CERN-IT/ADC

Oracle Tutorials

SQL
Structured Query Language
(1/2)

Giacomo Govi
IT/ADC

25 January 2005

Overview

- Goal:
  - Learn the basic for interacting with a RDBMS
- Outline
  - SQL generalities
  - Available statements
  - Restricting, Sorting and Aggregating data
  - Manipulating Data from different tables
  - SQL Functions

SQL Definition

Structured Query Language

- Non-procedural language to access a relational database
- Used to create, manipulate and maintain a relational database
- Official ANSI Standard

SQL as RDBMS interface

SQL provides statements for a variety of tasks, including:

Data Definition
- Creating, replacing, altering, and dropping objects

Data Manipulation
- Querying data
- Inserting, updating, and deleting rows in a table

Data Control
- Controlling access to the database and its objects
- Guaranteeing database consistency and integrity

SQL unifies all of the preceding tasks in one consistent language.

Available statements

Statement		Description
SELECT		Data retrieval
INSERT UPDATE DELETE	Rows	Data Manipulation Language (DML)
CREATE ALTER DROP RENAME TRUNCATE	Tables/Objects	Data Definition Language (DDL)
COMMIT ROLLBACK SAVEPOINT	Manages DML	Transaction Control
GRANT REVOKE		Data Control Language (DCL)

SQL & Tools

- SQL statements can be submitted via:
  - DB API's for programming languages (C, C++, Java, Python, PHP, …)
  - GUI applications (Excel, Access)
  - stored procedures (PL/SQL, Java)
  - Oracle tools (Reports, Forms, Designer…)
- SQL*Plus (Oracle!) is the basic tool to submit SQL commands (available on all CERN platforms).

To use SQL at CERN

- Direct connection to the database

  i.e. from lxplus

  sqlplus user@sid

- Benthic Software

  to install it, refer to:

  G:\Applications\Benthic\Benthic_license_CERN.html

  http://www.benthicsoftware.com/

Datatypes

Each value manipulated by a RDBMS has a datatype that defines the domain of values that each column can contain

- When you create a table, you must specify a datatype for each of its columns.

ANSI defines a common set

- Oracle has its set of built-in types
- User-defined types

Oracle Built-in Datatypes

`CHAR (size)`	fixed-length char array
`VARCHAR2(size)`	Variable-length char string
`NUMBER (precision, scale)`	any numeric
`DATE`	date
`TIMESTAMP`	date+time
`CLOB`	char large object
`BLOB`	binary large object
`BINARY_FLOAT`	32 bit floating point
`BINARY_DOUBLE`	64 bit floating point

… + some others

ANSI Data types translation

ANSI data type	*Oracle*
integer	NUMBER(38)
smallint	NUMBER(38)
numeric(p,s)	NUMBER(p,s)
varchar(n)	VARCHAR2(n)
char(n)	CHAR(n)
datetime	DATE
float	NUMBER
real	NUMBER

NULL value

NULL is a special value that means:

- unavailable
- unassigned
- unknown
- inapplicable

NULL value is not equivalent to

- zero
- blank space

Often used as default

Schema

A schema is a collection of logical structures of data, called schema objects.

- It is owned by a database user and has the same name of the user.
- Schema objects can be created and manipulated with SQL

Schema objects

-Tables
-Indexes
-Constraints
… but also (in ORACLE)
-Links
-Views
-Triggers
-Operators
-Sequences
-Stored functions
-Stored procedures
-Synonyms

…and more

Basic SQL

Aim: be able to perform the basic operation of the RDBMS data model:

- Create, Modify the layout of a table
- Remove a table from the user schema
- Insert data into the table
- Retrieve data from one or more tables
- Update/ Delete data in a table

Create a table

Define the table layout:

- Table identifier
- Column identifiers and data types
- Integrity/Consistency:
  - Column constraints, default values
  - Relational constraints

Create (and describe) a table

```
CREATE TABLE employees (
     id                    NUMBER(4),
     surname                 VARCHAR2(50),
     name                   VARCHAR2(100),
     hiredate              DATE DEFAULT SYSDATE,
     division              VARCHAR2(20),
     email                   VARCHAR2(20),
     citizenship             VARCHAR2(20)
   );
```

Retrieve the table layout:

```
DESCRIBE employees;
Name         Null?   Type
 ---------- ------ ----------
 ID                  NUMBER(4)
 ...
```

Using relational model

What's missing in the previous slide?

