1903. Bases de Datos
TEMA 8. ANEXO
ESTRUCTURA DE ORACLE SQL

MUY ÚTIL para las PRÁCTICAS en SQL

Anexo. Estructura de **Oracle SQL**

Contenidos

- **Cuestiones interesantes de Oracle SQL**
  - Tipos de sentencias
  - Operadores
  - Expresiones
  - Condiciones
  - Funciones

Tipos de sentencias SQL

- Control del sistema
- Control de la sesión
- Definición de datos
- Manipulación de datos
- Control de transacciones

Control del sistema

- **ALTER SYSTEM**…
  - Cambio de las propiedades de una instancia de BD en ejecución
  - Su efecto permanece mientras la BD está montada
  - Algunos ejemplos:

SQL> **ALTER SYSTEM SWITCH LOGFILE;**
SQL> **ALTER SYSTEM CHECKPOINT;**
SQL> **ALTER SYSTEM ENABLE RESTRICTED SESSION;**
SQL> **ALTER SYSTEM SET query_rewrite_enabled=true**
SQL> **ALTER SYSTEM FLUSH SHARED_POOL;**

Control de la sesión

- **ALTER SESSION…**
  - Cambio de las propiedades de una sesión de un usuario individual
  - Algunos ejemplos:

```
SQL> ALTER SESSION SET nls_date_format=
       'YYYY MM DD HH24:MI:SS';
SQL> ALTER SESSION SET nls_language=French;
SQL> ALTER SESSION SET sqltrace=true;
SQL> ALTER SESSION SET query_rewrite_enabled=true;
```

Definición de datos

- **Definición** y gestión de la estructura de bases de datos, esquemas y objetos (LDD)
  - **CREATE**, **ALTER**, **DROP**, **TRUNCATE**
  - Creación, alteración y eliminación de…
    - **bases de datos, tablespaces, esquemas,**
    - **tablas, índices, clusters, vistas, vistas materializadas,**
    - **secuencias, database links, sinónimos,**
    - **procedimientos, funciones, paquetes y disparadores**
  - Creación y manejo de **usuarios**
  - Concesión y revocación de **privilegios**
  - **Análisis** de los datos dentro de tablas o índices
- ***Son en sí mismas una transacción***
  - COMMIT previo y posterior (si tienen éxito)

Manipulación de datos

- Consulta y modificación de datos (LMD)
- Sentencias básicas:
  - **SELECT**, **INSERT**, **UPDATE**, **DELETE**
- Otras sentencias:
  - **MERGE**, **CALL**, **LOCK TABLE**, **EXPLAIN PLAN**, …

Control de transacciones

- Control de cambios hechos en la ejecución de sentencias LMD (INSERT, UPDATE, DELETE)
- Sentencias:
  - **COMMIT**,
    ROLLBACK,
    SAVEPOINT,
    SET TRANSACTION

SQL **Operadores**

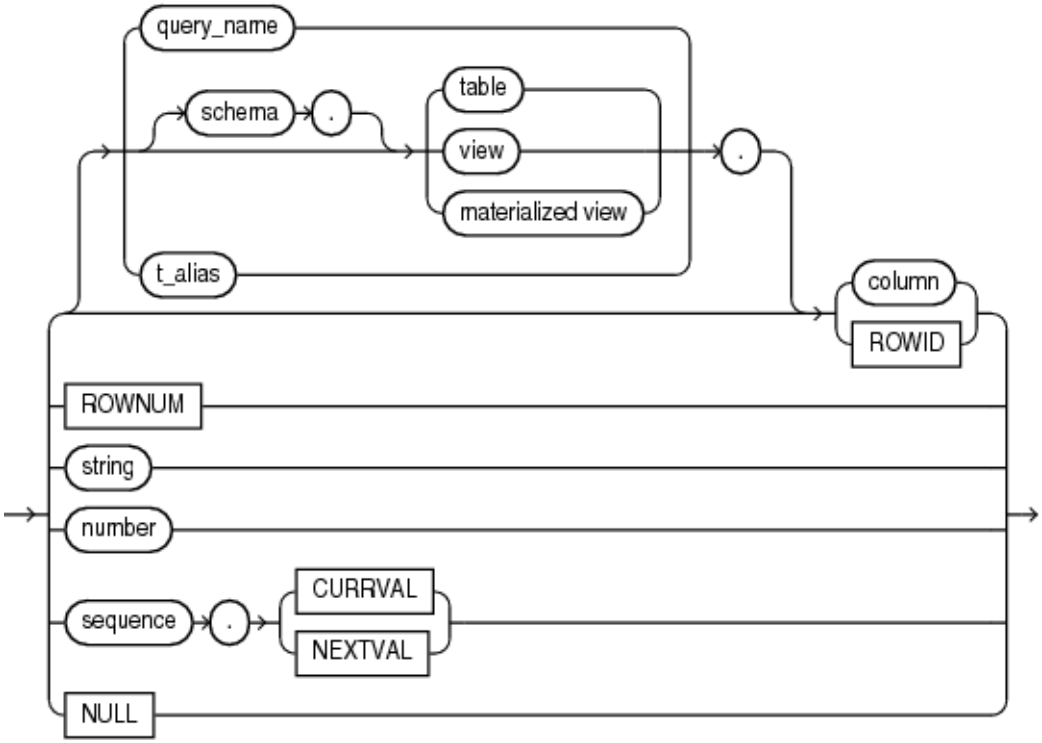
- Aritméticos: +, -, *, /
- Concatenación (CHAR, VARCHAR2): | |
- Operadores de conjuntos:
  - UNION, UNION ALL, INTERSECT, MINUS

