

Chapter 13 Indexes, Sequences, Views

What Objects are Present in Your Schema?

- **USER_OBJECTS**
 - a **Data Dictionary** view (Ch 14)
 - describes all objects owned by the current user
 - tables, views, sequences, indexes, synonyms, functions, procedures, triggers, packages, others

```
DESCR USER_OBJECTS

SELECT object_name, object_type,
       created, status
FROM USER_OBJECTS
ORDER BY object_type, object_name;
```

2

Index Preliminaries

- Analogy: a textbook's index
 - Where in our book is the topic GRANT?
 - How would you find GRANT if our book had no index?
 - What if the the book's last chapter was moved earlier in the book?

3

Index Concepts

- Is a named schema object
 - Can speed row retrieval by using ROWID pointers
 - **ROWID** is the unique address of each row in the database
 - the row can be **directly accessed** when the ROWID is known
- ```
SELECT ROWID, title, length FROM article;
```
- If a column is not indexed, a full **table scan** will occur
  - Is automatically used by Oracle when beneficial
  - Indexes take up space
  - Is automatically maintained by Oracle
    - when DML performed on an indexed column, Oracle must update each index
    - can slow DML operations, especially on bulk inserts, updates

4

## When Are Indexes Created?

- Automatically
  - A unique index is created automatically when you define a PRIMARY KEY or UNIQUE constraint
- Manually
  - Can create indexes on columns to speed up access to the rows
    - eg: index foreign key fields to speed join operations
    - eg: index fields that frequently appear in WHERE clauses to speed retrieval
  - Can create an index to enforce a column's uniqueness
    - Oracle recommends the unique CONSTRAINT approach

5

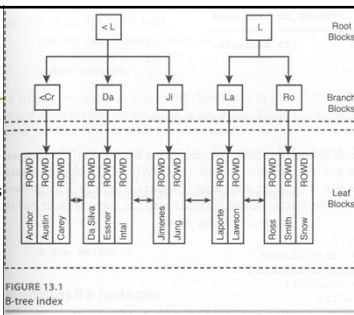
## Two Types of Indexes

- Balanced Tree (B-Tree)
  - the default type of index
  - best for columns with **high selectivity**
    - i.e., when a column has lots of distinct values (eg: articlenum)
  - best for exact match and range matches against both small and very large tables
- Bitmapped
  - best for columns with **low selectivity**
    - i.e., when a column has few distinct values (eg: freelancer, gender)

6

## B-Tree Indexes

- Structure (pg 573)
  - leaf blocks
    - one for each data value present in the column and its ROWID
  - root/branch blocks
    - enable quick searching
- How do they help speed row retrieval?
  1. seek the value in the index
  2. determine its ROWID
  3. directly read the row



```
SELECT ln, hiredate FROM emp WHERE ln = 'Essner';
SELECT ln, hiredate FROM emp WHERE ln < 'Essner';
```

7

## Creating an Index

```
CREATE [BITMAP] INDEX index
ON table (column[, column]...);
```

- You must have CREATE INDEX privilege
- Oracle must acquire a **table-level lock** before it can build the index
- Create an index for writer table's contact foreign key

```
CREATE INDEX writer_contact_idx
ON writer(contact);
```

8

## Practice Time

- Create an index for article table's length column

- Confirm the new indexes exist

```
SELECT object_name, created
FROM USER_OBJECTS
WHERE object_type = 'INDEX'
ORDER BY created;
```

- How can the index help speed row retrieval?

```
SELECT title FROM article WHERE length < 1500;
SELECT COUNT(*), MAX(length) FROM article;
```

9

## Creating a Composite Index

```
CREATE INDEX index
ON table (column[, column]...);
```

- Can include up to 32 columns per index
- Generally, the most commonly accessed or most selective columns go first
- Create an index for Grade table's composite foreign key of StudentID and SectionID

```
CREATE INDEX grade_studentidsectionid_idx
ON grade(student_id, section_id);
```

- what is the leading column of this index?
- what WHERE clauses could benefit from this index?

