SQL **Data Definition Language**

Dr. Xu Yang
Room. A323, Chi Un Building

Objectives

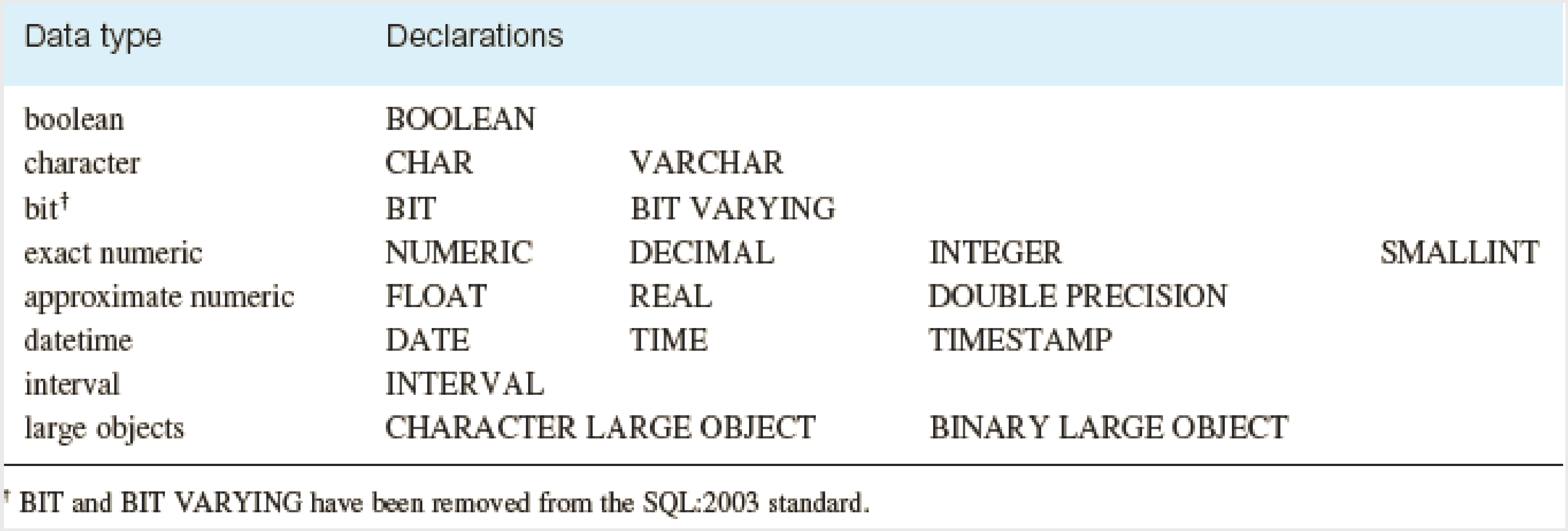
- Naming conventions
- Create and maintain tables by using the CREATE, ALTER, DROP, RENAME, and TRUNCATE statements
- Use the data dictionary to view and maintain information on tables
- Describe the data types that can be used when specifying column definitions
- Create and maintain integrity constraints

Naming Conventions

- **Table and column names: (Oracle 11g Release)**
  - Must begin with a letter
  - Can be no longer than 30 bytes;
  - Must contain only A–Z, a–z, 0–9, _, $, and #
  - Must not duplicate the name of another object owned by the same user
  - Must not be Oracle Server reserved words
  - Case insensitive, unless enclosed in double quotes. (avoid quotes)

http://docs.oracle.com/cd/B28359_01/server.111/b28286/sql_elements008.htm#SQLRF51110

ISO SQL data types



Data type	Declarations			
boolean	BOOLEAN			
character	CHAR	VARCHAR		
bit[†]	BIT	BIT VARYING		
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION	
datetime	DATE	TIME	TIMESTAMP	
interval	INTERVAL			
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT	

[†] BIT and BIT VARYING have been removed from the SQL:2003 standard.

Basic Data Types in Oracle

String Data Types			
• Fixed length	char(*length*)	`1 to 2000 characters`	`char(30)`
• Variable length	varchar2(*maximum-length*)	`1 to 4000 characters`	`varchar2(30)`
Numeric Data Types			

•Number	number(*overall, d*)
	Where overall = total number length,
	d = number of digits to the right of the decimal point
		`Overall: 1 to 38;`
		`d: -84 to 127`
			`number(5,2)`
			`Can not be larger than 999.99`

Date Data Types			
• Date	date		*MM/DD/YYYY*

http://docs.oracle.com/cd/B28359_01/server.111/b28286/sql_elements001.htm#SQLRF50950

Data Definition

- SQL DDL allows database objects such as schemas, domains, tables, views, and indexes to be created and destroyed.
- Main SQL DDL statements are:

CREATE SCHEMA	DROP SCHEMA
CREATE/ALTER DOMAIN	DROP DOMAIN
CREATE/ALTER TABLE	DROP TABLE
CREATE VIEW	DROP VIEW

- Many DBMSs also provide:

CREATE INDEX DROP INDEX

The Create Table Statement

```
CREATE  TABLE [schema.]table_name
     (column_name datatype [DEFAULT expr]
             [column_constraint],
                  ...
             [table_constraint]);
```

schema	is the same as the owner's name.
table/ column	is the name of the table/column
datatype	is the column's datatype and length.
DEFAULT expr	specifies a default value if a value is omitted in the INSERT statement.
column_constraint	is an integrity constraint as part of the column.
table_constraint	is an integrity constraint as part of the table definition.

CREATE TABLE

- Creates a table with one or more columns of the specified *dataType*.
- With NOT NULL, system rejects any attempt to insert a null in the column.
- Can specify a DEFAULT value for the column.
- Primary keys should always be specified as NOT NULL. (some platform only)
- FOREIGN KEY clause specifies FK along with the referential action.

Creating Tables

Create the table.

```
SQL> CREATE TABLE dept
            (deptno  NUMBER(2),
             dname   VARCHAR2(14),
             loc     VARCHAR2(13));
Table created.
```

Confirm table creation

```
SQL> DESCRIBE dept
```

```
    Name                             Null?     Type
 --------------------------- -------- ---------
 DEPTNO                                     NUMBER(2)
 DNAME                                   VARCHAR2(14)
 LOC                                     VARCHAR2(13)
```

The DEFAULT Option

You can specify a default value for a column used during an insert:

```
... hiredate DATE DEFAULT SYSDATE, ...
```

- The default datatype must match the column datatype.
- This option prevents null values from entering the columns if a row is inserted without a value for the column.

