Lab Week – 3
Creating Tables and Using Constraints

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Objectives

- Creating tables:
  - Describe the main database objects
  - Create tables
  - Describe the data types that can be used when specifying column definition
  - Alter table definitions
  - Drop, rename, and truncate tables
- Using Constraints
  - Describe constraints
  - Create and maintain constraints

Database Objects

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Generates primary key values
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

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 owned by the same user
- Must not be an Oracle Server reserved word

The CREATE TABLE Statement

- You must have :
  - CREATE TABLE privilege
  - A storage area

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

- You specify:
  - Table name
  - Column name, column data type, and column size

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Creating Tables

– Create the table.

```
CREATE TABLE dept
        (deptno      NUMBER(2),
         dname       VARCHAR2(14),
         loc         VARCHAR2(13),
         create_date DATE DEFAULT SYSDATE);
Table created.
```

– Confirm table creation.

```
DESCRIBE dept
```

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)
CREATE_DATE		DATE

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Data Types

Data Type	Description
CHAR(size)	Fixed-length character data
VARCHAR2	Variable-length character data
CLOB	Character data up to 8 Terabytes
NUMBER(p,s)	Variable-length numeric data
DATE	Date and Time values
BLOB	Binary data up to 8 Terabytes
RAW & LONG RAW	Raw binary data
BFILE	Binary data stored in operating-system files outside the database; up to 8 Terabytes

The DEFAULT Option

- Specify a default value for a column, to be used during an insert.
- Legal values are literal value, expression, or SQL function.
- The default datatype must match the column datatype.

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

Creating a table by using a sub-query

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

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

- Match the number of specified columns to the number of subquery columns.
- Define columns with column names and default values.

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Creating a table by using a sub-query

```
SQL> CREATE TABLE   dept30
  2  AS
  3    SELECT    empno, ename, sal*12 ANNSAL, hiredate
  4    FROM      emp
  5    WHERE     deptno = 30;

Table created.
```

```
SQL> DESCRIBE dept30
```

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

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

- List tables owned by the user.

```
SQL> SELECT   table_name
  2  FROM     user_tables;
```

- View distinct object types owned by the user.

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

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

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

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.

```
SELECT *
FROM tsg0781.Customer;
```

The ALTER TABLE Statement

- Use the ALTER TABLE statement to:
  - Add a new column
  - Modify an existing column
  - Define a default value for the new column

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

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

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Adding a Column

DEPT30

New column

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	
...				

"…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	
...				

Adding a Column

- You use the ADD clause to add columns.

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

- The new column becomes the last column.

```
    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
...
6 rows selected.
```

Modifying a Column

- You can change a column's data type, size, and default value.

```
ALTER TABLE dept30
MODIFY      (ename VARCHAR2(15));
Table altered.
```

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

Dropping a Table

- All data and structure in the table is deleted.
- Any pending transactions are committed.
- All indexes are dropped.
- You can not roll back this statement.

```
SQL> DROP TABLE dept30;

Table dropped.
```

Changing the Name of an Object

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

```
SQL> RENAME dept TO my_department;

Table renamed.
```

- You must be the owner of the object.

Truncating a Table

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

```
SQL> TRUNCATE TABLE dept;

Table truncated.
```

- You can not roll back row removal when using TRUNCATE.
- Alternatively, you can remove rows by using the DELETE statement – this will be covered next week.

Adding Comments to a Table

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

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

- Comments can be viewed through the data dictionary views.
  - ALL_COL_COMMENTS
  - USER_COL_COMMENTS
  - ALL_TAB_COMMENTS
  - USER_TAB_COMMENTS

What Are Constraints?

- Constraints enforce rules at the table level.
- Constraints prevent the deletion of a table if there are dependencies.
- The following constraint types are valid in Oracle:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK

Constraint Guidelines

- Name a constraint or the Oracle Server will generate a name by using the SYS_Cn format.
- 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.
- View a constraint in the data dictionary.

