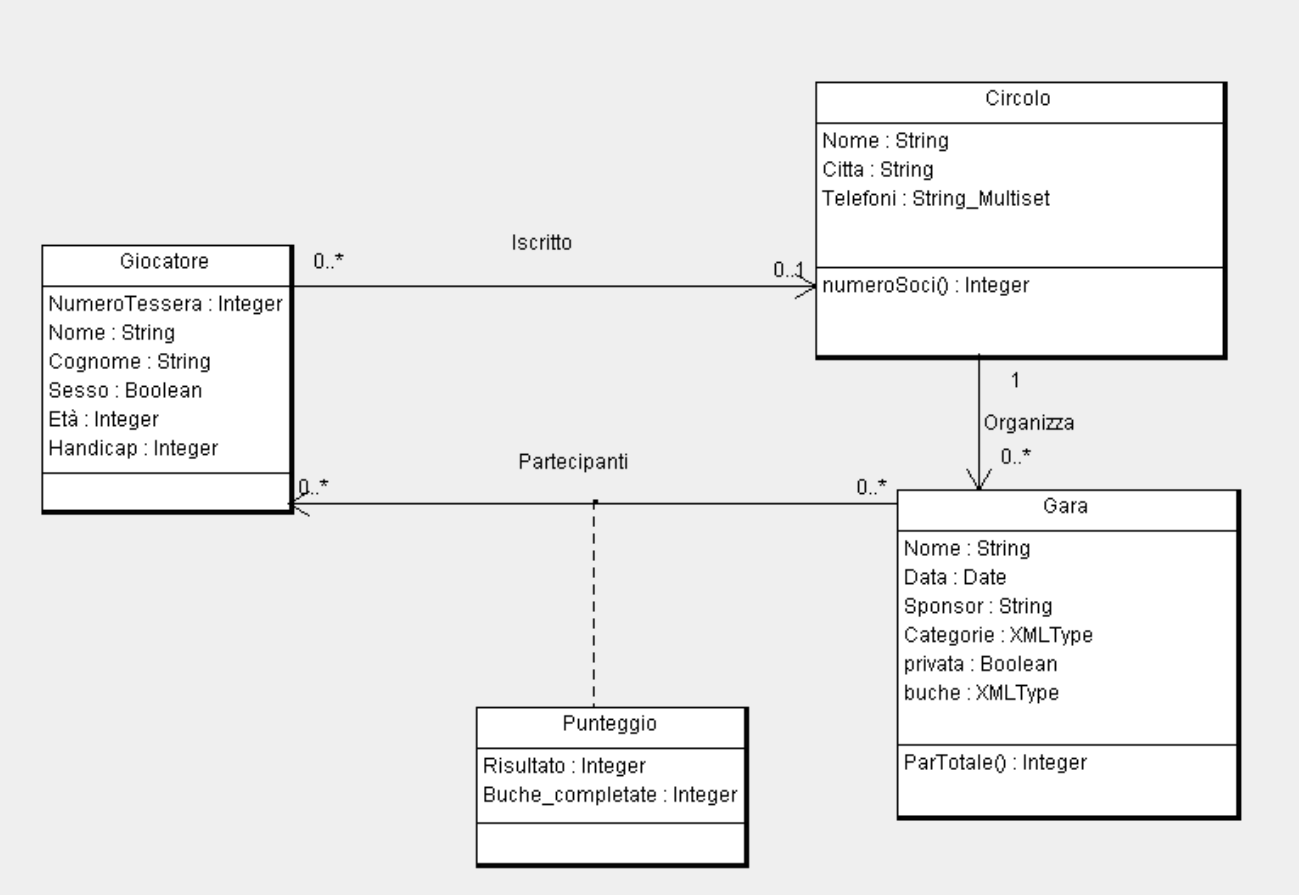
Progetto di modelli dei dati e DBMS di nuova generazione

Università degli Studi di Milano
A.A 2018-2019
F. Polvere – D. Bianda

1 PROGETTAZIONE DELLO SCHEMA

1.1 SCHEMA UML



1.2 ASSUNZIONI E VINCOLI

Dallo schema UML non è possibile esprimere alcuni vincoli che esporremo qui, in linguaggio naturale.

