

- Specify the HASHKEYS clause to create a hash cluster and specify the number of hash values for the hash cluster.
- Oracle Database rounds up the HASHKEYS value to the nearest prime number to obtain the actual number of hash values.

# Example

---

```
CREATE CLUSTER personnel  
  (department NUMBER(4));
```

```
CREATE INDEX idx_personnel ON CLUSTER personnel;
```

- Adding tables to the cluster

```
CREATE TABLE dept_10  
  CLUSTER personnel (department_id)  
  AS SELECT * FROM employees WHERE department_id = 10;
```

```
CREATE TABLE dept_20  
  CLUSTER personnel (department_id)  
  AS SELECT * FROM employees WHERE department_id = 20;
```

# [ table\_partitioning\_clauses ]

---

```
{ range_partitioning  
| hash_partitioning  
| list_partitioning  
}
```

# Range Partitioning

---

```
CREATE TABLE range_sales
  ( prod_id    NUMBER(6)
  , cust_id    NUMBER
  , time_id    DATE
  , channel_id CHAR(1)
  , promo_id   NUMBER(6)
  , quantity_sold NUMBER(3)
  , amount_sold    NUMBER(10,2)
  )
PARTITION BY RANGE (time_id)
(PARTITION SALES_Q1_1998 VALUES LESS THAN (TO_DATE('01-APR-
1998','DD-MON-YYYY')),
PARTITION SALES_Q2_1998 VALUES LESS THAN (TO_DATE('01-JUL-
1998','DD-MON-YYYY')),
PARTITION SALES_Q3_1998 VALUES LESS THAN (TO_DATE('01-OCT-
1998','DD-MON-YYYY')),
PARTITION SALES_Q4_1998 VALUES LESS THAN (TO_DATE('01-JAN-
1999','DD-MON-YYYY')),
PARTITION SALES_Q1_1999 VALUES LESS THAN (MAXVALUE));
```

# List Partitioning

---

```
CREATE TABLE list_customers
( customer_id      NUMBER(6)
, cust_first_name  VARCHAR2(20)
, cust_last_name   VARCHAR2(20)
, cust_address     CUST_ADDRESS_TYP
, nls_territory    VARCHAR2(30)
, cust_email       VARCHAR2(30))
PARTITION BY LIST (nls_territory) (
PARTITION asia VALUES ('CHINA', 'THAILAND'),
PARTITION europe VALUES ('GERMANY', 'ITALY',
'SWITZERLAND'),
PARTITION west VALUES ('AMERICA'),
PARTITION east VALUES ('INDIA'),
PARTITION rest VALUES (DEFAULT));
```

# Hash Partitioning

---

```
CREATE TABLE hash_products
  ( product_id      NUMBER(6)
    , product_name   VARCHAR2(50)
    , product_description VARCHAR2(2000)
    , category_id    NUMBER(2)
    , weight_class   NUMBER(1)
    , warranty_period INTERVAL YEAR TO MONTH
    , supplier_id    NUMBER(6)
    , product_status VARCHAR2(20)
    , list_price     NUMBER(8,2)
    , min_price      NUMBER(8,2)
    , catalog_url    VARCHAR2(50)
  )
PARTITION BY HASH (product_id)
PARTITIONS 5;
```



# Views

---

```
CREATE VIEW [ schema_name . ] view_name [  
    ( column [ ,...n ] ) ]  
AS select_statement [ ; ]  
[ WITH CHECK OPTION ]
```

# Updatable Views

---

- An updatable view is one you can use to insert, update, or delete base table rows.
- You can create a view to be inherently updatable, or you can create an INSTEAD OF trigger on any view to make it updatable.
- To learn whether and in what ways the columns of an inherently updatable view can be modified, query the USER\_UPDATABLE\_COLUMNS data dictionary view

# Updatable Views

---

- Specify `WITH CHECK OPTION` to indicate that Oracle Database prohibits any changes to the view that would produce rows that are not included in the subquery.

# Join Views

---

- A join view is one whose view subquery contains a join.
- Inherently updatable join view
  - the view must not be created WITH CHECK OPTION,
  - all columns into which values are inserted/updated must come from a key-preserved table.
- A key-preserved table is one for which every primary key or unique key value in the base table is also unique in the join view.

# Join Views

---

```
CREATE VIEW locations_view AS
  SELECT d.department_id, d.department_name,
         l.location_id, l.city
  FROM departments d, locations l
  WHERE d.location_id = l.location_id;
```

- The primary key `location_id` of the `locations` table is not unique in the `locations_view` view. Therefore, `locations` is not a key-preserved table and columns from that base table are not updatable.

# Join Views

---

```
SELECT column_name, updatable
FROM user_updatable_columns
WHERE table_name = 'LOCATIONS_VIEW';
```

COLUMN_NAME	UPD
-----	---
DEPARTMENT_ID	YES
DEPARTMENT_NAME	YES
LOCATION_ID	NO
CITY	NO

# Join Views

---

- Allowed

```
INSERT INTO locations_view (department_id,  
    department_name)  
VALUES (999, 'Entertainment');
```

# Materialized Views

---

- A materialized view physically stores in the database the results of the query

```
CREATE MATERIALIZED VIEW [ schema_name . ]  
    view_name [ ( column [ ,...n ] ) ]  
AS select_statement [ ; ]
```



# Indexes

---

- Type of indexes:
  - B+-tree indexes
  - B+-tree cluster indexes
  - Bitmap indexes
  - Bitmap join indexes:
    - Bitmap index for the join of two or more tables
    - Useful for datawarehousing

# Unique and Nonunique Indexes

---

- Unique indexes guarantee that no two rows of a table have duplicate values in the key column (or columns).
- Oracle recommends that unique indexes be created explicitly, using `CREATE UNIQUE INDEX`.

# CREATE INDEX

---

```
CREATE [ UNIQUE | BITMAP ] INDEX [ schema. ]index  
ON { cluster_index_clause  
    | table_index_clause } ;
```

- By default, Oracle Database creates B+-tree indexes.

# cluster\_index\_clause

---

CLUSTER [ schema. ] cluster

- Use the cluster\_index\_clause to identify the cluster for which a cluster index is to be created.
- You cannot create a cluster index for a hash cluster.

# table\_index\_clause

---

[ schema. ]table  
(column [ ASC | DESC ]  
[, column [ ASC | DESC ] ]...)

# Examples

---

```
CREATE BITMAP INDEX product_bm_ix  
ON products(list_price);
```