Defining Constraints

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

```
CREATE TABLE emp(
              empno   NUMBER(4),
              ename   VARCHAR2(10),
              ...
              deptno  NUMBER(7,2) NOT NULL,
              CONSTRAINT emp_empno_pk
                            PRIMARY KEY (EMPNO));
```

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Defining Constraints

- Column constraint level

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

- Table constraint level

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

The NOT NULL Constraint

- Ensures that null values are not permitted for the column

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 can contain a null value for this column)

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

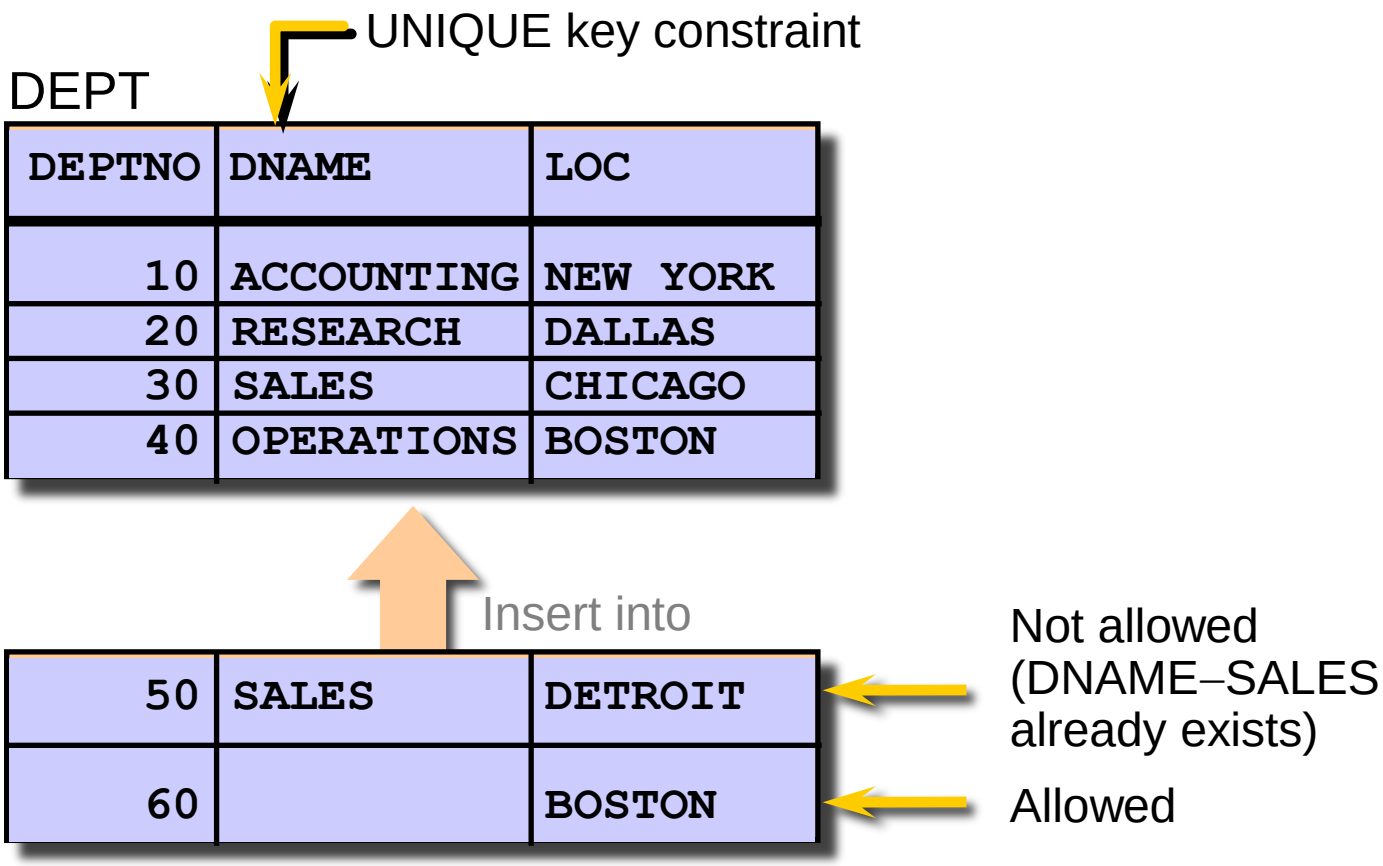
NOT NULL constraint

The NOT NULL Constraint

- Defined at the column level

```
SQL> CREATE TABLE emp(
  2      empno    NUMBER(4),
  3      ename    VARCHAR2(10) NOT NULL,
  4      job      VARCHAR2(9),
  5      mgr      NUMBER(4),
  6      hiredate DATE,
  7      sal      NUMBER(7,2),
  8      comm     NUMBER(7,2),
  9      deptno   NUMBER(7,2) NOT NULL);
```

