Chapter 2: Oracle Database Objects

Database Objects (Exam-Focused)

Oracle supports many objects, but **only 8 are tested** on the exam:

1. **TABLE** – Stores data in rows and columns
2. **INDEX** – Speeds up search queries on tables
3. **VIEW** – A virtual table that shows filtered data; stores no data
4. **SEQUENCE** – Auto-generates numbers (e.g., for primary keys)
5. **SYNONYM** – Alias for another object (table, view, etc.)
6. **CONSTRAINT** – Rules to enforce data integrity (e.g., NOT NULL)
7. **USERS** – Accounts that own schema objects
8. **ROLES** – Groups of privileges assigned to users

Schema vs. Nonschema Objects

- **Schema objects**: Owned by a user (e.g., tables, views, indexes)
- **Nonschema objects**: Not owned by a user (e.g., users, roles)

Schemas

- A **schema** is the collection of objects (like tables, views) owned by a **user**
- The **user account** and schema share the same name
- Multiple schemas = multiple users

Namespaces (Name Rules)

- Each object type has its own namespace
- **No duplicates** in the same namespace (e.g., can't have two tables named EMPLOYEES in the same schema)
- Different schemas **can** have objects with the same name

Naming Rules (Basic)

- 1–30 characters
- Must start with a letter
- Can include letters, numbers, _, $, #

- Not case-sensitive (unless quoted)
- Must not use reserved words (like SELECT, CREATE)

Quoted Names:

- Case-sensitive
- Can include spaces and reserved words
- Not recommended

Creating a Simple Table (Example)

```sql
CREATE TABLE cruises (
    cruise_id NUMBER PRIMARY KEY,
    cruise_type_id NUMBER,
    start_date DATE,
    end_date DATE,
    port VARCHAR2(50),
    status VARCHAR2(10) DEFAULT 'DOCK',
    captain_id NUMBER NOT NULL
);
```

Reviewing a Table's Structure

Use the SQL*Plus command:

```sql
DESC cruises
```

It shows:

- **Column name**
- **Nullable?**
- **Data type**

DATA TYPES

Character Data Types

- **CHAR(n)**: Fixed-length string. Pads with spaces. Max: 2000

- **VARCHAR2(n)**: Variable-length string. No padding. Max: 4000 characters (up to 32767 if MAX_STRING_SIZE = EXTENDED)

Numeric Data Types

- **NUMBER(n, m)**:
  - n: Precision (total digits, max 38)
  - m: Scale (digits to right of decimal; can be negative)
  - Defaults: If omitted, both max out
  - Excess precision → error; excess scale → rounds

Date/Time Data Types

- **DATE**: Stores year, month, day, hour, minute, second
- **TIMESTAMP(n)**: Adds fractional seconds to DATE. n = 0–9 (default 6)
- **TIMESTAMP(n) WITH TIME ZONE**: Stores full time zone
- **TIMESTAMP(n) WITH LOCAL TIME ZONE**: Time zone not stored; shown in user's local time zone
- **INTERVAL YEAR(n) TO MONTH**: Time span in years & months. n = 0–9 (default 2)
- **INTERVAL DAY(n1) TO SECOND(n2)**: Time span with days to fractional seconds
  - n1: Days (0–9, default 2)
  - n2: Seconds fraction (0–9, default 6)

Large Object (LOB) Types

- **BLOB**: Binary Large Object (e.g., images, video). No scale/precision
- **CLOB**: Character LOB
- **NCLOB**: Unicode CLOB

Restrictions:

- Cannot be in PRIMARY KEY, GROUP BY, ORDER BY, DISTINCT, or JOINs

CONSTRAINTS

Creating Constraints

- Constraints can be **in-line** (with column) or **out-of-line** (after all columns)
- Use CREATE TABLE or ALTER TABLE to define constraints

NOT NULL

- Requires column to always have a value
- **Cannot** be created out-of-line
- Defaults: All columns allow NULL unless specified otherwise

UNIQUE

- Ensures no duplicate values in column(s)
- Allows NULLs unless combined with NOT NULL
- Can be **composite** (multiple columns together must be unique)

PRIMARY KEY

- Combines **UNIQUE + NOT NULL**
- One per table. Can be single or composite
- Implies data must be unique and non-null

FOREIGN KEY

- Links child table to parent's PRIMARY or UNIQUE key
- Requires parent table/column to exist first
- Enforces referential integrity
- Add ON DELETE CASCADE or ON DELETE SET NULL as needed
- Best added via ALTER TABLE for flexibility and modularity

CHECK

- Restricts column to specific values or expressions
- Cannot reference other tables, subqueries, pseudocolumns, or certain functions (e.g., SYSDATE)
- Evaluates to **TRUE** or **NULL** to allow insert/update

Multiple Constraints

- Can combine multiple types on one table
- Only one PRIMARY KEY allowed per table

NULL Concept

- **NULL ≠ 0 or blank** — it means **unknown/absent**
- Any expression involving NULL → result is NULL

Data Type Restrictions (Constraints)

Constraint Type	BLOB/CLOB	TIMESTAMP WITH TIME ZONE
PRIMARY KEY	❌	❌
UNIQUE	❌	❌
FOREIGN KEY	❌	❌
CHECK	✅	✅
NOT NULL	✅	✅

Dropping Columns

Basic Syntax:

sql

```sql
ALTER TABLE table_name DROP COLUMN column_name;
```

Rules:

- A table must have **at least one column** remaining after dropping
- If a column is **referenced by a foreign key**, you **must use** CASCADE CONSTRAINTS:

  sql

  ```sql
  ALTER TABLE CRUISE_ORDERS DROP COLUMN CRUISE_ORDER_ID CASCADE CONSTRAINTS;
  ```

Foreign Key Impacts:

- Dropping a column that's part of a foreign key **drops the constraint too**
- This works **without** CASCADE CONSTRAINTS when dropping the column **from the referencing (child) table**
- If the foreign key is **composite** (uses multiple columns), you must either:
  - Drop **all involved columns at once**, or
  - Drop one column using CASCADE CONSTRAINTS

SET UNUSED

Purpose:

Hides column like a drop, but **faster** for large/active tables

```sql
ALTER TABLE table_name SET UNUSED COLUMN column_name;
```

Facts:

- Acts like a drop: column is **gone forever**
- **Cannot rollback**
- **Indexes and constraints** on the column are also removed
- **Same syntax as DROP**, just replace DROP with SET UNUSED
- You can set **multiple columns** as unused:

  ```sql
  ALTER TABLE table_name SET UNUSED (col1, col2);
  ```

Why Use It:

- Better **performance** for large tables
- Allows reuse of the column name
- Later, you can **fully drop** unused columns

Table Limit Warning:

- Max **1,000 columns** per table
- UNUSED columns **still count** toward the limit until dropped

Drop Unused Columns:

```sql
ALTER TABLE table_name DROP UNUSED COLUMNS;
```

Check Tables With Unused Columns:

- Use view: `USER_UNUSED_COL_TABS`
- Shows **table names** and **unused column count**, but **not column names**

External Tables (Exam Objective 2.07)

Definition:

- An **external table** is *read-only*

- **Metadata is stored in the database**, but **data is stored outside** (e.g., in a flat file)
- Cannot use INSERT, UPDATE, DELETE, INDEX, or CONSTRAINTS

Purpose:

- Bridge between SQL and non-database sources (flat files, spreadsheets, etc.)
- Works similarly to SQL*Loader and Data Pump, but allows querying via SELECT

Restrictions:

- Cannot contain LOB columns (CLOB, BLOB, etc.)
- Cannot define constraints
- Marking a column as UNUSED will drop it

Steps to Create an External Table:

1. **Create DIRECTORY Object:**

   ```sql
   CREATE OR REPLACE DIRECTORY dir_name AS 'path';
   GRANT READ ON DIRECTORY dir_name TO user;
   ```

   - dir_name: Name used in SQL
   - 'path': Must exist in server's OS (Oracle doesn't create it)
   - Grant access for users to read from this directory

2. **Create External Table:**

```sql
CREATE TABLE table_name (
    col1 CHAR(n),
    col2 CHAR(n)
)
ORGANIZATION EXTERNAL
(
    TYPE ORACLE_LOADER
    DEFAULT DIRECTORY dir_name
    ACCESS PARAMETERS (
        RECORDS DELIMITED BY NEWLINE
        SKIP 2
        FIELDS (
            col1 CHAR(n),
            col2 CHAR(n)
        )
    )
    LOCATION ('filename.txt')
);
```

- ORGANIZATION EXTERNAL: Required clause
- TYPE ORACLE_LOADER (or ORACLE_DATAPUMP): Choose based on format
- ACCESS PARAMETERS:
    - RECORDS DELIMITED BY NEWLINE: Each line = one record
    - SKIP 2: Skip header lines
    - FIELDS (...): Define fixed-length fields
- LOCATION: Name of the file in the specified directory

Usage:

- You can use SELECT like a normal table
- Use SQL functions to clean/transform imported data
- No DML (INSERT, UPDATE, DELETE) allowed

Related Concepts for the Exam

Object Types and Purpose

Object Type	Purpose
Table	Stores data
View	Virtual table, filters on one/more tables
Sequence	Auto-increment counter, often for IDs
Synonym	Alias for another object
Index	Speeds up queries
Constraint	Rule to control data validity
User	Represents a database user
Role	Set of privileges granted to users

Object Classification:

- **Schema Objects**: Table, View, Sequence, Private Synonym, Index, Constraint
- **Non-Schema Objects**: User, Role, Public Synonym

Namespaces:

- Indexes and Constraints: Own namespace
- Tables, Views, Sequences, Private Synonyms: Share namespace within schema
- Users & Roles: Shared namespace across database
- Public Synonyms: Own global namespace

Data Types Summary:

Type	Examples
Character	CHAR, VARCHAR2
Numeric	NUMBER, FLOAT
Date/Time	DATE, TIMESTAMP, INTERVAL
LOB	CLOB, BLOB, NCLOB

Constraints Summary:

- Types: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
- NOT NULL must be inline (with column definition)

TRUNCATE TABLE vs DELETE:

Feature	TRUNCATE TABLE	DELETE
DDL/DML	DDL (implicit commit)	DML (can rollback)
Triggers	Not fired	Fired
Speed	Faster	Slower
Index impact	No selective removal	Can remove selectively

Use `TRUNCATE TABLE ... CASCADE` if child rows exist with ON DELETE CASCADE.

Column Management:

- DROP COLUMN: Removes column (with CASCADE CONSTRAINTS if needed)
- SET UNUSED: Hides column without immediate drop (can drop later)