Utilizzo dei dati spaziali di Oracle

Claudio Rocchini
Istituto Geografico Militare

Introduzione

- Adesso che abbiamo un insieme interessante di dati, proviamo a fare qualche interrogazione.
- Per far questo utilizzeremo l'interfaccia SQL DOS.
- Ricordatevi di attivare spool: spool + nome_file
- Se volete, ingrandite la finestra.

Un database normale

- Per prima cosa un database spaziale è anche un database normale.
- Per questo possiamo fare delle query classiche per analizzare gli attributi delle tabelle.
- Ad esempio provate ad eseguire:

```
SELECT lab, count(*)
FROM lap030 GROUP BY lab;
```

- Cosa compare?
- L'attributo lab per le strade del db25 IGM rappresenta la classifica stradale: la query quindi conta il numero di strade per ogni codice di classifica.

Una query di tipo spaziale

- Proviamo adesso a misurare le aree degli edifici
- Per farlo useremo la funzione SDO_GEOM.SDO_AREA
- Questa funzione necessità però della specifica esplicita dei metadati della feature interessata.
- I metadati andranno recuperati nella tabella dei metadati
- Per accorciare il risultato, limiteremo il calcolo ai primi 10 eficici:

```
SELECT aal015.gid,
       SDO_GEOM.SDO_AREA(aal015.geom,
         user_sdo_geom_metadata.diminfo )
  FROM aal015,user_sdo_geom_metadata
  WHERE gid<=10
  AND user_sdo_geom_metadata.table_name='AAL015';
```

- La query risulta un po' complessa: è necessario eseguire una join con la tabella dei metadati per recuperare il campo diminfo.
- Nota che 'AAL015' fra apici va scritto maiuscolo perché in questo caso è una parola e non un nome di tabella, quindi è "case sensitive".

La strada più lunga

- Eseguiamo una query simile alla precedente: in questo caso operiamo sulle strade, e ne calcoliamo la lunghezza con la funzione SDO_GEOM.SDO_LENGTH.
- Non vogliamo stampare però tutte le lunghezza, ma solo sapere qual è il massimo (funzione di aggregazione max):

```
SELECT max( SDO_GEOM.SDO_LENGTH(lap030.geom,
  user_sdo_geom_metadata.diminfo ) )
FROM lap030, user_sdo_geom_metadata
WHERE user_sdo_geom_metadata.table_name='LAP030';
```

Aggregazione Spaziale: Ingombro di una feature class

- Non solo i valori numerici possono essere aggregati ma anche quelli geometrici.
- Esistono una serie di funzioni spaziali aggreganti: il primo esempio è SDO_AGGR_MBR che calcola il massimo ingombro totale di una serie di oggetti spaziali.

```
SELECT SDO_AGGR_MBR(geom)
FROM LAP030;
```

- Oppure delle sole strade di tipo L302;

```
SELECT SDO_AGGR_MBR(geom)
FROM LAP030
WHERE LAB='L302';
```

Altro esempio di aggregazione

- Invece del massimo ingombro, posso costruire il minimo poligono convesso (convex hull) che copre una serie di oggetti selezionati:

```
SELECT
  SDO_AGGR_CONVEXHULL(SDOAGGRTYPE(geom,1))
  FROM paq040 ;
```

- In questo caso il risultato è un poligono.

Esportazioni

- Esistono una serie di funzioni che esportano le geometrie in altri formati.
- Interessante è l'esportazione in GML (qui limitata alle prime 3 strade per brevità).

```
SELECT SDO_UTIL.TO_GMLGEOMETRY(geom)
FROM lap030 WHERE gid<4;
```

- Il GML risultato può essere recuperato dal file di spool di SQL (se lo avete attivato!)

Creazioni di Dati

- Vediamo adesso una procedura di diversa filosofia.
- Utilizzeremo le query spaziali di Oracle per creare nuove geometrie, come se si utilizzasse uno strumento di ArcGis.
- In questo caso costruiremo la buffer zone dei fossi irrigui.
- Per prima cosa si creerà la tabella spaziale
- In seguito la tabella sarà riempita con una query.

