Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Learning & Culture

SQL – Structured Query Language

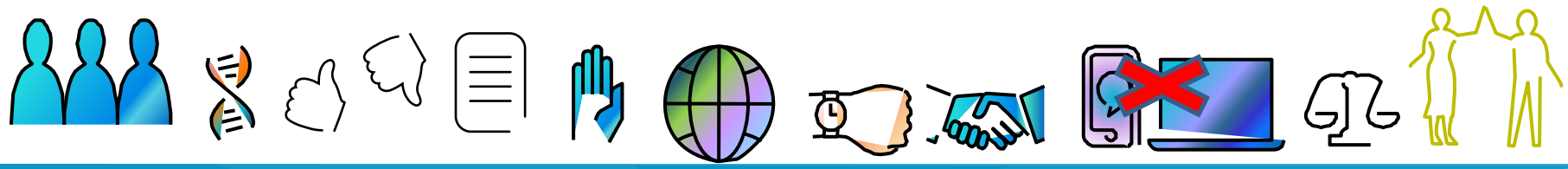
FLP - Java

Last updated: May, 21th, 2014

People matter, results count.

Ground Rules for Onsite Classrooms

- Everyone participates
- Respect individual opinions and diversities
- Be open and honest
- Give headlines, be concise
- One speaker at a time
- Make language a non-issue
- Stick to time contracts
- Seek first to understand, and then to be understood
- Clean desk / room policy
- No mobile phone, no computer
- Clients & Leadership are often here, please remember that
- Maintain a spirit of fun and enthusiasm



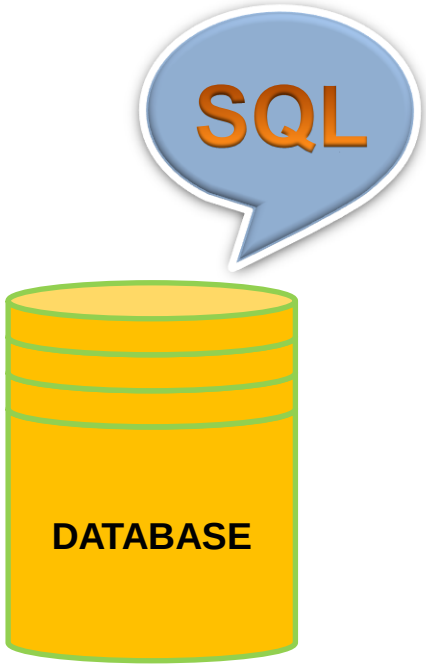
Objectives

By the end of this session, you will be able to:

- Understand what is Structured Query Language (SQL)
- List the categories of SQL commands
- Describe Schema Objects and Transactions
- Explain how to retrieve data and order data
- State the process to display data from Multiple Tables
- Describe working with SQL Functions

Introduction to Structured Query Language (SQL)

- Structured Query Language (SQL) is a database sub-language.
- It is used to interact with database in order to store, manipulate or retrieve data
- It is also pronounced as SQL or Sequel.
- It is a standard language for RDBMS.
- It is a non-procedural language.
- It is based on the relational model proposed by Dr. E.F. Codd.



Categories of SQL Commands

Data Definition Language (DDL)	Create, Alter, Drop
Data Manipulation Language (DML)	Insert , Update, Delete
Data query Language	Select
Data Control Statements	Grant, Revoke
Transaction Control Statements	Commit, Rollback, Savepoint, Set transaction

Categories of SQL Commands (contd.)

- **Data Definition Language (DDL)** statements are used to define the database structure or schema. Some examples:
  - **CREATE -** To create objects in the database.
  - **ALTER -** Alters the structure of the database.
  - **DROP -** Delete objects from the database.
  - **TRUNCATE -** Remove all records from a table, including all spaces allocated for the records are removed.

- **Data Manipulation Language (DML)** statements are used for managing data within schema objects. Some examples:
  - **INSERT -** Insert data into a table.
  - **UPDATE -** Updates existing data within a table.
  - **DELETE -** Deletes all records from a table.

Categories of SQL Commands (contd.)

- **Data Query Language (DQL)** allows you to access the data using:
  - **SELECT -** Retrieve data from the a database.
- **Data Control Language (DCL)** statements. Some examples:
  - **GRANT -** Gives user's access privileges to database.
  - **REVOKE -** Withdraw access privileges given with the GRANT command.
- **Transaction Control (TCL)** statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions:
  - **COMMIT -** Save work done.
  - **SAVEPOINT -** Identify a point in a transaction to which you can later rollback.
  - **ROLLBACK -** Restore database to original since the last COMMIT.