Syntax is correct, but we should impose a few more check to some of the columns:

- id is used to fully identify an employee
  - it must always have a value!
  - each employee (row) must have a different id
- name and surname must always have a value
- division and citizenship are not expected to accept ANY value…

The application filling the table could check all that…

Actually this is what the RDBMS is supposed to DO!

CONSTRAINT

-> column property defining range of values, relationship with other column inside the same or other tables

Giacomo.Govi@cern.ch

Create a *relational* table

```
CREATE TABLE employees  (
    id                      NUMBER(4)      NOT NULL,
    surname                 VARCHAR2(50)   NOT NULL,
    name                    VARCHAR2(100)  NOT NULL,
    hiredate                DATE DEFAULT SYSDATE,
    div_id                  NUMBER(2),
    email                   VARCHAR2(20)  UNIQUE,
    cit_id                  NUMBER(3),
    CONSTRAINT emp_pk PRIMARY KEY (id),
    CONSTRAINT emp_div_fk FOREIGN KEY(div_id)
                          REFERENCES divisions(id),
    CONSTRAINT emp_email_un UNIQUE(email)
  );
```

Object identifiers

Oracle cares about case sensitivity for quoted identifiers:

employees

"employees“

"Employees"

"EMPLOYEES"

Can reference different objects in the same schema!

employees

EMPLOYEES

"EMPLOYEES"

Reference the same object!

Coding Conventions

- SQL instructions are not case sensitive
- Careful with reserved words!

- Good practice for tables and column names is to prefix column names with a label from the table name:

```
CREATE TABLE divisions(
                    div_id NUMBER(4) NOT NULL,
                    div_name VARCHAR2(50) NOT NULL
);
```

Alter table

Modify the name:

```
ALTER TABLE employees RENAME TO newemployees;
```

Modify the layout:

```
ALTER TABLE employees ADD (salary NUMBER(7));
ALTER TABLE employees RENAME COLUMN div_id TO
   dep_id;
ALTER TABLE employees DROP (hiredate);
```

But also:

- Add/modify/drop constraints
- Enable/Disable constraints
- Modify more advanced properties…

Drop table

Remove the table from the user schema (recoverable in Oracle10g):

```
DROP TABLE employees;
```

->effects: the table is removed (or moved in the recycle bin) with all its data, and dependencies (indexes, etc…)

Remove the table from the database entirely (Oracle10g):

```
DROP TABLE employees PURGE;
```

Remove a table with referential constraints:

```
DROP TABLE employees CASCADE CONSTRAINTS;
```

Insert data in a table

Data are added in a table as new rows

Insertion following the table defined layout:

```
INSERT INTO employees VALUES(1369,'SMITH',
  TO_DATE('17-DEC-1980','DD-MON-YYYY`),20,NULL);
```

Insertion using a DEFAULT value

```
INSERT INTO employees VALUES   (1369, 'SMITH',
  DEFAULT,20,'john.smith@cern.ch');
```

Insertion specifying the column list:

```
INSERT INTO employees (id, name, div_id, email )
  VALUES(1369, 'SMITH', 20, 'john.smith@cern.ch');
```

Insertion in a table outside the current working schema:

```
INSERT INTO <schemaname>.employees …
```

Retrieve the table data (I)

How to query data from one or more tables

Retrieve all data available:

```
SELECT * FROM employees;
```

Full table id is needed outside the working schema:

```
SELECT * FROM <schemaname>.employees …
```

Retrieve a subset of the available columns:

```
SELECT id, name FROM employees;
```

Retrieve the distinguished column values:

```
SELECT DISTINCT div_id FROM employees;
```