SQL **Expresiones**

- Combinación de dos o más **valores, operadores** y **funciones** SQL, que se evalúa a un valor
  - Generalmente, asume el tipo de datos de sus componentes
  - Ejemplos
    - 2*2
      - Se evalúa a 4, y componentes y resultado son de tipo de datos NUMBER
    - TO_CHAR(TRUNC(SYSDATE+9))
      - Añade 9 días a la fecha actual, elimina el tiempo componente de la suma y convierte el resultado a un tipo de datos CHAR
- Se puede usar una expresión en…
  - La lista de selección de una sentencia SELECT
  - Una condición de las cláusulas WHERE y HAVING
  - Las cláusulas ORDER BY, CONNECT BY y START WITH
  - La cláusula VALUES de la sentencia INSERT
  - La cláusula SET de la sentencia UPDATE

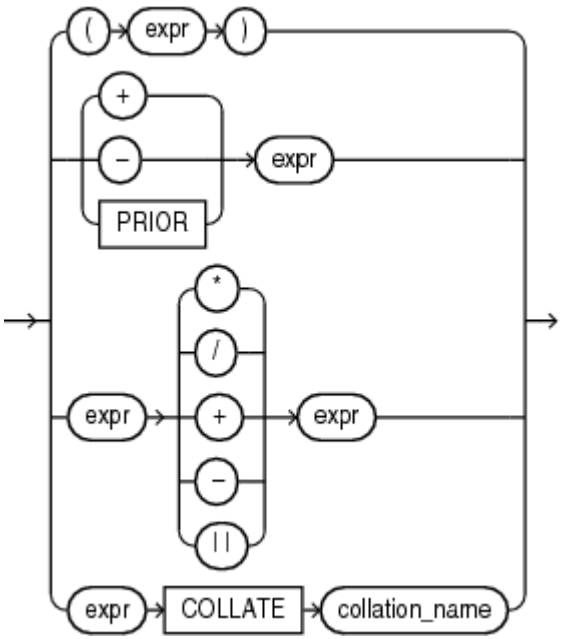
SQL Tipos de Expresiones

- **Expresión simple**
  - Especifica una columna, pseudocolumna, constante, número de secuencia, o nulo
  - Ejemplos
    - employees.last_name
    - 'this is a text string'
    - 10
    - N'this is an NCHAR string‘
    - NULL