10

## Guidelines for When to Index

- The column is used frequently in the WHERE clause
- The column is used frequently in a join condition
  - eg: a foreign key (Oracle automatically creates an index for a pk)
- The column contains a wide range of values
- The column contains unique values
- The column contains a large number of NULLs
- The table is large and most queries are expected to retrieve less than 5-15% of its rows
  - needle in the haystack

11

## Indexes: Pros and Cons

- Pros
  - 
  -
- Cons
  - 
  -
- Which fields should you consider indexing?

12

## Rebuilding an Index

```
ALTER INDEX indexname REBUILD;
```

- Can increase index performance & storage efficiency
  - especially after much DML has occurred since index was created
    - eg: more than 10% of the table's rows have been IUD

```
ALTER INDEX article_writerid_idx REBUILD;
Query Result x Statement Output x
ALTER INDEX article_writerid_idx succeeded.
```

13

## Removing an Index

```
DROP INDEX indexname;
```

- You must be the owner or have the DROP ANY INDEX privilege
- You cannot modify an index... must drop and recreate
- When a table is dropped, Oracle automatically drops all of the table's indexes

```
DROP INDEX article_writerid_idx;
DROP INDEX grade_studentidsectionid_idx;
Statement Output x
DROP INDEX article_writerid_idx succeeded.
DROP INDEX grade_studentidsectionid_idx succeeded.
```

14

Lab 13.2

## Sequence Concepts

- Generates a series of integer values
  - either an ascending or descending set of values
- Typically used to create value for a **synthetic** primary key
- Is a sharable object
  - can be shared by different users
  - can provide values for different tables
- Sequences are independent of tables
  - can modify or drop the sequence with no effect on table
  - can modify or drop a table with no effect on the sequence
  - the same sequence can be used for multiple tables

15

## Creating a Sequence

```
CREATE SEQUENCE sequence
[INCREMENT BY n]
[START WITH n]
[{MAXVALUE n | NOMAXVALUE}]
[{MINVALUE n | NOMINVALUE}]
[{CYCLE | NOCYCLE}]
[{CACHE n | NOCACHE}];
```

- The following was in the createissue25.sql script

```
CREATE SEQUENCE articlenum_seq
INCREMENT BY 1
START WITH 1
NOCACHE;
```

16

## Creating a Sequence

- **CYCLE | NOCYCLE**
  - specifies whether the sequence continues to generate values after reaching its MAXVALUE/MINVALUE
  - if use NOCYCLE, once MAXVALUE limit is reached, no additional values from the sequence will be generated
    - you will receive an error indicating that the sequence exceeds the MAXVALUE
  - beware of using CYCLE when the sequence is used to generate values for a primary key (or unique) field
    - why?
- **CACHE [20 | n] | NOCACHE**
  - specifies how many values Oracle will pre-generate and keep in memory

17

## Practice Time

- Create a sequence named `evens_seq` that provides values from 100 to 200, skipping by 2.
- Use SELECT statements to generate the first 5 values from the `evens_seq` sequence

```
SELECT _____
FROM _____;
```

18

## NEXTVAL and CURRVAL Pseudocolumns

- **NEXTVAL**
  - generates a new sequence number and places it in CURRVAL
  - you must qualify NEXTVAL with the sequence name
- **CURRVAL**
  - obtains the last value returned to the user's own session
    - NEXTVAL must be issued before CURRVAL contains an initial value
  - you must qualify CURRVAL with the sequence name

19

## Using a Sequence

- Sequences are generally used in INSERT and UPDATE statements
- Eg: insert a new article

```
INSERT INTO article
VALUES (articlenum_seq.NEXTVAL, 'War Sucks',
 'POL', TO_DATE('01-MAY-09'),
 1894, 'J525');
```

- What articlenum was used for that new article?

```
SELECT * FROM article WHERE title = 'War Sucks';

SELECT articlenum_seq.CURRVAL FROM DUAL;
```

20

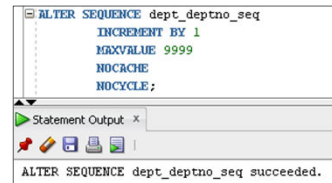
## Using a Sequence

- Gaps in sequence values can occur when:
  - a ROLLBACK occurs in a transaction that includes INSERT or UPDATE statements that used NEXTVAL
  - the Oracle server crashes
  - a sequence is used in another table
- But do gaps really matter?

