7

Managing Schema Objects

Objectives

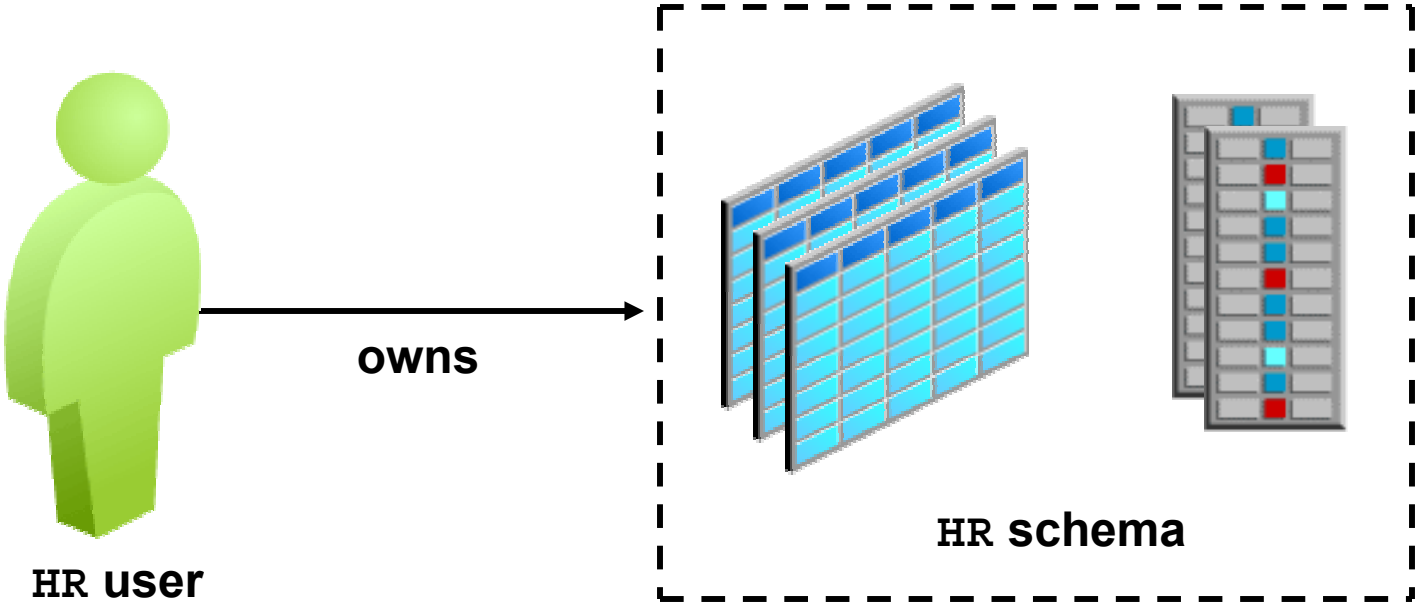
After completing this lesson, you should be able to do the following:

- Define schema objects and data types
- Create and modify tables
- Define constraints
- View the columns and contents of a table
- Create indexes
- Create views
- Create sequences
- Explain the use of temporary tables
- Use the data dictionary

ORACLE

What Is a Schema?

> **Schema**
Constraints
Indexes
Views
Sequences
Temp Tables
Data Dict



Accessing Schema Objects

Database Instance: orcl.oracle.com

Home | Performance | **Administration** | Maintenance

Schema

Database Objects

- Tables
- Indexes
- Views
- Synonyms
- Sequences
- Database Links
- Directory Objects
- Reorganize Objects

Programs

- Packages
- Package Bodies
- Procedures
- Functions
- Triggers
- Java Classes
- Java Sources

XML Database

- Configuration
- Resources
- Access Control Lists
- XML Schemas
- XMLType Tables
- XMLType Views

Users & Privileges

- Users
- Roles
- Profiles
- Audit Settings

Materialized Views

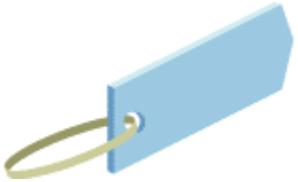
- Materialized Views
- Materialized View Logs
- Refresh Groups

BI & OLAP

- Dimensions
- Cubes
- OLAP Dimensions
- Measure Folders

Naming Database Objects

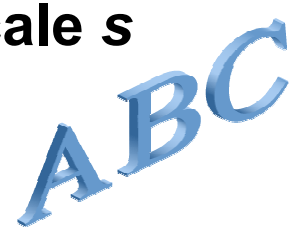
- The length of names must be from 1 to 30 bytes, with these exceptions:
  - Names of databases are limited to 8 bytes.
  - Names of database links can be as long as 128 bytes.
- Nonquoted names cannot be Oracle-reserved words.
- Nonquoted names must begin with an alphabetic character from your database character set.
- Quoted names are not recommended.