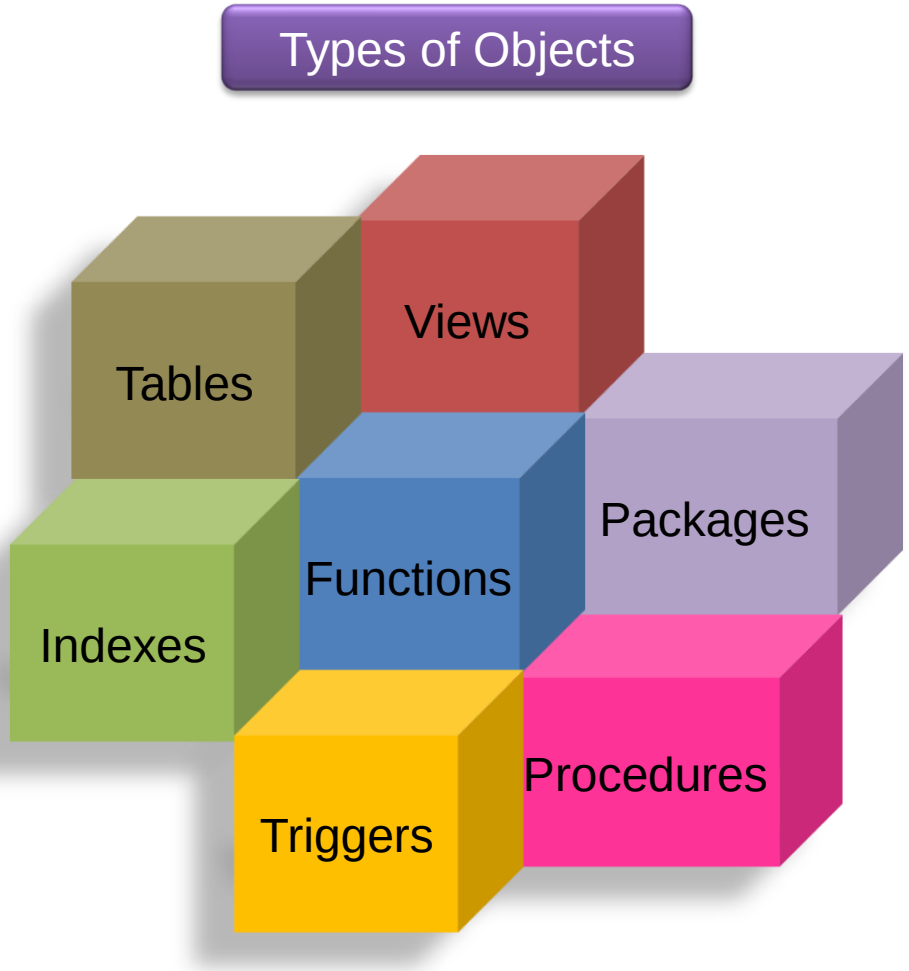
DATA Types

Data Types

Data Type	Description
VARCHAR2(*size*)	Variable-length character data
CHAR(*size*)	Fixed-length character data
NUMBER(*p*,*s*)	Variable-length numeric data
DATE	Date and time values
LONG	Variable-length character data up to 2 gigabytes
CLOB	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
ROWID	Hexadecimal string representing the unique address of a row in its table

Schema Objects

- A schema is a collection of logical structures of data, or schema objects.
- A schema is owned by a database user and has the same name as that user.
- Each user owns a single schema.
- Schema objects can be created and manipulated with SQL.



Operations on Tables

Create table	Create table employee (name varchar(20), age number);
Insert into Table	Insert into employee (name,age) values('VISHAL',29);
Update Table	Update emp set deptno = 10 where empno = 7900;
Drop Table	Drop table employee;
Truncate Table	Truncate table employee;
Delete Table	Delete table employee;

Creating a Simple Table

```
Create Table <table_name>
  ( column_name1 data_type(size),
    column_name2 data_type,
    column_name3 data_type(size),
     ……
    column_name10 data_type(size)
   );
```

Naming Conventions

- Names of object or column names cannot be greater than 30 bytes
- Special symbols like ; / *? not to be used.
- Names are not case sensitive.
- Must begin with alphabetic character unless surrounded by double quotation marks.
- Cannot be a reserved word.
- Give meaningful names.

Create Table

- Create Table Employee
  (empno number(3),
  ename varchar2(15),
  deptno number(2),
  grade char,
  salary number(8,2),
  join_date date);

- **Viewing Structure of table created above:**
  Desc <table_name>
  e.g
  Desc employee

Inserting Data

- **Insert into <table_name> (column1, column2, column3,…) values (column1_value,column2_value,…..)**

  e.g.

  Insert into Employee

  values(101,'John',10,'M',9000,'01-JAN-01');

- **Inserting Data –NULL values :**

  Insert into <table_name> values (column1_value, column2_value, NULL, column3_value,NULL, column4_value …..)

  e.g.

  Insert into Employee values(101,'John',**NULL,'M',9000,NULL**);

Inserting Data (contd.)

- **Inserting data through User Interaction:**

  Insert into <table_name> values (&var_name,&var_name,…..)

  e.g.

  Insert into Employee
  values(&emp_no,'&emp_name',&dept_no,'&grade',&salary,
  '&doj')
  /

- **Inserting data partially:**

  Insert into <table_name>(column_name1,column_name2)
  values (column_value1,column_value2);

  e.g.

  - Insert into Employee(empno,ename) values(105,'John R');
  - Insert into Employee(empno,ename) values(&emp_no,'&emp_name');

Insert – Character Strings and Date Values

- Character strings and date values are enclosed in single quotation marks.
- Character values are case sensitive.
- Date values are format sensitive.
- The default date format is DD-MON-RR.

Updating Data

- To make changes to existing rows:

  Update <table_name>
  Set col_name_1 = <new_value>,
  col_name_2 = <new_value>
  Where <condition>

- Examples:

  update emp set deptno = 10 where empno = 7900;
  update emp set deptno = 10, mgr = 7839 where empno = 7900;

Deleting Records and Dropping Table

- **Removing existing rows :**

  Delete from <table_name> Where <condition>;

  Example:

  Delete from dept where deptno = 40;

- **Dropping Table:**

  Drop table <table_name>;

  Example:

  Drop table dept;

- Specify CASCADE CONSTRAINTS to drop all referential integrity constraints that refer to primary and unique keys in the dropped table.