The UNIQUE Key Constraint

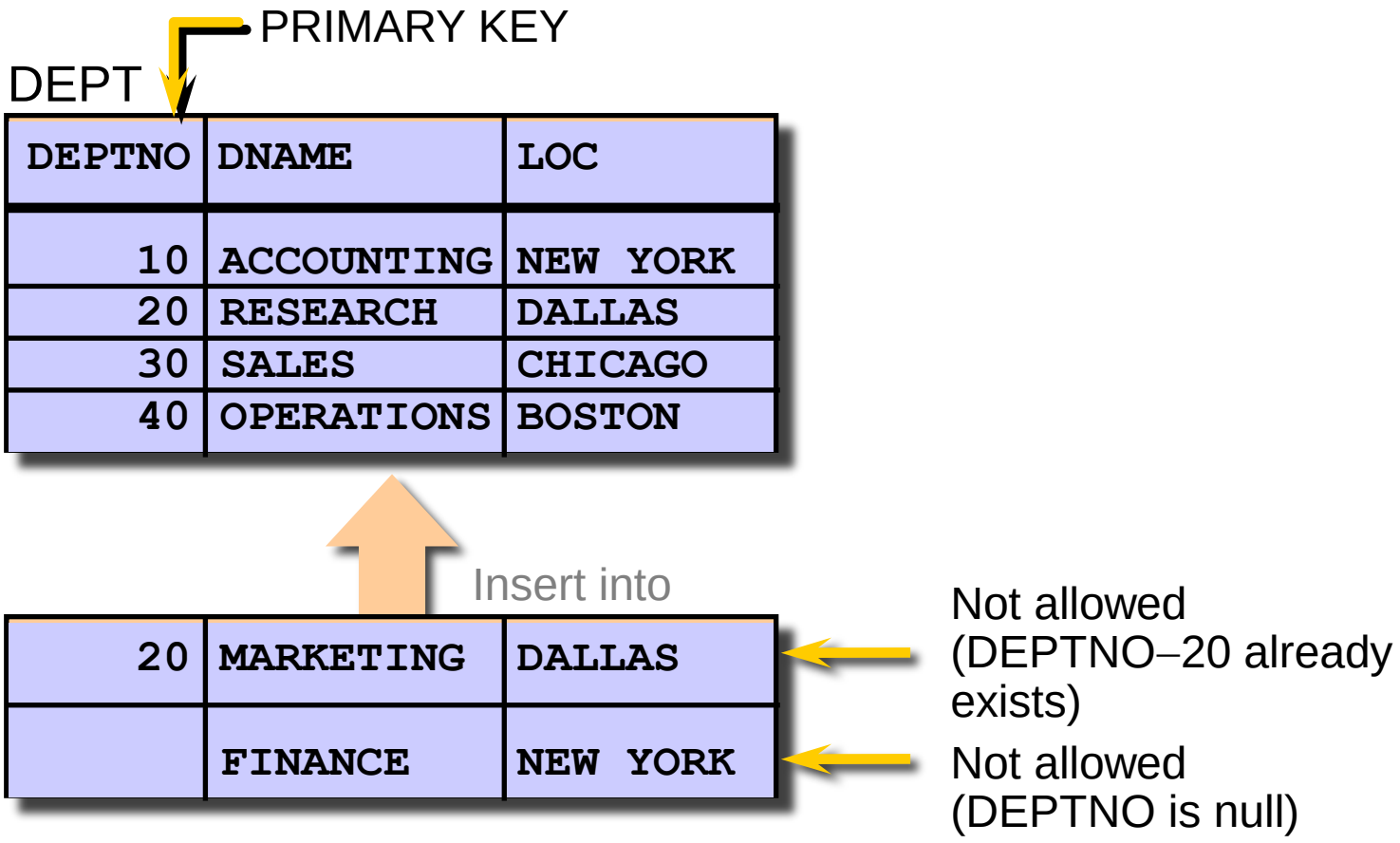


The UNIQUE Key Constraint

- May be defined at either the table level or the column level
- Defined at table level:

```
SQL> CREATE TABLE   dept(
  2      deptno     NUMBER(2),
  3      dname      VARCHAR2(14),
  4      loc        VARCHAR2(13),
  5      CONSTRAINT dept_dname_uk UNIQUE(dname));
```

