

# Chapter 11

## Create, Alter, and Drop Tables



### Object Naming Conventions

- Must begin with a letter
- Can be 1–30 characters long
- Must contain only A–Z, a–z, 0–9, \_, \$, and #
- Must not duplicate the name of another object in the same schema
- Must not be an Oracle Server reserved word



## Creating a Table

```
CREATE TABLE [schema.]table
(column datatype [DEFAULT expr][, ...]);
```

- Note: lots of optional clauses for creating tables we don't cover
- Two ways to create
  - specify each column's name and data type
  - base the new table on columns in another existing table
- User must have CREATE TABLE privilege (Ch 14)
- Column names are unique within a table
- Oracle permits up to 1000 columns per table
- DEFAULT *expr*
  - specifies a value to use when no field value is explicitly provided when a new row is INSERTed

3



## Common Data Types

Datatype	Description
<b>VARCHAR2(size)</b>	<b>Variable-length character data</b> ( ≤ 4000)
<b>CHAR(size)</b>	<b>Fixed-length character data</b> ( ≤ 2000)
<b>NUMBER(p,s)</b>	<b>Variable-length numeric data</b>
<b>DATE</b>	<b>Date with time values</b>
LONG ( <i>deprecated</i> )	Variable-length character data up to 2 gigabytes
CLOB	Single-byte character data up to 4 gigabytes
RAW and LONG RAW	Raw binary data
BLOB	Binary data up to 4 gigabytes
BFILE	Binary data stored in an external file; up to 4 gigabytes

4



## CREATE TABLE Example #1

```
CREATE TABLE writer(  
  writerid      CHAR(4)          CONSTRAINT writer_writer_id_nn  NOT NULL,  
  ln            VARCHAR2(30)     CONSTRAINT writer_ln_nn      NOT NULL,  
  fn            VARCHAR2(20)     CONSTRAINT writer_fn_nn      NOT NULL,  
  phone         VARCHAR2(14)     ,  
  lastcontact   DATE             ,  
  freelancer    CHAR(1)          ,  
  amount        NUMBER(6,2)      ,  
  contact       CHAR(4)          ,  
  CONSTRAINT writer_writerid_pk  PRIMARY KEY (writerid),  
  CONSTRAINT writer_contact_fk   FOREIGN KEY(contact) REFERENCES writer(writerid),  
  CONSTRAINT writer_freelancer_ck CHECK(freelancer IN ('Y', 'N')));
```



## Creating a Table From an Existing Table

- Columns in the new table inherit the same definition as the fields selected from the existing table
- Most constraints are not inherited
  - only NOT NULL constraints are inherited
  - must then define primary key, foreign key, check, unique constraints
- The new table can contain rows from the existing table(s)

```
SQL> CREATE TABLE freelancer AS  
      SELECT ln, fn, phone  
      FROM writer  
      WHERE freelancer = 'Y'
```

```
SQL> CREATE TABLE freelancer AS  
      SELECT ln, fn, phone  
      FROM writer  
      WHERE 'a' = 'g'
```



## Integrity Constraints

---

- For enforcing business rules
  - help assure data integrity
- Goal: prevent incomplete, incorrect data from being stored
- Two locations to specify
  - Column-level constraint
    - cited as the column is being defined
    - must be used for NOT NULL
  - Table-level constraint
    - cited at the end, after all of the columns have been defined
    - must be used for foreign keys and concatenated primary keys
  - Most constraints can be equivalently declared either location

7



## Integrity Constraints

---

- Primary Key Constraint
  - Requires that the column's values be non-null and unique
  - Oracle automatically creates an index for the primary key
    - the index disallows nulls and duplicate values
- Not Null Constraint
  - Prevents a new or modified record from being stored if the specified field is not assigned a value

8



# Integrity Constraints

- Foreign Key Constraint
  - used to enforce **referential integrity**
    - requires that each non-null foreign key match the value of a parent table row's primary key
  - must be declared as table-level constraints
  - the foreign key field must have the same data type as the primary key it references
  - **ON DELETE CASCADE** clause
    - indicates that when a parent row is deleted, the matching row(s) in the child table are to be deleted too
  - **ON DELETE SET NULL** clause
    - indicates that when a parent row is deleted, the foreign keys in the matching row(s) in the child table are to be set to Null
  - **ON DELETE RESTRICT** clause (default)
    - if neither clause is specified, the row in the parent table cannot be deleted when there are matching rows in the child table