In particolare, parlando dell'entità **giocatore**, abbiamo vincolato il valore di handicap tra 0 e 36, per il sesso del giocatore abbiamo ristretto i valori ai caratteri 'm' e 'f' così come l'età, compresa tra dei valori realistici. La chiave primaria è il numero della tessera. Abbiamo assunto che un giocatore inizialmente possa non appartenere ad un circolo, quindi avere la referenza nulla. Nel momento in cui viene inserito un punteggio viene controllato se la referenza è valida.

Per l'entità **circolo** abbiamo assunto che potesse avere più numeri di telefono, perciò abbiamo modellato l'attributo tramite una Nested Table di stringhe. La chiave primaria è il nome del circolo.

L'entità **gara** ha un attributo di tipo numerico, vincolato ai valori 0 ed 1, che è utilizzato come una variabile booleana per indicare se essa è privata. L'inserimento di una gara è soggetto alla validazione del documento che la rappresenta rispetto allo schema XML. La chiave primaria è il nome della gara.

L'entità **punteggio** non ha vincoli d'integrità a livello di database ma sono implementati tramite una procedura PL/SQL che valida l'aderenza allo schema del XML che rappresenta i punteggi, oltre a controllare che il giocatore indicato nel documento appartiene effettivamente ad un circolo.

Viene posto un vincolo alle stringhe di ogni entità che fa in modo di non avere delle stringhe vuote nel database.

Non vengono utilizzate chiavi esterne in quanto le associazioni tra entità sono basate sul modello ad oggetti.

Abbiamo interpretato diversamente l'XML dei punteggi, assumiamo che un giocatore si è ritirato da una gara se il numero di buche effettuate è minore di nove, ignorando l'elemento withdraw. Quindi l'attributo buche_completate riporta il numero di buche effettuate dal giocatore che è un valore compreso tra 0 e 9.

Inoltre abbiamo modificato l'XML schema della gara in modo da riportare anche le buche con i relativi par.

2 TRADUZIONE NEL MODELLO OBJECT-RELATIONAL DI ORACLE

Classe	Tabella	Tipo	Tabella tipata	Attributi
Punteggio		punteggio_t		giocatore: REF giocatore_t
		punteggi_set: NESTED TABLE punteggio_t		
Gara		gara_t	gare	punteggi: punteggi_set
Giocatore		giocatore_t	giocatori	circolo: REF circolo_t
		gare_set: NESTED TABLE REF gare_t		
		telefoni_set: NESTED TABLE VARCHAR		
Circolo		circolo_t	circoli	gare: gare_set telefoni: telefoni_set

Giocatore è stato implementato come tabella tipata perché alcune query richiedono l'accesso diretto alla tabella ed è necessario referenziarlo nelle istanze di punteggi. Contiene un riferimento ad un circolo per minimizzare la ridondanza dovuta al fatto che più giocatori sono soci dello stesso circolo.

Di conseguenza anche circolo è stato implementato tramite tabella tipata. Un'istanza di circolo contiene due nested table, una per memorizzare i numeri di telefono e l'altra per contenere le referenze alle gare ospitate.

Anche se una singola gara non è condivisa tra i circoli abbiamo scelto di implementarla come tabella tipata, perché alcune query richiedono l'accesso diretto alle tuple.

Ogni gara contiene una nested table di oggetti di tipo punteggio, per questi non è stata necessaria la creazione di una tabella in quanto gli accessi vengono sempre fatti a partire da una gara.

3 IMPLEMENTAZIONE

3.1 INTERROGAZIONI SQL

1) Creare il calendario delle gare di un circolo in un anno. Il calendario deve essere un documento XML.

```
SELECT XMLELEMENT("calendario",
            XMLATTRIBUTES(c.nome AS "circolo"),
            XMLAgg(
                XMLElement("gara",
                    XMLATTRIBUTES(value(g).data AS data, value(g).sponsor AS sponsor),
                    value(g).nome
                    )
            )
        ).getstringval()
FROM circoli c, table(c.gare) g
where c.nome='Golf Club Milano' AND extract(YEAR from value(g).data) = 2009
group by c.nome;
```

Viene effettuato il prodotto cartesiano tra i circoli e le gare organizzate e tramite la clausola where si filtrano le tuple riguardanti il circolo e l'anno di cui si vuole ricavare la classifica.
Per ogni tupla rimanente viene generato, tramite XMLELEMENT, un elemento gara composto dagli attributi data e sponsor e nome della gare come contenuto.
Ogni elemento appartiene ad una tupla diversa quindi è necessario aggregarle con XMLAGG e group by e ordinarle in base alla data
Il risultato ottenuto viene racchiuso in un altro elemento calendario che riporta come attributo il nome del circolo di cui si sta generando la classifica.