- Example: Drop table dept cascade constraints;

TRUNCATE Table

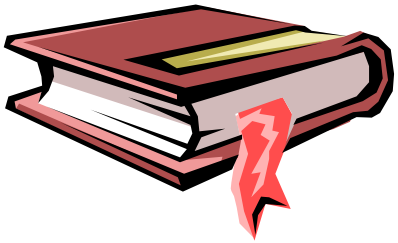
Truncate table <table_name>

- Deletes all the rows from the table.
- Frees the storage space allocated to the table.
- Cannot Rollback the changes.

Indexes

- Subset of the base table used for faster data acc.

Index



Through offset

Record

Index contains - *keys* (column values)
- *pointers* (address of data pages)

Indexes (contd.)

Types of Indexes

- Clustered Index
- Non-Clustered Index
- Composite Index

Suggestions on Indexes

- Create an index for frequently used columns based on the application query requirements.
- Small tables do not require indexes.

Clustered Index

Clustered Index

- Data storage is based on the clustered index.
- One to one between a table and a clustered index.
- Multiple columns can be used for the indexing.
- Efficient when the index is unique.
- CREATE [UNIQUE] CLUSTERED INDEX index_name ON table (column1, column2,…columnn)

Clustered Index

Non-Clustered Index

- Data storage is NOT based on the non-clustered index.
- Items are stored in the order of the key values mentioned.
- Data is stored separate from the index.
- Multiple columns can be used for the indexing.
- It is the default index that gets created.
- 256 non-clustered indexes may exist.
- CREATE [UNIQUE] NONCLUSTERED INDEX index_name ON table (column1, column2,…columnn)

Composite Index

Composite Index

- When two or more columns are specified at the time of index creation.
- Maximum of 16 columns can be specified.
- Can be both clustered and non-clustered.
- Can be unique.
- Used to search multiple columns.
- Smaller indexes are much more efficient.

Views

Views

- Views are virtual tables.
- Views are always based on the tables.
- Views are queries stored through which data goes in the underlying table.
- Using views we can hide the original table and give some conditional access of records to the user.
- Views can be based on one or more than one tables.
- If view is based on one table then directly insert, update or delete can be done by which the data will be affected in the table.
- But cannot do any DML directly on views which are based on more than one tables.

Views (contd.)

- create or replace view V1

  as

  select * from emp

  where sal >= 3500;

- **Insert into the view:**

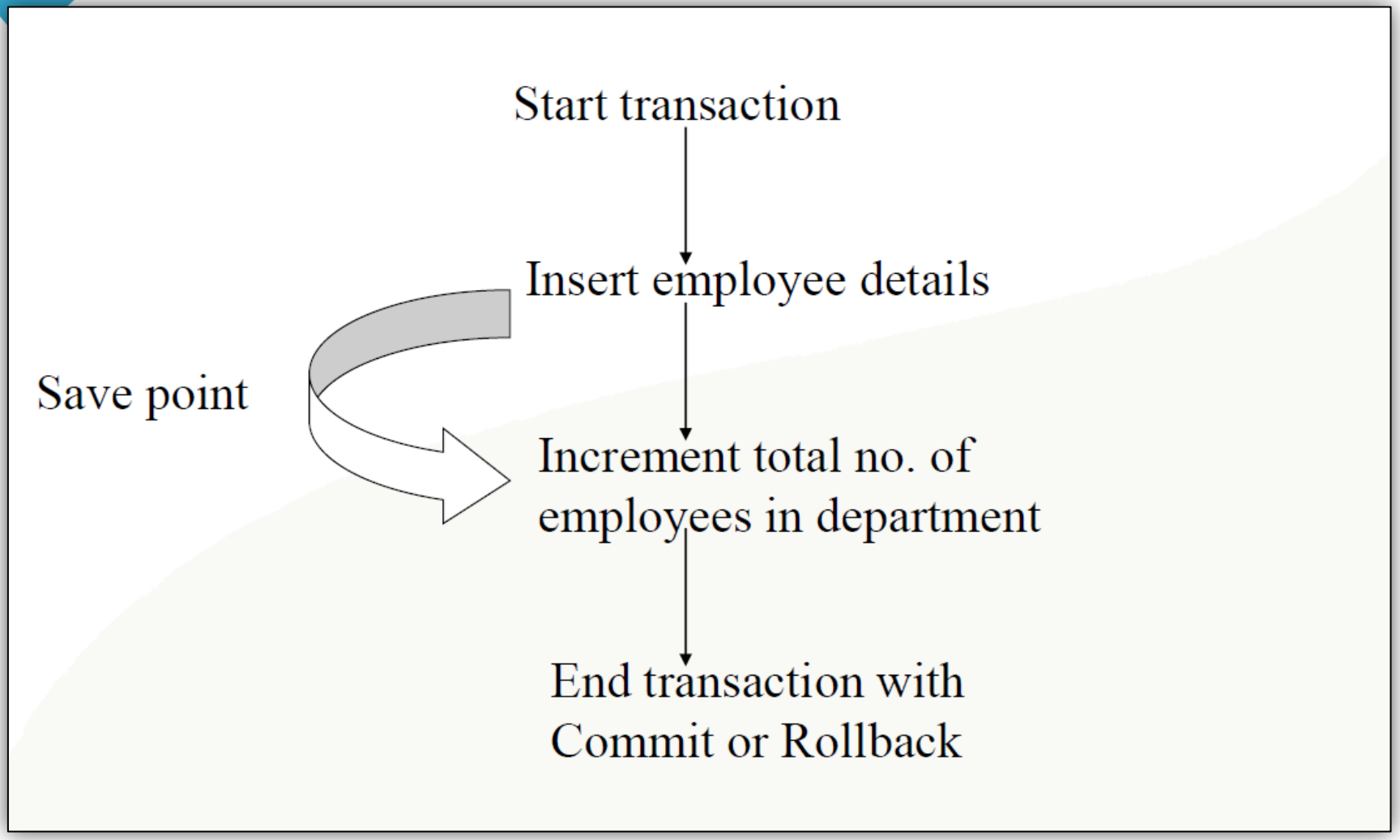
  Insert into v1(empno,ename,sal,deptno)

  values(100,'abc',10000,10);

Transactions

- **Transaction:** A set group of data manipulation commands is known as transaction.
- **Example:** When a employee joins, a transaction would be completed when his record is entered in employee table and total number of employees in department table is incremented by 1 for the respective department
  - **Commit** - Makes the DML changes of the transaction permanent.
  - **Rollback** - Undo the changes made through DML.
  - **Savepoint** - Intermediate points for rollback. A long transaction can be broken at different levels to ease COMMIT or ROLLBACK.