Example of CREATE TABLE Statement

```
SQL> create table computer_products
     (model_number            varchar2(12)   primary key,
     product_description      varchar2(50)   default 'N/A',
     list_price               number(6,2)    default 0,
     retail_price             number(6,2)    default 0,
     retail_unit              char(4)        default 'N/A',
     stock_on_hand            number(2,0)    default 0,
     stock_on_order           number(2,0)    default 0,
     last_shipment_received   date,
     manufacturer_code        varchar2(3));

     Table created.
```

Column name | Data type | Constraint

Integrity Constraints

- **Integrity constraints:**
  - **Required data-NOT NULL**
  - **Domain constraints-CHECK**
  - **Entity integrity-PRIMARY KEY**
  - **Referential integrity-FOREIGN KEY**
  - **General Constraints**
- These constraints can be defined in the CREATE and ALTER TABLE statements

Constraint Guidelines

- Name a constraint or the Oracle Server will generate a name by using the SYS_C*n* format.
  - Constraint names must follow the standard object-naming rules.
  - For example, name a NOT NULL constraint on the EMP table DEPTNO column, to EMP_DEPTNO_NN (or NN_EMP_DEPTNO).
- Create a constraint:
  - At the same time as the table is created
  - After the table has been created
- Define a constraint at the column or table level.

Defining Constraints

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

– Column-level constraint

```
column [CONSTRAINT constraint_name] constraint_type,
```

– Table-level constraint

```
column,...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

The Required Data Constraint-NOT NULL

- The NOT NULL constraint ensures that null values are not permitted for the column.

```
... ENAME VARCHAR2(30) CONSTRAINT EMP_ENAME_NN NOT NULL..
... ENAME VARCHAR2(30) NOT NULL
```

EMP

```
EMPNO  ENAME   JOB          ...   COMM   DEPTNO

 7839  KING    PRESIDENT                    10
 7698  BLAKE   MANAGER                      30
 7782  CLARK   MANAGER                      10
 7566  JONES   MANAGER                      20
                                ...
```

NOT NULL constraint (no row may contain a null value for this column)

Absence of NOT NULL constraint (any row can contain null for this column)

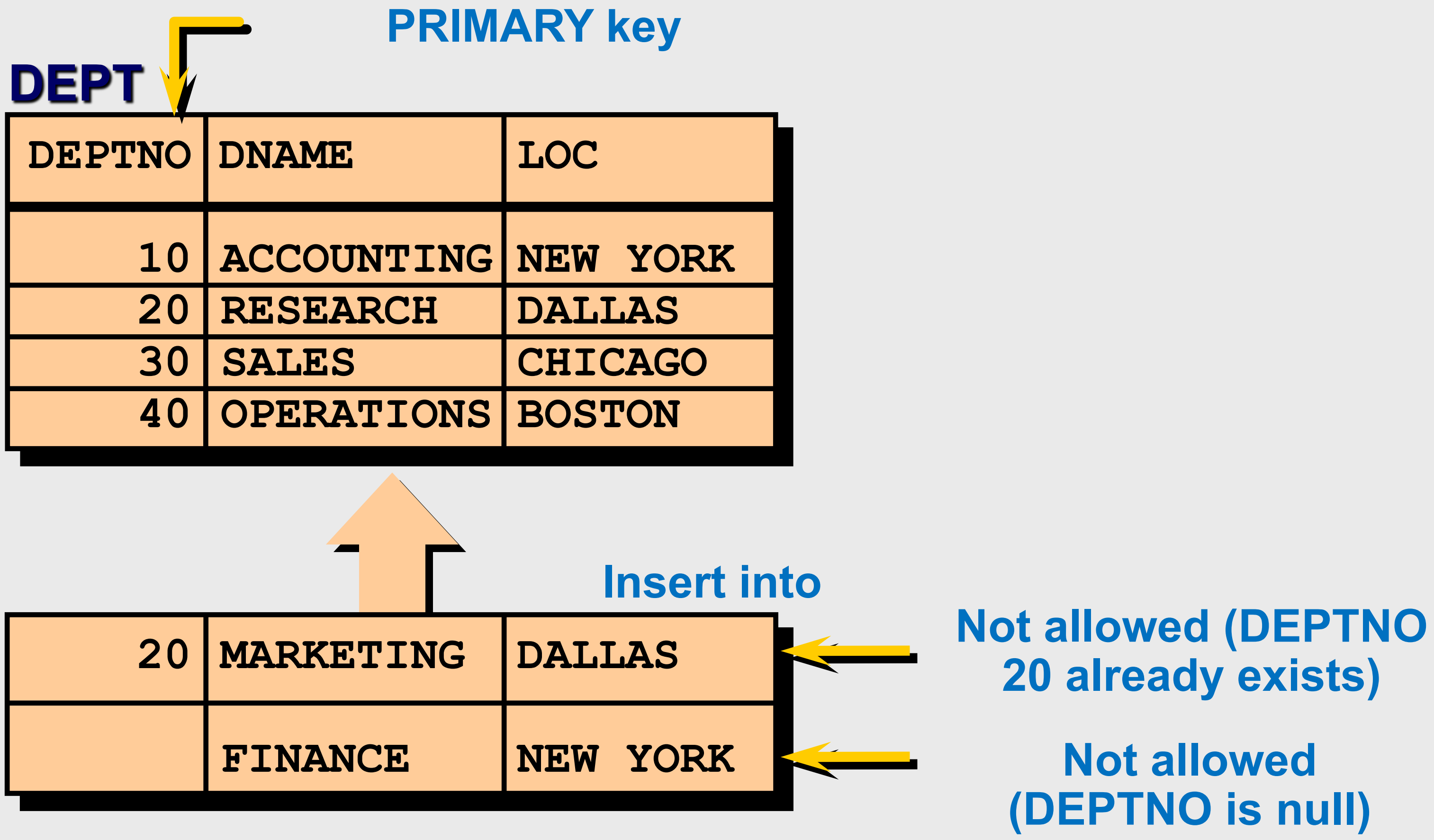
Domain Constraints-CHECK

Domain Constraints ensures that values assigned to a column must be from a defined domain

```
..., deptno   NUMBER(2),
     CONSTRAINT emp_deptno_ck
          CHECK (DEPTNO BETWEEN 10 AND 99),...
```

```
SQL> CREATE TABLE REVENUES
    (TRANSACTION_NUMBER ROWID PRIMARY KEY,
     TRANSACTION_DATE DATE NOT NULL,
     TRANSACTION_TYPE CHAR(1) CONSTRAINT  TRANS_TYPE_CK
     CHECK (TRANSACTION_TYPE IN('R','S','E','A','X')));

Table created.
```

The Entity Constraint-PRIMARY KEY


PRIMARY key
DEPT
Insert into
Not allowed (DEPTNO 20 already exists)
Not allowed (DEPTNO is null)

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

20	MARKETING	DALLAS
	FINANCE	NEW YORK

PRIMARY KEY

- **Primary key of a table must contain a unique, non-null value for each row.**

  Can only have one PRIMARY KEY clause per table. Can still ensure uniqueness for alternate keys using UNIQUE:

```
    SQL> CREATE TABLE dept(
               deptno        NUMBER(2),
               dname  VARCHAR2(14) NOT NULL,
               loc    VARCHAR2(13),
               CONSTRAINT dept_dname_uk UNIQUE(dname),
               CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno));