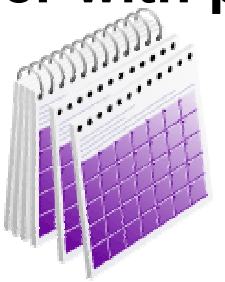


Specifying Data Types in Tables

Common data types:

- `CHAR(size [BYTE|CHAR])`: Fixed-length character data of *size* bytes or characters
- `VARCHAR2(size [BYTE|CHAR])`: Variable-length character string having a maximum length of *size* bytes or characters
- `DATE`: Valid date ranging from January 1, 4712 B.C. through A.D. December 31, 9999
- `NUMBER(p,s)`: Number with precision *p* and scale *s*





Creating and Modifying Tables

General | Constraints | Storage | Options | Partitions

* Name: jobs

Schema: shopowner

Tablespace: USERS

Organization: **Standard, Heap Organized**

Define Using: Column Specification

Specify the table name and schema.

Columns

Specify the column names, data types, and lengths.

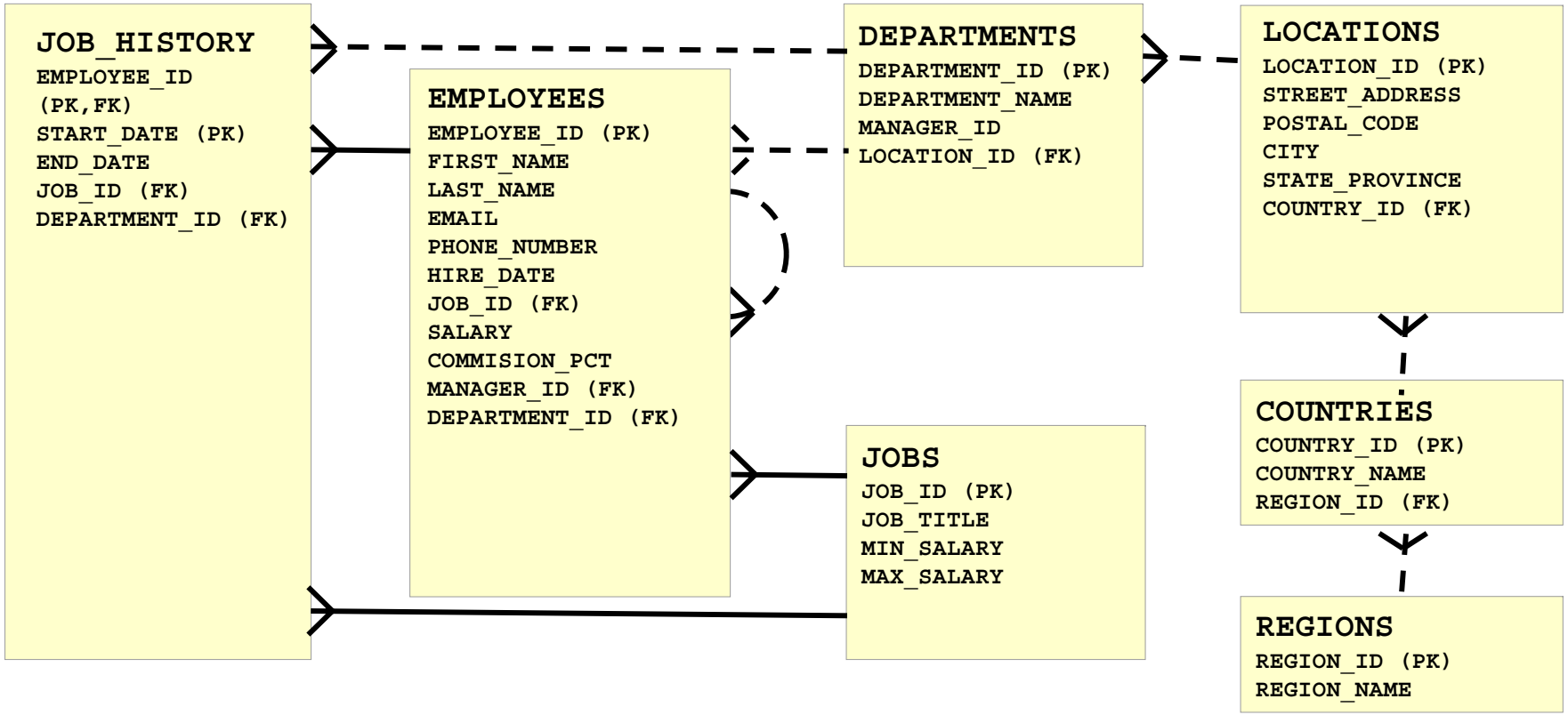
Advanced Attributes | Delete | Insert Col

Select		Name	Data Type	Size
◉		job_id	NUMBER	5
○		job_title	VARCHAR2	30
○		min_salary	NUMBER	6
○		max_salary	NUMBER	6
○			VARCHAR2	

Add 5 Table Columns

Understanding Data Integrity

Schema
> Constraints
Indexes
Views
Sequences
Temp Tables
Data Dict



Defining Constraints

Add UNIQUE Constraint

Cancel Continue

Up to 32 columns can make up a UNIQUE key constraint. The unique key columns constitute a unique identifier for each row in the table.

