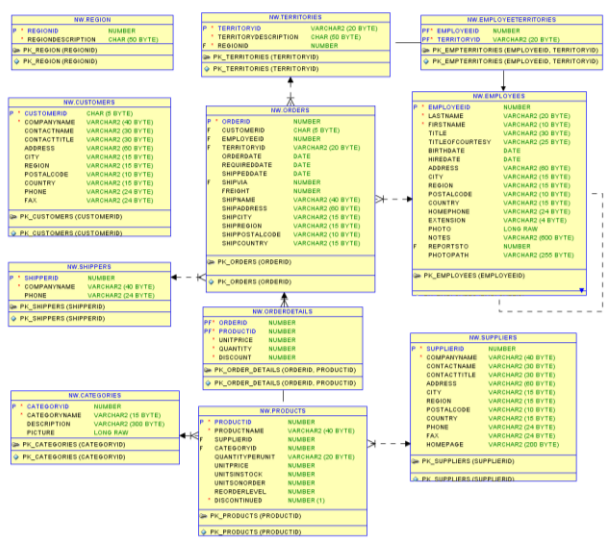
Laboratorio linguaggio SQL

Prof. Alessandra Lumini
Alma Mater Studiorum - Università di Bologna

Schema DB



L'istruzione SELECT

- È l'istruzione che permette di eseguire interrogazioni (*query*) sul DB

```
SELECT  A1,A2,..,Am
FROM    R1,R2,..,Rn
[WHERE   <condizione>]
[GROUP BY <listaAttributi>]
[HAVING <condizione>]
[ORDER BY <listaAttributi>]
```

  - ovvero:
    - SELECT (o TARGET) list (cosa si vuole come risultato)
    - clausola FROM (da dove si prende)
    - clausola WHERE (che condizioni deve soddisfare)
    - clausola GROUP BY (le colonne su cui raggruppare)
    - clausola HAVING (condizione relative ai gruppi)
    - clausola ORDER BY (ordinamento)

Il comando SELECT permette di realizzare le operazioni di selezione, proiezione, join, raggruppamento e ordinamento.

Selezione semplice

- Selezionare tutti i prodotti di categoria 'Beverages'

```
SELECT P.PRODUCTNAME
FROM NW.CATEGORIES C, NW.PRODUCTS P
WHERE C.CATEGORYID=P.CATEGORYID
AND C.CATEGORYNAME='Beverages';

SELECT P.PRODUCTNAME
FROM NW.CATEGORIES C INNER JOIN
NW.PRODUCTS P ON
C.CATEGORYID=P.CATEGORYID
WHERE C.CATEGORYNAME='Beverages';
```

1. Selezionare gli ordini gestiti dall'impiegata Davolio Nancy

Query aggregate

- Contare i prodotti di categoria ‘Beverages’

```
SELECT count(*)
FROM NW.CATEGORIES C, NW.PRODUCTS P
WHERE C.CATEGORYID=P.CATEGORYID
AND C.CATEGORYNAME='Beverages';
```

2. Contare gli ordini gestiti dall’impiegata Davolio Nancy

Query con raggruppamento

- Contare i prodotti di ciascuna categoria

```
SELECT C.CATEGORYNAME, count(*)
FROM NW.CATEGORIES C, NW.PRODUCTS P
WHERE C.CATEGORYID=P.CATEGORYID
group by C.CATEGORYNAME
order by 1;
```

3. Contare gli ordini gestiti da ciascun impiegato

Filtro 'Positivo'

- Selezionare tutti i prodotti presenti in almeno un ordine

```
SELECT PRODUCTNAME
FROM NW.PRODUCTS
where PRODUCTID in (SELECT PRODUCTID FROM
NW.ORDERDETAILS);

SELECT DISTINCT PRODUCTNAME
FROM NW.PRODUCTS P, NW.ORDERDETAILS O
where O.PRODUCTID = O.PRODUCTID;
```