Retrieve from more tables:

```
SELECT employees.name,visitors.name FROM
employees, visitors;
```

Retrieve the table data (II)

Assign pseudonyms to the columns to retrieve:

```
SELECT name AS emp_name FROM employees;
SELECT id "emp_id", name "emp_name" FROM
  employees;
```

Columns concatenation:

```
SELECT name || email AS name_email FROM
  employees;
SELECT 'employee ' || name || email FROM
  employees;
```

Treatment of NULL values (NVL operator):

```
SELECT NVL(email,'-') FROM employees;
SELECT NVL(salary,0) FROM employees;
```

Aggregating data

- Data can be grouped and some summary values can be computed
- Functions and clauses:
  - AVG, COUNT, MAX, MIN, STDDEV, SUM, VARIANCE
  - group by clause is used to define the grouping parameter
  - having clause can be used to limit the output of the statement

Group functions

Data can be grouped and some summary values can be computed

Retrieve the number of rows:

```
SELECT COUNT(*) FROM employees;
```

Retrieve the number of non-null values for a column:

```
SELECT COUNT(email) FROM employees;
```

Restrict to distinguished values:

```
SELECT COUNT(DISTINCT div_id) FROM employees;
```

Sum/Max/Min/Avg

```
SELECT SUM(salary) FROM employees;
```

Set operators

Combine multiple queries

Union without duplicates:

```
SELECT name, email FROM employees UNION
SELECT name, email FROM visitors;
```

Union with the whole row set:

```
SELECT cit_id FROM employees UNION ALL
SELECT cit_id FROM visitors;
```

Intersect:

```
SELECT  name FROM visitors INTERSECT
SELECT  name FROM former_employees;
```

Minus:

```
SELECT  name FROM visitors MINUS
SELECT  name FROM former_employees;
```

Restricting and sorting data

- Need to restrict and filter the rows of data that are displayed and/or specify the order in which these rows are displayed

- Clauses and Operators:
  - WHERE
  - Comparisons Operators (=, >, < …..)
  - BETWEEN, IN
  - LIKE
  - Logical Operators (AND,OR,NOT)

  - ORDER BY

Restricting data selection (I)

Filter the rows according to specified condition

Simple selections:

```
SELECT * FROM employees WHERE id = 30;
SELECT name FROM employees WHERE NOT div_id =
  2;
SELECT name FROM employees WHERE salary > 0;
SELECT * FROM employees WHERE hiredate <
  TO_DATE('01-01-2000', 'DD-MM-YYYY');
SELECT name FROM employees WHERE email IS
  NULL;
```

More Conditions (AND/OR):

```
SELECT * FROM employees WHERE div_id = 20 AND
  hiredate >  TO_DATE('01-01-2000',
 'DD-MM-YYYY');
```

Restricting data selection (II)

More selection operators

Use of wildcards

```
SELECT * FROM employees WHERE name LIKE 'C%';
```

Ranges

```
SELECT count(*) FROM employees WHERE salary
  BETWEEN 1000 and 2000;
```

Selection from a list

```
SELECT * FROM employees WHERE div_id IN
  (4,9,12);
```

List from an other selection

```
SELECT name FROM divisions WHERE id IN (SELECT
  div_id FROM employees WHERE salary > 2000);
```

Sorting selected data

Set the order of the rows in the result set:

```
SELECT name, div_id, salary FROM employees ORDER
  BY hiredate;
```

Ascending/Descending

```
SELECT name, div_id, salary FROM employees ORDER
  BY hiredate ASC;
SELECT name, div_id, salary FROM employees ORDER
  BY salary DESC, name;
NAME           DIV_ID SALARY
-------------- ------ ---------
Zzz            2      4000
Aaa            1      3000
Bbb            3      3000
```

Aggregating Clauses

Divide rows in a table into smaller groups:

```
SELECT column, group_function(column) FROM table
   [WHERE condition] GROUP BY group_by_expression;
```