Table created.
```

Entity Integrity

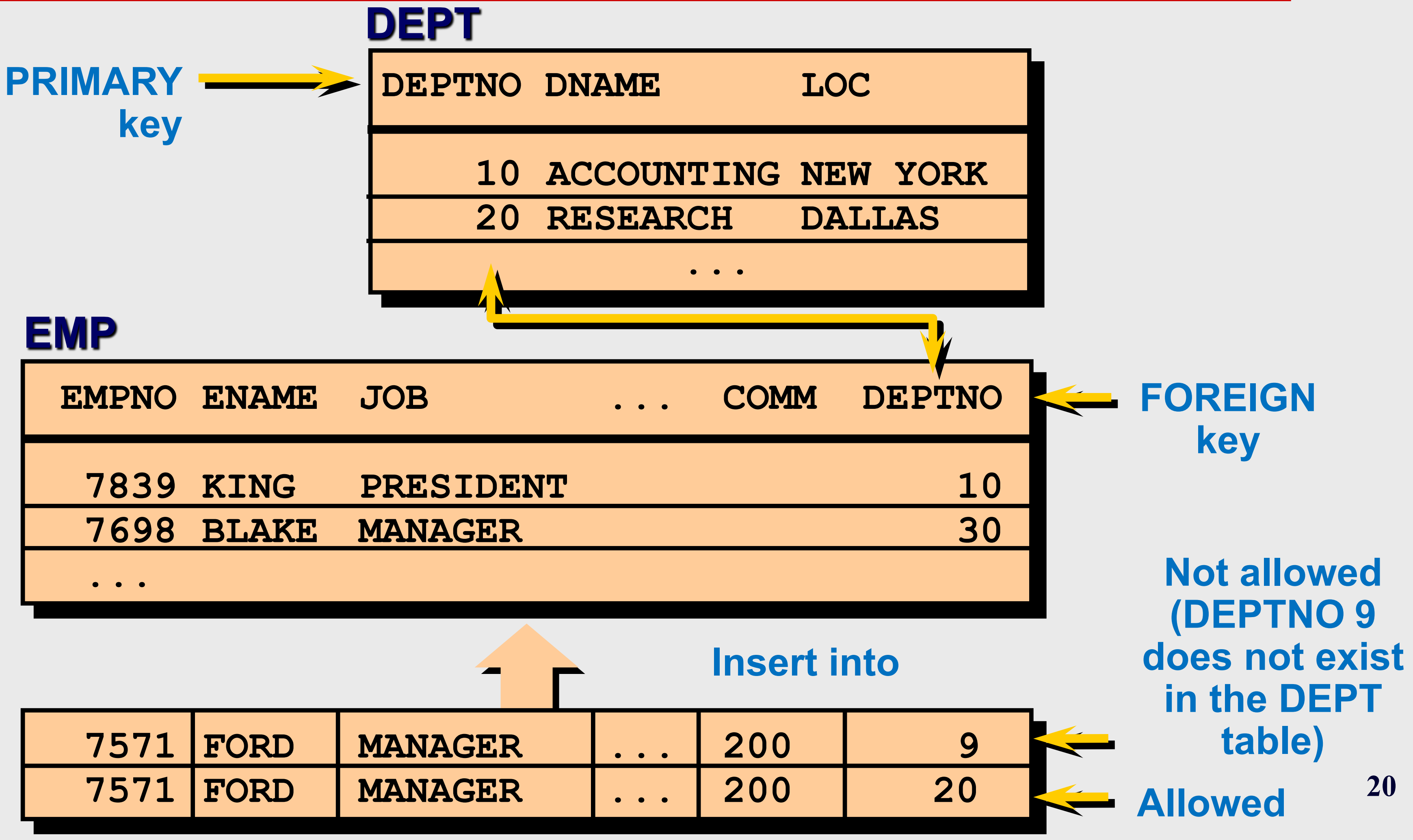
Composite Key

Neither column is unique by itself

```
SQL> create table invoice_items
     (invoice_number int,
     item_number int constraint  invoice_number_ck
        check (item_number in (1,2,3,4,5,6,7,8,9,10)),
     product_code varchar2(10),
     quantity int default 1,
     price float not null,
    constraint  invoice_items_pk primary key
       (invoice_number, item_number),
    constraint  invoice_number_fk
       foreign key (invoice_number)
       references invoices(invoice_number),
    constraint product_code_fk
       foreign key (product_code)
       references products(product_code));

Table created.
```

Composite key

Referential Integrity Constraint- FOREIGN KEY



DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
	...	

EMP

EMPNO	ENAME	JOB	...	COMM	DEPTNO
7839	KING	PRESIDENT			10
7698	BLAKE	MANAGER			30
...					

7571	FORD	MANAGER	...	200	9
7571	FORD	MANAGER	...	200	20

FOREIGN Key

- FK is column or set of columns that links each row in child table containing foreign FK to row of parent table containing the matching candidate key value.
- Referential integrity means that, if FK contains a value, that value must refer to existing row in parent table.