4. Selezionare gli impiegati che hanno gestito almeno un ordine spedito in 'Norway'

Filtro 'Negativo' = Sottrazione

- Selezionare tutti i prodotti che non sono presenti in nessun ordine spedito negli 'USA'

```
SELECT PRODUCTNAME
FROM NW.PRODUCTS
where PRODUCTID NOT IN (SELECT PRODUCTID
FROM NW.ORDERDETAILS OD join NW.ORDERS O
on (OD.ORDERID=O.ORDERID) WHERE
SHIPCOUNTRY='USA' );

--errata
SELECT DISTINCT PRODUCTNAME
FROM NW.PRODUCTS P, NW.ORDERDETAILS OD,
NW.ORDERS O
where OD.PRODUCTID = P.PRODUCTID and
OD.ORDERID=O.ORDERID and
SHIPCOUNTRY<>'USA';
```

5. Selezionare gli impiegati che non hanno gestito ordini spediti in 'Norway'

Filtri sui gruppi

- Selezionare gli impiegati che hanno gestito più di 50 ordini

```
SELECT A.LASTNAME, A.FIRSTNAME, count(*) as NumeroOrdini
FROM NW.ORDERS B, NW.EMPLOYEES A
WHERE A.EMPLOYEEID = B.EMPLOYEEID
group by A.EMPLOYEEID, A.LASTNAME, A.FIRSTNAME
having count(*)> 50
order by 3 desc;
```

6. Selezionare i prodotti che sono stati venduti a più di 10 clienti diversi

9

Filtro 'Universale' = Divisione

- Selezionare gli impiegati che hanno gestito ordini per tutti i paesi

```
--non ci sono paesi in cui non sia stato spedito un ordine gestito da quell'impiegato
SELECT A.LASTNAME, A.FIRSTNAME
FROM NW.EMPLOYEES A
WHERE NOT EXISTS
  (SELECT * FROM NW.ORDERS O WHERE NOT EXISTS
   (SELECT * FROM NW.ORDERS O1 WHERE A.EMPLOYEEID = O1.EMPLOYEEID AND
O.SHIPCOUNTRY=O1.SHIPCOUNTRY));

--numero paesi serviti= numero paesi presenti nel DB
SELECT A.LASTNAME, A.FIRSTNAME, count(distinct SHIPCOUNTRY) as PaesiDistinti
FROM NW.ORDERS B, NW.EMPLOYEES A
WHERE A.EMPLOYEEID = B.EMPLOYEEID
group by A.EMPLOYEEID, A.LASTNAME, A.FIRSTNAME
having count(distinct SHIPCOUNTRY)=
            (SELECT count(distinct SHIPCOUNTRY) FROM NW.ORDERS O);
```

7. Selezionare clienti che sono stati serviti da tutti i corrieri

10

Gestione delle date

- Funzioni di conversione Stringa-Data

TO_DATE(stringa **[,** formato**])**

```
TO_DATE('2003/07/09', 'yyyy/mm/dd')     Result: date value of July 9, 2003
TO_DATE('070903', 'MMDDYY')             Result: date value of July 9, 2003
TO_DATE('20020315', 'yyyymmdd')         Result: date value of Mar 15, 2002
```

TO_CHAR(value [, format_mask**])**

```
TO_CHAR(sysdate, 'yyyy/mm/dd')          Result: '2003/07/09'
TO_CHAR(sysdate, 'Month DD, YYYY')      Result: 'July 09, 2003'
TO_CHAR(sysdate, YYYY')                 Result: '2003'
```

- Selezionare gli ordini dal 1/1/1998 al 1/2/1998

```
SELECT ORDERID,ORDERDATE
FROM NW.ORDERS
WHERE ORDERDATE
BETWEEN TO_DATE ('01/01/1998', 'dd/mm/yyyy')
AND TO_DATE ('1998/02/01', 'yyyy/mm/dd');
```

8. Selezionare gli impiegati che hanno più di 60 anni

Gestione delle date

- Funzione di EXTRACT

```
EXTRACT (
{ YEAR | MONTH | DAY}
FROM date_value )
```

```
EXTRACT(YEAR FROM DATE '2003-08-22')    Result: 2003
EXTRACT(MONTH FROM DATE '2003-08-22')   Result: 8
EXTRACT(DAY FROM DATE '2003-08-22')     Result: 22
```