9



# CREATE TABLE Example #2

```
SQL> CREATE TABLE article (
  articlenum NUMBER(4) CONSTRAINT article_articlenum_nn NOT NULL
  title VARCHAR2(50) CONSTRAINT article_articlenum_pk PRIMARY KEY,
  type CHAR(3) CONSTRAINT article_title_nn NOT NULL,
  issue DATE,
  length NUMBER(5),
  writerid CHAR(4) CONSTRAINT article_writerid_nn NOT NULL,
  CONSTRAINT article_writerid_fk
  FOREIGN KEY(writerid) REFERENCES writer(writerid) ON DELETE CASCADE,
  CONSTRAINT article_type_fk
  FOREIGN KEY(type) REFERENCES type(type));
```

10



## Integrity Constraints

- Unique Constraint
  - No duplicate values allowed within the column
  - NULLs are allowed
  - Oracle automatically creates an index to enforce
  - Commonly created for \_\_\_\_\_ key columns
- Check Constraint
  - Used to limit the value entered into a field
  - An expression that is evaluated as either TRUE or FALSE
  - Oracle rejects the row if the constraint evaluates for FALSE

11



## Naming Constraints

- A Good Habit
  - makes it easier to understand constraint violation error messages
- General Form
  - tablename\_fieldname\_constrainttype
- Examples
  - article\_articlenum\_pk
  - article\_writerid\_fk
  - writer\_freelancer\_ck
- When you don't name a constraint, Oracle names it for you

```
SQL> CREATE TABLE x (f1 NUMBER(9) PRIMARY KEY);
SQL> SELECT constraint_name
       FROM USER_CONSTRAINTS WHERE table_name = 'X';
```

12



## Renaming Tables

```
RENAME current_name TO new_name
```

```
SQL> RENAME freelancer TO nonstaff
```

- Watch out for dependency problems
  - Views and PL/SQL program units (stored procedures, stored functions, packages) that use the table's old name become invalid (not usable)
  - Constraints are retained and are automatically adjusted
    - if the table you rename is referenced by foreign keys of another table, the foreign key constraint **definitions** are modified to refer to the table by its new name
    - the constraint **names** are not adjusted

13



## Dropping a Table

```
DROP TABLE table [CASCADE CONSTRAINTS][PURGE]
```

```
SQL> DROP TABLE freelancer;
```

- The table's structure and data rows are moved into the **Recycle Bin** (unless **PURGE** clause is used)
- All of the table's permissions, constraints, indexes, and triggers are also dropped
- You **cannot** roll back this statement!
- Any pending transaction is committed
  - each DDL statement generates an implicit COMMIT
- How different from TRUNCATE?

14



# Dropping a Table

```
DROP TABLE table [CASCADE CONSTRAINTS][PURGE]
```

```
SQL> DROP TABLE writer;  
ORA-02449: unique/primary keys in table referenced by foreign keys
```

- Watch out for dependency problems
  - Views and PL/SQL program units that depend on a dropped table remain, but become invalid (not usable)
  - If the table you are dropping is referenced by foreign keys of another table, you must include the `CASCADE CONSTRAINTS` option
    - Oracle will drop any foreign key constraints that referred to a column in the dropped table

15



# Recycle Bin (new 10g)

- Dropped tables are moved into the Recycle Bin
- Two ways to see Recycle Bin objects

```
SQL> show recyclebin
```

ORIGINAL NAME	RECYCLEBIN NAME	OBJECT TYPE	DROP TIME
X	BIN\$gphJmQV+R2mX2xvk25Ledw==\$0	TABLE	2007-11-09:14:10:41

```
SQL> SELECT object_name, type, droptime FROM user_recyclebin;
```

OBJECT_NAME	TYPE	DROPTIME
BIN\$HB+7xqpZRwS+Zxo/NMohgA==\$1	INDEX	2007-11-09:14:10:41
BIN\$gphJmQV+R2mX2xvk25Ledw==\$0	TABLE	2007-11-09:14:10:41

- Purge the Recycle Bin to remove dropped objects and free the disk space they occupied

```
SQL> PURGE RECYCLEBIN;
```

16

# Flashback Table Concepts (new 10g)

```
FLASHBACK TABLE table [, tablename...]  
TO {TIMESTAMP expr | BEFORE DROP [RENAME TO newtablename]}
```