Definition

Name <System Assigned 3>

Table Columns

Available Columns

COUNTRY_ID
REGION_ID

Move
Move All
Remove
Remove All

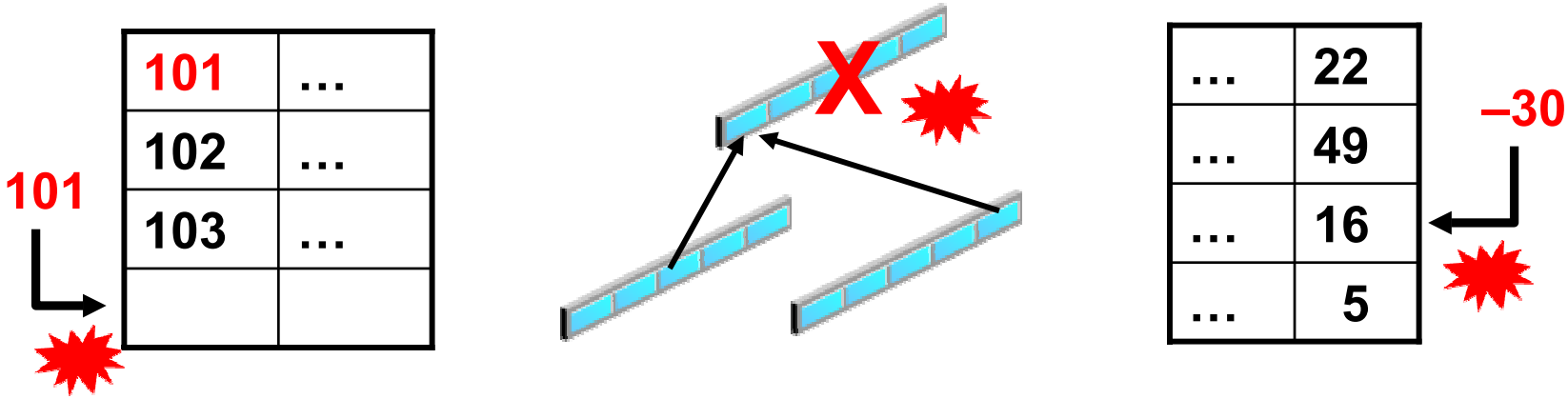
Selected Columns

COUNTRY_NAME

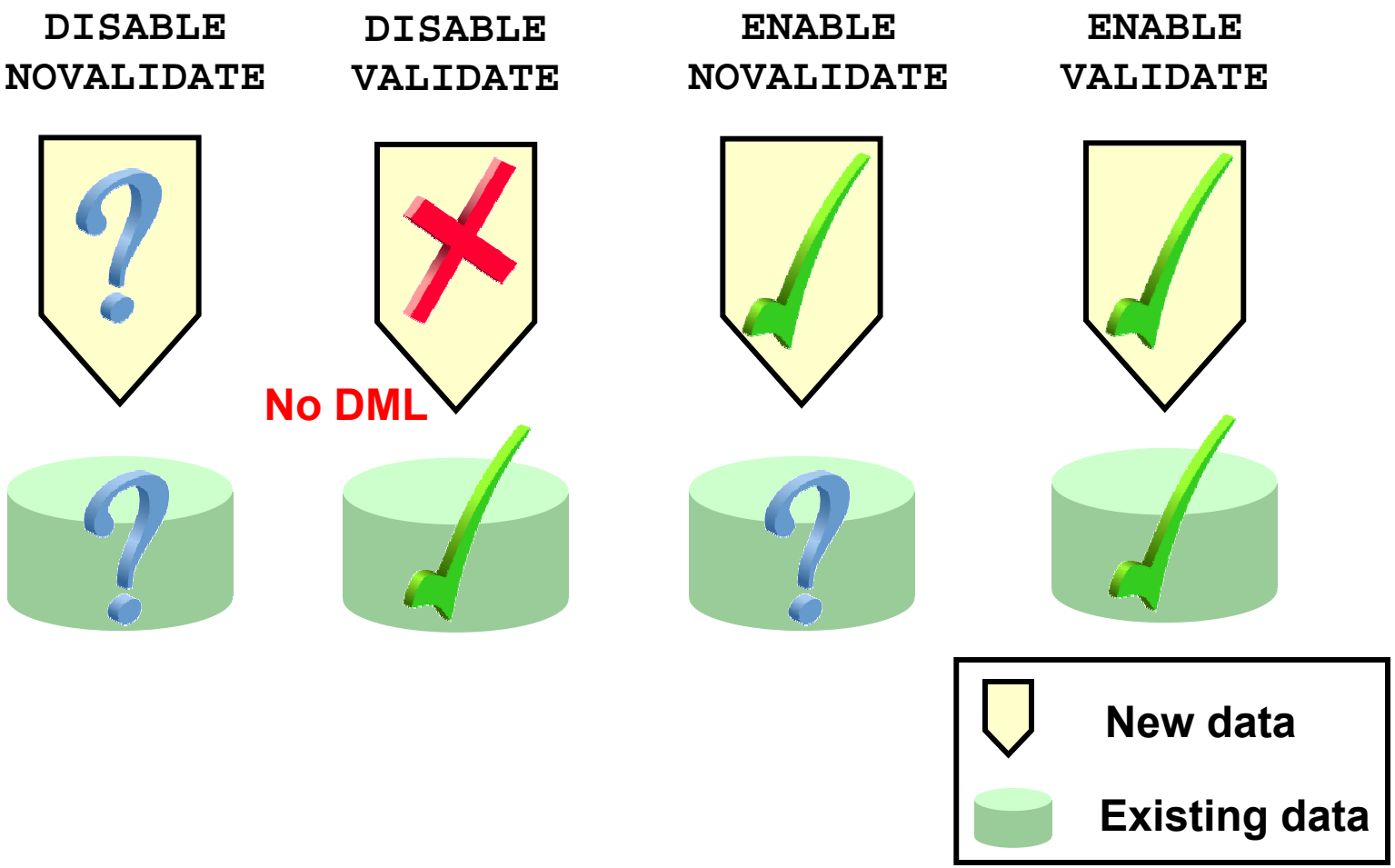
Constraint Violations

Examples of how a constraint can be violated are:

- Inserting a duplicate primary key value
- Deleting the parent of a child row in a referential integrity constraint
- Updating a column to a value that is out of the bounds of a check constraint



Constraint States



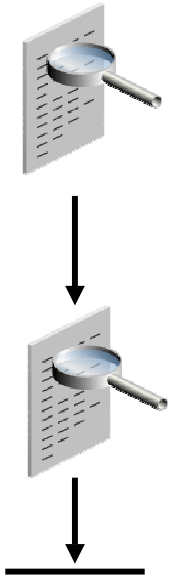
Constraint Checking

Constraints are checked at the time of:

- **Statement execution, for *nondeferred* constraints**
- **`COMMIT`, for *deferred* constraints**

Case: DML statement, followed by `COMMIT`

1. **Nondeferred constraints checked**
2. **`COMMIT` issued**
3. **Deferred constraints checked**
4. **`COMMIT` complete**



Creating Constraints with SQL: Examples

a

```
ALTER TABLE countries
ADD (UNIQUE(country_name) ENABLE NOVALIDATE);
```

b

```
ALTER TABLE employees ADD CONSTRAINT pk PRIMARY KEY
(employee_id)
```

c

```
CREATE TABLE t1 (pk NUMBER PRIMARY KEY, fk NUMBER, c1 NUMBER,
c2 NUMBER,
CONSTRAINT ri FOREIGN KEY (fk) REFERENCES t1, CONSTRAINT ck1
CHECK (pk > 0 and c1 > 0));
```

Viewing the Columns in a Table

View Table: HR.DEPARTMENTS

Actions Create Like Go Edit OK

General

Name	**DEPARTMENTS**
Schema	**HR**
Tablespace	**EXAMPLE**
Organization	**Standard, Heap Organized**

Columns