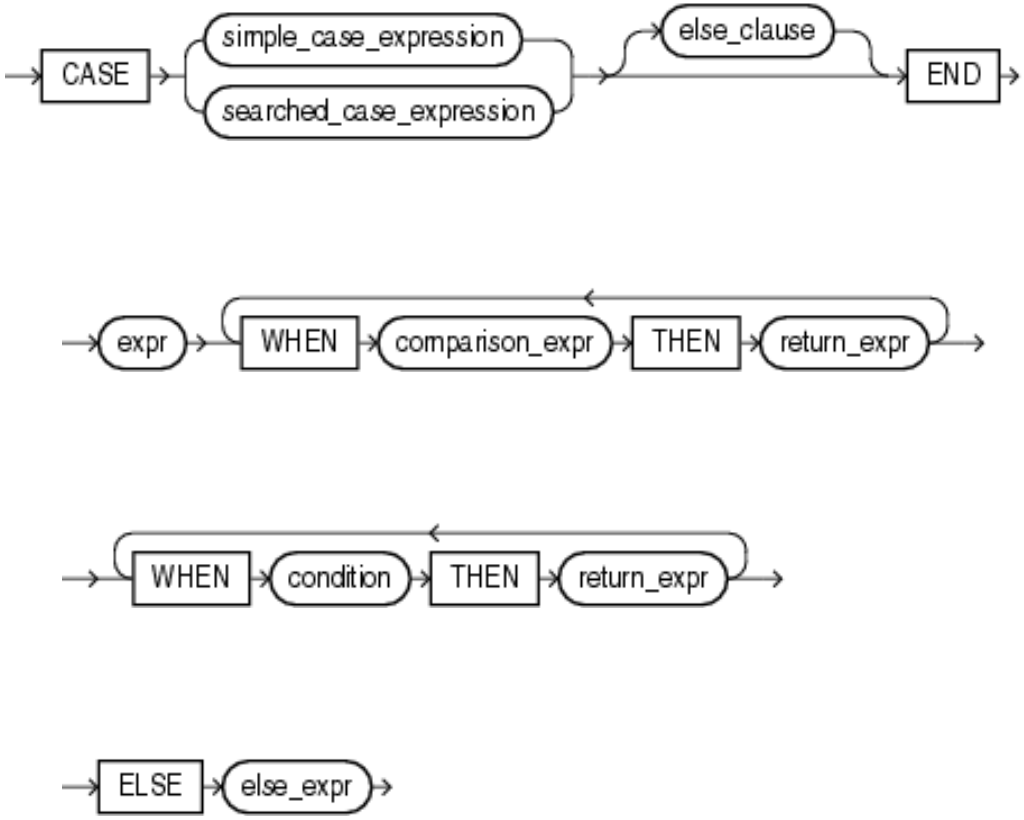
SQL Tipos de Expresiones

- **Expresión compuesta**
  - Combinación de otras expresiones
  - Ejemplos
    - ('CLARK' || 'SMITH')
    - LENGTH('MOOSE') * 57
    - SQRT(144) + 72
    - my_fun(TO_CHAR(sysdate,'DD-MMM-YY'))



SQL Tipos de Expresiones

- **Expresión CASE**
  - Permite usar lógica IF…THEN…ELSE en sentencias SQL sin tener que invocar procedimientos



SQL Tipos de Expresiones

- Expresión CASE

Ejemplo de *Simple CASE*

```
SELECT cust_last_name,
    CASE credit_limit
     WHEN 100 THEN 'Low'
     WHEN 5000 THEN 'High'
    ELSE 'Medium'
    END AS credit
FROM customers
ORDER BY cust_last_name, credit;
```

```
CUST_LAST_NAME CREDIT
-------------- ------
Adams          Medium
Adjani         Medium
Alexander      Medium
Allen          Medium
Altman         High
Anderson       Medium
. . .
```

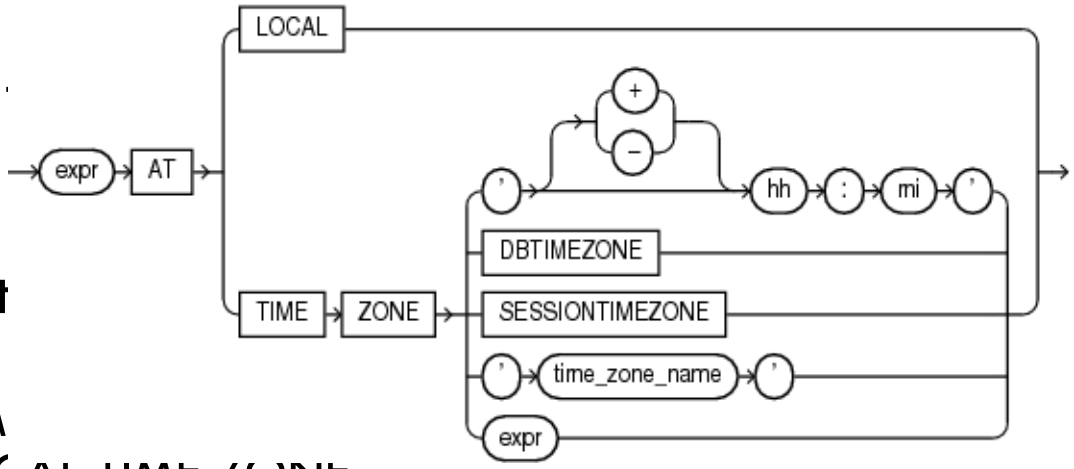
Ejemplo de *Searched CASE*

```
SELECT
  AVG(CASE WHEN e.salary > 2000
        THEN e.salary
      ELSE 2000 END) "Average Salary"
FROM employees e;
```

```
Average Salary
--------------
6461.68224
```