Creazione feature buffer

- Creiamo la feature buffer

```
CREATE TABLE buffer
(
   id NUMBER PRIMARY KEY,
   geom SDO_GEOMETRY
);
```

- Definiamone i metadati

```
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME,
   DIMINFO, SRID)
  VALUES ('buffer', 'geom',
    MDSYS.SDO_DIM_ARRAY
      (MDSYS.SDO_DIM_ELEMENT('X', 507000, 512000, 1),
       MDSYS.SDO_DIM_ELEMENT('Y', 5004000, 5010000, 1)
     , MDSYS.SDO_DIM_ELEMENT('Z', 0, 1000000, 1)) ,  3064);
```

- Infine costruiamo l'indice spaziale:

```
CREATE INDEX buffer_sp_idx
ON buffer(geom)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```


Creazione dei dati

- Vediamo adesso una nuova sintassi del comando INSERT: questo può inserire dati a partire dal risultato di una query.
- La query interessata utilizza la funzione SDO_BUFFER, per creare la buffer zone di 20 metri di raggio:

```
INSERT INTO BUFFER
SELECT
  gid,SDO_GEOM.SDO_BUFFER(geom,user_sdo_geom_meta
  data.diminfo,20)
FROM LBH030,user_sdo_geom_metadata
WHERE user_sdo_geom_metadata.table_name='LBH030';
```

- Anche in questo caso è necessario recuperare i metadati con una join su user_sdo_geom_metadata.

Visualizzazione con ArcGis

- Particolare della buffer zone creata, visto con ArcGis
- Per aggiungere una nuova feature alla connesione Interoperability è necessario riaprire i parametri di configurazione ed aggiungere la nuova tabella.
- Tramite questa feature potremmo individuare gli edifici soggetti ad inondazioni.

Esercizio non guidato:

- Seguendo le linee guida dell'esercizio sul buffer, potete provare a realizzare una versione semplificata della rete stradale, tramite la funzione SDO_UTIL.SIMPLIFY
- Questa funzione ha una sintassi molto simile a buffer.
- Bisogna ricordarsi di creare manualmente la feature class contenitore (tabella, metadati, indice spaziale).

Di palo in frasca (e ritorno)

- Adesso dobbiamo fare una serie di query complesse.
- Queste query riguardano i dati spaziali, ma non hanno come risultato dati spaziali, né devono creare altre feature.
- Per questi motivi è comodo ritornare all'interfaccia Web SQL (scusate ma non è colpa mia…)
- Lanciate Start>Oracle>"Vai alla home page del database". Quindi fate login e andate su SQL.

Nota sul "Query Planner" di Oracle

- I database classici esistono da almeno mezzo secolo: in tutto questo tempo sono stati studiati nei minimi dettagli.
- Grazie a questi studi, alla Oracle hanno realizzato il cosiddetto Query Planner: questo aggeggio si legge la query SQL che avete scritto e la "capisce" in profondità; cosa che permette ad Oracle di realizzare la query nel modo più efficiente possibile, applicando dove serve gli indici sui dati.
- I database spaziali sono nati ieri (in realtà sono ancora nel travaglio): Oracle non è ancora in grado di applicare gli indici spaziali, comunque sia scritta la query.
- In molti casi quindi si deve "guidare" Oracle verso la giusta direzione.

Relazioni Spaziali: introduzione

- Veniamo adesso alla parte più complessa e più utile delle query spaziali.
- Si tratta di tutte quelle query che mettono in relazione spaziale due feature fra di loro, oppure una feature con se stessa.
- Oracle spatial ha una serie di funzioni che controllano le relazioni spaziali. Una di queste è SDO_TOUCH.
- Queste funzioni ritornano la stringa 'TRUE' se il controllo è andato a buon fine, 'FALSE' altrimenti.
- Vedremo che il loro utilizzo banale è semplice, mentre un'utilizzo ottimizzato richiede qualche accortezza.

La prima join spaziale

- Proviamo adesso ad eseguire la seguente query, che ritorna l'elenco dei codici di ponte che toccano una risaia:

```
SELECT PAQ040.gid
FROM PAQ040, ABH135
WHERE
  SDO_TOUCH(ABH135.geom,PAQ040.geom) = 'TRUE';
```

- Notate che alla funzione SDO_TOUCH va passata la coppia di colonne geometriche da controllare. Inoltre il risultato va confrontato con la parola 'TRUE'
- Essendo una query su più tabelle, è obbligatorio specificare le colonne nella forma tabella.colonna (sia geom che gid sono presenti in entrambe le tabelle).

Risultati Doppi e velocità

- I più accorti avranno notato che alcuni numeri sono doppi: questi sono i ponti che toccano più di una risaia alla volta.
- Letta in italiano la query precedente suona una cosa del tipo: *seleziona i codici ponte per ogni ponte ed ogni risaia per cui il ponte tocca la risaia*. Ed Oracle ha fatto proprio questa cosa.
- Per ovviare a questo inconveniente potremmo aggiungere la parola DISTINCT dopo SELECT, per cancellare i risultati doppi.
- Un'altra cosa da notare è l'estreme lentezza dell'esecuzione: questo perché Oracle non riesce a capire che deve usare gli indici spaziali.

Una soluzione (1)

- In realtà l'errore logico è nostro: noi abbiamo chiesto: "*seleziona i codici ponte per ogni ponte ed ogni risaia per cui il ponte tocca la risaia".*
- In realtà volevamo chiedere: "*seleziona dei ponti per cui esistente una qualsiasi risaia che tocca il ponte*".
- In SQL è possibile tradurre la richiesta precedente, nel seguente modo:

Una soluzione (2)

```
SELECT PAQ040.gid
FROM PAQ040
WHERE EXISTS
(SELECT * FROM ABH135
 WHERE SDO_TOUCH(ABH135.geom,PAQ040.geom) = 'TRUE' );
```

- Questa nuova query ha una struttura che non abbiamo mai visto.
- Nel filtro si utilizza il costrutto EXISTS, che va seguito da una nuova query intera, racchiusa fra parentesi (in blu). Questa struttura si chiama sotto-query.
- Questa nuova query non produce risultati doppi, inoltre è molto veloce perché Oracle riesce a capire che deve usare gli indici spaziali.

Non così perspicace

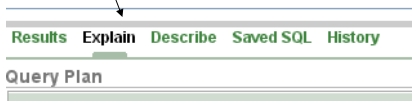
- Oracle non è però veramente così intelligente: in realtà sono io che sono stato molto bravo nel guidarlo.
- La funzione TOUCH è assolutamente simmetrica: vale a dire che se un ponte tocca una risaia, la stessa risaia tocca il ponte.
- Nella query quindi potevo (logicamente) scambiare di posto i due parametri (provate):