	Name	Data Type	Size	Scale	Not NULL	Default Value
✓	DEPARTMENT_ID	NUMBER	4		☑	
	DEPARTMENT_NAME	VARCHAR2	30		☑	
	MANAGER_ID	NUMBER	6		☐	
	LOCATION_ID	NUMBER	4		☐	

✓ Indicates a Primary Key column

Viewing the Contents of a Table

View Data for Table: HR.REGIONS

Refine Query OK

Query

```
SELECT "REGION_ID", "REGION_NAME" FROM "HR"."REGIONS"
```

Result

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa

Refine Query OK

Actions with Tables

Actions | Create Index | Go

- Create Like
- Create Index
- Create Synonym
- Create Trigger
- Generate DDL
- Object Privileges
- Manage Optimizer Statistics
- Reorganize
- Run Segment Advisor
- Shrink Segment
- Show Dependencies
- View Data
- Flashback Table
- Flashback Versions Query

Create Index

Show SQL | Schedule Job | Cancel | OK

General | Storage | Options | Partitions | Statistics

* Name

Schema: HR

Tablespace: <Default> | Estimate Index Size

Index Type: Standard - B-tree | Bitmap

Indexed Table Object

Index On: Table | Cluster

* Table Name: HR.EMPLOYEES | Populate Columns

TIP The indexed columns and their orders are indicated by the Order field

Table Columns

Column Name	Data Type	Sorting Order	Order
EMPLOYEE_ID	NUMBER	ASC	
FIRST_NAME	VARCHAR2	ASC	
LAST_NAME	VARCHAR2	ASC	
EMAIL	VARCHAR2	ASC	

Dropping a Table

Dropping a table removes:

- Data
- Table structure
- Database triggers
- Corresponding indexes
- Associated object privileges

```
DROP TABLE hr.employees PURGE;
```

Optional clauses for the `DROP TABLE` statement:

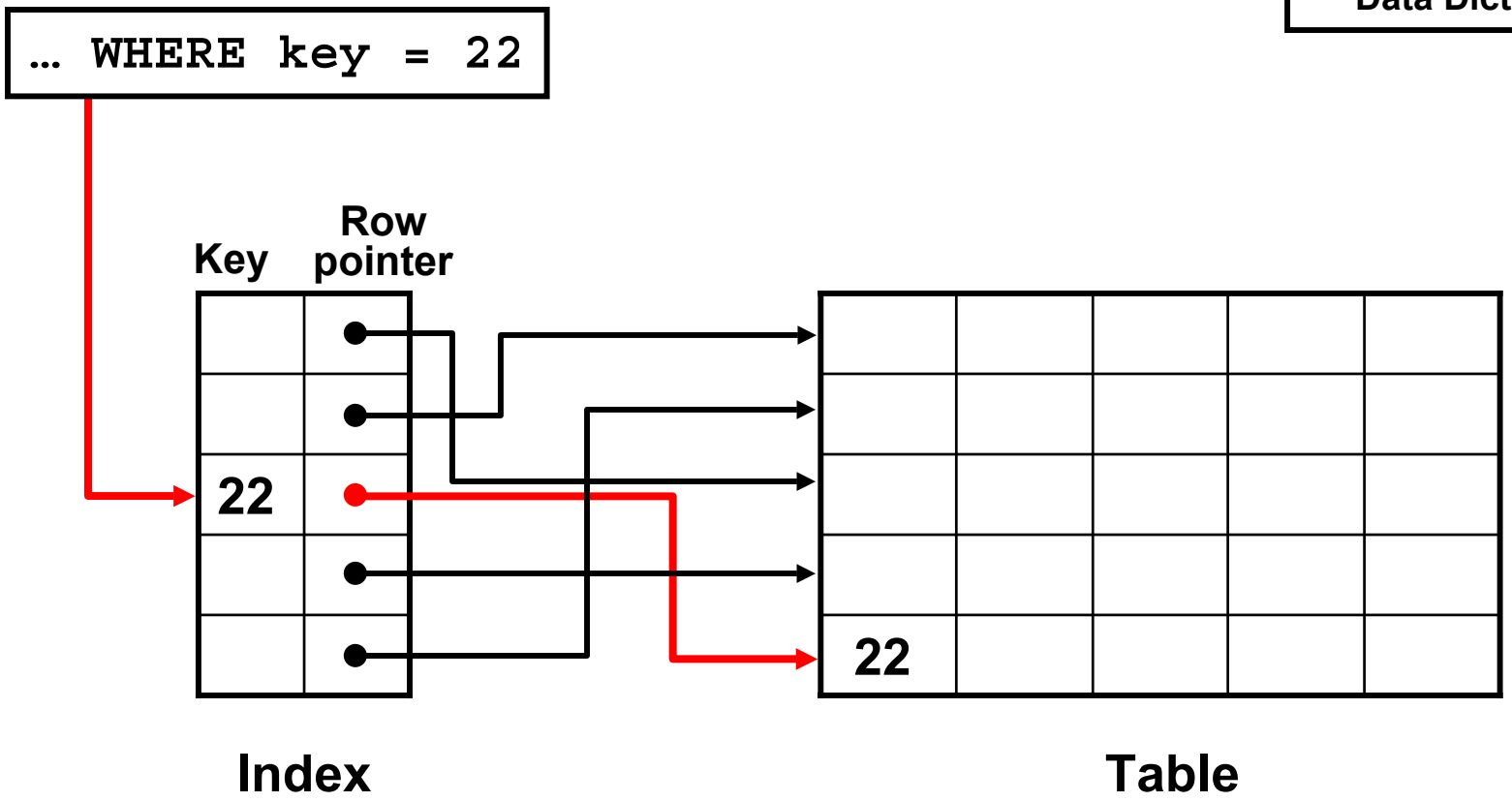
- `CASCADE CONSTRAINTS`: Dependent referential integrity constraints
- `PURGE`: No flashback possible

Truncating a Table

```
TRUNCATE TABLE hr.employees;
```

- Truncating a table makes its row data unavailable, and optionally releases used space.
- Corresponding indexes are truncated.

Indexes



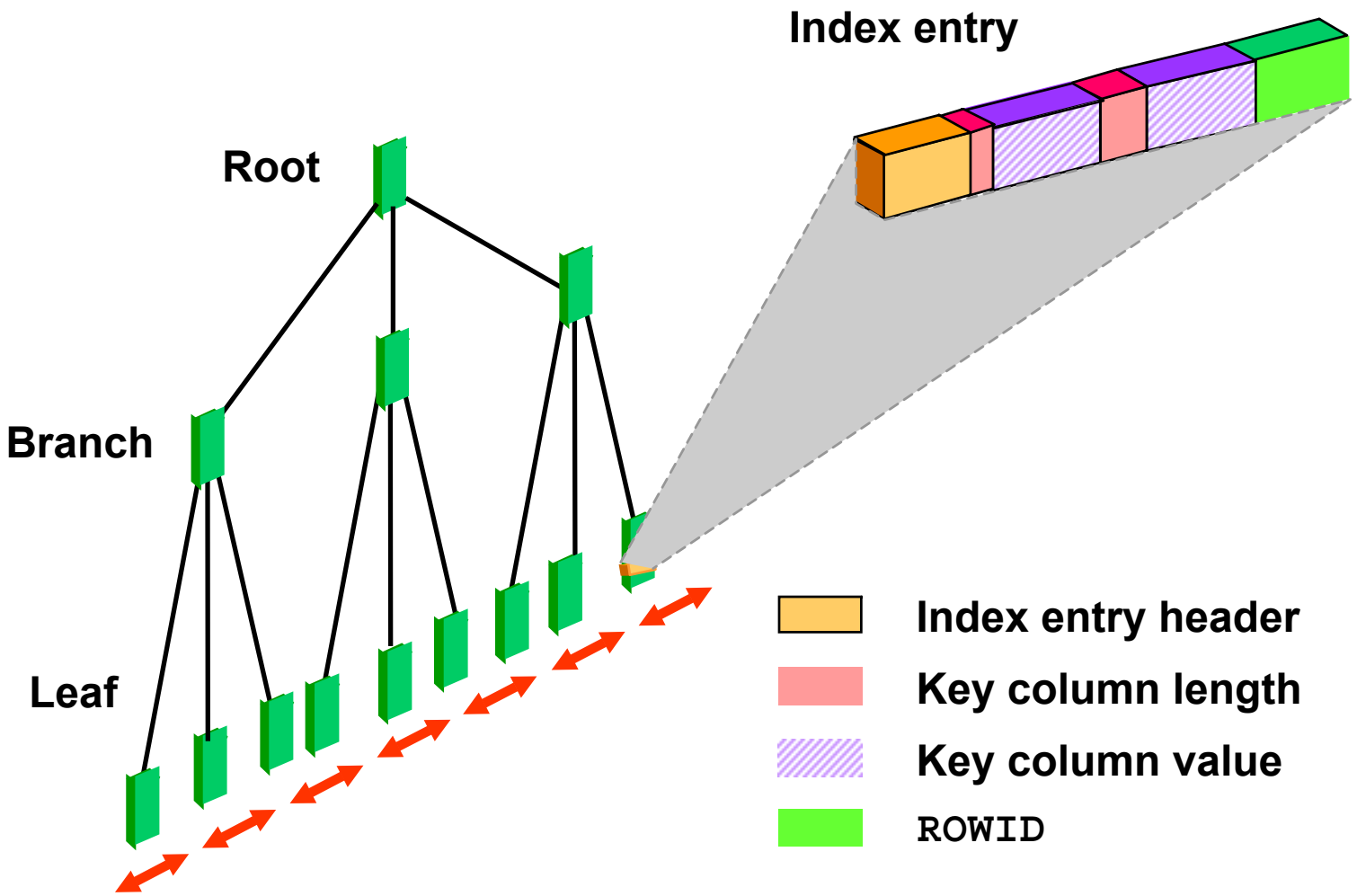
Types of Indexes

These are several types of index structures available to you, depending on the need:

- A B-tree index is in the form of a binary tree and is the default index type.
- A bitmap index has a bitmap for each distinct value indexed, and each bit position represents a row that may or may not contain the indexed value. This is best for low-cardinality columns.

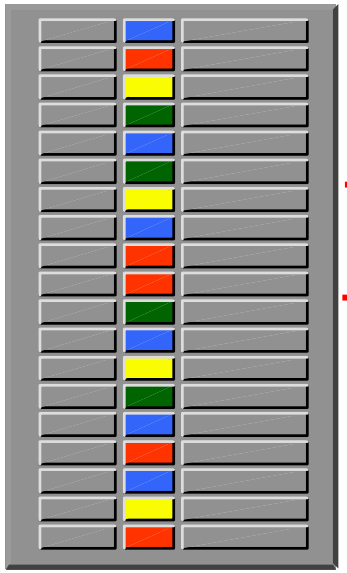
ORACLE

B-Tree Index



Bitmap Indexes

Table



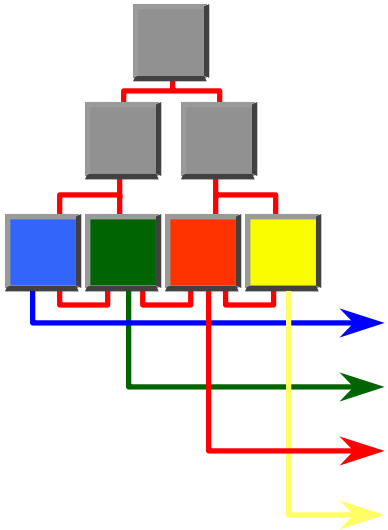
File 3

Block 10

Block 11

Block 12

Index



Key	Start ROWID	End ROWID	Bitmap
<Blue,	10.0.3,	12.8.3,	100010010001001010 0>
<Green,	10.0.3,	12.8.3,	000101000010010000 0>
<Red,	10.0.3,	12.8.3,	010000001100000100 1>
<Yellow,	10.0.3,	12.8.3,	001000100000100001 0>

Index Options

- A unique index ensures that every indexed value is unique.
- An index can have its key values stored in ascending or descending order.
- A reverse key index has its key value bytes stored in reverse order.
- A composite index is one that is based on more than one column.
- A function-based index is an index based on a function's return value.
- A compressed index has repeated key values removed.

Creating Indexes

Create Index

Show SQL | Cancel | OK

General | Storage | Options | Partitions

* Name

Schema HR

Tablespace <Default> Estimate Index Size

Index Type Standard - B-tree Bitmap

Indexed Table Object

* Table Name HR.EMPLOYEES Populate Columns

TIP The indexed columns and their orders are indicated by the Order field

Table Columns

Column Name	Data Type	Sorting Order	Order
EMPLOYEE_ID	NUMBER	ASC	
FIRST_NAME	VARCHAR2	ASC	

```
CREATE INDEX my_index ON
employees(last_name, first_name);
```

What Is a View?

Schema
Constraints
Indexes
> Views
...

LOCATION table

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	CO
2200	12-98 Victoria Street	2901	Sydney	New South Wales	AU
2800	Rua Frei Caneca 1360	01307-002	Sao Paulo	Sao Paulo	BR
1000	1297 Via Cola di Rie	00989	Roma		IT
1100	93091 Calle della Testa	10934	Venice		IT

COUNTRY table

CO	COUNTRY_NAME	REGION_ID
AR	Argentina	2
AU	Australia	3
BE	Belgium	1
BR	Brazil	2

View

LOCATION_ID	COUNTRY_NAME
2200	Australia
2800	Brazil

```
CREATE VIEW v AS SELECT location_id, country_name FROM
locations l, countries c
WHERE l.country_id = c.country_id AND c.country_id in
('AU','BR');
```

Creating Views

Database Instance: orcl.oracle.com > Views > Create View Logged in As SYS

Create View

Show SQL Cancel OK

General | Options | Object

* Name: STAFF

* Schema: HR

Aliases:

```
CREATE OR REPLACE VIEW "HR"."STAFF" AS SELECT EMPLOYEE_ID,
LAST_NAME, JOB_ID, MANAGER_ID, DEPARTMENT_ID
FROM EMPLOYEES
```

☑ Replace the view if exists

* Query Text:

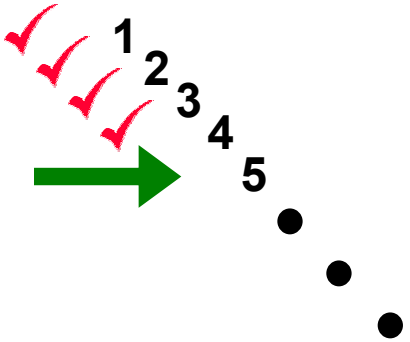
```
SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, MANAGER_ID,
DEPARTMENT_ID
FROM EMPLOYEES
```

Sequences

Schema
Constraints
Indexes
Views
> Sequences
Temp Tables
Data Dict

A sequence is a mechanism for automatically generating integers that follow a pattern.

- A sequence has a name, which is how it is referenced when the next value is requested.
- A sequence is not associated with any particular table or column.
- The progression can be ascending or descending.
- The interval between numbers can be of any size.
- A sequence can cycle when a limit is reached.



Creating a Sequence

Create Sequence

Show SQL | Cancel | OK

General

* Name ABC_SEQ
* Schema HR

Values

* Maximum Value ◉ Value 100 ○ Unlimited
* Minimum Value ◉ Value 1 ○ Unlimited
* Interval 5
* Initial 10

Options

☑ Cycle Values - Sequence will wrap around on reaching limit
☐ Order Values - Sequence numbers will be generated in order

Cache Options

☑ Use Cache
Cache Size 20

Show SQL

Return

```
CREATE SEQUENCE "HR"."ABC_SEQ" CYCLE NOORDER CACHE 20
MAXVALUE 100 MINVALUE 1 INCREMENT BY 5 START WITH 10
```

Using a Sequence

Workspace

Enter SQL, PL/SQL and SQL*Plus statements.

Clear

```
INSERT INTO local_temp VALUES
(local_temp_id.nextval, sysdate, 8, 20);
```

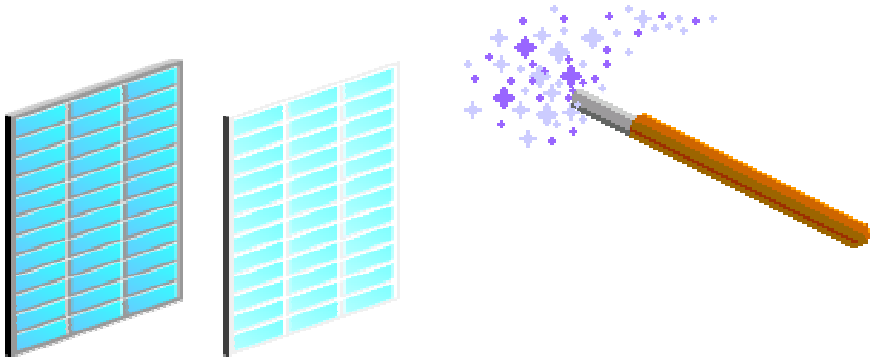
Execute | Load Script | Save Script | Cancel

1 row created.

Temporary Tables

A temporary table:

- Provides storage of data that is automatically cleaned up when the session or transaction ends
- Provides private storage of data for each session
- Is available to all sessions for use without affecting each other's private data



Temporary Tables: Considerations

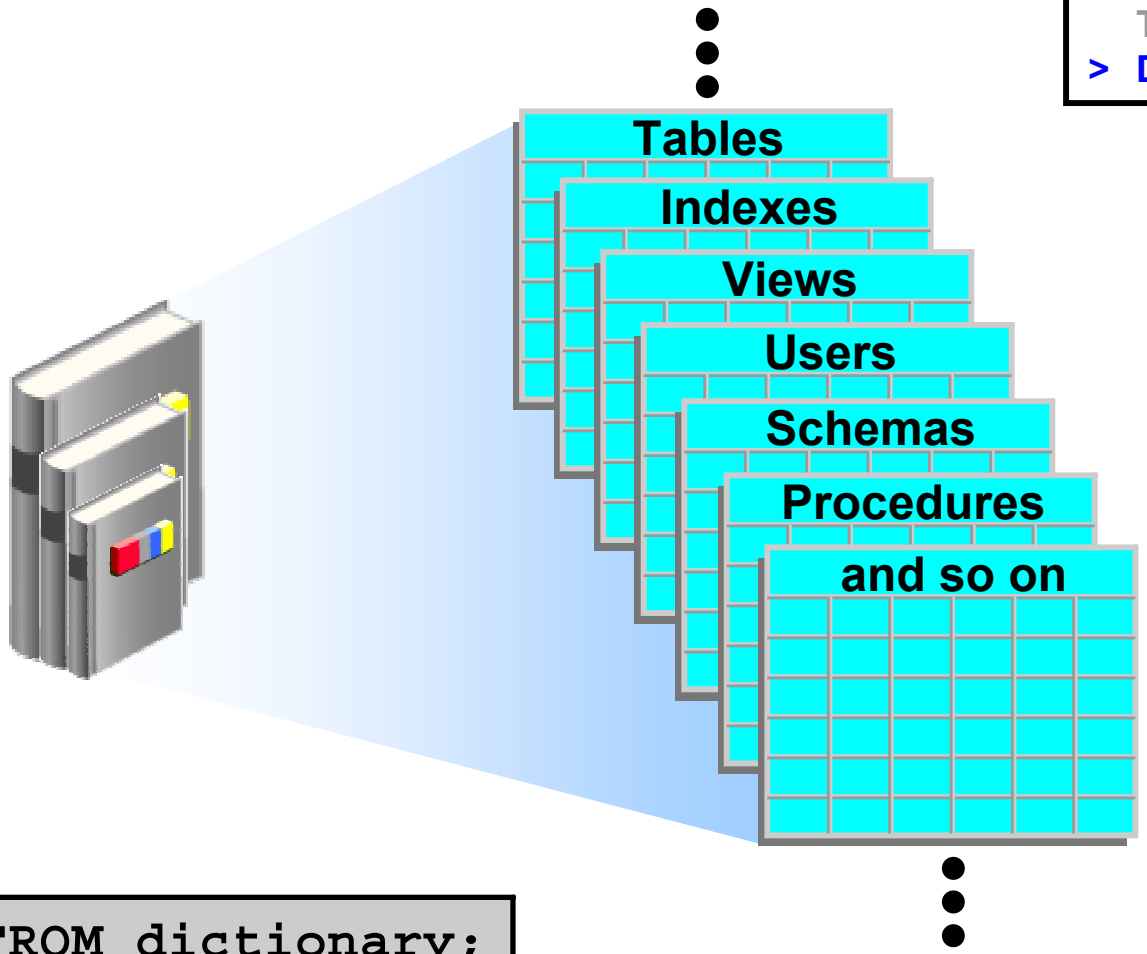
- Use the `GLOBAL TEMPORARY` clause to create temporary tables:

```
CREATE GLOBAL TEMPORARY TABLE employees_temp
ON COMMIT PRESERVE ROWS
AS SELECT * FROM employees;
```

- Use the `TRUNCATE TABLE` command to delete the contents of the table.
- You can create the following on temporary tables:
  - Indexes
  - Views
  - Triggers

Data Dictionary: Overview

- Schema
- Constraints
- Indexes
- Views
- Sequences
- Temp Tables
- > Data Dict




```
SELECT * FROM dictionary;
```

Data Dictionary Views

	Who Can Query	Contents	Subset of	Notes
`DBA_`	DBA	Everything	N/A	May have additional columns meant for DBA use only
`ALL_`	Everyone	Everything that the user has privileges to see	`DBA_` views	Includes user's own objects
`USER_`	Everyone	Everything that the user owns	`ALL_` views	Is usually the same as `ALL_` except for the missing `OWNER` column. Some views have abbreviated names as `PUBLIC` synonyms.

Data Dictionary: Usage Examples

a

```
SELECT table_name, tablespace_name FROM
user_tables;
```

b

```
SELECT sequence_name, min_value, max_value,
increment_by FROM all_sequences WHERE
sequence_owner IN ('MDSYS','XDB');
```

c

```
SELECT USERNAME, ACCOUNT_STATUS FROM
dba_users WHERE ACCOUNT_STATUS = 'OPEN';
```

d

```
DESCRIBE dba_indexes;
```

Summary

In this lesson, you should have learned how to:

- Define schema objects and data types
- Create and modify tables
- Define constraints
- View the columns and contents of a table
- Create indexes
- Create views
- Create sequences
- Explain the use of temporary tables
- Use the data dictionary

Practice Overview: Administering Schema Objects

This practice covers the following topics:

- Creating tables with columns
- Creating constraints:
  - Primary Key
  - Foreign Key
  - Check constraint
- Creating indexes