SQL Tipos de Expresiones

- **Expresión Datetime**
  - Devuelve un valor de
    de datos *datetime*
  - La *expr* inicial debe
    devolver un valor de t
    - TIMESTAMP,
      TIMESTAMP WITH TIM
      TIMESTAMP WITH LOCAL TIME ZONE
    - DATE no está soportado



- Ejemplo

```
FROM_TZ(CAST(TO_DATE('1999-12-01 11:00:00',
'YYYY-MM-DD HH:MI:SS') AS TIMESTAMP), 'America/New_York')
```

SQL Tipos de Expresiones

- **Expresión Función**
  - Función **nativa** SQL

    LENGTH('BLAKE')

    ROUND(1234.567*43)

    SYSDATE
  - Función **definida por el usuario**

    circle_area(radius)

    payroll.tax_rate(empno)

    hr.employees.comm_pct@remote(dependents, empno)

    DBMS_LOB.getlength(column_name)

    my_function(a_column)
    - Se permite ***notación posicional, nombrada y mixta***:

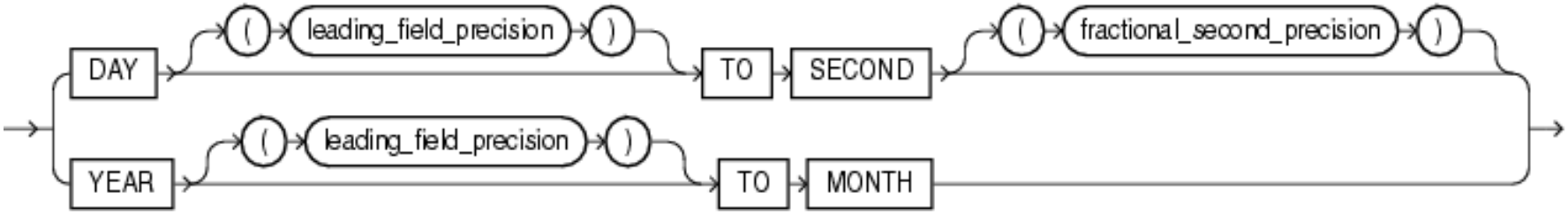
      CALL my_function(3, 4) ...

      CALL my_function(arg2 => 4, arg1 => 3) ...

      CALL my_function(3, arg2 => 4) …

SQL Tipos de Expresiones

- **Expresión Intervalo**
  - Devuelve un valor de tipo INTERVAL YEAR TO MONTH

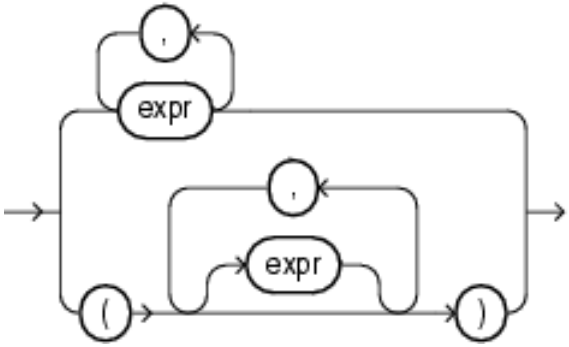


  - En la expresión…
    - Las expresiones *expr1* y *expr2* deben evaluarse a un DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE o TIMESTAMP WITH LOCAL TIME ZONE
    - *Leading_field_precision* y *fractional_second_precision* deben ser enteros entre 0 y 9. Por omisión, valen 2 y 6
  - Ejemplo

```
SELECT (SYSTIMESTAMP - order_date) DAY(9) TO SECOND
FROM orders WHERE order_id = 2458;
```

SQL Tipos de Expresiones

- **Expresión Lista**
  - Es una lista de expresiones, separadas por comas y entre ( )
  - Puede aparecer en comparaciones y condiciones de pertenencia en la cláusula WHERE



    - Llamado a veces constructor de filas de valores *(row value constuctor)*

```
(10, 20, 40)
('SCOTT', 'BLAKE', 'TAYLOR')
SELECT * FROM employees
WHERE (first_name, last_name, email) IN
    (('Guy', 'Himuro','GHIMURO'),('Karen', 'Colmenares', 'KCOLMENA'))
```

  - También pueden aparecer en cláusulas GROUP BY

```
SELECT department_id, MIN(salary) min, MAX(salary) max
FROM employees
GROUP BY (department_id, salary)
```

SQL **Condiciones**

- Aparecen…
  - En la cláusula WHERE de las sentencias DELETE, SELECT y UPDATE
  - En estas cláusulas de la sentencia SELECT:
    - WHERE, START WITH, CONNECT BY, HAVING
- Se evalúa a TRUE o FALSE
- Se puede usar operadores lógicos AND, OR para combinar varias condiciones en una
- Ejemplos

```
NVL(salary, 0) + NVL(salary + (salary*commission_pct, 0) ) > 25000
name = 'SMITH'
employees.department_id = departments.department_id
hire_date > '01-JAN-08'
job_id IN ('SA_MAN', 'SA_REP')
salary BETWEEN 5000 AND 10000
commission_pct IS NULL AND salary = 2100
```