21

## Modifying a Sequence

- You must be the owner or have the ALTER ANY SEQUENCE privilege
- Can change the increment, maximum value, minimum value, cycle option, or cache option
  - can't change the START value
    - the sequence must be dropped and re-created



```
ALTER SEQUENCE dept_deptno_seq
INCREMENT BY 1
MAXVALUE 9999
NOCACHE
NOCYCLE;
```

Statement Output x

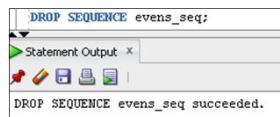
```
ALTER SEQUENCE dept_deptno_seq succeeded.
```

22

## Removing a Sequence

```
DROP SEQUENCE sequencename;
```

- You must be the owner of the sequence or have the DROP ANY SEQUENCE privilege
- Once removed, the sequence can no longer be referenced



```
DROP SEQUENCE evens_seq;
```

Statement Output x

```
DROP SEQUENCE evens_seq succeeded.
```

23

Lab 13.3

## View Concepts

- A stored SELECT statement
- Used to present subsets or combinations of data
- A **virtual** table based on a table or another view
  - the tables on which a view is based are called **base tables**
- Contains no data of its own
  - is like a window through which table data can be viewed/changed
  - can't be indexed

24

## View Example #1

### ■ Create a View to Summarize Data

```
CREATE OR REPLACE VIEW writer_activity AS
 SELECT ln || ', ' || fn author, article_count, avg_length
 FROM writer INNER JOIN
 (SELECT writerid, count(*) article_count,
 avg(length) avg_length
 FROM article
 GROUP BY writerid) stats
 ON writer.writerid = stats.writerid;
```

### ■ Use the writer\_activity View

```
DESC writer_activity
SELECT * FROM writer_activity;
SELECT author, avg_length
 FROM writer_activity WHERE article_count >= 3;
```

25

## Why Use Views?

### ■ To make complex queries easy

- a view is **prebuilt query**
- a user can use the view without knowing how to:
  - use record selection criteria
  - write complex join expressions
  - derive calculated columns
  - perform sorting
  - perform grouping

### ■ To present different views of the same data

- one view to provide a detailed look at rows, others can present summaries grouped/sorted in various ways

### ■ To restrict data access

- a view can display only those rows or columns you want it to
- privileges can be granted on the view without granting privileges on the base tables

26

## Practice Time: Using Views

```
DESC article_writer

SELECT * FROM article_writer;

SELECT title, author FROM article_writer;

SELECT text
 FROM USER_VIEWS
 WHERE view_name = 'ARTICLE_WRITER';
```

### ■ Where did this **article\_writer** view come from?

- createissue25.sql script

27

## Creating a View

```
CREATE [OR REPLACE] VIEW viewname
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraintname]]
[WITH READ ONLY];
```

### ■ OR REPLACE

- overwrites any existing view of the same name
- avoids having to drop the view, recreate it, and re-grant privileges to use it

### ■ subquery

- a complete SELECT statement
- can use aliases for the columns
- can use joins to involve multiple tables (a join view)

28

## Creating a View

```
CREATE [OR REPLACE] VIEW viewname
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraintname]]
[WITH READ ONLY];
```

- **WITH CHECK OPTION**
  - specifies that only rows that meet the view's WHERE clause can be inserted, updated, or deleted
  - although a view's WHERE clause restricts the rows selected when the view runs, it does not restrict DML unless you use WITH CHECK OPTION
- **constraintname**
  - is the name assigned to the CHECK OPTION constraint
- **WITH READ ONLY**
  - ensures that no DML operations can be performed using the view

29

## Practice Time

- The `writer` table's `amount` column is considered too sensitive. Create a view named `writer_info` that shows every column from the `writer` table except `amount`. The view must not allow inserts, updates, or deletes.
- After you've created the `writer_info` view
  - use `DESCR` to view its structure
  - use `SELECT *` to see what it produces
  - use your `writer_info` view to change `writerid` C200's amount to 300... what happens?