```
SQL> CREATE TABLE emp(
         empno     NUMBER(4),
         ename     VARCHAR2(10) NOT NULL,
         job       VARCHAR2(9),
         mgr       NUMBER(4),
         hiredate  DATE,
         sal       NUMBER(7,2),
         comm      NUMBER(7,2),
         deptno    NUMBER(2) NOT NULL,
         CONSTRAINT emp_deptno_fk FOREIGN KEY (deptno)
                    REFERENCES dept (deptno));
```

FOREIGN Key

- Any INSERT/UPDATE attempting to create FK value in child table without matching Candidate Key value in parent is rejected.
- Action taken attempting to update/delete a Candidate Key value in parent table with matching rows in child is dependent on **referential action** specified using ON UPDATE and ON DELETE subclauses:
  - **CASCADE**
  - **SET NULL**
  - **SET DEFAULT**
  - **NO ACTION**

Referential Actions

CASCADE: Delete row from parent and delete matching rows in child, and so on in cascading manner.

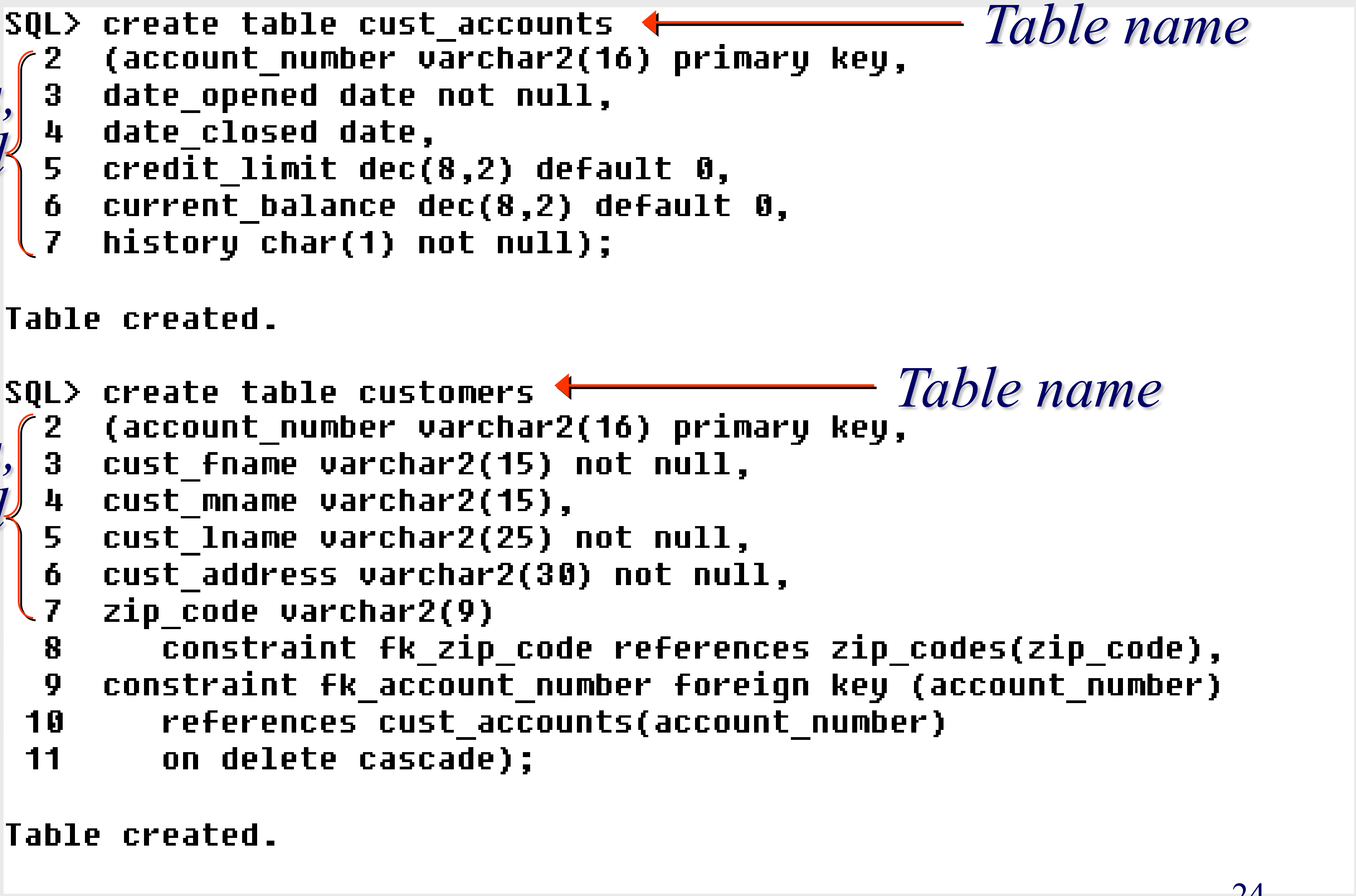
SET NULL: Delete row from parent and set FK column(s) in child to NULL. Only valid if FK columns do not have the NOT NULL qualifier specified.

SET DEFAULT: Delete row from parent and set each component of FK in child to specified default. Only valid if DEFAULT specified for FK columns.

NO ACTION: Reject delete from parent. Default.

CONSTRAINT *index-name* FOREIGN KEY (*column-name*)
REFERENCES *table-name*(*key-name*)
ON DELETE CASCADE

An Example of Creating a Table with Referential Integrity Constraints



```
SQL> create table cust_accounts
  2  (account_number varchar2(16) primary key,
  3  date_opened date not null,
  4  date_closed date,
  5  credit_limit dec(8,2) default 0,
  6  current_balance dec(8,2) default 0,
  7  history char(1) not null);

Table created.

SQL> create table customers
  2  (account_number varchar2(16) primary key,
  3  cust_fname varchar2(15) not null,
  4  cust_mname varchar2(15),
  5  cust_lname varchar2(25) not null,
  6  cust_address varchar2(30) not null,
  7  zip_code varchar2(9)
  8     constraint fk_zip_code references zip_codes(zip_code),
  9  constraint fk_account_number foreign key (account_number)
 10     references cust_accounts(account_number)
 11     on delete cascade);

Table created.
```

Table name

Column names, data types and constraints

Table name

Column names, data types and constraints

①

②

Tables in the Oracle Database

- User tables
  - Collection of tables created and maintained by the user
  - Contain user information
- Data dictionary
  - Collection of tables created and maintained by the Oracle Server
  - Contain database information

Querying the Data Dictionary

- Describe tables owned by the user.

```
SQL> SELECT   *
     FROM     user_tables;
```

- View distinct object types owned by the user.

```
SQL> SELECT   DISTINCT object_type
     FROM     user_objects;
```

- View tables, views, synonyms, and sequences owned by the user.

```
SQL> SELECT   *
     FROM     user_catalog;
```

Viewing Constraints

- Query the USER_CONSTRAINTS table to view all constraint definitions and names.

```
SQL>   SELECT  constraint_name, constraint_type,
               search_condition
       FROM    user_constraints
       WHERE   table_name = 'EMP';
```

- View the columns associated with the constraint names in the USER_CONS_COLUMNS view.

```
SQL> SELECT   constraint_name, column_name
     FROM     user_cons_columns
     WHERE    table_name = 'EMP';
```

Referencing Another User’s Tables

- Tables belonging to other users are not in the user’s schema.
- You should use the owner’s name as a prefix to the table.
- Constraints must reference tables in the same database.

Creating a Table by Using a Subquery

- Create a table and insert rows by combining the CREATE TABLE statement and AS *subquery* option.

```
CREATE TABLE table
        [column(, column...)]
AS subquery;
```

  - *Subquery* is the SELECT statement that defines the set of rows to be inserted into the new table.
  - Match the number of specified columns to the number of subquery columns.
  - Define columns with column names and default values and integrity constraints, not the datatype or referential integrity constraints (Foreign Key).

Creating a Table by Using a Subquery

```
SQL> CREATE TABLE dept30
     AS
       SELECT  empno, ename, sal * 12 ANNSAL, hiredate
       FROM    emp
       WHERE   deptno = 30;
Table created.
```

```
SQL> DESCRIBE dept30
```

```
 Name                            Null?    Type
 ------------------------------- -------- -----
 EMPNO                           NOT NULL NUMBER(4)
 ENAME                                    VARCHAR2(10)
 ANNSAL                                   NUMBER
 HIREDATE                                 DATE
```

The Alter Table Statement

- Use the ALTER TABLE statement to:
  - Add, modify, or remove columns
  - Add or remove constraints
  - Enable or disable constraints
  - Define a default value for the new column

```
ALTER TABLE table
ADD (column datatype [DEFAULT expr] [NOT NULL]
    [, column datatype]...);
```

```
ALTER TABLE table
MODIFY (column datatype [DEFAULT expr] [NOT NULL]
       [, column datatype]...);
```

Adding a Column

```
SQL> ALTER TABLE dept30
2    ADD (job VARCHAR2(9));
```

DEPT30

EMPNO	ENAME	ANNSAL	HIREDATE
7698	BLAKE	34200	01-MAY-81
7654	MARTIN	15000	28-SEP-81
7499	ALLEN	19200	20-FEB-81
7844	TURNER	18000	08-SEP-81
...			

New column

JOB

"…add a new column into DEPT30 table…"

DEPT30

EMPNO	ENAME	ANNSAL	HIREDATE	JOB
7698	BLAKE	34200	01-MAY-81	
7654	MARTIN	15000	28-SEP-81	
7499	ALLEN	19200	20-FEB-81	
7844	TURNER	18000	08-SEP-81	
...				

Modify a Column

- Modify a column definition by using the ALTER TABLE statement with the MODIFY clause.
  - You can change a column's datatype, size, default value, and NOT NULL column constraint.