SQL Condiciones

- **Condición de Comparación**
  - Comparan una expresión con otra
  - Devuelve TRUE, FALSE o UNKNOWN

Tipo de condición	Propósito	Ejemplo
=	Test de igualdad	SELECT * FROM employees WHERE salary = 2500 ORDER BY employee_id;
! = ^= <>	Test de desigualdad. *Algunas pueden no estar disponibles para ciertas plataformas*	SELECT * FROM employees WHERE salary != 2500 ORDER BY employee_id;
> < >= <=	Tests mayor-que, menor_que, mayor-o-igual-que, menor-o-igual-que	SELECT * FROM employees WHERE salary <= 2500 ORDER BY employee_id;

SQL Condiciones

condición	Propósito	Ejemplo
ANY **SOME**	Compara un valor con cada uno de los valores en una lista o los devueltos por una subconsulta. Debe ir precedido de =, !=, <, >, <=, >=. Seguido de una expresión o subconsulta que devuelve uno o más valores. Se evalúa a FALSE si la consulta no devuelve filas.	SELECT * FROM employees WHERE salary **= ANY** (SELECT salary FROM employees WHERE department_id = 30) ORDER BY employee_id;
ALL	Compara un valor con todos los valores en una lista o los devueltos por una subconsulta. Debe ir precedido de =, !=, <, >, <=, >=. Seguido de una expresión o subconsulta que devuelve uno o más valores Se evalua a TRUE si la consulta no devuelve filas.	SELECT * FROM employees WHERE salary **>= ALL** (1400, 3000) ORDER BY employee_id;

SQL Condiciones

- **Condición Lógicas**
  - Combina los resultados de dos condiciones para
    - Invertir el resultado de una condición
    - Producir un único resultado basado en ambas condiciones

Condición	Operación	Ejemplo
NOT	Devuelve TRUE si la condición que le sigue es FALSE y viceversa. Si es UNKNOWN, devuelve UNKNOWN	SELECT * FROM employees WHERE NOT (job_id = 'PU_CLERK') ORDER BY employee_id;
AND	Devuelve TRUE si ambas condiciones son TRUE, FALSE si alguna es FALSE y UNKNOWN en otro caso	SELECT * FROM employees WHERE job_id = 'PU_CLERK' **AND** departament_id = 30;
OR	Devuelve TRUE si alguna condición es TRUE, FALSE si ambas son FALSE y UNKNOWN en otro caso	SELECT * FROM employees WHERE job_id = 'PU_CLERK' **OR** departament_id = 10;

SQL Condiciones

- **Condición LIKE (correspondencia de patrones)**
  - Permiten comparaciones por igualdad que involucran correspondencia de patrones
  - Sintaxis: ***char1*** **[NOT] LIKE** ***char2*** **[ESCAPE** ***esc_char*****]**
    - Search value:
      - *char1* es una expresión de caracteres (una columna CHAR, VARCHAR2, NCHAR, NVARCHAR2)
    - Pattern:
      - *char2* es una expresión de caracteres (un literal)
      - Puede contener los comodines _ y %
    - Escape character:
      - *esc_char* es una expresión de caracteres (un literal) que permite incluir '_' y '% en el patrón y que no sean considerados comodines, sino caracteres
  - Distingue mayúsculas de minúsculas

SQL Condiciones

- Condición LIKE
  - Ejemplos

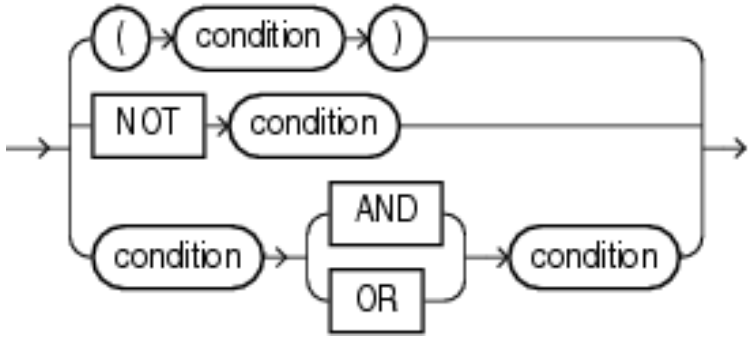
```
last_name LIKE 'Ma%‘
last_name LIKE 'SMITH_‘

SELECT last_name
FROM employees
WHERE last_name LIKE '%A\_B%' ESCAPE '\'
ORDER BY last_name;

SELECT salary
FROM employees
WHERE last_name LIKE 'R%'
ORDER BY salary;
```