Transactions (contd.)



Transactions (contd.)

- Begin when the first DML SQL statement is executed.
- End with one of the following events:
  - A COMMIT or ROLLBACK statement is issued.
  - A DDL or DCL statement executes (automatic commit).

- Example of Savepoint:

  Insert ….

  Insert ....

  Savepoint this_is_insert

  Update….

  Update….

  Savepoint this_is_update

  Delete ….

  Rollback to this_is_insert

Implicit Transaction Processing

- An automatic commit occurs under the following circumstances:
  - DDL statement is issued.
  - DCL statement is issued.
  - Normal exit from session (SQL*Plus), without explicitly issuing COMMIT or ROLLBACK statements.
- An automatic rollback occurs under an abnormal termination of SQL*Plus or a system failure.

Retrieving Data

‘Select’ statement in a Query

- Retrieves rows and columns.
- May include more than one table or more than one database.
- Defines the columns required by the query as a series of expressions.
- Resulting columns are in the same order as in the query.

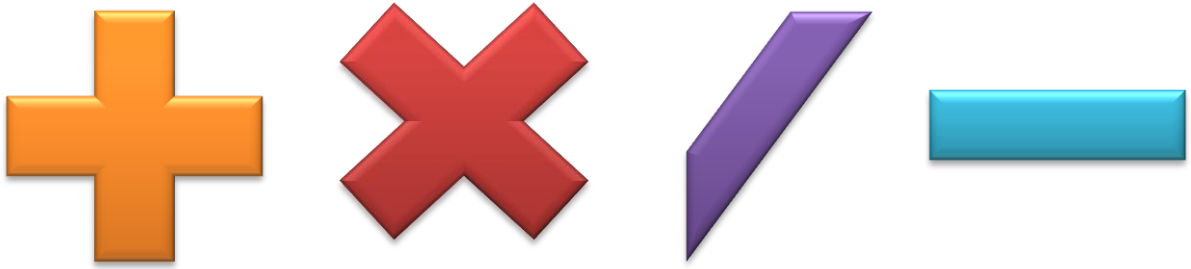
Retrieving Data (contd.)

- Select Distinct /*/<column_list>
  From <table_name>
  Where <conditions>
  Group By <column_list>
  Having <conditions>
  Order By <column_list>

- Simple retrieval examples:
  - Select * from emp;
  - Select empno,ename,deptno,sal from emp;
  - Select dname from dept;
  - Select empno,ename deptno, sal from emp where empno = 7900;

Performing Arithmetic Calculation on Columns

- Airthmetic Opeartors **+** , ***** , ***/*** , **-** can be used on numeric or date datatype columns:
  - Select empno,ename,sal, **sal*12 from emp;**
  - Select ename,hiredate, **hiredate + 30 from emp;**
  - Select ename,hiredate, **sysdate – hiredate from emp;**
  - Select empno,ename,sal,comm, **sal + comm from emp;**



Operator Precedence

- * / + -
- Multiplication and division take priority over addition and subtraction.
- Operators of the same priority are evaluated from left to right.
- Parentheses are used to force prioritized evaluation and to clarify statements.

Handling NULL Values

- Select empno, ename , sal,sal + comm from emp;
- Arithmetic manipulation by NULL value will always be NULL.
- NULL values can be handled by using NVL function.
- NVL(<column_name/value>,<value_substituted>).
- NVL function helps to substitute NULL value with a different value of the same datatype.
- Select empno,ename,sal,sal + nvl(comm,0) from emp.

Concatenating Columns / Literals

- ||(Pipe) is used to join two columns or strings of same or different data-type.
- Resultant column is of character datatype.
  - select ename ||job from emp;
  - select ename || ' -' ||job from emp;
  - select 'Employee '||ename ||' is working since ' ||hiredate from emp;

Giving Alias name to Columns

- Select empno,ename,sal,sal*12 annual_salary from emp;
- Select empno,ename,sal,sal*12 ASannual_salary from emp;
- Select empno,ename,sal,comm "Commission Earned“ from emp;
- Select 'Employee ' || ename || ' is working since ' || hiredate Employee_Information from emp;

Eliminating Duplicate Rows

Distinct displays unique values for the columns:

- select distinct job from emp;
- select distinct deptno , jobfrom emp;

Relational Operators

=	Checks for equality comparison between two values
<> , != , ^=	Checks for in-equality comparison between two values
>	Checks for greater than value
<	Checks for less than value
>=	Checks for greater than and equal to value
<=	Checks for less than and equal to value
[NOT] Between…And	Checks for value in Between the Range of Values , inclusive at both ends
[NOT] In	Checks for specific list of values
[NOT] Like	Checks for matching pattern using wildcards - % for all characters and _ for a single character
Is [NOT] NULL	Checks for NULL value