```
SQL> ALTER TABLE emp
     MODIFY     (job VARCHAR2(50));
Table altered.
```

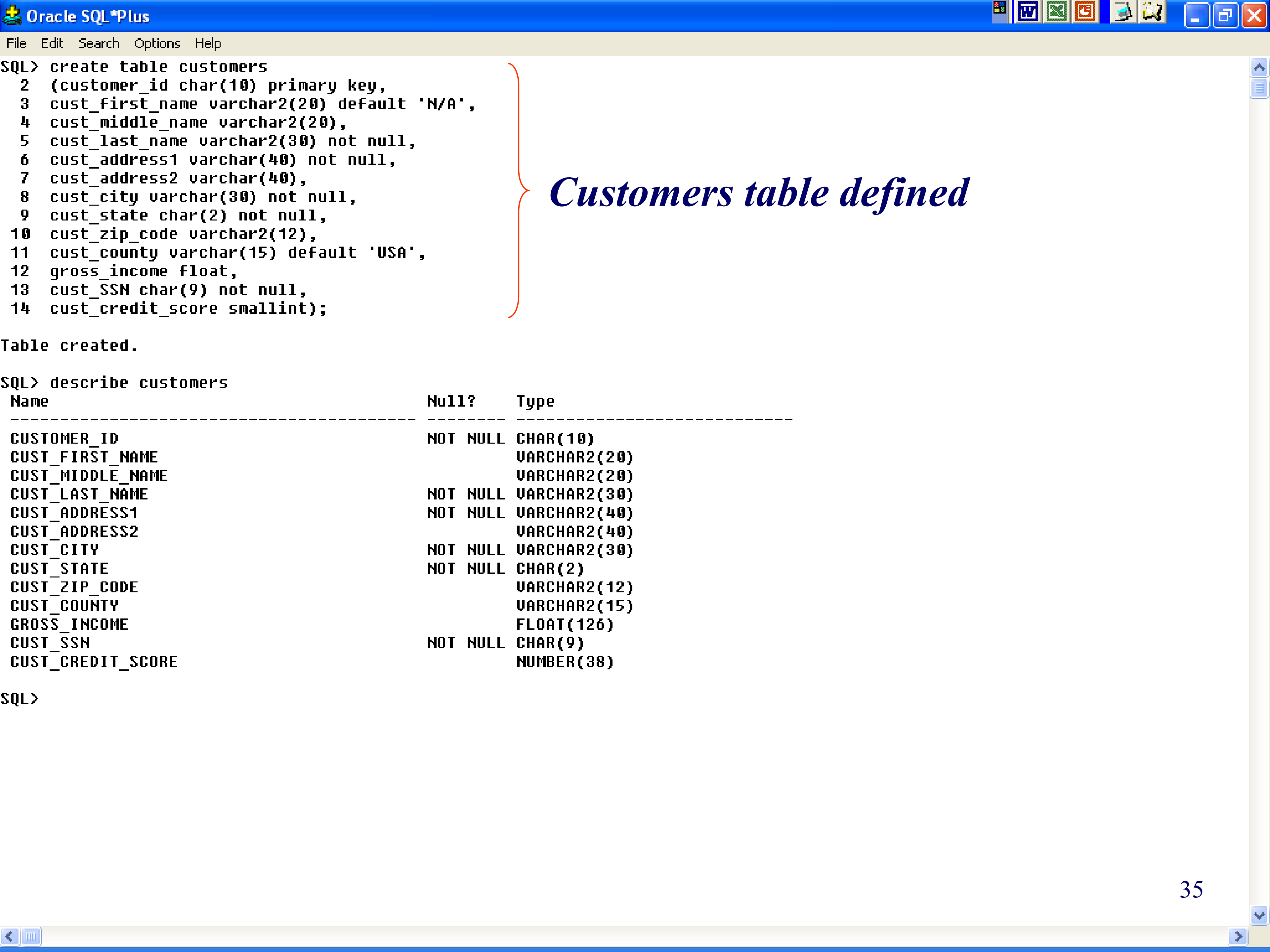
  - A change to the default value affects only subsequent insertions to the table.

Dropping a Column

- You use the DROP COLUMN clause TO free space in the database by dropping columns you no longer need.

```
SQL> ALTER TABLE     dept30
     DROP   COLUMN   job ;
Table altered.
```

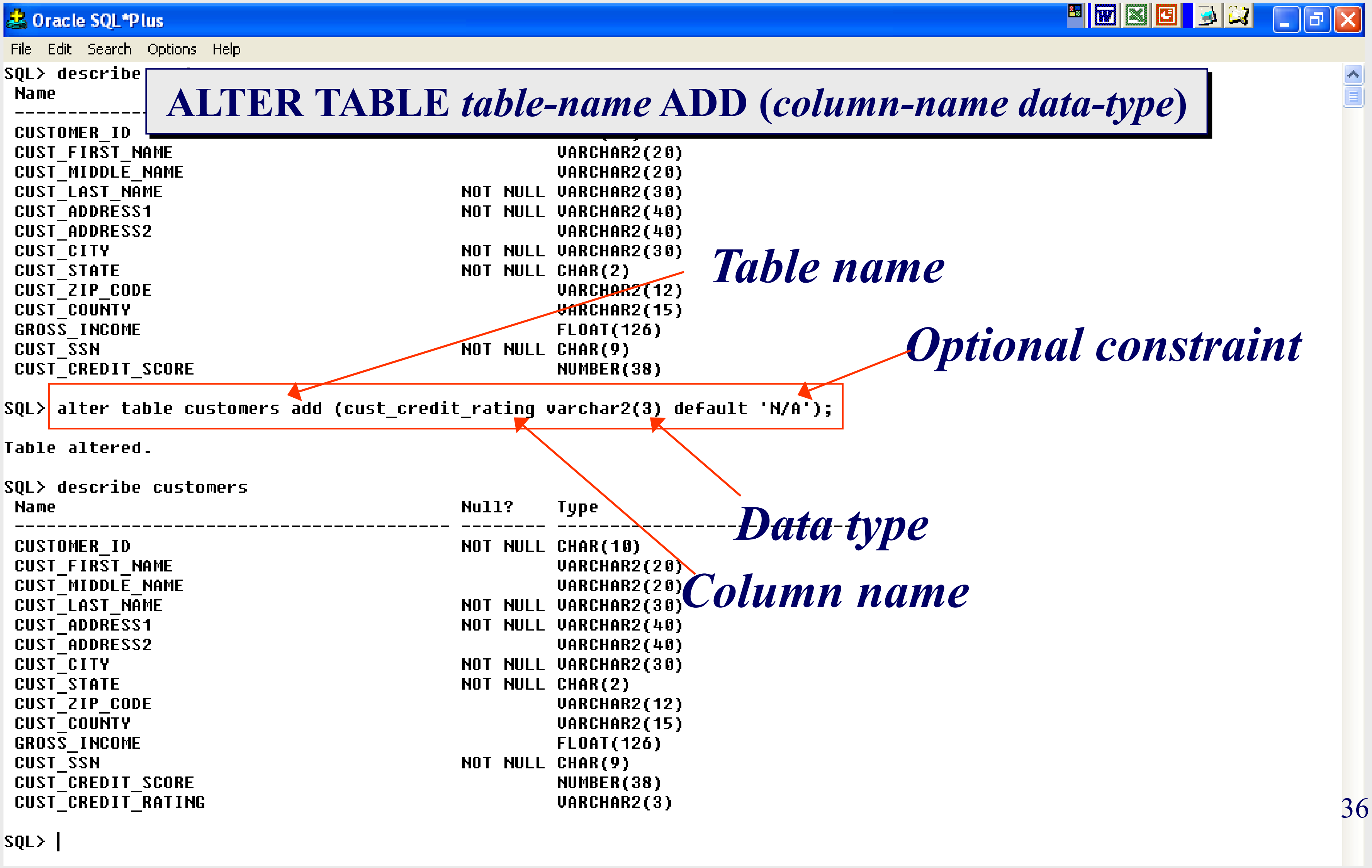
- – The column may or may not contain data.
- – Only one column can be dropped at a time.
- – The table must have at least one column remaining in it after it is altered.
- – Once a column is dropped, it cannot be recovered.