SQL Condiciones

- **Condición NULL**
  - IS NULL devuelve TRUE si la expresión es NULL
  - IS NOT NULL devuelve TRUE si la expresión no es NULL
  - Ejemplo

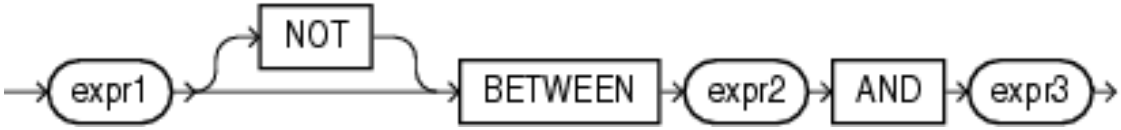
```
SELECT last_name
FROM employees
WHERE commission_pct IS NULL
ORDER BY last_name;
```

- **Condición Compuesta**
  - Combinación de otras condiciones



SQL Condiciones

- **Condición BETWEEN**
  - Determina si el valor de una expresión está en un intervalo definido por otras dos expresiones



  - Las tres expresiones han de ser del mismo tipo de datos (numérico, caracteres o datetime)
    - Si no, Oracle las convierte a un tipo de datos común, pero si no puede hacerlo, devuelve error
  - Ejemplo

```
SELECT *
FROM employees
WHERE salary BETWEEN 2000 AND 3000
ORDER BY employee_id;
```

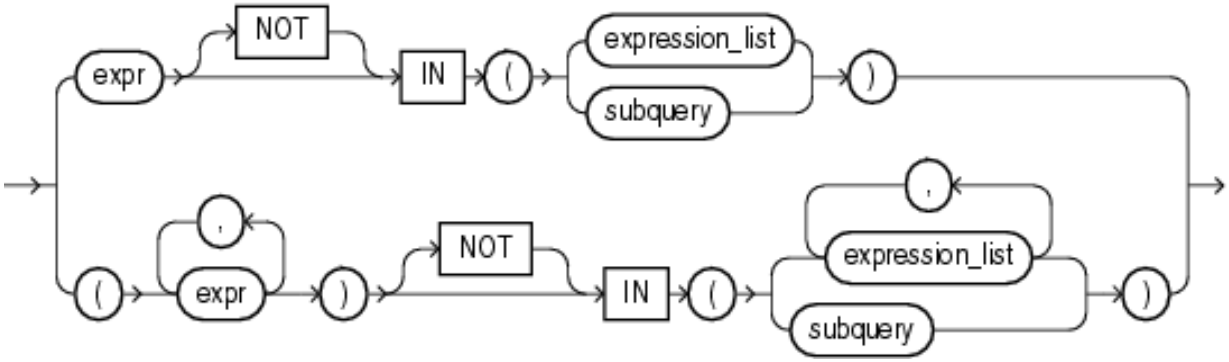
SQL Condiciones

- **Condición EXISTS**
  - Comprueba la existencia de filas en una subconsulta
    - Se evalúa a TRUE si devuelve al menos una fila
  - Ejemplo

```
SELECT department_id
FROM departments d
WHERE EXISTS
   (SELECT * FROM employees e
    WHERE d.department_id = e.department_id)
ORDER BY department_id;
```

SQL Condiciones

- **Condición IN**
  - Comprueba que un valor pertenece a una *lista de valores* o a una *subconsulta*



  - Ejemplos

```
SELECT * FROM employees
   WHERE job_id IN ('PU_CLERK','SH_CLERK');

SELECT * FROM employees
   WHERE salary NOT IN
      (SELECT salary  FROM employees
       WHERE department_id =30);
```

SQL Condiciones

- Condición IN
  - Si algún item en la lista o subconsulta que sigue al NOT IN se evalúa a NULL, entonces todas las condiciones devuelven FALSE o UNKNOWN, y no se devuelven filas:
    - Devuelve una fila por cada fila que cumple la condición:

      SELECT last_name FROM employees
      WHERE departament_id **NOT IN (10,20)**;
    - NO devuelve ninguna fila:

      SELECT last_name FROM employees
      WHERE departament_id **NOT IN (10,20, NULL)**;
      - Porque la cláusula WHERE evalúa:

        departament_id !=10 AND departament_id !=20
        AND departament_id !=NULL
      - La última condición devuelve UNKNOWN, por lo que la expresión completa es FALSE