The PRIMARY KEY Constraint

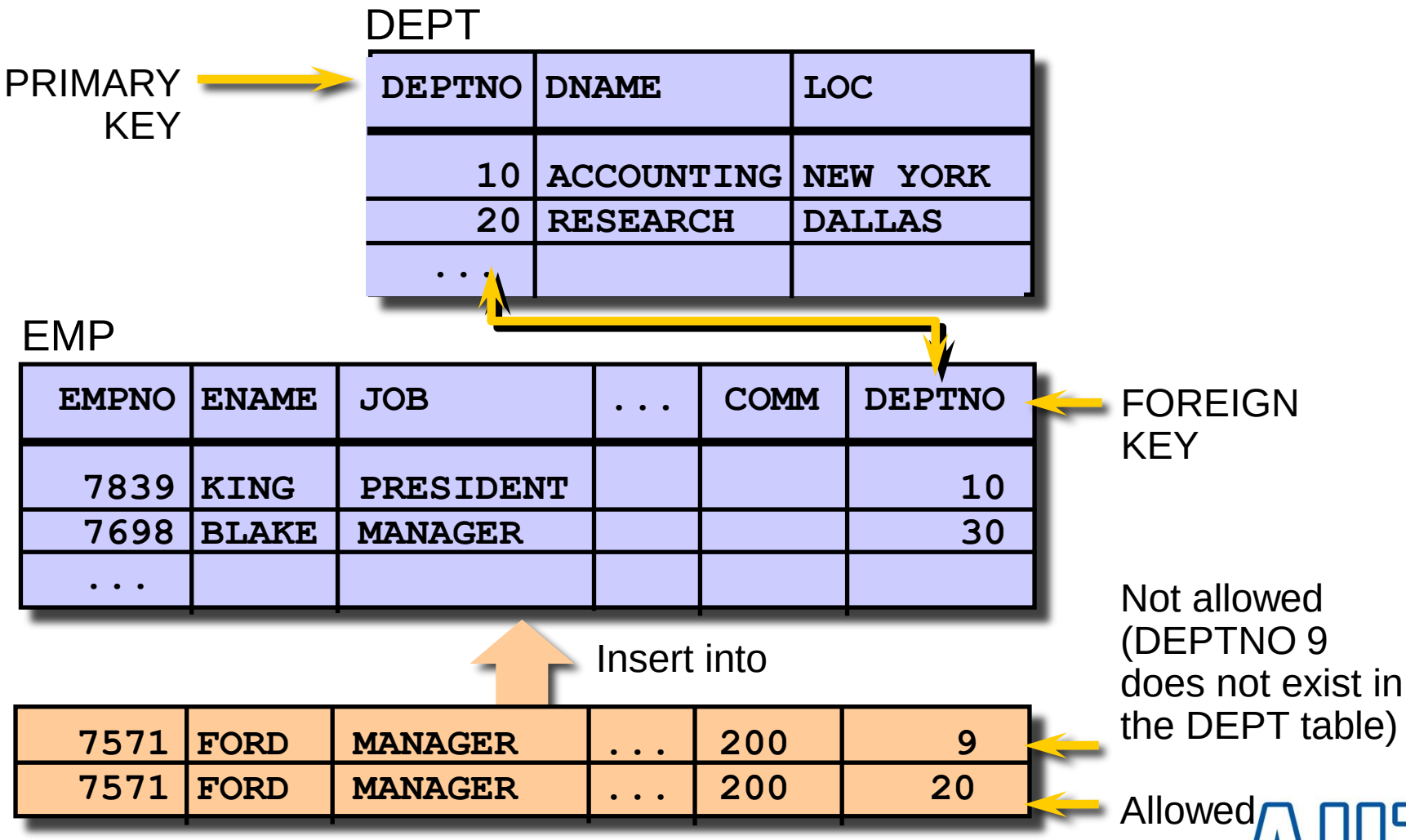


The PRIMARY KEY Constraint

- Defined at either the table level or the column level

```
SQL> CREATE TABLE   dept(
  2       deptno      NUMBER(2),
  3       dname       VARCHAR2(14),
  4       loc         VARCHAR2(13),
  5       CONSTRAINT dept_dname_uk UNIQUE (dname),
  6       CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno));
```

The FOREIGN KEY Constraint



The FOREIGN KEY Constraint

- Defined at either the table level or the column level

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

FOREIGN KEY Constraint Keywords

- FOREIGN KEY

  Defines the column in the child table at the table constraint level
- REFERENCES

  Identifies the table and column in the parent table
- ON DELETE CASCADE

  Allows deletion in the parent table and deletion of the dependent rows in the child table
- Other ON DELETE Options
  - ON DELETE {CASCADE | NO ACTION | SET DEFAULT}

The CHECK Constraint

- Defines a condition that each row must satisfy
- Expressions that are not allowed:
  - References to CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudocolumns
  - Calls to SYSDATE, UID, USER, and USERENV functions
  - Queries that refer to other values in other rows

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

Adding a Constraint

- Add or drop, but not modify, a constraint
- Enable or disable constraints
- Add a NOT NULL constraint by using the MODIFY clause

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

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
  2  ADD CONSTRAINT   emp_mgr_fk
  3            FOREIGN KEY(mgr) REFERENCES emp(empno);
Table altered.
```

Dropping a Constraint

- Remove a foreign key constraint from the EMP table.

```
SQL> ALTER TABLE       emp
  2  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
  2  DROP PRIMARY KEY CASCADE;
Table altered.
```

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

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
  2  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
  2  ENABLE CONSTRAINT        emp_empno_pk;
Table altered.
```

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

Viewing Constraints

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

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

```
CONSTRAINT_NAME           C SEARCH_CONDITION
------------------------- - --------------------------
SYS_C00674                C EMPNO IS NOT NULL
SYS_C00675                C DEPTNO IS NOT NULL
EMP_EMPNO_PK              P
...
```

Viewing the Columns Associated with Constraints

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

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

```
CONSTRAINT_NAME              COLUMN_NAME
---------------------------- --------------------
EMP_DEPTNO_FK                DEPTNO
EMP_EMPNO_PK                 EMPNO
EMP_MGR_FK                   MGR
SYS_C00674                   EMPNO
SYS_C00675                   DEPTNO
```

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Summary (Slide 1-42)

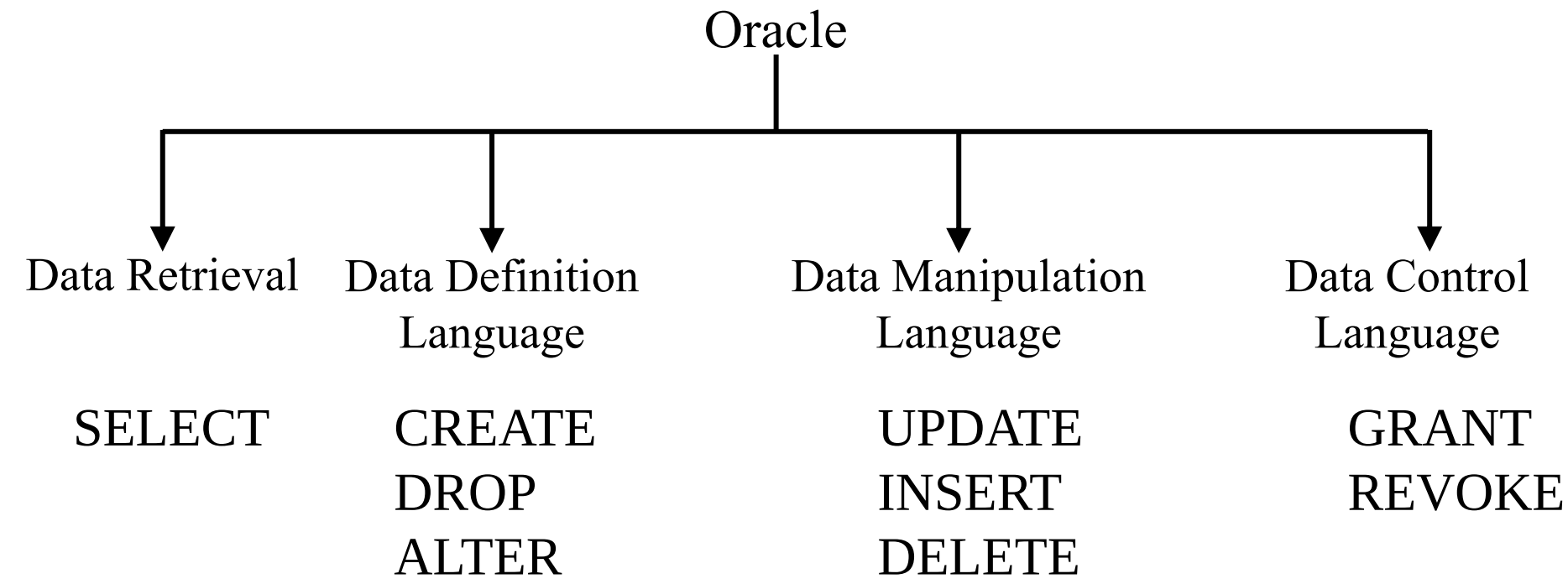
- Create table, Alter table, Drop table
- Create the following types of constraints:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK
- Query the USER_CONSTRAINTS table to view all constraint definitions and names.

Data Manipulation Language

Manipulating Data

- In this session:
  - Describe each DML statement
  - Insert rows into a table
  - Update rows in a table
  - Delete rows from a table

Data Access Language



- DML - Commands used to maintain and query data in a DB

Data Manipulation Language

- A DML statement is executed when you:
  - Add new rows to a table
  - Modify existing rows in a table
  - Remove existing rows from a table

Adding a New Row to a Table

50	DEVELOPMENT	DETROIT

New row

DEPT

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

"…insert a new row into DEPT table…"

DEPT

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

The INSERT Statement

- Add new rows to a table by using the INSERT statement.

```
INSERT INTO   table [(column [, column...])]
VALUES        (value [, value...]);
```

- Only one row is inserted at a time with this syntax.

Inserting New Rows

- Insert a new row containing values for each column.
- List values in the default order of the columns in the table.
- Optionally list the columns in the INSERT clause.

```
SQL> INSERT INTO     dept (deptno, dname, loc)
  2  VALUES           (50, 'DEVELOPMENT', 'DETROIT');
1 row created.
```

- Enclose character and date values within single quotation marks.

Inserting Rows with Null Values

- Implicit method: Omit the column from the column list.

```
SQL> INSERT INTO     dept (deptno, dname )
  2  VALUES           (60, 'MIS');
1 row created.
```

- Explicit method: Specify the NULL keyword.

```
SQL> INSERT INTO     dept
  2  VALUES           (70, 'FINANCE', NULL);
1 row created.
```

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Inserting Special Values

- The SYSDATE function records the current date and time.

```
SQL> INSERT INTO     emp (empno, ename, job,
  2                  mgr, hiredate, sal, comm,
  3                  deptno)
  4  VALUES          (7196, 'GREEN', 'SALESMAN',
  5                  7782, SYSDATE, 2000, NULL,
  6                  10);
1 row created.
```