- Restores a table to an earlier state
  - also retrieves the table's indexes, triggers, and constraints except for referential integrity constraints that reference other tables
- Oracle cannot restore a table to an state earlier than the most recent DDL operation that changed the table's structure
- Oracle cannot restore a table to an state earlier than the undo data available
- Must have sufficient privileges (Ch14)
- To flashback a table to an earlier **timestamp**, you must have SELECT, INSERT, DELETE, and ALTER privileges on the table and **row movement** must be enabled for the table
  - **TO TIMESTAMP** is not available in Oracle Express Edition

```
SQL> ALTER TABLE x ENABLE ROW MOVEMENT;
```

# Flashback Table Examples (new 10g)

```
FLASHBACK TABLE table [, tablename...]  
TO {TIMESTAMP expr | BEFORE DROP [RENAME TO newtablename]}
```

```
SQL> FLASHBACK TABLE freelancer  
      TO BEFORE DROP;  
Flashback complete.
```

- **TO TIMESTAMP** is not available in Oracle Express Edition

```
SQL> FLASHBACK TABLE timecard  
      TO TIMESTAMP(SYSTIMESTAMP-INTERVAL '5' minute);  
Flashback complete.
```



## Documenting Tables and Columns

- Use COMMENT to add a comment about a table, view, materialized view, or column into the **Data Dictionary** (ch 13)
- To drop a comment, set it to an empty string ''

```
SQL> COMMENT ON TABLE course IS 'Information for a course.';
SQL> COMMENT ON COLUMN course.course_no
      IS 'The unique ID for a course.';
SQL> COMMENT ON COLUMN course.cost
      IS 'The dollar amount charged for enrollment.';
SQL> COMMENT ON COLUMN course.cost IS '';
```

```
SQL> SELECT *
      FROM USER_TAB_COMMENTS
      WHERE table_name = 'COURSE'
```

```
SQL> SELECT *
      FROM USER_COL_COMMENTS
      WHERE table_name = 'COURSE'
```

19



## Adding Columns

```
ALTER TABLE table
ADD      (column datatype [DEFAULT expr]
         [, column datatype]...);
```

- Can only add a new NOT NULL column when either:
  - the column has a DEFAULT value OR
  - the table has no existing rows

```
SQL> ALTER TABLE writer
      ADD (sex CHAR(1) DEFAULT 'M'
          CONSTRAINT writer_gender_nn NOT NULL
          CONSTRAINT writer_gender_ck CHECK (sex IN ('M', 'F')))
```

20



## Renaming a Column

```
ALTER TABLE table
RENAME column TO column
```

- Used to change a column's name
- The column's data remains intact
- Watch out for dependency problems

```
SQL> ALTER TABLE writer
      RENAME COLUMN sex to gender
```

21



## Dropping Columns

```
ALTER TABLE table
DROP (column, [column]...)
```

- Removes the column(s) and removes its data from each row of the table, freeing space in the data blocks on disk

```
SQL> ALTER TABLE writer DROP COLUMN gender
```

- Can't drop the primary key column(s)
- At least one column must remain

22



## Dropping Columns

- Dropping a column can take considerable time
  - can require lots of disk I/O
  - a quicker alternative is to mark a column as unused, then later use DROP UNUSED COLUMNS to remove those columns that have been marked UNUSED

```
SQL> ALTER TABLE writer SET UNUSED (gender)
```

```
SQL> ALTER TABLE writer DROP UNUSED COLUMNS
```

23



## Modifying a Column

```
ALTER TABLE table  
MODIFY      (column datatype [DEFAULT expr]  
            [, column datatype]...);
```

- Used to change a column's:
  - width
  - datatype
  - default value

```
SQL> ALTER TABLE writer  
      MODIFY (amount NUMBER(7,2)  
             CONSTRAINT writer_amount_nn NOT NULL,  
             ln VARCHAR2(40))
```

24



## Changing a Text Column's Size

- You can **increase** the size of a text column
- You can **reduce** the size of a text column provided it does not cause data loss
  - Oracle scans the existing data values in the column and returns an error if data exists that exceeds the new limit
  - <ver 9i, columns containing data couldn't be decreased (ORA-01441 error)
- Text column examples

```
SQL> ALTER TABLE writer
        MODIFY (phone VARCHAR2(4));
ERROR at line 2:
ORA-01441: cannot decrease column length because value too big.

SQL> ALTER TABLE writer
        MODIFY (ln VARCHAR2(38));
Table altered.
```