SQL **Funciones**

- Manipulan elementos de datos y devuelven un resultado
- Pueden operar con cero, uno, dos o más argumentos
  - function(argument, argument, ...)
- Nativas de Oracle y pueden emplearse en sentencias SQL
- Tipos de funciones
  - **single_row_function**
  - **aggregate_function**
  - *analytic_function*
  - *object_reference_function*
  - *model_function*
  - user_defined_function
  - *OLAP_function*
  - *data_cartridge_function*

SQL **Funciones de fila**

- ***Single-row functions***
  - Devuelven una sola fila resultado para cada fila de la tabla o vista consultada
  - Pueden aparecer en…
    - Listas select
    - Cláusula WHERE
    - Cláusulas START WITH y CONNECT BY
    - Cláusulas HAVING
- Las veremos clasificadas en función de los tipos de datos de argumentos y resultado

SQL Funciones de fila **numéricas**

- Aceptan un numérico y devuelven valores numéricos
- La mayoría devuelven valores NUMBER con precisión 38 dígitos decimales.
  - Las funciones COS, COSH, EXP, LN, LOG, SIN, SINH, SQRT, TAN y TANH tienen 36 dígitos decimales de precisión
  - Las funciones ACOS, ASIN, ATAN y ATAN2 tienen 30 dígitos decimales de precisión

ABS	EXP	SIGN
ACOS	FLOOR	SIN
ASIN	LN	SINH
ATAN	LOG	SQRT
ATAN2	MOD	TAN
BITAND	NANVL	TANH
CEIL	POWER	**TRUNC (number)**
COS	REMAINDER	WIDTH_BUCKET
COSH	**ROUND (number)**	

SQL Funciones de fila de **carácter**

- Reciben CHAR o VARCHAR2 y **devuelven valores de tipo CHAR o VARCHAR2**
- La longitud del valor devuelto está limitado por la máxima longitud de su tipo de datos
  - En funciones que devuelven CHAR o VARCHAR2, si la longitud del valor devuelto excede el límite, Oracle Database lo trunca y devuelve el resultado sin mensaje de error
  - Para funciones que devuelven valores CLOB, si la longitud de los valores devueltos excede el límite, Oracle levanta un error y no devuelve datos

CHR	NCHR	REGEXP_REPLACE	SOUNDEX
CONCAT	NLS_INITCAP	REGEXP_SUBSTR	**SUBSTR**
INITCAP	NLS_LOWER	REPLACE	TRANSLATE
LOWER	NLS_UPPER	**RPAD**	TRANSLATE ... USING
LPAD	NLSSORT	RTRIM	TRIM
LTRIM			**UPPER**

SQL Funciones de fila de **carácter**

- Reciben CHAR o VARCHAR2 y **devuelven valores NUMBER**

ASCII
INSTR
LENGTH
REGEXP_COUNT
REGEXP_INSTR

SQL Funciones de fila **datetime**

- Operan sobre datos DATE, timestamp o intervalo

Acepta **DATE** o **timestamp.** Devuelve **DATE**	Acepta **DATE** o **timestamp.** Devuelve **NUMBER**	Sólo acepta **DATE** Devuelve **DATE**
ADD_MONTHS **CURRENT_DATE** **LAST_DAY** **NEW_TIME** **NEXT_DAY**	**MONTHS_BETWEEN**	**ROUND (date)** **TRUNC (date)**
Acepta **DATE, timestamp** e **intervalo.** Devuelve el **mismo** tipo		
CURRENT_DATE CURRENT_TIMESTAMP DBTIMEZONE **EXTRACT (datetime)** FROM_TZ LAST_DAY LOCALTIMESTAMP NEW_TIME	NEXT_DAY NUMTODSINTERVAL NUMTOYMINTERVAL ORA_DST_AFFECTED ORA_DST_CONVERT ORA_DST_ERROR SESSIONTIMEZONE SYS_EXTRACT_UTC	**SYSDATE** SYSTIMESTAMP **TO_CHAR (datetime)** TO_DSINTERVAL TO_TIMESTAMP TO_TIMESTAMP_TZ TO_YMINTERVAL TZ_OFFSET

SQL Funciones de fila para **comparación general**

- Determinan el valor más grande o más pequeño en un conjunto de valores

GREATEST
LEAST

```
SELECT GREATEST(1, '3.925', '2.4') "Greatest"
FROM DUAL;
Greatest
--------
3.925

SELECT LEAST('HARRY', 'HARRIOT', 'HAROLD') "Least"
FROM DUAL;
Least
--------
HAROLD
```