Examples

- Select * from emp where deptno **=10;**
- Select * from emp where sal **>3000;**
- Select * from emp where deptno **in(10,30);**
- Select * from emp where deptno **>=110;**
- Select * from emp where deptno **<=100;**
- Select * from emp where sal **between 1000 and 3000;**
- Select * from emp where hiredate **between '01-JAN-85' and '01-JAN-95‘;**
- Select * from emp where comm **is not null;**
- Select * from emp where ename **like 'B%’;**
- Select * from emp where ename **like '_L%‘;**

Logical Operators

AND	Used to join two conditions with AND logical operator. Both of the conditions should evaluate to TRUE
OR	Used to join two conditions with OR logical operator. Either of the condition can be TRUE or FALSE
NOT	Used for negating the result of the condition

Examples

- Select * from emp where sal > 2000 **and** deptno = 10;
- Select ename,sal,comm,sal+nvl(comm,0) from emp where sal > 1500 **OR** sal+nvl(comm,0) > 1500;
- Select * from emp where **notsal > 2000;**

Rules of Precedence

Order Evaluated	Operator
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT logical condition
7	AND logical condition
8	OR logical condition

- **Note:** Use parentheses to override rules of precedence.

Ordering Data

- Data is randomly retrieved in any order.
- ORDER BY can be used to order the display of data in a specific order.
- Last clause in the Select statement.

- **ORDER BY <column> ASC/DESC , <column> ASC/DESC**
  - Select * from emp order by ename;
  - Select * from emp order by deptno,ename

Order By

- Calculated columns can be used to order data.
- Column alias names can be used to order data.
- Columns not included in the select list can be used to order data.
- Null values appear last in an ascending sort and first in a descending sort.

- **Example:**
- Select ename,sal,comm, **SAL+NVL(COMM,0)** from emp where sal > 1500 order by **SAL+NVL(COMM,0)**

- Select ename,sal,comm, **SAL+NVL(COMM,0)total_salary** from emp where sal > 1500 order by **total_salary**

Table Alias Name

- An alias name which is a short name can be give to a table by specifying it after the table name.
- The alias name and column name can be used as short-cuts to table name and column name specification.

- Example:
  - Select empno, ename, deptno from emp e;
  - Select e.empno, e.ename, e.deptno from emp e;

Displaying Data from Multiple Tables

- **Department and Employee Data**

Department Table

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

Employee Table

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	9-Jun-81	2450		10
7839	KING	PRESIDENT		17-Nov-81	5000		10
7934	MILLER	CLERK	7782	23-Jan-82	1300		10
7369	SMITH	CLERK	7902	17-Dec-80	800		20
7876	ADAMS	CLERK	7788	12-Jan-83	1100		20
7902	FORD	ANALYST	7566	3-Dec-81	3000		20
7788	SCOTT	ANALYST	7566	9-Dec-82	3000		20
7566	JONES	MANAGER	7839	2-Apr-81	2975		20
7499	ALLEN	SALESMAN	7698	20-Feb-81	1600	300	30
7698	BLAKE	MANAGER	7839	1-May-81	2850		30
7654	MARTIN	SALESMAN	7698	28-Sep-81	1250	1400	30
7900	JAMES	CLERK	7698	3-Dec-81	950		30
7844	TURNER	SALESMAN	7698	8-Sep-81	1500	0	30
7521	WARD	SALESMAN	7698	22-Feb-81	1250	500	30

Displaying Data from Multiple Tables (contd.)

- Data can be displayed from more than one table by joining the tables on the common values.
- Oracle performs a join query when more than one table appears in the FROM list.
- The join is performed by using a condition in the **WHERE clause which is known as join condition**.
- Oracle combines pairs of rows, each containing one row from each table, for which the join condition evaluates to TRUE.
- **select empno,ename,sal,dname**

  from emp, dept

  where emp.deptno= dept.deptno

Joins

- The column names can be different in the tables used for joining.
- The data-type should match.
- Joins are performed on the actual values.
- Without a Join condition , the data displayed would be Cartesian product of all the tables used in the FROM clause.
- The number of joins would be one less than the number of tables.
- Types Of Joins:
  - Equi
  - Non-Equi
  - Self
  - Outer

Equi Join

- The default type of join used is Equi join or Cartesian join.
- Based on **Equality join condition.**

  Examples:
- Select empno,ename,dname
  from emp e, dept d
  where e.deptno=d.deptno;

- Select e.empno,e.ename,dname,pl,sl,cl
  from emp e, dept d, leave_balance l
  where e.deptno = d.deptno
  and e.empno = l.empno
  and d.deptno in (10,30)
  and hiredate < '01-JAN-82‘
  Order By e.ename;

Non-Equi Join

- A join based on any other operator except the equi join is known as Non-Equi Join.
- The tables involved in the join do not have a common column to be joined.

Examples:

- select empno,ename,losal, hisal, sal,grade
  from emp, salgrade
  where sal **between losal and hisal**

Self Join

- A type of Equi join which joins to the same table itself.

  Example:
- List the employee names along with their manager.

- select x.empno, x.ename, y.empno "mgr no ", y.ename "mgr name”
  from emp x, emp y
  where x.mgr =y.empno

Self Join

- A type of Equi join which joins to the same table itself.

  Example:
- List the employee names along with their manager.