```
SELECT PAQ040.gid
FROM PAQ040
WHERE EXISTS
(SELECT * FROM ABH135
 WHERE SDO_TOUCH(PAQ040.geom, ABH135.geom) = 'TRUE' );
```

- In questo caso la query non funziona nemmeno. Addirittura Oracle sostiene che ci siamo dimenticati di fare gli indici spaziali.

Un inciso: le spiegazioni del Query Planner

- Se si sta utilizzando l'interfaccia Web (come in questo caso), è possibile chiedere al Query Planner di Oracle, il suo piano di attacco della query.
- Per farlo digitate la query da analizzare nella finestra e premete il tasto Explain
- Nota: la struttura del query planner è molto complessa ed esce dagli scopi di questo corso. Ne vogliamo dare solo un breve cenno, senza entrare nei dettagli



Analisi della query "brutta"

- Digitate la prima versione della query (quella lenta con i risultati doppi) ed fatevela spiegare.
- Non compaiono indici: "NESTED LOOP" vuol dire una cosa brutta: che Oracle di scandisce le tabelle un numero di volte al quadrato (es. 1000x1000 edifici = un milione di scansioni)

Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes
SELECT STATEMENT			62	4	291	11.160
NESTED LOOPS			62	4	291	11.160
TABLE ACCESS	FULL	PAQ040	30	1	3	4.500
TABLE ACCESS	FULL	ABH135	2	1	10	60

* Unindexed columns are shown in red

Analisi della query "bella"

- Ridigitate ora la query bella (quella con EXISTS) e fatevela spiegare:
- Compare il nome dell'indice spaziale e non ci sono "NESTED LOOP": ok tutto va bene.
- Si capiva anche dalla velocità di esecuzione: ma in casi reali, un nested loop con milioni di oggetti può richiedere un tempo di esecuzione di milioni di anni (letteralmente!).

Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes
SELECT STATEMENT			2	1	5	300
FILTER						
TABLE ACCESS	FULL	PAQ040	30	1	3	4.500
DOMAIN INDEX		ABH135_SP_IDX				

* Unindexed columns are shown in red

Altri operatori

- Ovviamente ci sono moltissimi operatori relazionali.
- Eccone alcuni:
  - **SDO_CONTAINS, SDO_INSIDE**
  - **SDO_COVEREDBY, SDO_COVERS**
  - **SDO_EQUAL**
  - **SDO_OVERLAPS**
  - **SDO_ANYINTERACT**

Un inciso: soprannomi delle tabelle

- Per rendere più leggibile una query, SQL ha la possibilità di dare un soprannome ad una tabella. Questo si realizza scrivendo il soprannome dopo il nome della tabella nel comando FROM. Fatto questo possiamo sostituire il soprannome a quello della tabella. Es:

```
SELECT p.gid
FROM PAQ040 p
WHERE EXISTS
(SELECT * FROM ABH135 r
 WHERE SDO_TOUCH(r.geom,p.geom) = 'TRUE' );
```

- Questo meccanismo è utile per accorciare le tabelle.
- Il soprannome è invece obbligatorio, nel caso in cui facciamo una query con due volte la stessa tabella, come nel caso seguente.

Controllo query con se stessa

- Vogliamo controllare che non ci siano aree di coltura duplicate: possiamo scrivere un controllo di una tabella con se stessa, utilizzando i soprannomi di tabella.

```
SELECT c1.gid
FROM AEA010 c1
WHERE EXISTS
(
  SELECT c2.gid
  FROM AEA010 c2
  WHERE SDO_EQUAL(c2.geom,c1.geom)<>'FALSE'
  AND c1.gid<>c2.gid
);
```

- L'ultimo controllo `c1.gid<>c2.gid` è fondamentale per evitare di controllare una coltura con sé stessa! Provate ad eliminare questo controllo: otterrete l'elenco di tutte le colture, perché ogni coltura è uguale a se stessa.

Un esempio complesso

- Vogliamo trovare il numero di edifici che si trova entro 10 metri da una qualsiasi risaia. Utilizziamo la funzione SDO_WITHIN_DISTANCE. Curiosamente il parametro distanza non è un numero ma una parola nella forma 'distance=x':

```
SELECT count(AAL015.gid)
FROM AAL015
WHERE EXISTS
(SELECT * FROM PAQ040
WHERE
  SDO_WITHIN_DISTANCE(PAQ040.geom,AAL015.geom,'distance
  =10')='TRUE'
);
```

- La query è abbastanza veloce, ma non moltissimo, questo perché Oracle, sebbene il numero di case è molto superiore a quello delle risaie, partendo la query dalle case, preferisce comunque scandire tutte le case e poi le risaie. In questo caso guidare Oracle verso la giusta soluzione non è semplice (continua…)