```
SQL> create table customers
(customer_id char(10) primary key,
cust_first_name varchar2(20) default 'N/A',
cust_middle_name varchar2(20),
cust_last_name varchar2(30) not null,
cust_address1 varchar(40) not null,
cust_address2 varchar(40),
cust_city varchar(30) not null,
cust_state char(2) not null,
cust_zip_code varchar2(12),
cust_county varchar(15) default 'USA',
gross_income float,
cust_SSN char(9) not null,
cust_credit_score smallint);

Table created.

SQL> describe customers
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                               NOT NULL CHAR(10)
 CUST_FIRST_NAME                                    VARCHAR2(20)
 CUST_MIDDLE_NAME                                   VARCHAR2(20)
 CUST_LAST_NAME                            NOT NULL VARCHAR2(30)
 CUST_ADDRESS1                             NOT NULL VARCHAR2(40)
 CUST_ADDRESS2                                      VARCHAR2(40)
 CUST_CITY                                 NOT NULL VARCHAR2(30)
 CUST_STATE                                NOT NULL CHAR(2)
 CUST_ZIP_CODE                                      VARCHAR2(12)
 CUST_COUNTY                                        VARCHAR2(15)
 GROSS_INCOME                                       FLOAT(126)
 CUST_SSN                                  NOT NULL CHAR(9)
 CUST_CREDIT_SCORE                                  NUMBER(38)

SQL>
```

Customers table defined

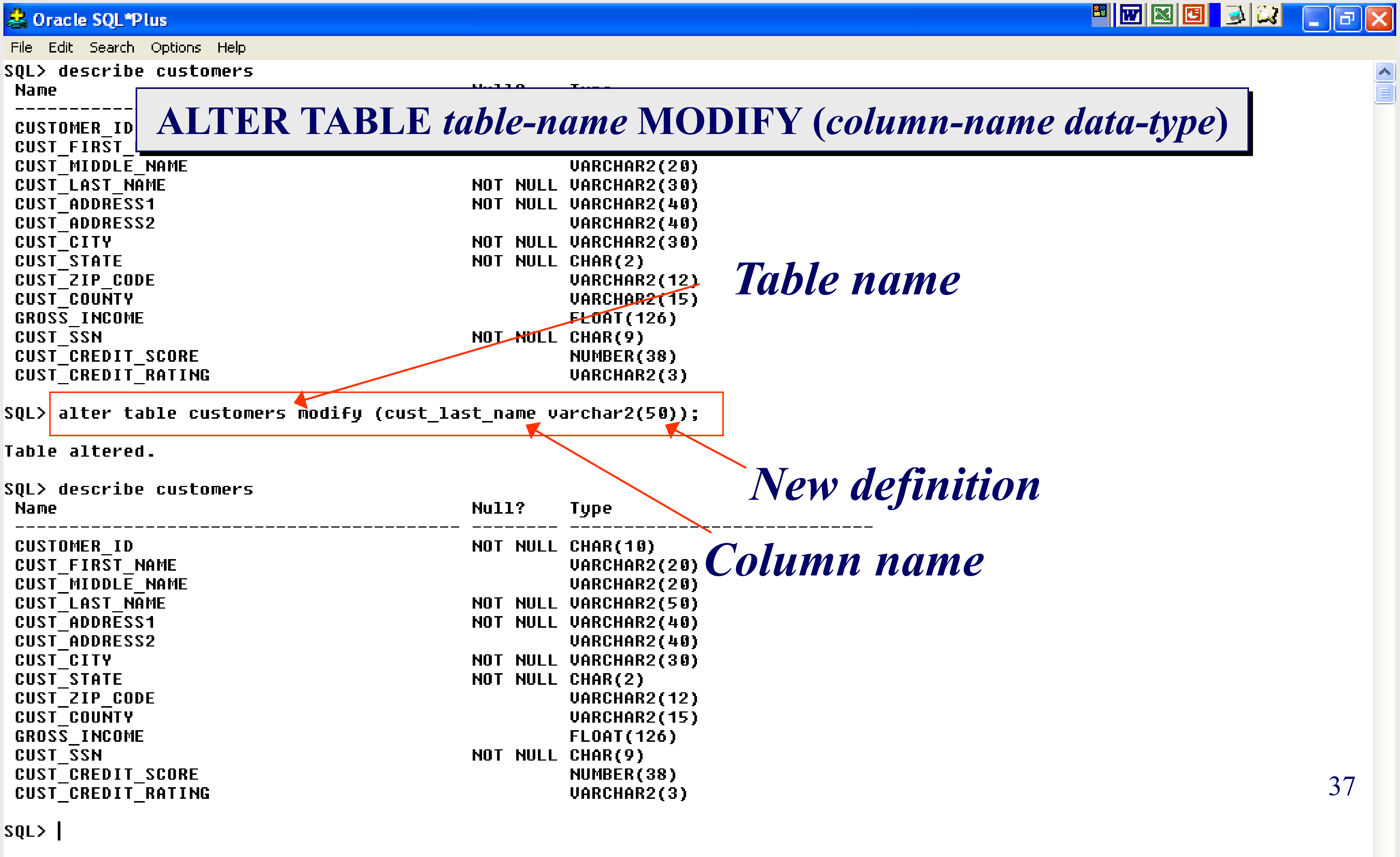
Adding a Column



ALTER TABLE *table-name* ADD (*column-name data-type*)