- select x.empno, x.ename, y.empno "mgr no ", y.ename "mgr name”
  from emp x, emp y
  where x.mgr =y.empno

Self Join Data

Employee X Table

EMPNO	ENAME	JOB	MGR	SAL
7369	SMITH	CLERK	7902	800
7499	ALLEN	SALESMAN	7698	1600
7521	WARD	SALESMAN	7698	1250
7566	JONES	MANAGER	7839	2975
7654	MARTIN	SALESMAN	7698	1250
7698	BLAKE	MANAGER	7839	2850
7782	CLARK	MANAGER	7839	2450
7788	SCOTT	ANALYST	7566	3000
7839	KING	PRESIDENT		5000
7844	TURNER	SALESMAN	7698	1500
7876	ADAMS	CLERK	7788	1100
7900	JAMES	CLERK	7698	950
7902	FORD	ANALYST	7566	3000
7934	MILLER	CLERK	7782	1300

Employee Y Table

EMPNO	ENAME	JOB	MGR	SAL
7902	FORD	ANALYST	7566	3000
7698	BLAKE	MANAGER	7839	2850
7839	KING	PRESIDENT		5000
7566	JONES	MANAGER	7839	2975
7788	SCOTT	ANALYST	7566	3000
7782	CLARK	MANAGER	7839	2450
7369	SMITH	CLERK	7902	800
7499	ALLEN	SALESMAN	7698	1600
7521	WARD	SALESMAN	7698	1250
7654	MARTIN	SALESMAN	7698	1250
7844	TURNER	SALESMAN	7698	1500
7876	ADAMS	CLERK	7788	1100
7900	JAMES	CLERK	7698	950
7934	MILLER	CLERK	7782	1300

Self Join Example

- List the employees earning more than BLAKE.

- Select x.empno, x.ename, y.empno “EmpNo.ofBlake",
  y.sal “SalaryofBlake", x.sal "Empsalary"
  from empx, empy
  where y.ename = 'BLAKE‘
  And x.sal > y.sal

Employee X Table

EMPNO	ENAME	JOB	MGR	SAL
7369	SMITH	CLERK	7902	800
7499	ALLEN	SALESMAN	7698	1600
7521	WARD	SALESMAN	7698	1250
7566	JONES	MANAGER	7839	2975
7654	MARTIN	SALESMAN	7698	1250
7698	BLAKE	MANAGER	7839	2850
7782	CLARK	MANAGER	7839	2450
7788	SCOTT	ANALYST	7566	3000
7839	KING	PRESIDENT		5000
7844	TURNER	SALESMAN	7698	1500
7876	ADAMS	CLERK	7788	1100
7900	JAMES	CLERK	7698	950
7902	FORD	ANALYST	7566	3000
7934	MILLER	CLERK	7782	1300

Employee Y Table

EMPNO	ENAME	JOB	MGR	SAL
7902	FORD	ANALYST	7566	3000
7698	BLAKE	MANAGER	7839	2850
7839	KING	PRESIDENT		5000
7566	JONES	MANAGER	7839	2975
7788	SCOTT	ANALYST	7566	3000
7782	CLARK	MANAGER	7839	2450
7369	SMITH	CLERK	7902	800
7499	ALLEN	SALESMAN	7698	1600
7521	WARD	SALESMAN	7698	1250
7654	MARTIN	SALESMAN	7698	1250
7844	TURNER	SALESMAN	7698	1500
7876	ADAMS	CLERK	7788	1100
7900	JAMES	CLERK	7698	950
7934	MILLER	CLERK	7782	1300

Self Join Example (contd.)

- List the employees in the same dept as BLAKE.
- Select x.empno,x.ename, y.deptno "Dept No of Blake",
  x.deptno "Emp Dept"
  from emp x, emp y
  where y.ename = 'BLAKE'
  and x.deptno = y.deptno

Outer Join

- A join which returns the rows for unmatched rows of the joined tables.
- An outer join returns all rows that satisfy the join condition and those rows from one table for which no rows from the other satisfy the join condition.
- **(+)** is used to specify Outer join.
- Outer join is placed on the column of the table (transaction) which would return rows for the unmatched values from the other joined table (master).

  Example:
- Listing dept with or without employees
- select d.deptno,dname, empno

  from emp e, dept d

  where e.deptno**(+) = d.deptno;**

Department and Employee Data – Outer Join

Department Table

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

Employee Table

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	9-Jun-81	2450		10
7839	KING	PRESIDENT		17-Nov-81	5000		10
7934	MILLER	CLERK	7782	23-Jan-82	1300		10
7369	SMITH	CLERK	7902	17-Dec-80	800		20
7876	ADAMS	CLERK	7788	12-Jan-83	1100		20
7902	FORD	ANALYST	7566	3-Dec-81	3000		20
7788	SCOTT	ANALYST	7566	9-Dec-82	3000		20
7566	JONES	MANAGER	7839	2-Apr-81	2975		20
7499	ALLEN	SALESMAN	7698	20-Feb-81	1600	300	30
7698	BLAKE	MANAGER	7839	1-May-81	2850		30
7654	MARTIN	SALESMAN	7698	28-Sep-81	1250	1400	30
7900	JAMES	CLERK	7698	3-Dec-81	950		30
7844	TURNER	SALESMAN	7698	8-Sep-81	1500	0	30
7521	WARD	SALESMAN	7698	22-Feb-81	1250	500	30

Working with SQL Functions

- Functions work on the arguments provided to manipulate data value and return a result.
- SQL Functions are built-in functions provided by Oracle to be used by SQL statements.
- Types of Functions:
  - Single Row Functions :
    - String / Character
    - Date and Time
    - Number
    - Conversion
    - Common
  - Aggregate -Group/ Columnar Functions
  - Analytical

Single-Row Function

- Single-row functions return a single result row for every row of a queried table.
- The function can be used in select lists and where clause.

Single-Row String Functions

Lower(str)	Converts the string into lower case
Upper(str)	Converts the string into upper case
Initcap(str)	Converts the string into proper case
Length(str)	Returns the number of characters in the string
Lpad(str,length,char_set)	Pads the character specified by character set upto the length of the string on the left side
Rpad(str,length,char_set)	Pads the character specified by character set upto the length of the string on the right side

Single-Row String Functions (contd.)