Example:

```
SELECT div_id, MIN(salary), MAX (salary) FROM
   employees GROUP BY div_id;
```

- All columns in the SELECT that are not in the group function must be included in the GROUP BY clause
- GROUP BY column does not have to be in the SELECT

Restrict the groups:

```
SELECT div_id, MIN(salary), MAX (salary) FROM
   employees GROUP BY division HAVING MIN(salary)
   < 5000;
```

Update data in a table

Aim: change existing values in a table

With no clause all the rows will be updated:

```
UPDATE employees SET salary=1000;
```

A single result select can be used for update:

```
UPDATE employees SET salary=(SELECT MAX(salary));
```

The previous value can be used for the update:

```
UPDATE employees SET salary=salary+1000;
```

In order to update a specific row(s), a WHERE clause can be provided:

```
UPDATE employees SET salary=5000 WHERE
  name=smith;
UPDATE employees SET salary=5000 WHERE div_id=3;
```

The syntax for the WHERE clause is the same as for the SELECT statements…

Delete data from a table

Aim: remove existing data from a table

With no clause all the rows will be deleted:

```
DELETE FROM employees;
```

In order to delete a specific row(s), a WHERE clause can be provided:

```
DELETE FROM employees WHERE name=smith;
DELETE FROM employees WHERE div_id=3;
```

The syntax for the WHERE clause is the same as for the SELECT statements…

Manipulating data from more tables

In RDBMS data model, to ensure consistency:

Row idenfication -> Primary Key

Constrained relationship with other table row-> Foreign Key

In general, entries for a given table column might be related to other table columns…

JOIN:

Retrieve data from more tables defining a condition for the row association

- Natural usage on foreign key constrained columns

Types of join

Equijoin	Values in the two corresponding columns of the different tables must be equal
Non-Equijoin	The relationship between the columns of the different tables must be other than equal
Outerjoin	It returns also the rows that does not satisfy the join condition
SelfJoin	Joining data in a table to itself

Equijoin

Foreign Key

EMP.NAME	EMP.DIV_ID
KING	10
BLAKE	30
CLARK	10

Primary Key

DIV.ID	DIV.NAME
10	ACCOUNTING
30	SALES
20	OPERATIONS

EMP.NAME	EMP.DIV_ID	DIV.NAME
KING	10	ACCOUNTING
BLAKE	30	SALES
CLARK	10	ACCOUNTING

Outerjoin

Foreign Key

EMP.NAME	EMP.DIV_ID
KING	10
BLAKE	NULL
CLARK	10
MARTIN	20
TURNER	10
JONES	NULL

Primary Key

DIV.ID	DIV.NAME
10	ACCOUNTING
30	SALES
20	OPERATIONS

EMP.NAME	EMP.DIV_ID	DIV.NAME
KING	10	ACCOUNTING
BLAKE	NULL	NULL
CLARK	10	ACCOUNTING
MARTIN	20	OPERATIONS
TURNER	10	ACCOUNTING
JONES	NULL	NULL

Join Examples Syntax

Equijoins:

ANSI syntax:

```
SELECT employees.name, divisions.name FROM employees INNER
   JOIN divisions ON employees.div_id=divisions.id;
```

Oracle:

```
SELECT employees.name, divisions.name FROM employees,
   divisions WHERE employees.div_id=divisions.id;
```

Outerjoins:

ANSI syntax (LEFT,RIGHT,FULL)

```
SELECT employees.name, divisions.name FROM employees
  FULL OUTER JOIN divisions
  ON employees=division.id;
```

Oracle:

```
SELECT employees.name, divisions.name FROM employees,
   divisions WHERE employees.div_id(+)=divisions.id;
```

SQL Functions

Oracle provides a set of SQL functions for manipulation of column and constant values

- Use the functions as much as possible in the *where* clauses instead of making the selection in the host program (it may invalidate the use of an index)

Type	Functions
CHAR	concat, length, lower, upper, trim, substr
NUMBER	trunc, mod, round, logical comparison, arithmetic
DATE	to_date, to_char, -, +, trunc, months_between
…others	to_char, to_number, decode, greatest, least, vsize