```
SQL> describe
 Name
 ----------------
 CUSTOMER_ID
 CUST_FIRST_NAME                                    VARCHAR2(20)
 CUST_MIDDLE_NAME                                   VARCHAR2(20)
 CUST_LAST_NAME                            NOT NULL VARCHAR2(30)
 CUST_ADDRESS1                             NOT NULL VARCHAR2(40)
 CUST_ADDRESS2                                      VARCHAR2(40)
 CUST_CITY                                 NOT NULL VARCHAR2(30)
 CUST_STATE                                NOT NULL CHAR(2)
 CUST_ZIP_CODE                                      VARCHAR2(12)
 CUST_COUNTY                                        VARCHAR2(15)
 GROSS_INCOME                                       FLOAT(126)
 CUST_SSN                                  NOT NULL CHAR(9)
 CUST_CREDIT_SCORE                                  NUMBER(38)

SQL> alter table customers add (cust_credit_rating varchar2(3) default 'N/A');

Table altered.

SQL> describe customers
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                               NOT NULL CHAR(10)
 CUST_FIRST_NAME                                    VARCHAR2(20)
 CUST_MIDDLE_NAME                                   VARCHAR2(20)
 CUST_LAST_NAME                            NOT NULL VARCHAR2(30)
 CUST_ADDRESS1                             NOT NULL VARCHAR2(40)
 CUST_ADDRESS2                                      VARCHAR2(40)
 CUST_CITY                                 NOT NULL VARCHAR2(30)
 CUST_STATE                                NOT NULL CHAR(2)
 CUST_ZIP_CODE                                      VARCHAR2(12)
 CUST_COUNTY                                        VARCHAR2(15)
 GROSS_INCOME                                       FLOAT(126)
 CUST_SSN                                  NOT NULL CHAR(9)
 CUST_CREDIT_SCORE                                  NUMBER(38)
 CUST_CREDIT_RATING                                 VARCHAR2(3)

SQL> |
```

Modifying a Column Definition



ALTER TABLE *table-name* MODIFY (*column-name data-type*)

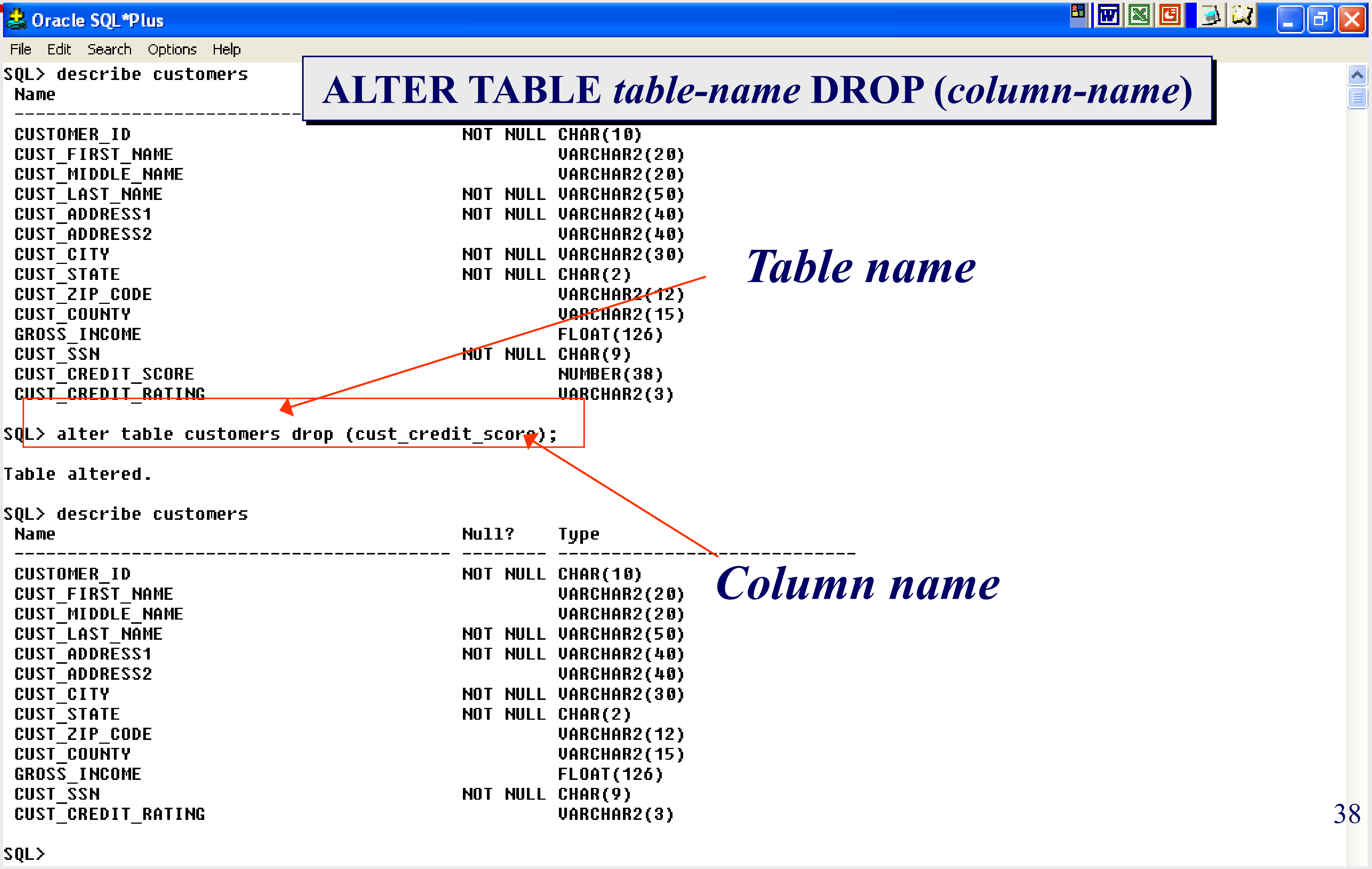
```
SQL> describe customers
 Name                                      Null?    Type
 ------------
 CUSTOMER_ID
 CUST_FIRST_
 CUST_MIDDLE_NAME                                   VARCHAR2(20)
 CUST_LAST_NAME                            NOT NULL VARCHAR2(30)
 CUST_ADDRESS1                             NOT NULL VARCHAR2(40)
 CUST_ADDRESS2                                      VARCHAR2(40)
 CUST_CITY                                 NOT NULL VARCHAR2(30)
 CUST_STATE                                NOT NULL CHAR(2)
 CUST_ZIP_CODE                                      VARCHAR2(12)
 CUST_COUNTY                                        VARCHAR2(15)
 GROSS_INCOME                                       FLOAT(126)
 CUST_SSN                                  NOT NULL CHAR(9)
 CUST_CREDIT_SCORE                                  NUMBER(38)
 CUST_CREDIT_RATING                                 VARCHAR2(3)

SQL> alter table customers modify (cust_last_name varchar2(50));

Table altered.