Substr(str,starting_position, no_of_char)	Returns the characters from the position specified by starting position upto the number of characters specified
Replace(str,search_str, replace_str)	Replaces the characters in string specified in the search string with the replace string
Rtrim(str,characters)	Trims the specified characters from the right side of the string.By Default trims blank characters/spaces
Ltrim(str,characters)	Trims the specified characters from the left side of the string.By Default trims blank characters/spaces
Trim(str)	Trims spaces from the string

Examples of Single-Row Functions

- select ename, **lower(ename)**, **upper(ename)**, **initcap(ename)**, **length(ename)** from emp;
- select **rpad(job,20,'*')** from emp;
- select ename, **ltrim(ename)**, **ltrim(ename,'SM')**, **rtrim(ename,'NS')** from emp;
- select ename, **substr(ename,2,5)** from emp;
- select ename, **replace(ename,'AM','*#')** from emp;

Date Functions

Add_months(date, no_of_months_to_add)	Returns a date after adding number of specified months to the date
Last_day(date)	Returns the last date of the month for the given date
Months_between(date1,date2)	Returns the number of months between two dates

Examples of Date functions

- Select sysdate, last_day(sysdate) from dual;
- Select ename,hiredate, months_between(sysdate,hiredate) "worked",
  add_months(hiredate,6) "confirm"
  from emp;

Numeric Functions

ABS(number)	Returns the absolute value of the number
Mod(number,divisor)	Returns modules of the number divided by the divisor
Power(number,exponent)	Returns the value raised to an exponent power
Sqrt(number)	Returns the square root of the number
Sign(number)	Returns 1 if number is positive or negative or 0 if zero

Examples on Numeric functions

select abs(-100),mod(36,5),power(2,4),sqrt(121) from dual;

Common Functions

Round(value,format_specifier) For – date, number	Rounds the value of number/date to the specified precision/format in the format specifier
Truncate(value,format_specifier) For - date,number	Truncates the value of number/date to the specified precision/format in the format specifier
NVL(value,substitute_value) For – date,number,character	Substitutes the NULL value with the specified substitute value
Greatest(value1,value2,value3..) For – date,number	Returns the greatest value from the list specified
Least(value1,valu2,value3...) For – date,number	Returns the least value from the list specified

Examples for Common Functions

- select **sysdate,round(sysdate,'month')** from dual;
- select **sysdate, trunc(sysdate,'month')** from dual;
- select **round(186.956, 2), round(186.956,-2)** from dual;
- select **trunc(186.956, 2), trunc(186.956,-2)** from dual;
- select ename,sal,comm, **greatest(sal,comm), least(sal,comm)** from emp
- select ename,sal, **nvl(comm,0)** from emp;
- select ename,sal,comm, **greatest(sal,nvl(comm,0))** from emp;
- select **greatest('01-JAN-85','31-MAR-99','01-APR-95')** from dual;
- select **nvl(hiredate,'01-JAN-00‘)** from emp;

Conversion Functions

To_char(number,format_specifier)	Converts a numeric value into character value according to the format specifier
To_char(date,format_specifier)	Converts the date into character value according to the format specifier
To_Number(character_number, format_specifier)	Converts a numeric value of character type as specified by format specifier to number
To_date(character_date, format-specifier)	Converts a character date as specified by the format specifier to date in default format

Examples of Conversion functions

- Select **to_char(123,'$999.999')** from dual;
- Select ename, **to_char(hiredate,'dd/mm/yyyy ')** from emp;

Group / Aggregate Functions

- Aggregate functions return a single result row based on groups of rows.
- They are commonly used with the GROUP BY clause in a SELECT statement.

Count(*/column)	Counts the number of rows for the specified column for group of rows
Max(column)	Returns the maximum value for the specified column for group of rows
Min(column)	Returns the minimum value for the specified column for group of rows
Sum(column)	Returns the total of all values for the specified column for group of rows
Avg(column)	Returns the average value for the specified column for group of rows

Examples

- **Display total number of employees, total , maximum , minimum and average salaries paid**

  select **count(*),sum(sal),max(sal),min(sal),avg(sal)** from emp;

- **Display maximum , minimum and average salaries paid in department 30 / SALES**

  select **max(sal),min(sal),avg(sal)** from emp
  where deptno = 30

  select **max(sal),min(sal),avg(sal)**
  from emp e, dept d
  where e.deptno = d.deptno
  and dname **=** 'SALES‘

- **Find out the difference between maximum and minimum salaries paid**

  select **max(sal)-min(sal)** "Difference” from emp;

- **Find out how many distinct jobs are held**

  select **count(distinct job)** from emp

Grouping Data

- Data can be grouped to obtain summary information for each group.
- Aggregate / Group functions like -count, max, min can be used to get summarized values for the group.
- A **GROUP BY** clause of SELECT statement is used to group data.
- A single row is returned for each group.
- The largest group is the table itself.
- The select column list can have only the columns used for grouping data and the aggregate function.
- Individual columns / single –row columns cannot be used in the select column list.
- Group functions cannot be used in the WHERE clause.