25



## Changing a Numeric Column's Size

- You can only **reduce** the precision or scale of a numeric column when all rows contain Null
  - Oracle won't let you decrease a numeric column's width even if doing so would not result in data loss
- You can **increase** the precision or scale of a numeric column
- Numeric column examples

```
SQL> DESC writer
SQL> ALTER TABLE writer
        MODIFY (amount NUMBER(9,2));
Table altered.
SQL> ALTER TABLE writer
        MODIFY (amount NUMBER(8,2));
ORA-01440: column to be modified must be empty to
decrease precision or scale
```

- Workaround to decrease a numeric size to save space when there is data is present in the column?

26



## Changing a Column's Datatype

- Permitted under two circumstances:
  - the column is empty
  - changing to a **compatible** datatype
    - eg: VARCHAR2 to CHAR

```
SQL> ALTER TABLE writer
      MODIFY phone CHAR(14)
```

27



## Changing Column's Default Value

- Has no effect on existing rows
  - affects subsequent INSERTs into the table
- Can also eliminate a NOT NULL restriction

```
SQL> ALTER TABLE writer
      MODIFY (gender DEFAULT 'F',
             ln NULL)
```

28

# Adding/Dropping Constraints

- Add new constraint
  - a constraint is automatically enabled when it is created
  - can't add a constraint when the table's existing data violates the constraint

```
SQL> ALTER TABLE writer
      ADD CONSTRAINT writer_writerid_pk PRIMARY KEY(writerid)

SQL> ALTER TABLE writer
      ADD CONSTRAINT writer_gender_ck
      CHECK(gender IN ('M', 'F'))
```

- Drop an existing constraint

```
SQL> ALTER TABLE writer
      DROP CONSTRAINT writer_gender_ck;
```

29

# Composite PK and FK Constraints

- Composite keys can be declared in a CREATE TABLE statement by placing them in table-level constraints at the bottom of the statement
- Composite key constraints can also be created after the table's creation, preferably before any rows are INSERTed

```
SQL> ALTER TABLE enrollment
      ADD CONSTRAINT enr_pk
      PRIMARY KEY (student_id, section_id);
```

```
SQL> ALTER TABLE grade
      ADD CONSTRAINT gr_enr_fk
      FOREIGN KEY (student_id, section_id)
      REFERENCES enrollment (student_id, section_id)
```

GRADE		
STUDENT_ID (FK)(FK)	NUMBER(8,0)	NOT NULL
SECTION_ID (FK)(FK)	NUMBER(8,0)	NOT NULL
GRADE_TYPE_CODE (FK)(FK)	CHAR(2)	NOT NULL
GRADE_CODE_OCCURRENCE (FK)	NUMBER(38,0)	NOT NULL
NUMERIC_GRADE	NUMBER(3,0)	NOT NULL
COMMENTS	VARCHAR2(2000)	NULL
CREATED_BY	VARCHAR2(30)	NOT NULL
CREATED_DATE	DATE	NOT NULL
MODIFIED_BY	VARCHAR2(30)	NOT NULL
MODIFIED_DATE	DATE	NOT NULL



ENROLLMENT		
STUDENT_ID (FK)(FK)	NUMBER(8,0)	NOT NULL
SECTION_ID (FK)(FK)	NUMBER(8,0)	NOT NULL
ENROLL_DATE	DATE	NOT NULL
FINAL_GRADE	NUMBER(3,0)	NULL
CREATED_BY	VARCHAR2(30)	NOT NULL
CREATED_DATE	DATE	NOT NULL
MODIFIED_BY	VARCHAR2(30)	NOT NULL
MODIFIED_DATE	DATE	NOT NULL

30



## Enabling/Disabling Constraints

- Generally dangerous to disable constraints
- eg: importing rows
  - do data cleansing in source table
  - disable constraints in destination table
  - import rows from source table into destination table
  - enable constraints in destination table

```
SQL> ALTER TABLE writer  
      DISABLE CONSTRAINT writer_gender_ck;
```

```
SQL> ALTER TABLE writer  
      ENABLE CONSTRAINT writer_gender_ck;
```

31



## Determining Which Rows Violate a Constraint

- A table was created but neglected to create constraints to protect the integrity of its data. How can you determine which rows violate a:
  - primary key constraint?
  - unique constraint?
  - check constraint?
  - foreign key constraint?

32