Character manipulation Functions

String concatenation:

```
SELECT CONCAT(CONCAT(name, ' email is '), email)
  FROM employees WHERE id = 152;
```

String length:

```
SELECT LENGTH(email) FROM employees WHERE
citizenship = 5;
```

Set the Case (LOWER/UPPER):

```
SELECT CONCAT(LOWER(name),'@cern.ch') FROM
  employees;
```

More operators:

TRIM,LTRIM,RTRIM Remove characters from the string start/end

SUBSTR Extract a specific portion of the string

Numeric functions (I)

SQL Function for numeric types (column value or expression):

ABS(p)

- Returns the absolute value of the column or the expression

CEIL(p)

- Returns the smalles integer greater then or equal to the parameter value

FLOOR(p)

- Returns largest integer equal to or less than the parameter value

MOD(m, n)

- Returns the remainder of *m* divided by *n* (or *m* if *n* is 0)

POWER(p, n)

- Returns *p* raised to the *n*th power

Numeric functions (II)

ROUND(p,n)
- Returns *p* rounded to *n* places to the right of the decimal point (default n=0)

SIGN(p)
- Returns the sign of *p*

SQRT(p)
- Returns the square root of *p*.

TRUNC(m, n)
- Returns *n* truncated to *m* decimal places

POWER(m, n)
- Returns *m* raised to the *n*th power (default n=0)

More Math functions:
ACOS, ASIN, ATAN, ATAN2, COS,
COSH, EXP, LN, LOG, SIN, SINH, TAN, TANH

Date operation

Functions to form or manipulate a Date datatype:

SYSDATE

- Returns the current operating system date and time

NLS_DATE_FORMAT

- Session Parameter for the default Date format model

```
ALTER SESSION SET NLS_DATE_FORMAT = 'yy.mm.dd';
```

TO_DATE(s [,format [,'nlsparams']])

- Converts the character string *s* (CHAR, VARCHAR2) to a value of DATE datatype. *format* is a datetime model format.

ROUND(date,format)

- Returns *date* rounded to the unit specified by the format model *format*

TRUNC(date,format)

- Returns *date* with the time portion of the day truncated to the unit specified by the format model *format*

Other functions:

NEXT_DAY(date,day),LAST_DAY(date)

Other functions

Conversion functions:

TO_CHAR(p,[format])

- Converts *p* to a value of VARCHAR2 datatype
- *p* can be character, numeric, Date datatype
- *format* can be provided for numeric and Date.

TO_NUMBER(expr,[format]))

- Converts *expr* to a value of NUMBER datatype.
- *expr* can be BINARY_FLOAT, BINARY_DOUBLE or CHAR, VARCHAR2 in the format specified by *format*

More useful functions:

DECODE
VSIZE
GREATEST
LEAST

The DUAL table

Table automatically created by Oracle Database in the schema of SYS user.

- Accessible (read-only) to all users.

By selecting from the DUAL table one can:

- Compute constant expressions with functions:

```
SELECT ABS(-15) FROM DUAL;
ABS(-15)
----------
15
```

- Retrieve some Environment parameters:

```
SELECT UID, USER FROM DUAL;
       UID  USER
---------  -------------
       578  GOVI
```

Summary

- What is SQL, for what and how do we use it
- ANSI and Oracle-specific SQL Datatypes
- User's schema
- Basic SQL for :
  - Create, Modify, Delete a table
  - Insert data into a table
  - Select data from one or more tables with/without conditions
  - Update or delete data from a table
- SQL functions
- The Oracle DUAL table

Documentation

- Oracle SQL: The essential reference
  David Kreines, Ken Jacobs
  O'Reilly & Associates; ISBN: 1565926978; (October 2000)

- Mastering Oracle SQL
  Sanjay Mishra, Alan Beaulieu
  O'Reilly & Associates; ISBN: 0596001290; (April 2002)

- http://otn.oracle.com

Questions & Answers