SQL Funciones de fila de **conversión**

- Convierten un valor de un tipo de datos a otro tipo de datos

ASCIISTR	SCN_TO_TIMESTAMP	TO_NCHAR (datetime)
BIN_TO_NUM	TIMESTAMP_TO_SCN	TO_NCHAR (number)
CAST	TO_BINARY_DOUBLE	TO_NCLOB
CHARTOROWID	TO_BINARY_FLOAT	**TO_NUMBER**
COMPOSE	TO_BLOB	TO_SINGLE_BYTE
CONVERT	**TO_CHAR (character)**	TO_TIMESTAMP
DECOMPOSE	**TO_CHAR (datetime)**	TO_TIMESTAMP_TZ
HEXTORAW	**TO_CHAR (number)**	TO_YMINTERVAL
NUMTODSINTERVAL	TO_CLOB	TREAT
NUMTOYMINTERVAL	**TO_DATE**	UNISTR
RAWTOHEX	TO_DSINTERVAL	
RAWTONHEX	TO_LOB	
ROWIDTOCHAR	TO_MULTI_BYTE	
ROWIDTONCHAR	TO_NCHAR (character)	

SQL Funciones de fila para **codificación** y **NULL-related**

- Funciones de **codificación y decodificación**
  - Permiten inspeccionar y decodificar datos en la base de datos

DECODE
DUMP
ORA_HASH
STANDARD_HASH
VSIZE

- Funciones **relacionadas con NULL**
  - Facilitan el manejo del NULL

COALESCE
LNNVL
NANVL
NULLIF
NVL
NVL2

SQL Funciones de fila para **identificación y entorno**

- Proporcionan información sobre la instancia y la sesión

CON_DBID_TO_ID
CON_GUID_TO_ID
CON_NAME_TO_ID
CON_UID_TO_ID
ORA_INVOKING_USER
ORA_INVOKING_USERID
SYS_CONTEXT
SYS_GUID
SYS_TYPEID
UID
USER
USERENV

SQL Funciones de fila. OTRAS…

- *Large Object Functions*
- *Collection Functions*
- *Hierarchical Functions*
- *Data Mining Functions*
- *XML Functions*
- *JSON Functions*

SQL **Funciones de agregados**

- ***Aggregate function***
  - Devuelve una sola fila resultado, con base en un grupo de filas (y no un resultado para cada fila)
  - Puede aparecer en una *lista select*
    - Si la SELECT incluye cláusula GROUP BY, la función se aplica a cada grupo de filas y devuelve un valor por grupo
    - Si se omite GROUP BY, se aplica a la tabla completa y obtiene un solo valor
  - También en la cláusula HAVING
    - Permite descartar del resultado grupos completos de filas que no cumplen una condición basada en este tipo de funciones
  - Y en la cláusula ORDER BY

SQL Funciones de agregados

- Lista de todas las funciones de agregados

APPROX_COUNT_DISTINCT	LAST	STATS_MW_TEST
AVG	LISTAGG	STATS_ONE_WAY_ANOVA
COLLECT	**MAX**	STATS_T_TEST_*
CORR	MEDIAN	STATS_WSR_TEST
CORR_*	**MIN**	STDDEV
COUNT	PERCENT_RANK	STDDEV_POP
COVAR_POP	PERCENTILE_CONT	STDDEV_SAMP
COVAR_SAMP	PERCENTILE_DISC	**SUM**
CUME_DIST	RANK	SYS_OP_ZONE_ID
DENSE_RANK	STATS_BINOMIAL_TEST	SYS_XMLAGG
FIRST	STATS_CROSSTAB	VAR_POP
GROUP_ID	STATS_F_TEST	VAR_SAMP
GROUPING	STATS_KS_TEST	VARIANCE
GROUPING_ID	STATS_MODE	XMLAGG

SQL Funciones de agregados

- Muchas **funciones con un solo argumento** admiten…
  - **DISTINCT** y **UNIQUE** (sinónimos)
    - La función sólo considera valores distintos (no las repeticiones)
    - Ejemplo: la media DISTINCT de 1, 1, 1, 3 es 2
  - **ALL**
    - Opción por omisión
    - La función considera todos los valores, duplicados incluidos
    - Ejemplo: la media ALL de 1, 1, 1, 3 es 1.5
- Otras cuestiones
  - Todas ignoran el NULL en su argumento salvo COUNT(*), GROUPING y GROUPING_ID
  - COUNT y REGR_COUNT devuelven un número o cero, nunca NULL
  - Si el conjunto de datos no contiene filas, o contiene sólo filas con nulos en los argumentos de la función, ésta devuelve NULL
  - Se pueden anidar: MAX(COUNT(*)), AVG(MAX(salary))