SQL> describe customers
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                               NOT NULL CHAR(10)
 CUST_FIRST_NAME                                    VARCHAR2(20)
 CUST_MIDDLE_NAME                                   VARCHAR2(20)
 CUST_LAST_NAME                            NOT NULL VARCHAR2(50)
 CUST_ADDRESS1                             NOT NULL VARCHAR2(40)
 CUST_ADDRESS2                                      VARCHAR2(40)
 CUST_CITY                                 NOT NULL VARCHAR2(30)
 CUST_STATE                                NOT NULL CHAR(2)
 CUST_ZIP_CODE                                      VARCHAR2(12)
 CUST_COUNTY                                        VARCHAR2(15)
 GROSS_INCOME                                       FLOAT(126)
 CUST_SSN                                  NOT NULL CHAR(9)
 CUST_CREDIT_SCORE                                  NUMBER(38)
 CUST_CREDIT_RATING                                 VARCHAR2(3)

SQL> |
```

Table name

New definition

Column name

Deleting a Column



ALTER TABLE ***table-name*** **DROP (*column-name*)**

Table name

Column name

```
SQL> describe customers
 Name
 ----------------------------
 CUSTOMER_ID                               NOT NULL CHAR(10)
 CUST_FIRST_NAME                                    VARCHAR2(20)
 CUST_MIDDLE_NAME                                   VARCHAR2(20)
 CUST_LAST_NAME                            NOT NULL VARCHAR2(50)
 CUST_ADDRESS1                             NOT NULL VARCHAR2(40)
 CUST_ADDRESS2                                      VARCHAR2(40)
 CUST_CITY                                 NOT NULL VARCHAR2(30)
 CUST_STATE                                NOT NULL CHAR(2)
 CUST_ZIP_CODE                                      VARCHAR2(12)
 CUST_COUNTY                                        VARCHAR2(15)
 GROSS_INCOME                                       FLOAT(126)
 CUST_SSN                                  NOT NULL CHAR(9)
 CUST_CREDIT_SCORE                                  NUMBER(38)
 CUST_CREDIT_RATING                                 VARCHAR2(3)

SQL> alter table customers drop (cust_credit_score);

Table altered.

SQL> describe customers
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                               NOT NULL CHAR(10)
 CUST_FIRST_NAME                                    VARCHAR2(20)
 CUST_MIDDLE_NAME                                   VARCHAR2(20)
 CUST_LAST_NAME                            NOT NULL VARCHAR2(50)
 CUST_ADDRESS1                             NOT NULL VARCHAR2(40)
 CUST_ADDRESS2                                      VARCHAR2(40)
 CUST_CITY                                 NOT NULL VARCHAR2(30)
 CUST_STATE                                NOT NULL CHAR(2)
 CUST_ZIP_CODE                                      VARCHAR2(12)
 CUST_COUNTY                                        VARCHAR2(15)
 GROSS_INCOME                                       FLOAT(126)
 CUST_SSN                                  NOT NULL CHAR(9)
 CUST_CREDIT_RATING                                 VARCHAR2(3)

SQL>
```

Adding a Constraint

- Add constraints to a table or column by using the ALTER TABLE statement with the ADD clause:

```
ALTER TABLE table
ADD [CONSTRAINT constraint] type (column);
```

– *table*	is the name of the table.
– *constraint*	is the name of the constraint.
– *type*	is the constraint type.
– *column*	is the name of the column affected by the constraint

– You can add or drop but not modify the structure of a constraint.

– You can enable or disable constraints.

– You can add a NOT NULL constraint by using the MODIFY clause.

Adding a Constraint

- Add a FOREIGN KEY constraint to the EMP table, indicating that a manager must already exist as a valid employee in the EMP table.

```
SQL> ALTER TABLE     emp
     ADD CONSTRAINT emp_mgr_fk
             FOREIGN KEY(mgr) REFERENCES emp(empno);
Table altered.
```

Dropping a Constraint

- Remove the manager constraint from the EMP table.

```
SQL> ALTER TABLE      emp
     DROP CONSTRAINT emp_mgr_fk;
Table altered.
```

- Remove the PRIMARY KEY constraint on the DEPT table and drop the associated FOREIGN KEY constraint on the EMP.DEPTNO column.

```
SQL> ALTER TABLE      dept
     DROP PRIMARY KEY CASCADE;
Table altered.
```

» The CASCADE option of the DROP clause causes any dependent constraints also to be dropped.

Disabling Constraints

- Execute the DISABLE clause of the ALTER TABLE statement to deactivate an integrity constraint.
- Apply the CASCADE option to disable dependent integrity constraints.

```
SQL> ALTER TABLE emp
     DISABLE CONSTRAINT emp_empno_pk CASCADE;
Table altered.
```

Enabling Constraints

- Activate an integrity constraint currently disabled in the table definition by using the ENABLE clause.

```
SQL> ALTER TABLE emp
     ENABLE CONSTRAINT emp_empno_pk;
Table altered.
```

- A UNIQUE or PRIMARY KEY index is automatically created if you enable a UNIQUE or PRIMARY KEY constraint.

Cascading Constraints

- You use the CASCADE CONSTRAINTS option to drop all referential integrity constraints that refer to the primary and unique keys on the dropped columns.

Changing the Name of an Object

- To change the name of a table, view, sequence, or synonym, execute the RENAME statement.

```
SQL> RENAME dept TO department;
Table renamed.
```

Truncating a Table

- The TRUNCATE TABLE statement:
  - Removes all rows from a table
  - Releases the storage space used by that table

```
SQL> TRUNCATE TABLE department;
Table truncated.
```

  - You cannot roll back row removal when using TRUNCATE.
  - Alternatively, you can remove rows by using the DELETE statement.
  - The DELETE statement can also remove all rows from a table, but it does not release storage space.

Dropping a Table

- The DROP TABLE statement:
  - Deletes all data and the table structure
  - Commits any pending transactions
  - Drops all indexes

```
SQL> DROP TABLE dept30;
Table dropped.
```

- You cannot roll back this statement

Adding Comments to a Table

- You can add comments to a table or column by using the COMMENT statement.

```
SQL> COMMENT ON TABLE emp
     IS 'Employee Information';
Comment created.
```

- Comments can be viewed through the following data dictionary views:
  – ALL_COL_COMMENTS
  – USER_COL_COMMENTS
  – ALL_TAB_COMMENTS
  – USER_TAB_COMMENTS

Summary

- Create and maintain tables by using the following statements:

Statement	Description
CREATE TABLE	Creates a table
ALTER TABLE	Modifies table structures
DROP TABLE	Removes the rows and table structure
RENAME	Changes the name of a table, view, sequence, or synonym
TRUNCATE	Removes all rows from a table and releases the storage space
COMMENT	Adds comments to a table or view

Summary

- Create the following types of constraints:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK