ORACLE SQL

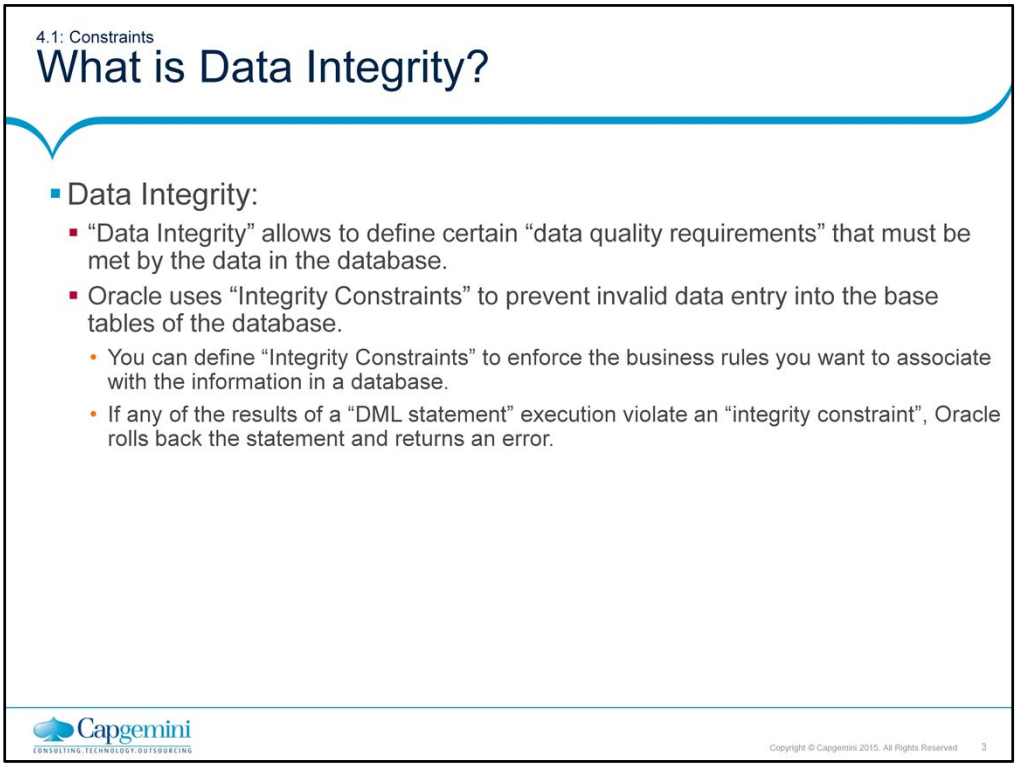
Lesson 04: Constraints, Adv. Group by, Adv. Subqueries, Managing other db objects..

Lesson Objectives

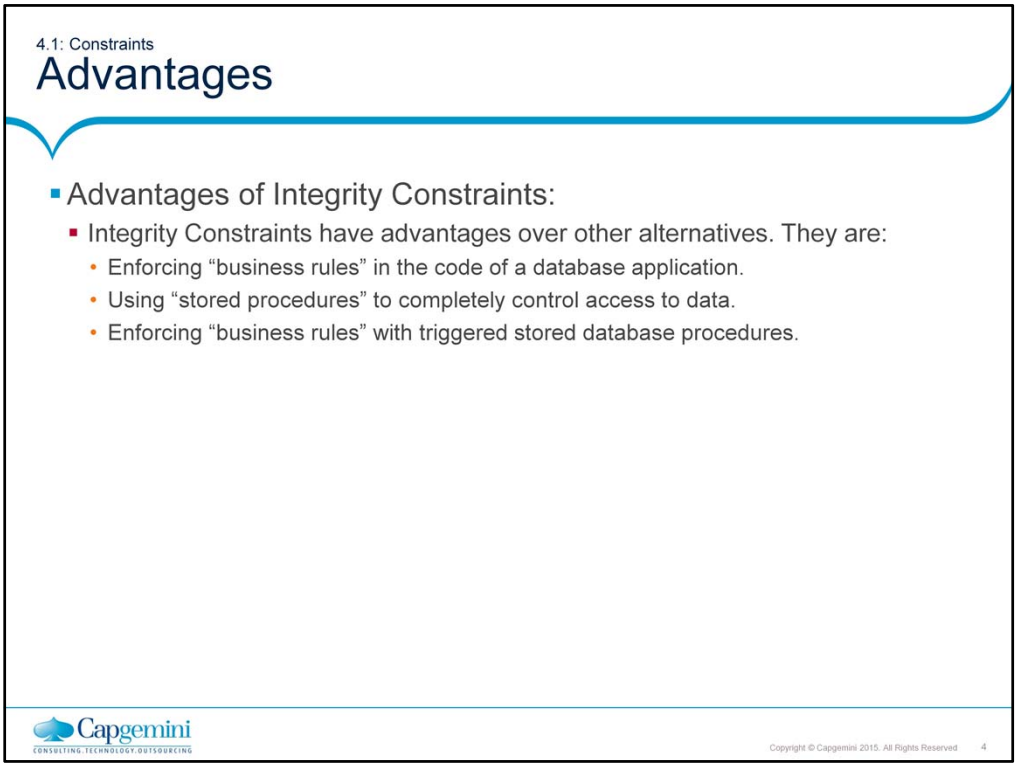
- To understand the following topics:
  - Constraints
  - Adv Group by
  - Adv subqueries
  - Other DB objects

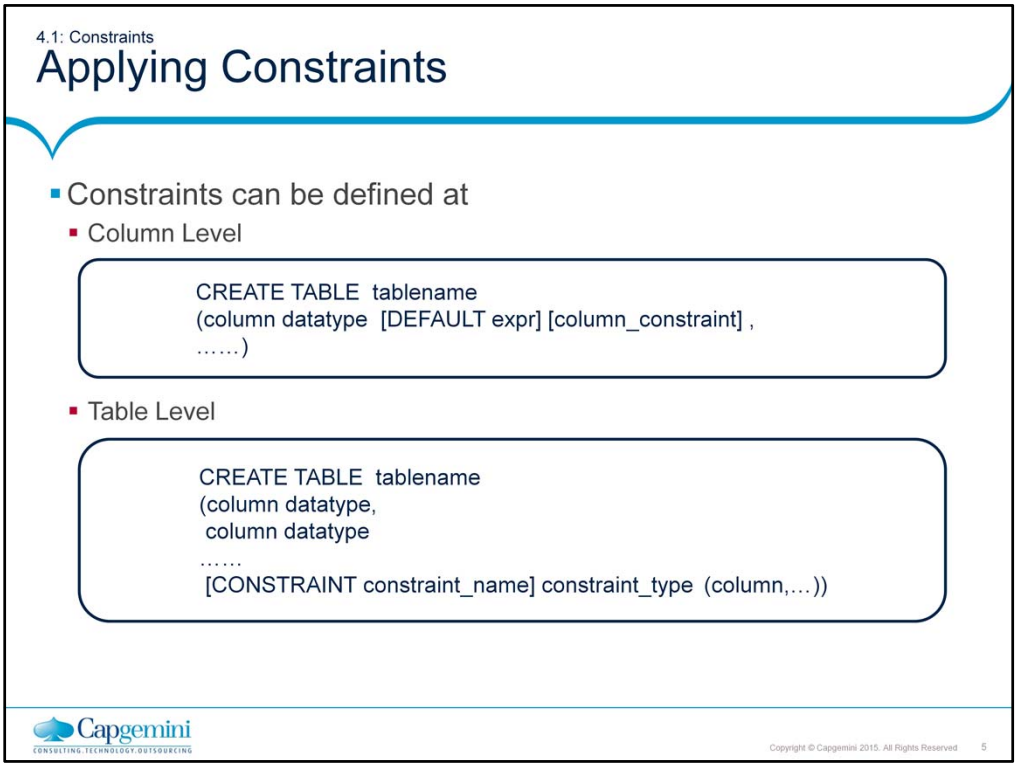
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

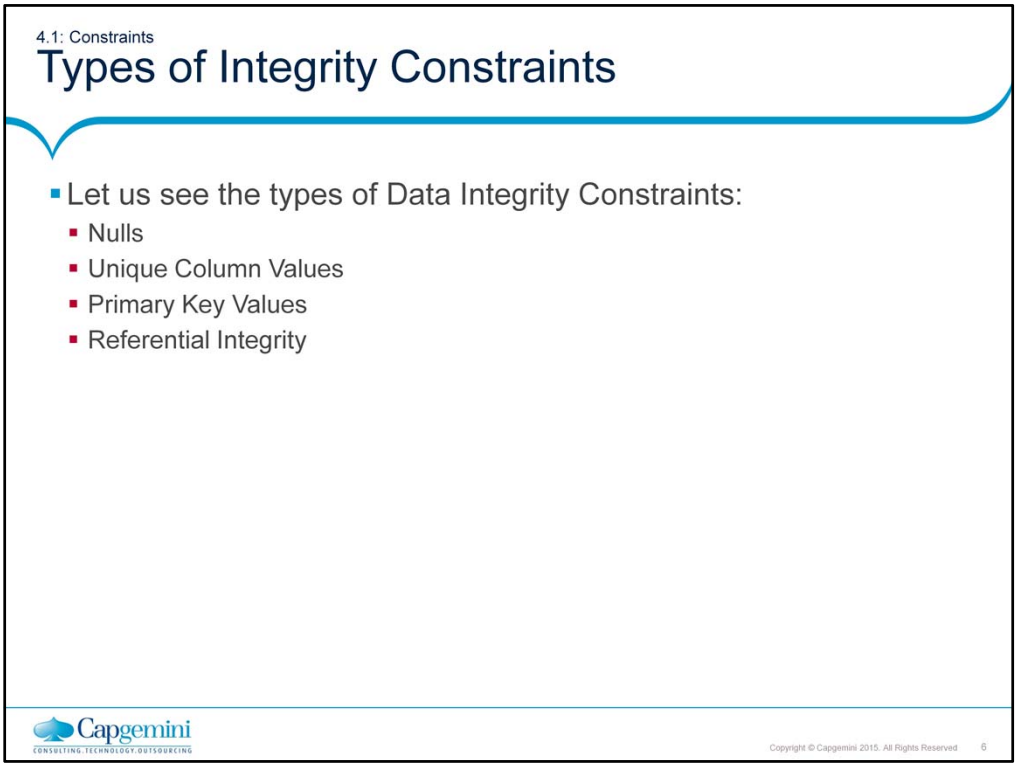


Data Integrity: Integrity Constraint
An Integrity Constraint is a declarative method of defining a rule for a column of a table.
Example of Data Integrity:
Assume that you define an "Integrity Constraint" for the Staff_Sal column of the Staff_Master table.
This Integrity Constraint enforces the rule that "no row in this table can contain a numeric value greater than 10,000 in this column".
If an INSERT or UPDATE statement attempts to violate this "Integrity Constraint", Oracle rolls back the statement and returns an "information error" message.


4.1: Constraints
Advantages
Advantages of Integrity Constraints:
Integrity Constraints have advantages over other alternatives. They are:
Enforcing “business rules” in the code of a database application.
Using “stored procedures” to completely control access to data.
Enforcing “business rules” with triggered stored database procedures.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
4



Applying Constraints:
In Oracle you can apply constraints
Column Level - References a single column and is defined within a specification for the owning column; can be any type of integrity constraint
Table Level - References one or more columns and is defined separately from the definitions of the columns in the table; can define any constraints except NOT NULL



Types of Data Integrity:
Oracle supports the following Integrity Constraints:

NOT NULL constraints for the rules associated with nulls in a column. "Null" is a rule defined on a single column that allows or disallows, inserts or updates on rows containing a "null" value (the absence of a value) in that column.

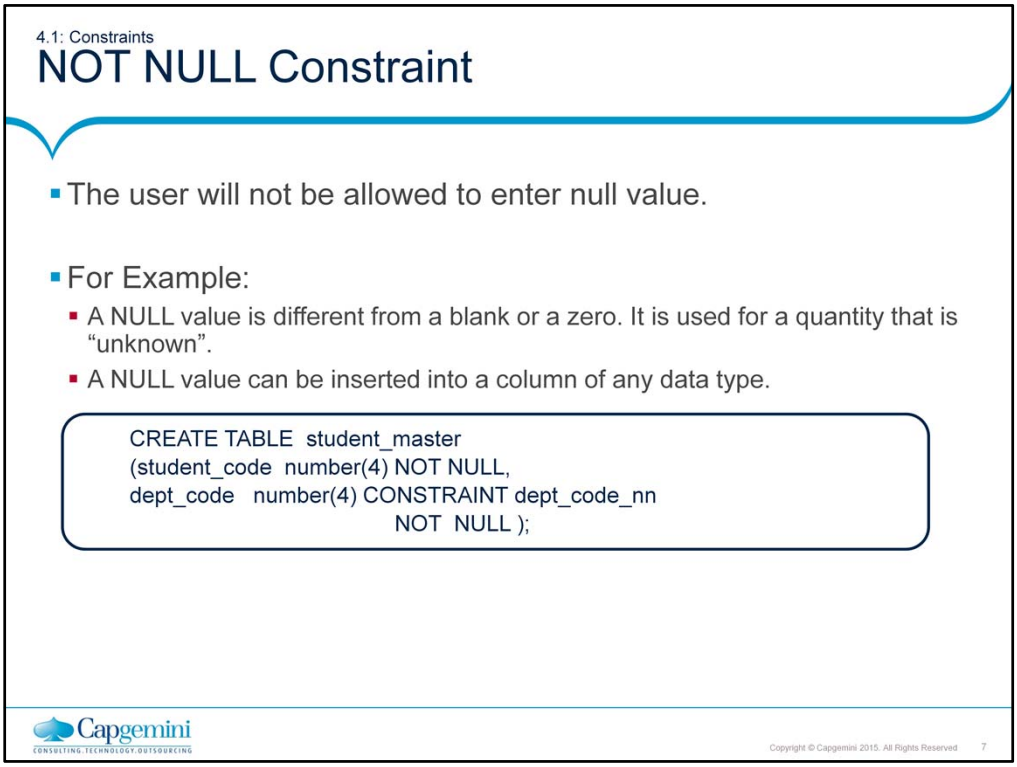
UNIQUE key constraints for the rule associated with unique column values. A "unique value" rule defined on a column (or set of columns) allows inserting or updating a row only if it contains a "unique value" in that column (or set of columns).

PRIMARY KEY constraints for the rule associated with primary identification values. A "primary key" value rule defined on a key (a column or set of columns) specifies that "each row in the table can be uniquely identified by the values in the key".

FOREIGN KEY constraints for the rules associated with referential integrity. A "Referential Integrity" rule defined on a key (a column or set of columns) in one table guarantees that "the values in that key, match the values in a key in a related table (the referenced value)". Oracle currently supports the use of FOREIGN KEY integrity constraints to define the referential integrity actions, including:

- update and delete No Action
- delete CASCADE
- delete SET NULL

CHECK constraints for complex integrity rules



NOT NULL constraint:
Often there may be records in a table that do not have values for every field.
This could be because the information is not available at the time of the data entry or because the field is not applicable in every case.
If the column is created as NULLABLE, in the absence of a user-defined value the DBMS will place a NULL value in the column.
A NULL value is different from a blank or a zero. It is used for a quantity that is "unknown".
"Null" is a rule defined on a single column that allows or disallows, inserts or updates on rows containing a "null" value (the absence of a value) in that column.
A NULL value can be inserted into a column of any data type.
Principles of NULL values:
Setting a NULL value is appropriate when the "actual value" is unknown, or when a value is not meaningful.
A NULL value is not equivalent to the value of "zero" if the data type is number, and it is not equivalent to "spaces" if the data type is character.
A NULL value will evaluate to NULL in any expression
For example: NULL multiplied by 10 is NULL.
NULL value can be inserted into columns of any data type.
If the column has a NULL value, Oracle ignores any UNIQUE, FOREIGN KEY, and CHECK constraints that may be attached to the column.

4.1: Constraints

DEFAULT clause

- If no value is given, then instead of using a “Not Null” constraint, it is sometimes useful to specify a default value for an attribute.

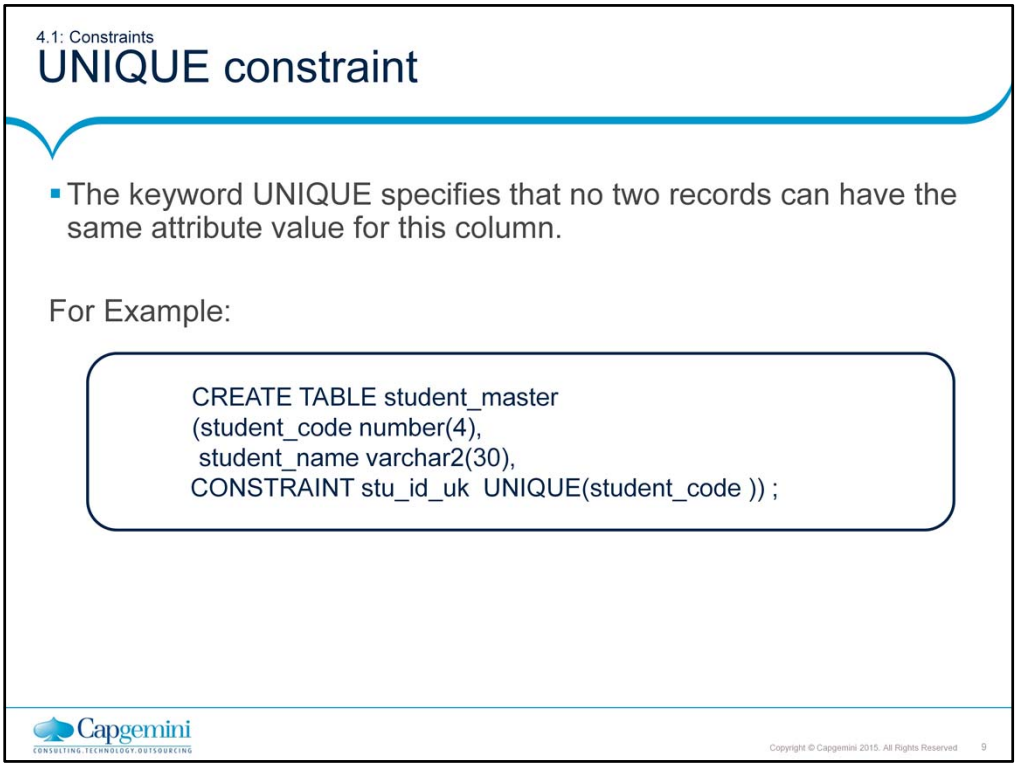
For Example:

- When a record is inserted the default value can be considered.

```
CREATE TABLE  staff_master(
Staff_Code number(8) PRIMARY KEY,
Staff_Name varchar2(50) NOT NULL,
Staff_dob date,
Hiredate date DEFAULT sysdate,
.....)
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

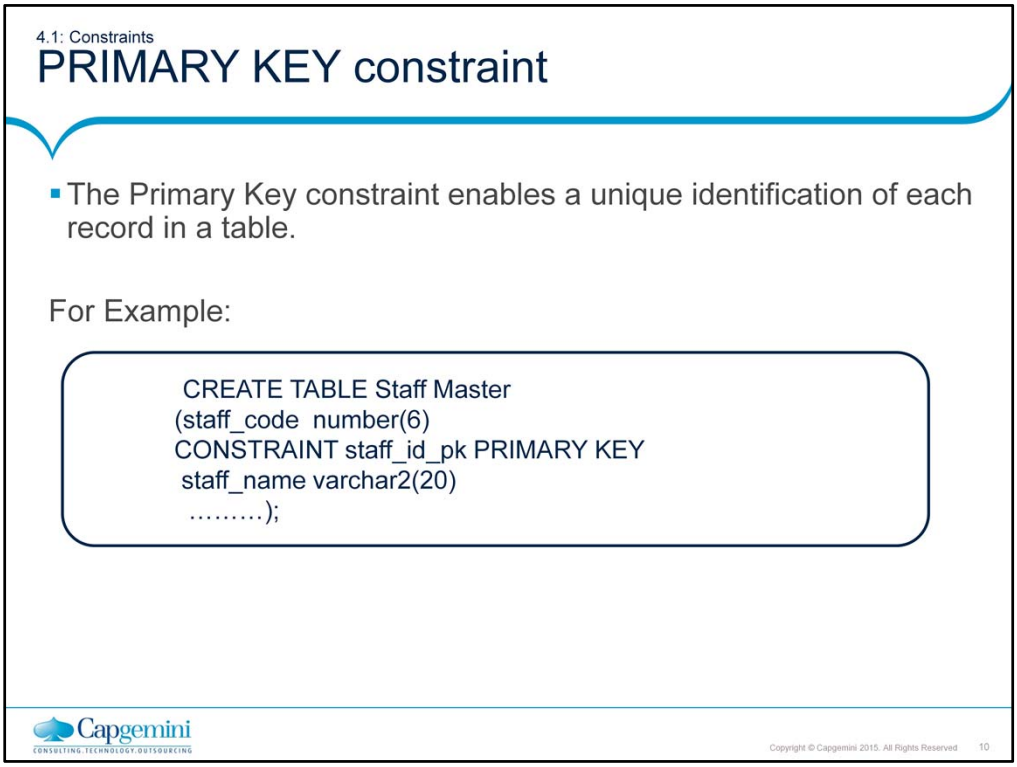


UNIQUE constraint:
The UNIQUE constraint does not allow duplicate values in a column.

If the UNIQUE constraint encompasses two or more columns, then two equal combinations are not allowed.

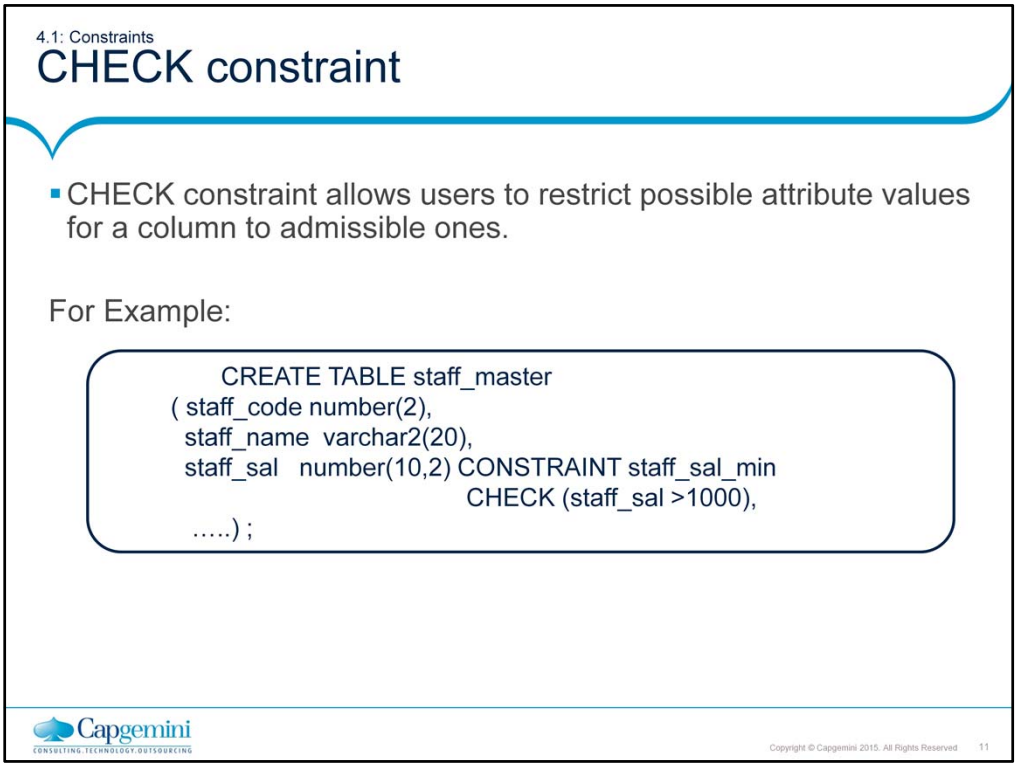
However, if a column is not explicitly defined as NOT NULL, then NULLS can be inserted multiple number of times.

A UNIQUE constraint can be extended over multiple columns.
A “unique value” rule defined on a column (or set of columns) allows inserting or updating a row only if it contains a “unique value” in that column (or set of columns).

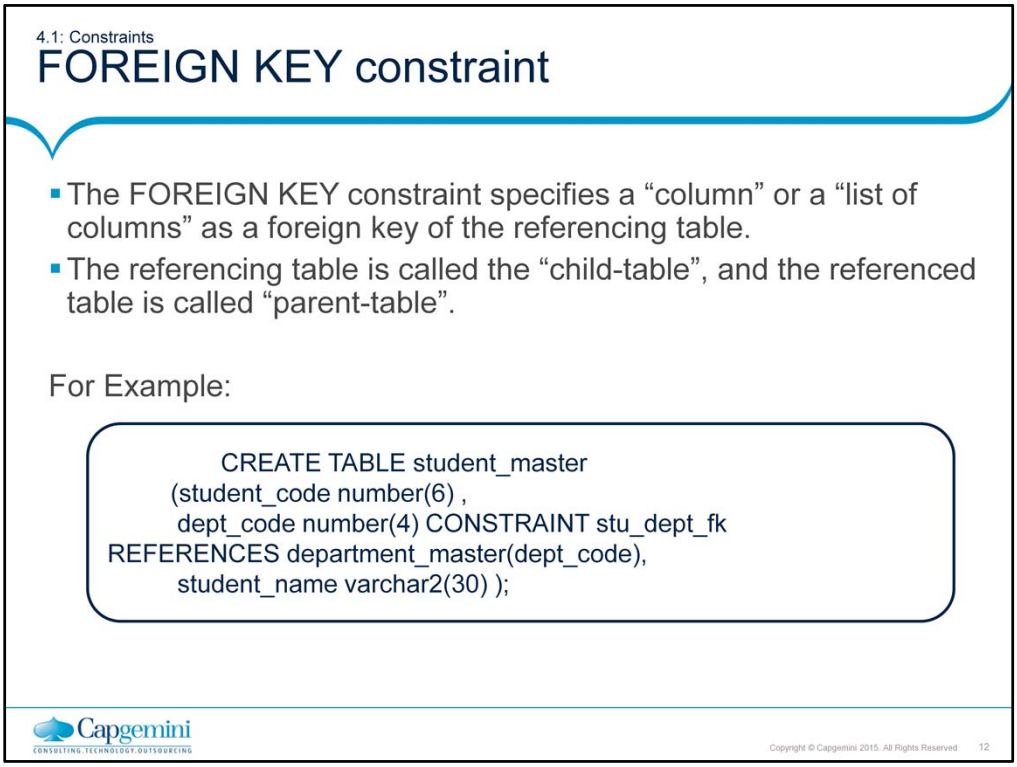


PRIMARY KEY constraint:
On a technical level, a PRIMARY KEY combines a UNIQUE constraint and a NOT NULL constraint.
Additionally, a table can have at the most one PRIMARY KEY.
After creating a PRIMARY KEY, it can be referenced by a FOREIGN KEY.
A “primary key” value rule defined on a key (a column or set of columns) specifies that “each row in the table can be uniquely identified by the values in the key”.
The example on the slide defines the primary key constraint at the column level.
The same example is seen below with the constraint defined at table level

```
CREATE TABLE Staff Master
(staff_code number(6)
 staff_name varchar2(20)
 ……..)
CONSTRAINT staff_id_pk PRIMARY KEY (staff_code)
;
```



CHECK constraint:
A CHECK constraint allows to state a minimum requirement for the value in a column.
If more complicated requirements are desired, an INSERT trigger must be used.

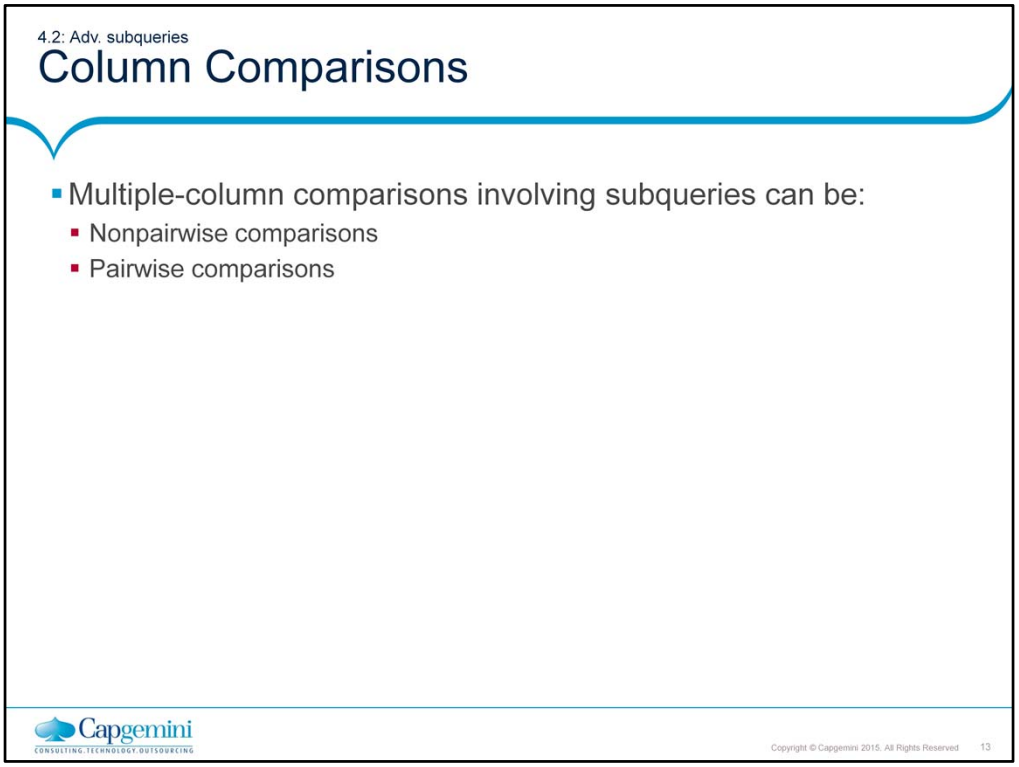


FOREIGN KEY Constraint Keywords:
Given below are a few Foreign Key constraint keywords:

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: Deletes the dependent rows in the child table when a row in the parent table is deleted.
ON DELETE SET NULL: Converts dependent FOREIGN KEY values to NULL.

You can query the USER_CONSTRAINTS table to view all constraint definitions and names.
You can view the columns associated with the constraint names in the USER_CONS_COLUMNS view.

In EMP table, for deptno column, if we want to allow only those values that already exist in deptno column of the DEPT table, we must enforce what is known as REFERENTIAL INTEGRITY. To enforce REFERENTIAL INTEGRITY, declare deptno field of DEPT table as PRIMARY KEY, and deptno field of EMP table as FOREIGN KEY as follows

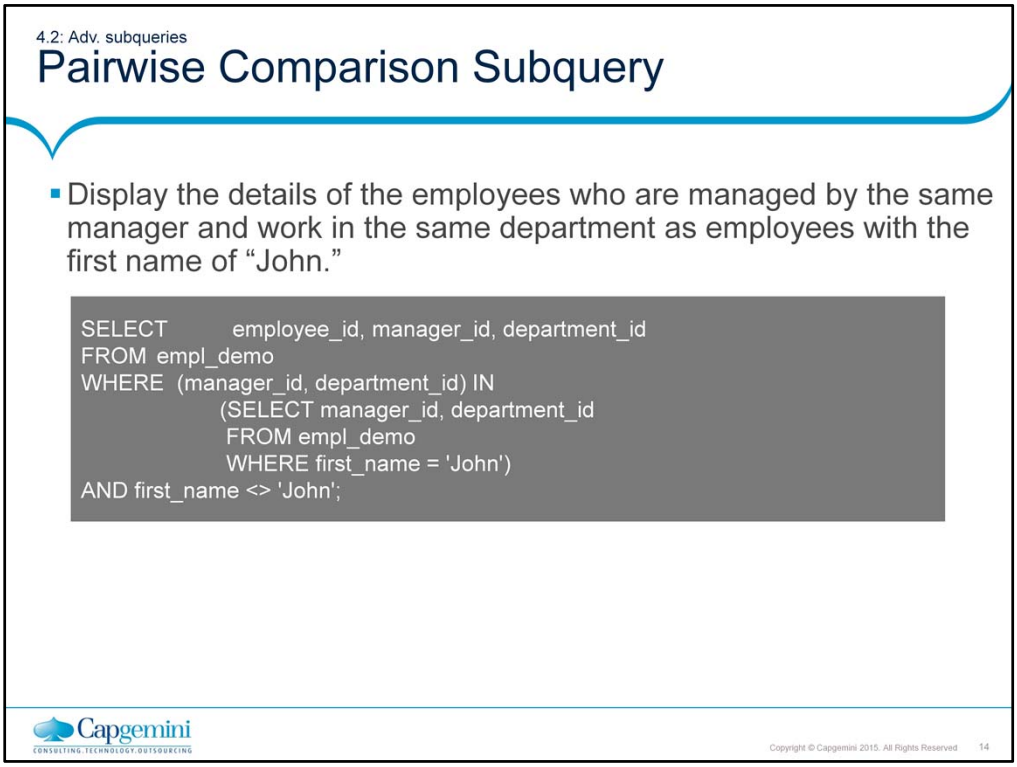


Pairwise Versus Nonpairwise Comparisons

Multiple-column comparisons involving subqueries can be nonpairwise comparisons or pairwise comparisons. If you consider the example "Display the details of the employees who work in the same department, and have the same manager, as 'Daniel'? ," you get the correct result with the following statement:

```
SELECT first_name, last_name, manager_id,
department_id
FROM empl_demo
WHERE manager_id IN (SELECT manager_id
                     FROM empl_demo
                     WHERE first_name = 'Daniel')
AND department_id IN (SELECT department_id
                      FROM empl_demo
                      WHERE first_name = 'Daniel');
```

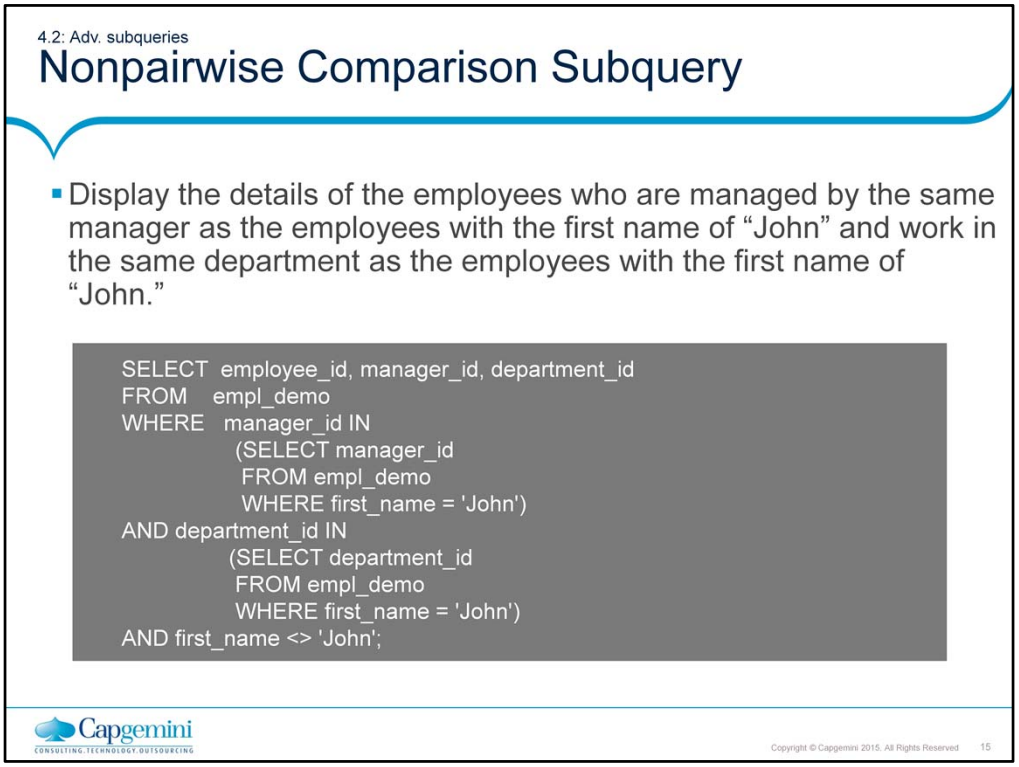
There is only one "Daniel" in the `EMPL_DEMO` table (Daniel Faviet, who is managed by employee 108 and works in department 100). However, if the subqueries return more than one row, the result might not be correct. For example, if you run the same query but substitute "John" for "Daniel," you get an incorrect result. This is because the combination of `department_id` and `manager_id` is important. To get the correct result for this query, you need a pairwise comparison.



Pairwise Comparison Subquery

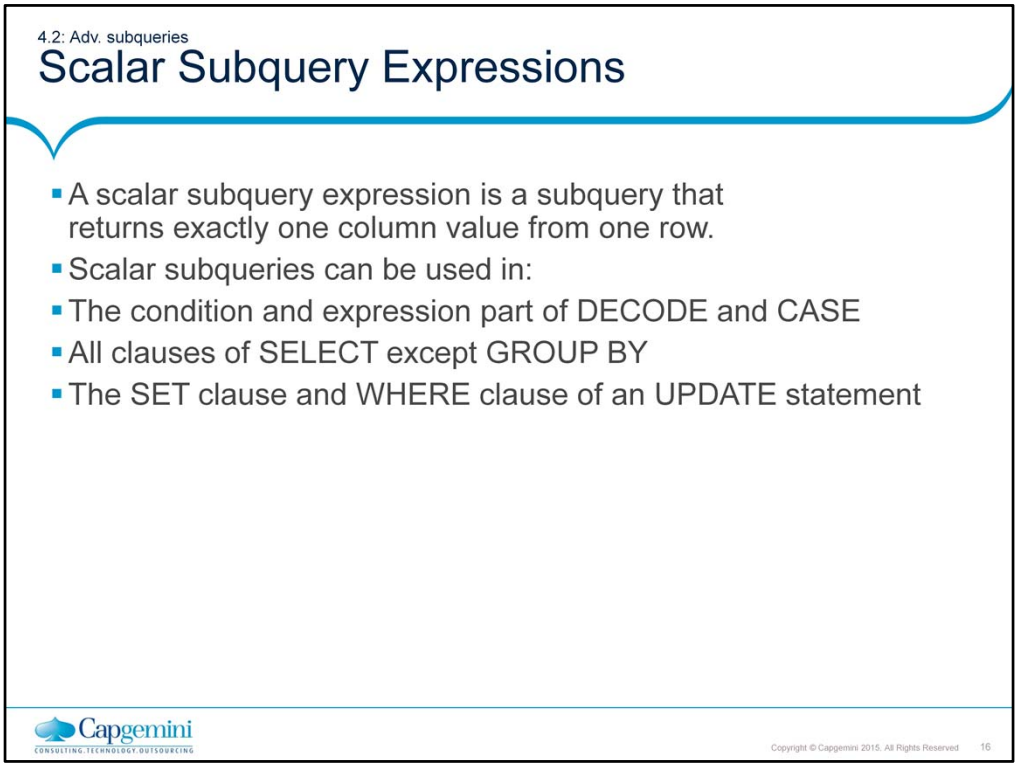
The example in the slide compares the combination of values in the MANAGER_ID column and the DEPARTMENT_ID column of each row in the EMPL_DEMO table with the values in the MANAGER_ID column and the DEPARTMENT_ID column for the employees with the FIRST_NAME of “John.” First, the subquery to retrieve the MANAGER_ID and DEPARTMENT_ID values for the employees with the FIRST_NAME of “John” is executed. This subquery returns the following:

	MANAGER_ID	DEPARTMENT_ID
1	108	100
2	123	50
3	100	80



Nonpairwise Comparison Subquery

The example shows a nonpairwise comparison of the columns. First, the subquery to retrieve the `MANAGER_ID` values for the employees with the `FIRST_NAME` of “John” is executed. Similarly, the second subquery to retrieve the `DEPARTMENT_ID` values for the employees with the `FIRST_NAME` of “John” is executed. The retrieved values of the `MANAGER_ID` and `DEPARTMENT_ID` columns are compared with the `MANAGER_ID` and `DEPARTMENT_ID` columns for each row in the `EMPL_DEMO` table. If the `MANAGER_ID` column of the row in the `EMPL_DEMO` table matches with any of the values of `MANAGER_ID` retrieved by the inner subquery and if the `DEPARTMENT_ID` column of the row in the `EMPL_DEMO` table matches with any of the values of `DEPARTMENT_ID` retrieved by the second subquery, the record is displayed.



Scalar Subqueries in SQL

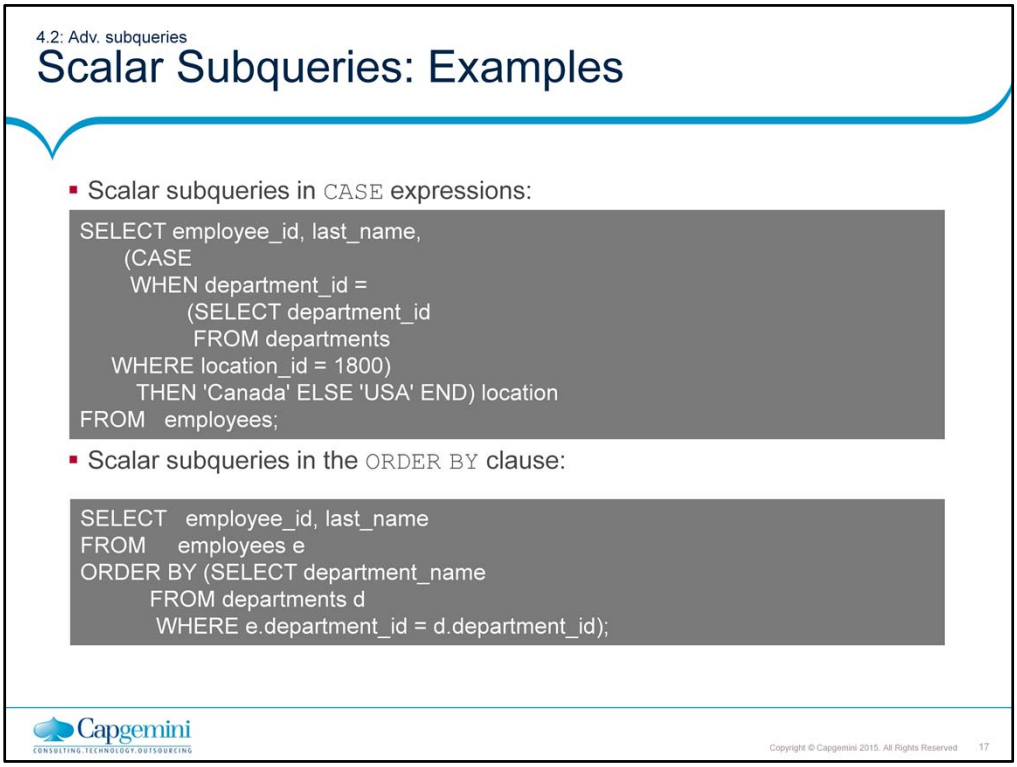
A subquery that returns exactly one column value from one row is also referred to as a scalar subquery. Multiple-column subqueries that are written to compare two or more columns, using a compound `WHERE` clause and logical operators, do not qualify as scalar subqueries.

The value of the scalar subquery expression is the value of the select list item of the subquery. If the subquery returns 0 rows, the value of the scalar subquery expression is `NULL`. If the subquery returns more than one row, the Oracle server returns an error. The Oracle server has always supported the usage of a scalar subquery in a `SELECT` statement. You can use scalar subqueries in:

- The condition and expression part of `DECODE` and `CASE`
- All clauses of `SELECT` except `GROUP BY`
- The `SET` clause and `WHERE` clause of an `UPDATE` statement

However, scalar subqueries are not valid expressions in the following places:

- As default values for columns and hash expressions for clusters
- In the `RETURNING` clause of data manipulation language (DML) statements
- As the basis of a function-based index
- In `GROUP BY` clauses, `CHECK` constraints, and `WHEN` conditions
- In `CONNECT BY` clauses
- In statements that are unrelated to queries, such as `CREATE PROFILE`



Scalar Subqueries: Examples

The first example in the slide demonstrates that scalar subqueries can be used in CASE expressions. The inner query returns the value 20, which is the department ID of the department whose location ID is 1800. The CASE expression in the outer query uses the result of the inner query to display the employee ID, last names, and a value of Canada or USA, depending on whether the department ID of the record retrieved by the outer query is 20 or not.

The following is the result of the first example in the slide:

	EMPLOYEE_ID	LAST_NAME	LOCATION
1	198	OConnell	USA
2	199	Grant	USA
3	200	Whalen	USA
4	201	Hartstein	Canada
5	202	Fay	Canada
6	203	Mavris	USA

4.2: Adv. subqueries

Correlated Query

- When a sub query references a column from the table referred to in the parent query it is known as correlated query
- A correlated subquery answers a multiple-part question whose answer depends on the value in each row processed by the parent statement

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

4.2: Adv. subqueries

Correlated Query Example

- List the employees earning more than average salaries in their own department
- select ename,sal,deptno
- from emp a
- where sal > [A query which returns the avg salary of the department in which the employee of the outer query is working]

- select ename,sal,deptno
- from emp a
- where sal > (select avg(sal) from emp b
- where b.deptno = a.deptno)
- order by deptno

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

4.2: Adv. subqueries

Using Exists and Not Exists

Exists	Returns TRUE if the subquery returns a single row satisfying the condition
Not Exists	Returns TRUE if the subquery does not return any row

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

4.2: Adv. subqueries

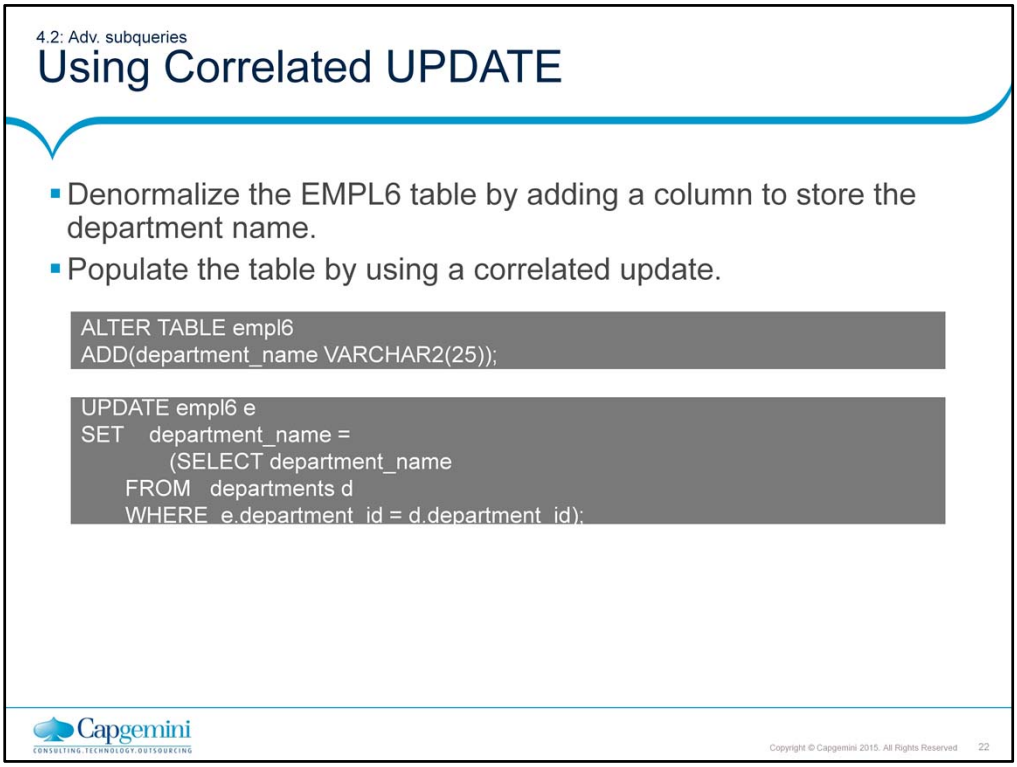
Example

- List the departments without employees
- select * from dept d
- where not exists (select deptno from emp e where e.deptno=d.deptno)

- List the departments with employees
- select * from dept d
- where exists (select deptno from emp e where e.deptno=d.deptno)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21



Correlated UPDATE (continued)

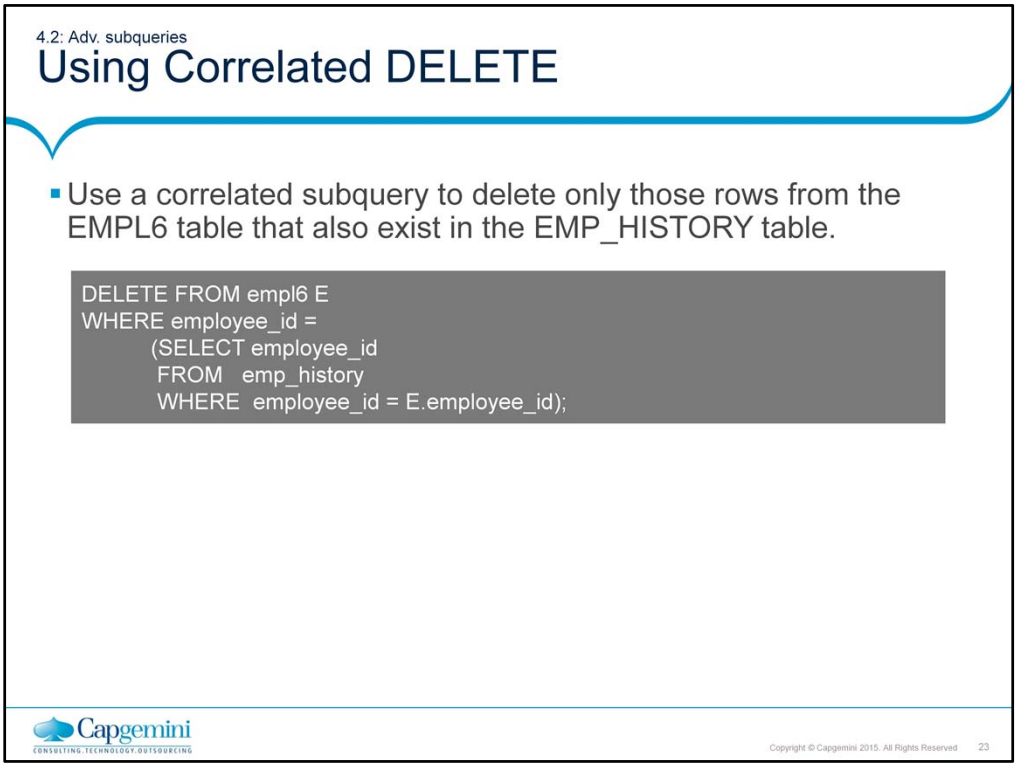
The example in the slide denormalizes the EMPL6 table by adding a column to store the department name and then populates the table by using a correlated update.

Following is another example for a correlated update.

Problem Statement

The REWARDS table has a list of employees who have exceeded expectations in their performance. Use a correlated subquery to update rows in the EMPL6 table based on rows from the REWARDS table:

```
UPDATE empl6
SET    salary = (SELECT empl6.salary +
rewards.pay_raise
                 FROM   rewards
                 WHERE  employee_id  =
                      empl6.employee_id
                 AND  payraise_date =
                    (SELECT MAX(payraise_date)
                    FROM   rewards
                    WHERE  employee_id  =
empl6.employee_id))
WHERE  empl6.employee_id
IN     (SELECT employee_id FROM rewards);
```



Correlated DELETE (continued)

Example

Two tables are used in this example. They are:

- The EMPL6 table, which provides details of all the current employees
- The EMP_HISTORY table, which provides details of previous employees

EMP_HISTORY contains data regarding previous employees, so it would be erroneous if the same employee's record existed in both the EMPL6 and EMP_HISTORY tables. You can delete such erroneous records by using the correlated subquery shown in the slide.

4.2: Adv. subqueries

Inline view

- A subquery in the FROM or column list clause of the Select statement is known as inline view
- Columns of inline view can be used in the outer query

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

4.2: Adv. subqueries

Examples of inline view

- List the employees earning more than the average salary. Also display the average salary

- select ename, sal, average_salary
- from emp , (select avg(sal) average_salary from emp)
- where sal > average_salary

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25

4.2: Adv. subqueries

Views

- A view is a stored query
- A view takes the output of the query and treats it as a table.
- Used for storing complex queries for easy representation
- Oracle stores the definition of view
- It does not contain data
- It is known as virtual table
- The definition is expanded at runtime when it is used

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26

4.2: Adv. subqueries

Views

- Create or Replace view <viewname) (column_list)
- as <query>
- with check option constraint
- with READ ONLY

- Create view emp_view as
- select empno,ename,deptno,sal from emp;

- Create view emp_dept_view as
- select empno,ename,job,dname
- from emp e, dept d
- where e.deptno = d.deptno

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 27

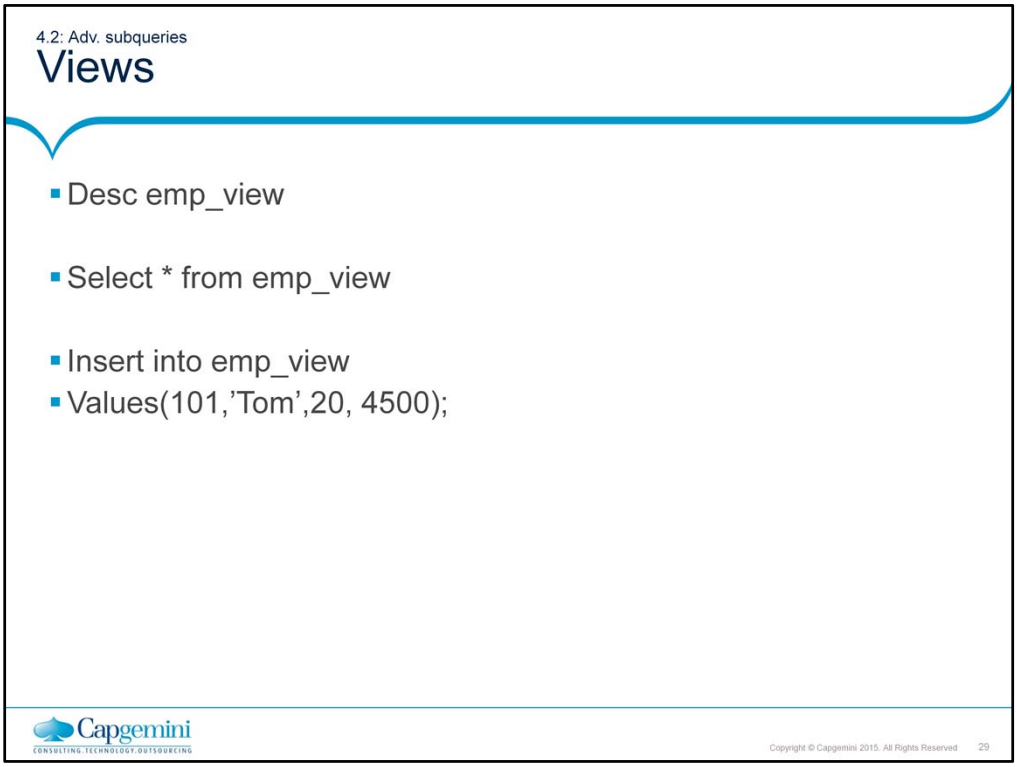
4.2: Adv. subqueries

Views

- Views are derived from base tables and hence have many similarities.
- They can be described and queried
- With some restrictions we can insert into, update or delete data from views
- All the operations are performed on the base tables of the view and they affect the actual data of the base table subject to integrity constraint and triggers

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 28


4.2: Adv. subqueries
Views
Desc emp_view
Select * from emp_view
Insert into emp_view
Values(101,'Tom',20, 4500);
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
29

4.2: Adv. subqueries

View Examples

- Create view dept_summary(dept_name,emp_count,total_salary, maximum_sal, minimum_sal) as
- select dname,count(*),sum(sal),max(sal),min(sal)
- from emp e, dept d
- where d.deptno = e.deptno
- **group by dname;**

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 30

4.2: Adv. subqueries

How Views are used

- To provide table level security by restricting data to predetermined rows or columns
- Hides complexity
- Simplifies statements for users
- Save complex queries

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 31

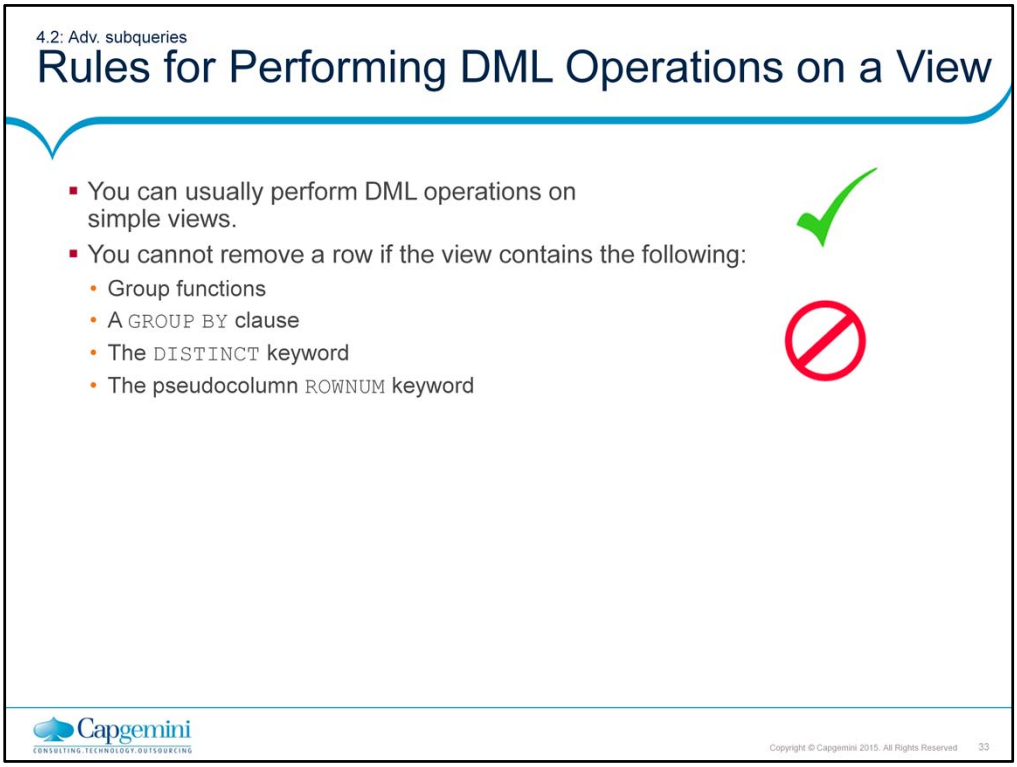
4.2: Adv. subqueries

Views – Check option and Read Only

- Create view emp_dept_10 as
- select * from emp
- where deptno = 10
- with check option
- DML must confirm to condition specified in where clause
- Create OR REPLACE view emp_dept_10 as
- select * from emp
- where deptno = 10
- with READ ONLY;
- DML cannot be performed

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 32

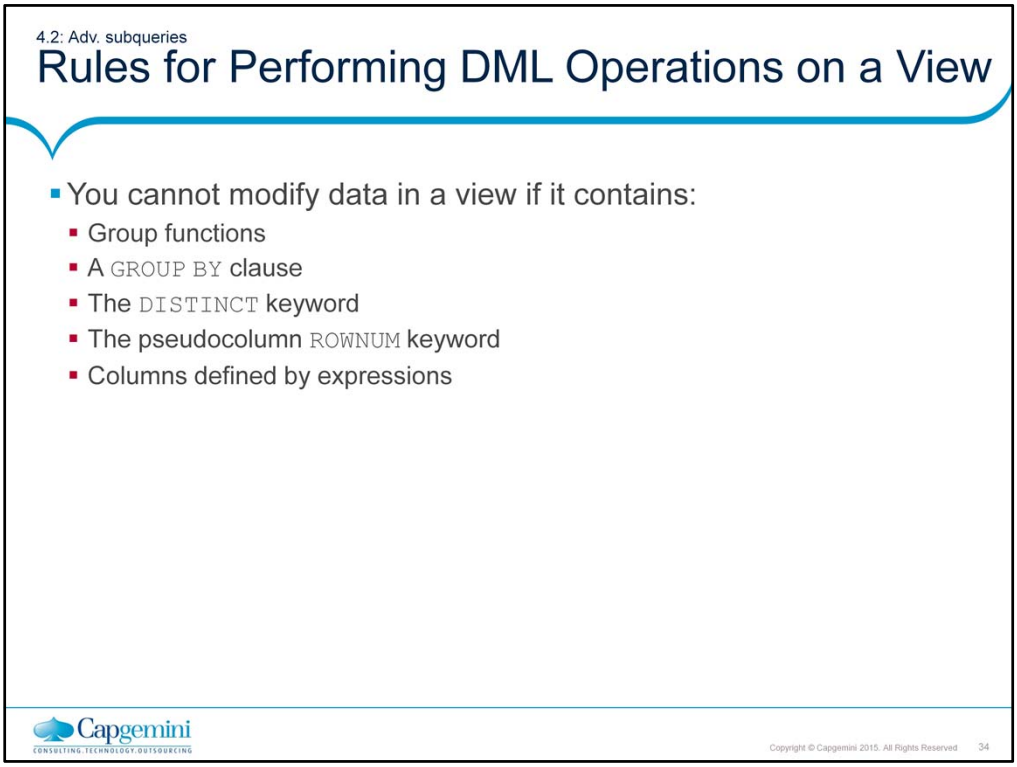


Rules for Performing DML Operations on a View

You can perform DML operations on data through a view if those operations follow certain rules.

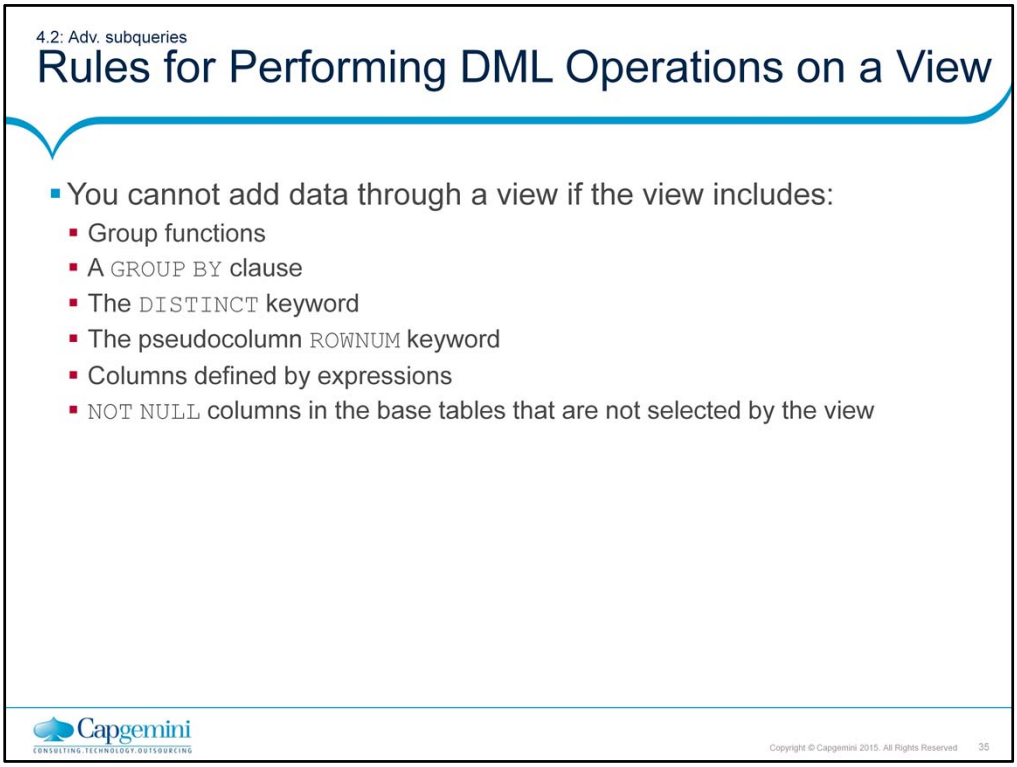
You can remove a row from a view unless it contains any of the following:

- Group functions
- A GROUP BY clause
- The DISTINCT keyword
- The pseudocolumn ROWNUM keyword



Rules for Performing DML Operations on a View (continued)

You can modify data through a view unless it contains any of the conditions mentioned in the previous slide or columns defined by expressions (for example, SALARY * 12).



Rules for Performing DML Operations on a View (continued)

You can add data through a view unless it contains any of the items listed in the slide. You cannot add data to a view if the view contains NOT NULL columns without default values in the base table. All the required values must be present in the view. Remember that you are adding values directly to the underlying table through the view.

For more information, see the section on “CREATE VIEW” in Oracle Database SQL Language Reference 11g, Release 1 (11.1).

4.2: Adv. subqueries

Removing a View

- Drop view <view_name>
- Drop view dept_summary;

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 36

4.2: Adv. subqueries

Updateable View Restrictions

- A view can be updateable if it does not contain :
  - Set operator
  - Distinct clause
  - Aggregate or Analytical functions
  - Group by clause
  - Subquery in select list
  - Joins (with some exceptions)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 37

4.2: Adv. subqueries

Rules for updateable join view

- The DML statement must affect only one table underlying in the join (known as key-preserved table)
- For an UPDATE statement, all columns updated must be extracted from a key-preserved table.
- For a DELETE statement, the join can have one and only one key-preserved table
- For an INSERT statement, all columns into which values are inserted must come from a key-preserved table

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 38

4.2: Adv. subqueries

Updatable view example

- create or replace view emp_dept_upd
- as select EMPNO, ENAME,JOB,MGR,HIREDATE, SAL,
- COMM, e.DEPTNO, dname
- from emp e, dept d
- where e.deptno = d.deptno;
- SELECT column_name, updatable
- FROM user_updatable_columns
- WHERE table_name = 'EMP_DEPT_UPD‘;
- insert into emp_DEPT_UPD (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL,COMM, DEPTNO)
- values(1,'HAPPY','CLERK',7782,'01-JAN-02',1500,NULL,10);

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 39

4.3: DB Objects

Index

- Index helps to locate information faster
- Indexes can be created on column(s) of a table to speed up execution of SQL statements on that table
- Oracle index provides a faster access path to table data
- Indexes are the primary means of reducing disk I/O when properly used.
- A useful tool for application tuning used by developers and DBA's

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 40

4.3: DB Objects

Indexes…

- Oracle provides several indexing schemes , the most common used is B-tree structure
- Indexes are automatically created with the same name by Oracle when Primary and Unique constraints are created.
- Oracle maintains and uses indexes on its own
- Columns containing NULL values are not indexed

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 41

4.3: DB Objects

Index examples

- Create unique index <index_name> on table(column_list) asc/desc
- Create index empidx on emp1(empno);
- Create index emp_dept on emp1(deptno,ename);
- create index emp_job on emp(job)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 42

4.3: DB Objects

When to Create an Index

- A column contains a wide range of values
- A column contains a large number of null values
- One or more columns are frequently used together in a Where clause or a join condition
- The table is large and most queries are expected to retrieve less than 2 to 4 percent of the rows

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 43

4.3: DB Objects

When not to create an index

- The table is small
- The columns are not often used as a condition in the query
- Most queries are expected to retrieve more than 2 to 4 percent of the rows in the table
- The table is updated frequently
- The indexed columns are referenced as part of an expression

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 44

4.3: DB Objects

Removing an Index

- Drop index <index_name>
- Drop index empidx;

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 45

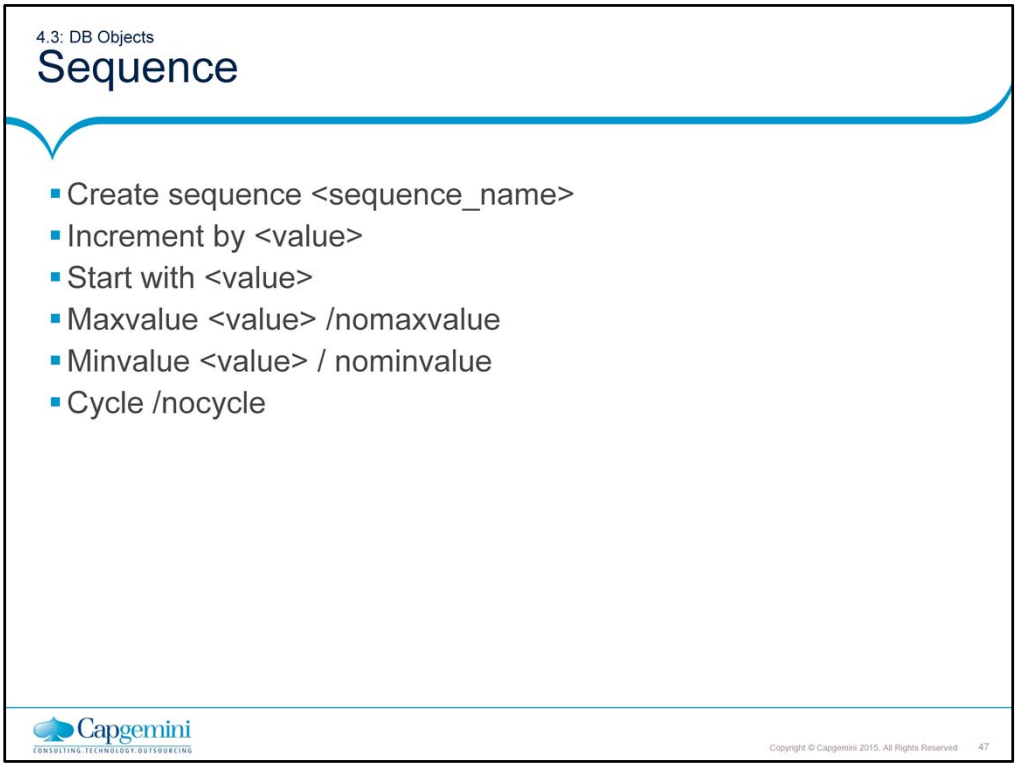
4.3: DB Objects

Sequences

- Sequences are used for generating unique sequential series of numbers
- Useful in multi-user environment
- Reduces serialization where the statements of two transactions must generate sequential numbers at the same time
- A new sequence number can be generated or the current sequence number can be used by using NEXTVAL or CURRVAL
- They are generated independently of tables
- Used for generating unique primary keys
- The sequence number is incremented independent of transaction committing or rolling back

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 46


4.3: DB Objects
Sequence
Create sequence <sequence_name>
Increment by <value>
Start with <value>
Maxvalue <value> /nomaxvalue
Minvalue <value> / nominvalue
Cycle /nocycle
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
47

4.3: DB Objects

Sequence example

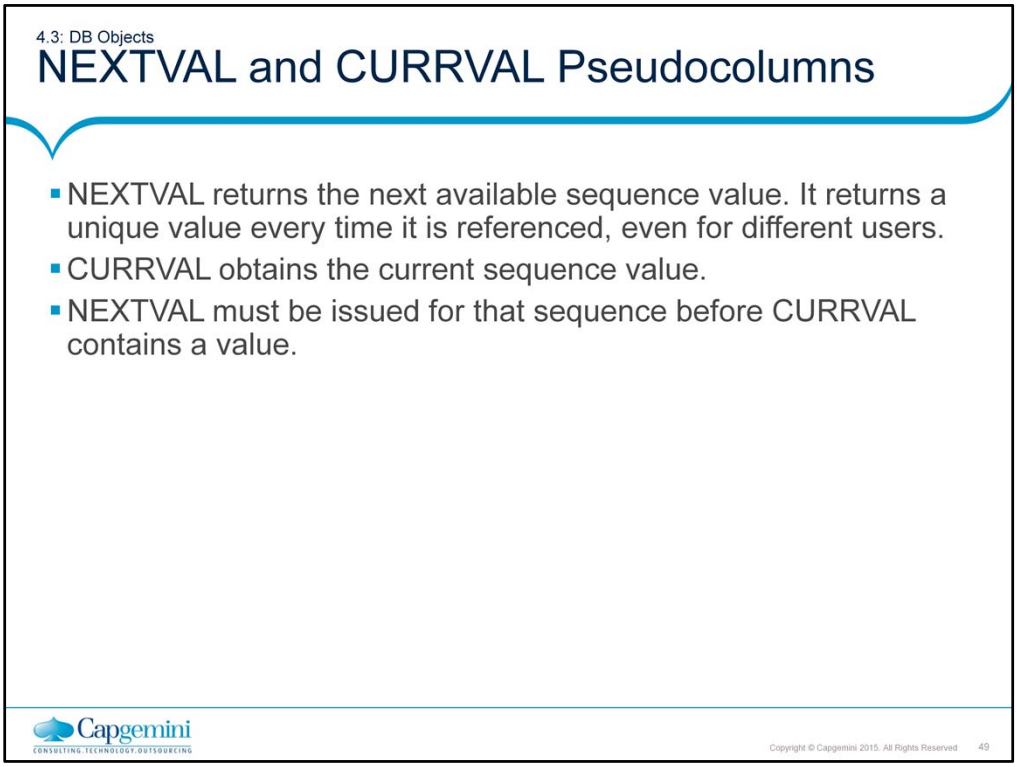
- create sequence seq_deptno
- start with 50
- increment by 10
- maxvalue 500 ;

- insert into dept VALUES (seq_deptno.NEXTVAL,'HUMAN RESOURCE','NEW YORK');

- select seq_deptno.CURRVAL from dual;

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 48

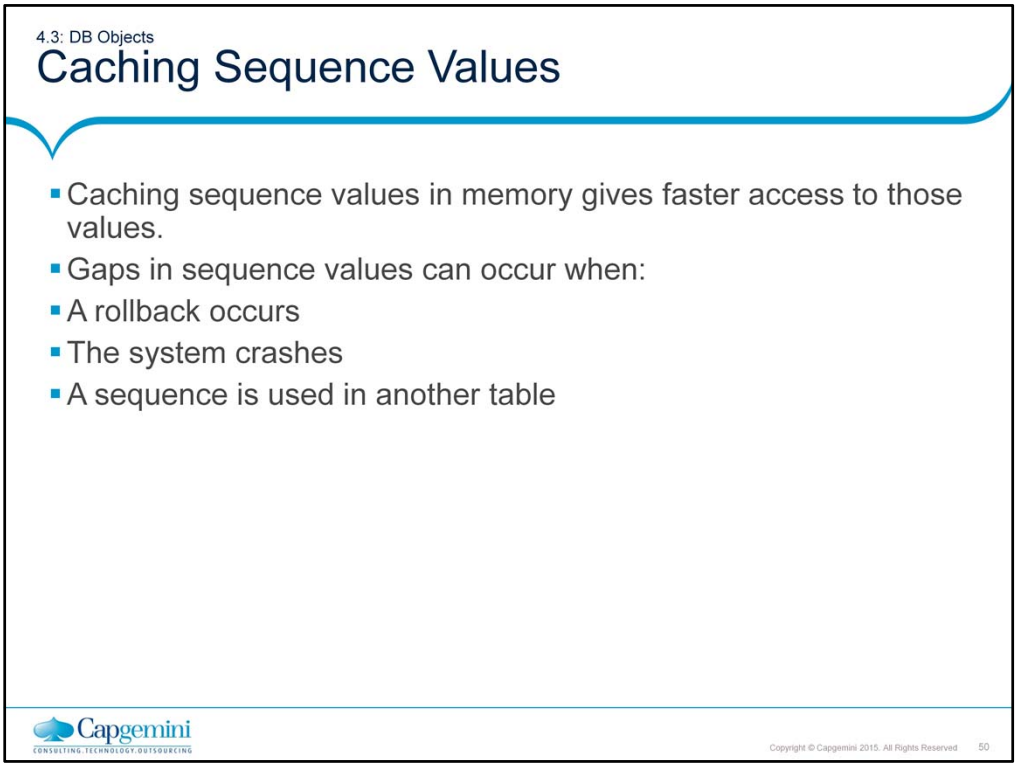


NEXTVAL and CURRVAL Pseudocolumns

After you create your sequence, it generates sequential numbers for use in your tables. Reference the sequence values by using the NEXTVAL and CURRVAL pseudocolumns.

The NEXTVAL pseudocolumn is used to extract successive sequence numbers from a specified sequence. You must qualify NEXTVAL with the sequence name. When you reference sequence.NEXTVAL, a new sequence number is generated and the current sequence number is placed in CURRVAL.

The CURRVAL pseudocolumn is used to refer to a sequence number that the current user has just generated. However, NEXTVAL must be used to generate a sequence number in the current user's session before CURRVAL can be referenced. You must qualify CURRVAL with the sequence name. When you reference sequence.CURRVAL, the last value returned to that user's process is displayed.



Caching Sequence Values

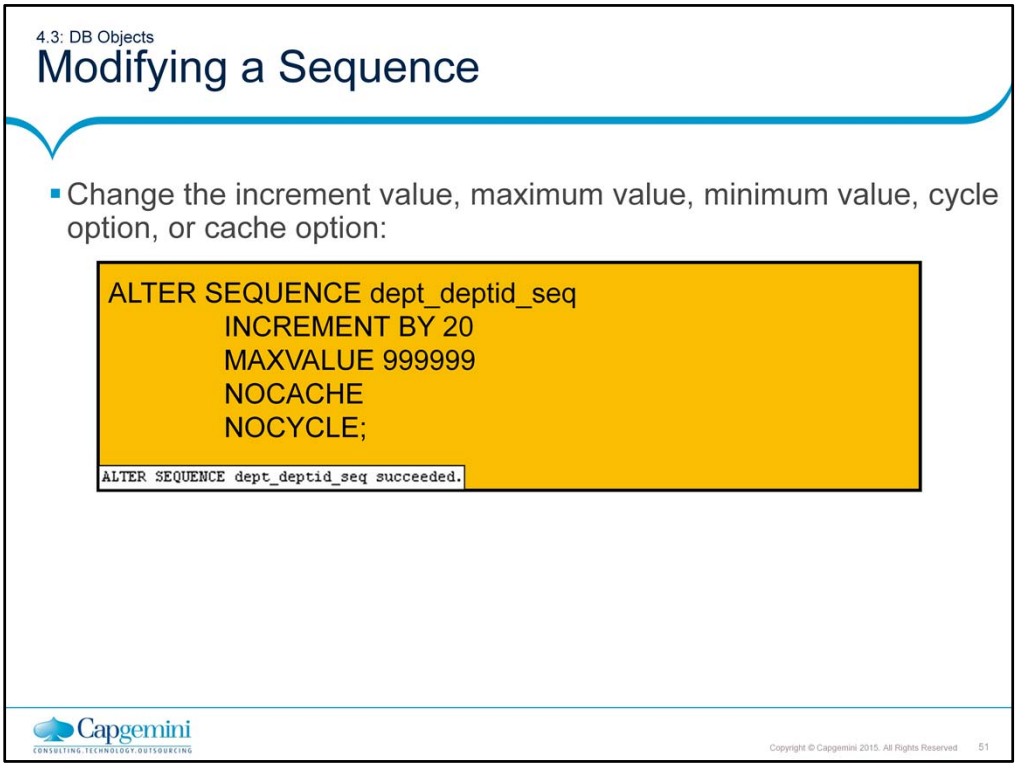
You can cache sequences in memory to provide faster access to those sequence values. The cache is populated the first time you refer to the sequence. Each request for the next sequence value is retrieved from the cached sequence. After the last sequence value is used, the next request for the sequence pulls another cache of sequences into memory.

Gaps in the Sequence

Although sequence generators issue sequential numbers without gaps, this action occurs independent of a commit or rollback. Therefore, if you roll back a statement containing a sequence, the number is lost.

Another event that can cause gaps in the sequence is a system crash. If the sequence caches values in memory, those values are lost if the system crashes.

Because sequences are not tied directly to tables, the same sequence can be used for multiple tables. However, if you do so, each table can contain gaps in the sequential numbers.



Modifying a Sequence

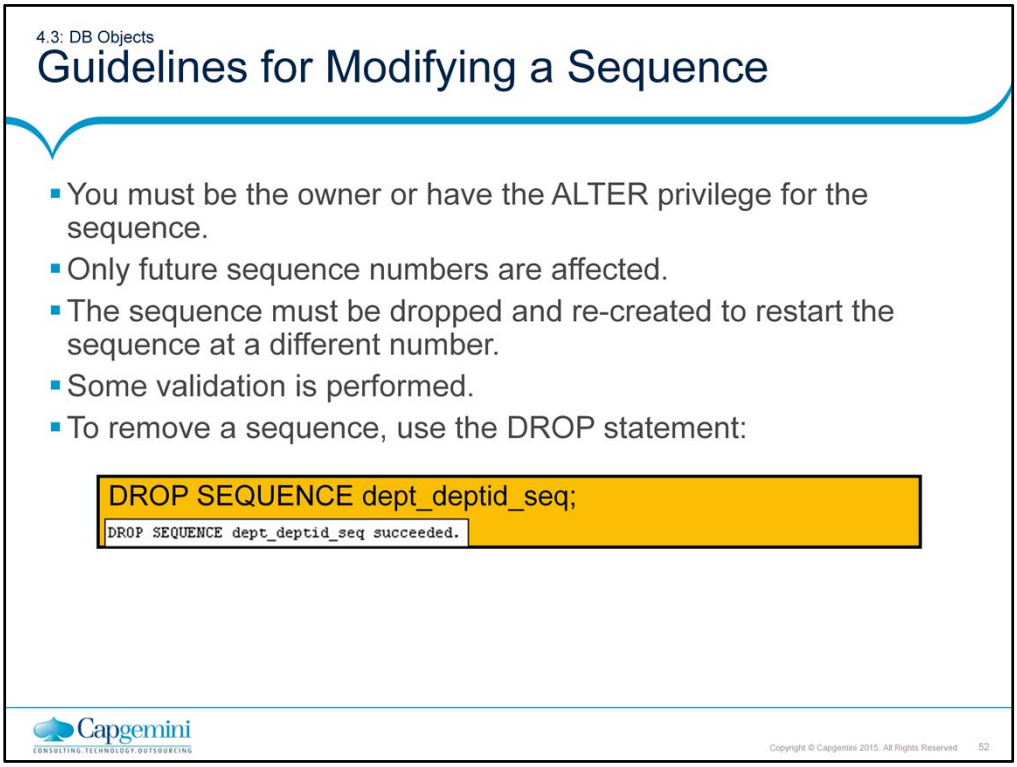
If you reach the MAXVALUE limit for your sequence, no additional values from the sequence are allocated and you will receive an error indicating that the sequence exceeds the MAXVALUE. To continue to use the sequence, you can modify it by using the ALTER SEQUENCE statement.

Syntax

```
ALTER  SEQUENCE   sequence
    [INCREMENT BY n]
    [{MAXVALUE n | NOMAXVALUE}]
    [{MINVALUE n | NOMINVALUE}]
    [{CYCLE | NOCYCLE}]
    [{CACHE n | NOCACHE}];
```

In the syntax, sequence is the name of the sequence generator.

For more information, see the section on “ALTER SEQUENCE” in Oracle Database SQL Language Reference 11g, Release 1 (11.1).



Guidelines for Modifying a Sequence

You must be the owner or have the ALTER privilege for the sequence to modify it. You must be the owner or have the DROP ANY SEQUENCE privilege to remove it.

Only future sequence numbers are affected by the ALTER SEQUENCE statement.

The START WITH option cannot be changed using ALTER SEQUENCE. The sequence must be dropped and re-created to restart the sequence at a different number.

Some validation is performed. For example, a new MAXVALUE that is less than the current sequence number cannot be imposed.

```
ALTER SEQUENCE dept_deptid_seq
    INCREMENT BY 20
    MAXVALUE 90
    NOCACHE
    NOCYCLE;
```

The error:

```
Error report:
SQL Error: ORA-04009: MAXVALUE cannot be made to be less than the current value
04009. 00000 -  "MAXVALUE cannot be made to be less than the current value"
*Cause:    the current value exceeds the given MAXVALUE
*Action:   make sure that the new MAXVALUE is larger than the current value
```

4.3: DB Objects

Synonyms

- Synonyms are alias name for table,view,sequence,procedures,functions, package,snapshots
- Hides the owner and name of the object
- Provides location transparency in distributed databases.
- Simplifies usage of SQL statements by the users
- Provides data independence
- Synonyms can be Public or Private

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 53

4.3: DB Objects

Synonym

- Create [public] synonym <synonym_name>
- For <object_name>

- create synonym balance for leave_balance;

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 54

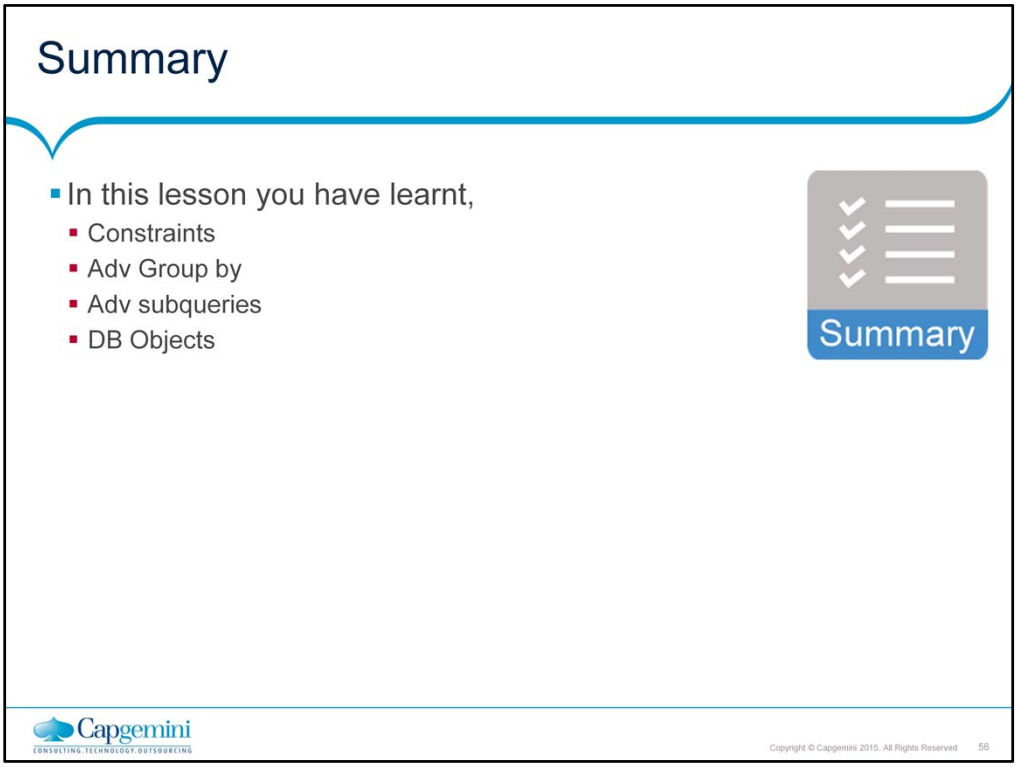
4.3: DB Objects

Removing Synonym

- Drop synonym <synonym_name>
- Drop synonym balance ;

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 55


Summary
In this lesson you have learnt,
Constraints
Adv Group by
Adv subqueries
DB Objects
Summary
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
56

Review Question

- Question 1: Which constraint will not allow to enter null values?
- Question 2: Indexes can be created _____ or _____
- Question 3: _____ obtains the current sequence value

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 57

Add the notes here.