30

## View Example #2

```
CREATE OR REPLACE VIEW freelancers AS
SELECT * FROM writer WHERE freelancer = 'Y'
WITH CHECK OPTION CONSTRAINT freelancers_freelancer_ck;
```

```
SELECT * FROM freelancers;
```

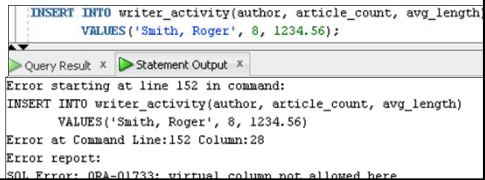
```
UPDATE freelancers
SET lastcontact = SYSDATE WHERE writerid = 'J525';
1 row updated.
```

```
INSERT INTO freelancers
VALUES('S100', 'Smith', 'Roger', '(666) 666-6666',
to_date('22-DEC-02'), 'Y', 0, NULL);
1 row created.
```

```
INSERT INTO freelancers
VALUES('J543', 'Jones', 'Jane', '(777) 777-7777',
to_date('22-DEC-02'), 'N', 0, NULL);
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

31

## DML Using Views

- DML can only involve columns present in the view's definition
    - as demonstrated in the previous slide
  - When using a view to insert/update/delete rows, the base table's constraints must be satisfied
  - For DML, a view must not contain
    - expressions (eg: `sal + 100`)
    - multi-row (aggregate) functions (eg: `SUM`, `AVG`)
    - single-row functions (eg: `RTRIM`, `LENGTH`)
    - set operators (eg: `INTERSECT`, `UNION`)
    - `DISTINCT`
    - `GROUP BY`
    - `ORDER BY`
- 

## DML with a Join View

- **Key-Preserved** base table
  - when the base table's primary key is **unique in the results of the join view**
- DML may only be performed on a key-preserved table
  - DML may only affect a **child base table** (many side of the relationship)

```
CREATE OR REPLACE VIEW article_author AS
SELECT articleum, title, type, length, a.writerid, ln, phone
FROM article a, writer w
WHERE a.writerid = w.writerid;

INSERT INTO article_author(articleum, title, length, writerid)
VALUES (articleum_seq.NEXTVAL, 'Economy Heads South',99,'J525');

INSERT INTO article_author(ln, phone)
VALUES ('Hemingway', '(555) 555-1212');

CREATE OR REPLACE VIEW succeeded.
1 rows inserted
```

```
Error starting at line 9 in command:
INSERT INTO article_author(ln, phone)
VALUES ('Hemingway', '(555) 555-1212')
Error at Command Line:9 Column:27
Error report:
ORA-01770: cannot modify a column which does not appear in preserved table
```

33

## Which Columns are Updatable?

- **USER\_UPDATABLE\_COLUMNS** (pg 611)
  - **data dictionary** view that indicates which columns can participate in DML

```
SELECT column_name, updatable, insertable, deletable
FROM USER_UPDATABLE_COLUMNS
WHERE table_name = 'ARTICLE_AUTHOR';
```

Statement Output x Query... x

All Rows Fetched: 7 in 0.031 seconds

|   | COLUMN_NAME | UPDATABLE | INSERTABLE | DELETABLE |
|---|-------------|-----------|------------|-----------|
| 1 | ARTICLENUM  | YES       | YES        | YES       |
| 2 | TITLE       | YES       | YES        | YES       |
| 3 | TYPE        | YES       | YES        | YES       |
| 4 | LENGTH      | YES       | YES        | YES       |
| 5 | WRITERID    | YES       | YES        | YES       |
| 6 | LN          | NO        | NO         | NO        |
| 7 | PHONE       | NO        | NO         | NO        |

ARTICLE is a key preserved table

34

## Altering a View

```
ALTER VIEW viewname COMPILE;
```

- Views can't be modified
  - use CREATE OR REPLACE to redefine the view
    - egs: add another join, additional columns
- Invalid Views
  - altering/dropping a base table **invalidates** views that refer to the table
  - the next time you try to use the view, Oracle attempts to revalidate the view by compiling it
- Use ALTER VIEW to explicitly **recompile** a view
  - explicit recompilation lets you locate errors before run time
  - eg: recompile a view after altering one of its base tables to ensure that the table modification does not affect the view
- view\_status\_demo.sql

35

## Renaming and Dropping Views

```
RENAME oldname TO newname;
```

- Same syntax as renaming a table
- The view works the same
- Privileges to use the view remain intact
- Any dependent objects marked as invalid

```
DROP VIEW viewname;
```

- Removes the view's definition from the database
- Has no effect on the base tables
- Objects that are based on a deleted view become invalid

36