Un esempio di documento XML prodotto dall'esecuzione della query è il seguente.

```
<?xml version="1.0" encoding="UTF-8"?>
<calendario circolo="Golf Club Milano">
  <gara DATA="2009-06-10" SPONSOR="Nike">Coppa del mondo</gara>
  <gara DATA="2009-12-25" SPONSOR="Lavazza S.p.A">Coppa del presidente</gara>
</calendario>
```

2) Determinare il circolo a cui è iscritto un dato golfista.

```
SELECT DEREF(circolo).nome from giocatori where numerotessera='MC1001';
```

Dato un numero di tessera che identifica unicamente il giocatore, si ricava la sua tupla contenente la referenza al circolo.
Dereferenziandola è possibile ricavare il nome univoco associato al circolo.

3) Determinare i golfisti che partecipano solo a gare organizzate dal proprio circolo solo per i propri soci.

```
SELECT DISTINCT VALUE(p).giocatore.numeroTessera FROM gare g, TABLE(g.punteggi) p WHERE privata = 1
MINUS
SELECT VALUE(p).giocatore.numeroTessera
FROM gare g, TABLE(g.punteggi) p
WHERE privata = 0
```

La prima select si occupa di ricavare tutti i giocatori che partecipano ad una gara privata, usando distinct per evitare le ripetizioni.
Al risultato di questa query vengono rimossi quelli derivanti dalla seconda select. Questa seleziona tutti i giocatori che hanno partecipato a gare pubbliche.
Il risultato finale della differenza sono i giocatori che partecipano solamente a gare private del proprio circolo.

4) Per ogni gara, determinare i golfisti ritirati. Ordinare la lista in base al numero di buche che sono state completate.

```
SELECT g.nome, XMLElement("ritirati", XMLAgg(
                XMLElement("giocatore",
                 XMLAttributes(s.buche_completate AS "buche_completate"),
                 deref(s.giocatore).nome)
               order by s.buche_completate DESC)
             ).getstringval()  ritirati
from gare g, table(g.punteggi) s
where s.buche_completate<9 --ritirato
group by g.nome
```

La query utilizza le funzioni SQL/XML di publishing per generare il documento XML che contiene, per ogni gara, i giocatori ritirati.
L'utilizzo di XMLAGG permette di aggregare i giocatori appartenenti ad una stessa gara.
La clausola where permette di filtrare i giocatori ritirati, ossia quelli che hanno meno di nove buche completate.
Tramite XMLAGG il risultato viene ordinato in base al numero di buche completate.
Dall'esecuzione della query si ottiene un record per ogni gara, questo conterrà il suo nome ed un attributo di tipo XML rappresentante l'elenco dei giocatori ritirati ordinati.

Un esempio di risultato del documento XML relativa ad un dato torneo è mostrato nella seguente figura.

```
<?xml version="1.0" encoding="UTF-8"?>
<ritirati>
  <giocatore buche_completate="5">Annika</giocatore>
  <giocatore buche_completate="4">Ariya</giocatore>
</ritirati>
```

5) Creare un documento XML contenente la classifica di una gara. Il documento deve contenere i vincitori per ogni categoria prevista per quella gara. E' ammesso che un giocatore possa essere vincitore in più categorie (ad esempio, una signora con 5 di handicap con di più di 40 anni, può essere seconda in prima categoria e vincere il premio Lady e Over 40). Occorre indicare quali giocatori hanno giocato sotto il "par".

```
SELECT XMLQUERY('
<classifica>{
    for $c in $cat/category_list/category
    let $nprizes := $c/@numPrize
    let $giocatori := if($c/@type = "Over") then(
                            $doc//classifica/giocatore[@eta>=$c/@age]
                        )else if($c/@type = "Lady") then(
                            $doc//classifica/giocatore[@sesso="f"]
                        )else(
                            $doc//classifica/giocatore[@handicap>$c/@from and @handicap<$c/@to]
                        )
    return
        <categoria tipo="{$c/@type}" posti="{$nprizes}">{
            (for $g in $giocatori
            let $ord := $g/@punteggio
            order by $ord ascending
            return $g)[position()<=$nprizes]
        }</categoria>
    }</classifica>
    '
    PASSING itab1.doc AS "doc", itab2.categorie AS "cat"
    RETURNING CONTENT) classifica
FROM ( SELECT nome, XMLELEMENT(
                "classifica", XMLAGG(XMLELEMENT(
                    "giocatore",
                    XMLATTRIBUTES(
                        value(p).giocatore.numerotessera AS "numero_tessera",(CASE
                           WHEN(value(p).risultato - value(p).giocatore.handicap) < g.partotale() THEN 'true'
                           ELSE 'false'
                       END) AS "sotto_par",(value(p).risultato - value(p).giocatore.handicap) AS "punteggio", value(p).giocatore
                       .sesso AS "sesso", value(p).giocatore.eta AS "eta",
                       value(p).giocatore.handicap AS "handicap"
                    ), value(p).giocatore.nome
                      || ' '
                      || value(p).giocatore.cognome

                ) ORDER BY giocatore DESC )
            ) doc
      FROM  gare g, TABLE ( g.punteggi ) p
      WHERE nome = 'Coppa del presidente' AND value(p).buche_completate = 9
      GROUP BY nome
   ) itab1, gare itab2
where itab2.nome = itab1.nome
```

La query per generare la classifica è composta da una prima parte che estrae tutti i giocatori di una data gara che non si sono ritirati indipendentemente dalla categoria a cui appartengono. Il risultato è un documento XML che viene successivamente processato tramite XMLQuery che si occupa di suddividere i giocatori in base alla categoria ed ordinarli secondo il punteggio.

Di seguito vengono descritte in modo più approfondito le due parti principali partendo dalla prima.

```
SELECT nome, XMLELEMENT(
                "classifica", XMLAGG(XMLELEMENT(
                    "giocatore",
                    XMLATTRIBUTES(
                        value(p).giocatore.numerotessera AS "numero_tessera",(CASE
                            WHEN(value(p).risultato - value(p).giocatore.handicap) < g.partotale() THEN 'true'
                            ELSE 'false'
                        END) AS "sotto_par",(value(p).risultato - value(p).giocatore.handicap) AS "punteggio", value(p).giocatore
                        .sesso AS "sesso", value(p).giocatore.eta AS "eta",
                        value(p).giocatore.handicap AS "handicap"
                    ), value(p).giocatore.nome
                      || ' '
                      || value(p).giocatore.cognome

                ) ORDER BY giocatore DESC )
            ) doc
        FROM  gare g, TABLE ( g.punteggi ) p
        WHERE nome = 'Coppa del presidente' AND value(p).buche_completate = 9
        GROUP BY nome
```

A partire da una data gara si ricavano tutte le tuple riguardanti i singoli punteggi dei giocatori, rimuovendo quelli che si sono ritirati e hanno quindi un numero di buche minore di nove.

Tramite le funzioni di publishing viene creato un documento XML chiamato classifica che conterrà a sua volta tanti elementi giocatore quanti sono i golfisti.

Ognuno di questi singoli elementi contiene informazioni riguardante il giocatore, in particolare, viene ricavato il punteggio effettivo applicando l'handicap e calcolato se è sotto il par del campo.

Inoltre abbiamo deciso di riportare età, sesso ed handicap in modo da potere poi classificare i giocatori secondo la categoria corretta.