Grouping Data-Group by Deptno

Employee Table

EMPNO	ENAME	JOB	MGR	SAL	DEPTNO	
7369	SMITH	CLERK	7902	800	10	
7499	ALLEN	SALESMAN	7698	1600	10	
7521	WARD	SALESMAN	7698	1250	10	10
7698	BLAKE	MANAGER	7839	2850	20	
7566	JONES	MANAGER	7839	2975	20	
7782	CLARK	MANAGER	7839	2450	20	20
7654	MARTIN	SALESMAN	7698	1250	20	
7839	KING	PRESIDENT		5000	30	
7844	TURNER	SALESMAN	7698	1500	30	
7900	JAMES	CLERK	7698	950	30	30
7934	MILLER	CLERK	7782	1300	30	

Examples of Group By

- **Number of employees department wise**

```
select deptno,count(*) from emp
group by deptno;

select dname,count(*) from emp e, dept d
where d.deptno = e.deptno
group by dname;
```

HAVING Clause

- The HAVING clause is used in combination with the GROUP BY clause. It can be used in a SELECT statement to filter the records that a GROUP BY returns.

- **The syntax for the HAVING clause is:**

```
SELECT column1, column2, ... column_n, aggregate_function
(expression)
FROM tables
WHERE predicates
GROUP BY column1, column2, ... column_n
HAVING condition1 ... condition_n;
```

Example: HAVING Clause

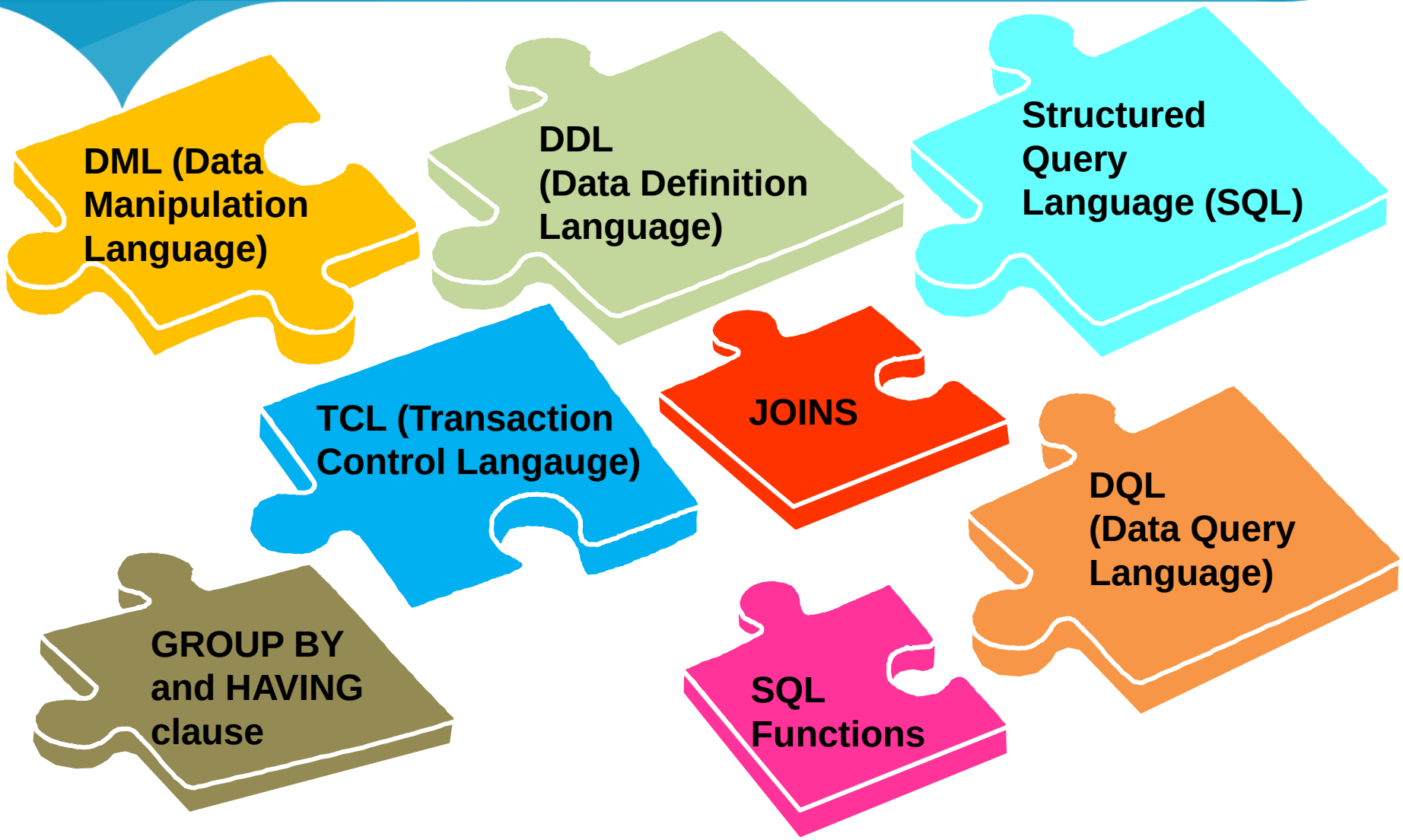
- SUM function can be used to return the name of the department and the total sales (in the associated department). The HAVING clause will filter the results so that only departments with sales greater than $1000 will be returned:

```
SELECT department, SUM(sales) as "Total sales"
FROM order_details
GROUP BY department
HAVING SUM(sales) > 1000;
```

- COUNT function can be used to return the name of the department and the number of employees (in the associated department) that make over $25,000 / year. The HAVING clause will filter the results so that only departments with more than 10 employees will be returned:

```
SELECT department, COUNT(*) as "Number of employees"
FROM employees
WHERE salary > 25000
GROUP BY department
HAVING COUNT(*) > 10;
```

Recap



Thank You

About Capgemini

With more than 120,000 people in 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The group reported 2011 global revenues of EUR 9.7 billion.
Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Rightshore® is a trademark belonging to Capgemini

www.capgemini.com

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

People matter, results count.