UNITÀ LESSICALI

- Simboli, identificatori, costanti, commenti
- Simboli: +,-,*,/,=,<,>,”:”,%,..,@,’,”,:,**,<>,!=,~=...
- Identificatori: lettere, numeri, $, _, # ($,_,#: uno solo, in mezzo), oppure "caratteri ascii"(fino a 30)
- Costanti:
  - numeriche: interi e reali
  - caratteri e stringhe:
    - `'a', 'abc', 'url="http', 'l''amico', q'!l'amico!'`
    - Ma non: `'l'amico', 'et&agrave;'`
  - bool: TRUE, FALSE, NULL
- Commenti:

```
select *     -- commento su singola linea
from         /* commenti su più
                linee                */
```

UN DETTAGLIO CRUCIALE SULLE STRINGHE: attenti all'&

- Per il compilatore, **`&abc`** è una metavariabile
- Quindi, non scrivete mai:
  - **`print('proc?par1=aa&par2=bb');`**
    - Immettere un valore per par2:
    - vecchio 11: print('proc?par1=aa&par2=bb');
    - nuovo 11: print('proc?par1=aa=bb');
  - **`print('sar&agrave; fatto');`**
- Scrivete invece:
  - **`print('proc?par1=aa&' || 'par2=bb');`**
  - **`print('sar&' || 'agrave; fatto');`**

TIPI PL-SQL: TIPI NUMERICi

- BINARY_INTEGER (sottotipi: NATURAL, NATURALN, POSITIVE, POSITIVEN, sinonimo: PLS_INTEGER)
- NUMBER: floating point; sottotipi: Double precision, Float, Real
- NUMBER(cifre) o NUMBER(cifre:=38,posvirgola:=0)]: virgola fissa; sottotipi: Dec, Decimal, Numeric, Integer, Int, Smallint
  - Ex: Number(8), Number(8,4)

TIPI PL-SQL: STRINGHE

- CHAR[(max length:=1)] dimensione fissa (max<256 per le colonne, 32767 per le variabili); sub: character
- LONG: dim variabile (<2MB per le colonne, 32760 per le variabili)
- RAW(max length): non interpretati; max colonna: 255; variabile: 32767
- LONG RAW: long+raw
- UROWID: memorizza il valore della pseudocolonna rowid
- VARCHAR2(max length); max col<4000, var<32767; dimensione variabile; sub: string, varchar
- NCHAR / NVARCHAR2: stringhe unicode
- CHAR – VARCHAR2: attenti ai blank nei confronti

TIPI PL-SQL: DATE e BOOLEANI

- Boolean: contiene *true*, *false* e *null*. Solo variabili, non nella basi di dati
- Date: contengono anche ore, minuti e secondi

SUBTYPES

- Servono a dare un altro nome ad un tipo:
  - DECLARE SUBTYPE Accumulator IS NUMBER;
- Possono aggiungere vincoli:
  - DECLARE
    - temp NUMBER(4);
    - SUBTYPE InteroQuattro IS temp%TYPE;
    - SUBTYPE Intero IS NUMBER(4);
    - SUBTYPE InteroTre IS NUMBER(3) NOT NULL;

CONVERSIONI

- TO_CHAR: data, numero a stringa
- TO_DATE: stringa, numero a data:
  - to_date('11-03-1988','dd-mm-yyyy')
- TO_NUMBER: stringa a numero
- stringa e raw-rowid: ROWTOHEX, HEXTORAW, CHARTOROWID, ROWIDTOCHAR
- Conversioni implicite: ogni conversione immaginabile; non usarle

DICHIARAZIONI DI VARIABILI

- Possono avere come inizializzatore un'espressione complessa, o non averlo (NULL):
  - nome tipo;
  - nome tipo [not null] := expr;
    (oppure: nome tipo [not null] DEFAULT expr;)
  - nome CONSTANT tipo [not null] := expr;
- Un tipo si può specificare: variabile%TYPE, colonna%TYPE (non riceve in questo caso il NOT NULL)
- tabella%ROWTYPE, cursore%ROWTYPE: tipo record
  - selezione e scrittura come var.field
  - assegnamento totale solo tra due record il cui tipo è tratto dalla stessa tabella o cursore, o usando SELECT * INTO reg FROM tab, se reg ha tipo tab%ROWTYPE

NOMI

- Classi di nomi:
  - Semplici: emp
  - Qualificati: luigi.emp
  - Remoti: emp@rome
  - Qualificati e remoti: luigi.emp@rome
- Non è permessa ambiguità tra gli identificatori dichiarati nello stesso livello; quelli dichiarati internamente coprono quelli esterni, ma non i nomi di colonna
- Negli identificatori maiuscole e minuscole sono uguali

Assegnamenti

- ide := expr
- select { col1,..,coln | * } into { var1,..,varn | rcdvar} from ... where
  - Fallisce se la select non trova ennuple (NO_DATA_FOUND) o ne trova più di una (TOO_MANY_ROWS)
  - per ogni colonna, ci deve essere una variabile con tipo compatibile

ESPRESSIONI

- Operatori:
  - ** (esp.), NOT ; +, -; *, / ; +, -, || (concatenazione);
  - =, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN
  - AND; OR
- AND/OR: logica a tre valori non stretta (null or true = true
- stringa LIKE pattern:
  - _: un carattere; %: 0 o più caratteri; case sensitive
- a BETWEEN 10 AND 20
- a IN (3,5,8): attenzione ai null (semantica bizzarra)
- NULL = NULL dà NULL, non TRUE. if a then b else c end if esegue c se a è null.
- Una stringa vuota vale NULL, quindi X = Y dà sempre null se Y è una stringa vuota

FUNZIONI BUILT-IN

- Numeriche: trigonometriche, logaritmiche, trunc-round-floor-ceil...
- Char: char, concat, lower, lpad, ltrim, replace, substr, translate, upper; ascii, instr, length
- Date: add_months, last_day, months_between, next_day, sysdate
- Conversione
- NVL(e1,e2): ritorna e1 se non è null, altrimenti e2
- user, uid
- Funzioni di gruppo:
  - accettano distinct o (default) all
  - ignorano i null, tranne count(*)
  - avg, count(*), max, min, stddev, sum, variance

FLUSSO DEL CONTROLLO: IF THEN ELSE

- IF cond THEN seq of stat;
  [ELSIF cond THEN seq of stat;]*
  [ELSE seq of stat;]
  END IF;
- I rami *then* sono valutati solo se la condizione è *true*; *false* e *null* non valutano il *then*, ma valutano l'*else*.

LOOP EXIT

- [WHILE cond] LOOP seq of stat; END LOOP;
- Per uscire:
  - Se cond è false-null, non viene (più) eseguito
  - EXIT
  - EXIT WHEN cond: uguale a IF cond THEN EXIT END IF
- Loop etichettati:
  - <<ciclo>> LOOP seq of stat; END LOOP ciclo;
  - EXIT ciclo: può uscire da più cicli in un colpo
- FOR ide IN [REVERSE] small..big
  LOOP ....
  - se small>big il loop non è eseguito; si può uscire con EXIT
  - ide è dichiarato implicitamente, è costante, ha il loop come scope

GOTO

- GOTO label può saltare solo verso l’esterno