Qui di seguito viene mostrato il risultato intermedio della query

```
<?xml version="1.0" encoding="UTF-8"?>
<classifica>
 <giocatore numero_tessera="MC1001" sotto_par="false" punteggio="40" sesso="m" eta="26" handicap="10">Dustin Johnson</giocatore>
 <giocatore numero_tessera="MC1003" sotto_par="true" punteggio="33" sesso="m" eta="53" handicap="2">Rory McIlroy</giocatore>
 <giocatore numero_tessera="FC1003" sotto_par="false" punteggio="38" sesso="f" eta="43" handicap="8">Stacy Lewis</giocatore>
 <giocatore numero_tessera="MC1004" sotto_par="true" punteggio="31" sesso="m" eta="23" handicap="27">Francesco Molinari</giocatore>
 <giocatore numero_tessera="FC1001" sotto_par="true" punteggio="34" sesso="f" eta="22" handicap="4">Annika Sorenstam</giocatore>
 <giocatore numero_tessera="FC1002" sotto_par="true" punteggio="32" sesso="f" eta="36" handicap="17">Ariya Jutanugarn</giocatore>
 <giocatore numero_tessera="MC1002" sotto_par="false" punteggio="39" sesso="m" eta="42" handicap="20">Sergio Garcia Fernandez</giocatore>
 <giocatore numero_tessera="FC1004" sotto_par="false" punteggio="36" sesso="f" eta="23" handicap="35">Paula Creamer</giocatore>
</classifica>
```

```
XMLQUERY('
<classifica>{
    for $c in $cat/category_list/category
    let $nprizes := $c/@numPrize
    let $giocatori := if($c/@type = "Over") then(
                            $doc//classifica/giocatore[@eta>=$c/@age]
                        )else if($c/@type = "Lady") then(
                            $doc//classifica/giocatore[@sesso="f"]
                        )else(
                            $doc//classifica/giocatore[@handicap>=$c/@from and @handicap<=$c/@to]
                        )
    return
        <categoria tipo="{$c/@type}" posti="{$nprizes}">{
            (for $g in $giocatori
            let $ord := $g/@punteggio
            order by $ord ascending
            return $g)[position()<=$nprizes]
        }</categoria>
    }</classifica>
    '
    PASSING itab1.doc AS "doc", itab2.categorie AS "cat"
    RETURNING CONTENT)
```

La seconda query è composta da codice XQuery a cui vengono passati il documento XML riguardante le categorie della gara e il documento XML appena generato.

Tramite una espressione FLWR si itera su tutte le categorie presenti in quella gara, si ricava ad ogni iterazione il numero di premi e la sequenza di nodi giocatore che appartengono a quella categoria. L'appartenenza è determinata tramite operatori condizionali xpath.

L'espressione FLWR più interna itera invece sulla sequenza filtrata di giocatori ordinandoli in base al punteggi. Di questi poi si mantengono solo i primi n dove n è il numero di premi per quella categoria.

Di seguito viene riportato il risultato finale per una gara.

```
<?xml version="1.0" encoding="UTF-8"?>
<classifica>
 <categoria tipo="First" posti="3">
  <giocatore numero_tessera="MC1003" sotto_par="true" punteggio="33" sesso="m" eta="53" handicap="2">Rory McIlroy</giocatore>
  <giocatore numero_tessera="FC1001" sotto_par="true" punteggio="34" sesso="f" eta="22" handicap="4">Annika Sorenstam</giocatore>
  <giocatore numero_tessera="FC1003" sotto_par="false" punteggio="38" sesso="f" eta="43" handicap="8">Stacy Lewis</giocatore>
 </categoria>
 <categoria tipo="Second" posti="3">
  <giocatore numero_tessera="MC1004" sotto_par="true" punteggio="31" sesso="m" eta="23" handicap="27">Francesco Molinari</giocatore>
  <giocatore numero_tessera="FC1002" sotto_par="true" punteggio="32" sesso="f" eta="36" handicap="17">Ariya Jutanugarn</giocatore>
  <giocatore numero_tessera="FC1004" sotto_par="false" punteggio="36" sesso="f" eta="23" handicap="35">Paula Creamer</giocatore>
 </categoria>
 <categoria tipo="Lady" posti="1">
  <giocatore numero_tessera="FC1002" sotto_par="true" punteggio="32" sesso="f" eta="36" handicap="17">Ariya Jutanugarn</giocatore>
 </categoria>
 <categoria tipo="Over" posti="1">
  <giocatore numero_tessera="MC1003" sotto_par="true" punteggio="33" sesso="m" eta="53" handicap="2">Rory McIlroy</giocatore>
 </categoria>
</classifica>
```