Inserting Values by Using Substitution Variables

- Create an interactive script by using SQL*Plus substitution parameters. & is a placeholder for the variable value.

```
SQL> INSERT INTO     dept (deptno, dname, loc)
  2  VALUES          (&department_id,
  3                  '&department_name', '&location');
```

```
Enter value for department_id: 80
Enter value for department_name: EDUCATION
Enter value for location: ATLANTA

1 row created.
```

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Copying Rows from Another Table

- Write your INSERT statement with a subquery.

```
SQL> INSERT INTO managers(id, name, salary, hiredate)
  2                 SELECT empno, ename, sal, hiredate
  3                 FROM   emp
  4                 WHERE  job = 'MANAGER';
3 rows created.
```

- Do not use the VALUES clause.
- Match the number of columns in the INSERT clause to those in the subquery.

Changing Data in a Table

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

EMP

"…update a row in EMP table…"

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		20
7566	JONES	MANAGER		20
...				

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

The UPDATE Statement

- Modify existing rows with the UPDATE statement.

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE      condition];
```

- Update more than one row at a time, if required.

Updating Rows in a Table

- Specific row or rows to be modified in the WHERE clause.

```
SQL> UPDATE   emp
  2  SET      deptno = 20
  3  WHERE    empno = 7782;
1 row updated.
```

- All rows in the table are modified if you omit the WHERE clause.

```
SQL> UPDATE   employee
  2  SET      deptno = 20;
14 rows updated.
```

Updating with Multiple-Column Sub-query

- Update employee 7698's job and department to match that of employee 7499.

```
SQL> UPDATE   emp
  2  SET      (job, deptno) =
  3                (SELECT job, deptno
  4                FROM    emp
  5                WHERE   empno = 7499)
  6  WHERE    empno = 7698;
1 row updated.
```

Updating Rows Based on Another Table

- Use subqueries in UPDATE statements to update rows in a table based on values from another table.

```
SQL> UPDATE   employee
  2  SET      deptno =  (SELECT   deptno
  3                     FROM     emp
  4                     WHERE    empno = 7788)
  5  WHERE    job    =  (SELECT   job
  6                     FROM     emp
  7                     WHERE    empno = 7788);
2 rows updated.
```

Updating Rows: Integrity Constraint Error

```
SQL> UPDATE   emp
  2  SET      deptno = 55
  3  WHERE    deptno = 10;
```

```
UPDATE emp
       *
ERROR at line 1:
ORA-02291: integrity constraint (USR.EMP_DEPTNO_FK)
violated - parent key not found
```

- Department number 55 does not exist

Removing a Row from a Table

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	
...		

"…delete a row from DEPT table…"

DEPT

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

The DELETE Statement

- You can remove existing rows from a table by using the DELETE statement.

```
DELETE [FROM]   table
[WHERE          condition];
```

Deleting Rows from a Table

- Specific rows are deleted when you specify the WHERE clause.

```
SQL> DELETE FROM     department
  2  WHERE           dname = 'DEVELOPMENT';
1 row deleted.
```

- All rows in the table are deleted if you omit the WHERE clause.

```
SQL> DELETE FROM     department;
4 rows deleted.
```

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Deleting Rows Based on Another Table

- Use subqueries in DELETE statements to remove rows from a table based on values from another table.

```
SQL> DELETE FROM     employee
  2  WHERE           deptno =
  3                         (SELECT    deptno
  4                          FROM      dept
  5                          WHERE     dname ='SALES');
6 rows deleted.
```

Deleting Rows: Integrity Constraint Error

```
SQL> DELETE FROM     dept
  2  WHERE           deptno = 10;
```

```
DELETE FROM dept
            *
ERROR at line 1:
ORA-02292: integrity constraint (USER.EMP_DEPTNO_FK)
violated - child record found
```

- You cannot delete a row
- that contains a primary key
- that is used as a foreign key
- in another table

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

COMMIT and ROLLBACK

- Use COMMIT to permanently save the changes made to the database
- Use ROLLBACK to undo the previous operation.
- Note:
  - If you close SQL*Plus using the x in top right hand corner-your work will be lost unless you have specifically committed your DML
  - If you type ‘Exit’ to close SQL*Plus, your work will be saved

Lab Activities

- Complete exercise handed out in lab