- Ordinare gli impiegati per anno di nascita (dal più giovane)

```
SELECT EMPLOYEEID, LASTNAME, FIRSTNAME, EXTRACT( YEAR FROM BIRTHDATE)
FROM NW.EMPLOYEES
ORDER BY 4 DESC;
```

```
SELECT EMPLOYEEID, LASTNAME, FIRSTNAME, to_char(BIRTHDATE, 'YYYY')
FROM NW.EMPLOYEES
ORDER BY 4 DESC;
```

9. Selezionare gli ordini per cui l'anno di spedizione è diverso da quello di ordine

Self-join

- Selezionare le coppie di impiegati della stessa città (senza duplicati)

```
SELECT e1.EMPLOYEEID,e2.EMPLOYEEID,e1.CITY
FROM NW.EMPLOYEES e1,NW.EMPLOYEES e2
where e1.EMPLOYEEID<e2.EMPLOYEEID and e1.CITY=e2.CITY;
```

- Selezionare gli impiegati che hanno un superiore che abita nella stessa città

```
SELECT e1.EMPLOYEEID as Impiegato ,e2.EMPLOYEEID as Superiore, e1.CITY
FROM NW.EMPLOYEES e1,NW.EMPLOYEES e2
where e1.REPORTSTO =e2.EMPLOYEEID and e1.CITY=e2.CITY;
```

10.1 Selezionare le coppie di clienti della stessa nazione (senza duplicati)

10.2 Selezionare gli impiegati che hanno un superiore della stessa età (anno di nascita)

13

Outer-join

- Selezionare gli impiegati e il relativo superiore, inclusi gli impiegati che non hanno superiore

```
SELECT e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME, e.REPORTSTO as Sup,
s.LASTNAME as cognomeSup, s.FIRSTNAME as nomeSup
FROM NW.EMPLOYEES e left outer join NW.EMPLOYEES s on (
e.REPORTSTO=s.EMPLOYEEID);
```

11. Ordinare i clienti in base al numero di ordini effettuati nel 1996 (includere anche i clienti che non hanno fatto ordini)

14

Max(Count())

- Selezionare l'impiegato che ha gestito il maggior numero di ordini

```
SELECT e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME,
count(ORDERID) as NumOrdini
FROM NW.EMPLOYEES E join  NW.ORDERS o on
(e.EMPLOYEEID=o.EMPLOYEEID)
group by e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME
having count(ORDERID)>=ALL (SELECT  count(ORDERID) as
NumOrdini
FROM  NW.ORDERS
group by EMPLOYEEID);
```

12.1 Selezionare il cliente che ha effettuato il maggior numero di ordini

12.2 Selezionare il cliente che ha generato il più alto introito nel 1996

Table expressions

- Table expressions = viste create dinamicamente in memoria su cui si possono poi effettuare delle query.

```
WITH <alias_name> AS (subquery_sql_statement)
SELECT <column_name_list> FROM <alias>;
```

- Selezionare l'impiegato che ha gestito il maggior numero di ordini

```
WITH imp1 AS (SELECT LASTNAME, FIRSTNAME, count(ORDERID) as NumOrdini
FROM NW.EMPLOYEES E join  NW.ORDERS o on (e.EMPLOYEEID=o.EMPLOYEEID)
group by e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME
order by 3 desc)

SELECT LASTNAME, FIRSTNAME, NumOrdini
FROM imp1
where NumOrdini>=ALL(SELECT NumOrdini FROM imp1);
```

- Oppure si può formulare la query direttamente nella clausola FROM

```
SELECT <column_name_list> FROM (<query>);
```

ROWNUM

- La funzione ROWNUM restituisce l'ordinale di una riga in una tabella (la prima riga ha ROWNUM 1, ecc.)
- Se la tabella è ordinata si può filtrare su ROWNUM per selezionare il primo valore
- Selezionare l'impiegato che ha gestito il maggior numero di ordini

```
SELECT EMPLOYEEID, LASTNAME, FIRSTNAME, NumOrdini
FROM (SELECT e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME, count(ORDERID) as NumOrdini
FROM NW.EMPLOYEES E join  NW.ORDERS o on (e.EMPLOYEEID=o.EMPLOYEEID)
group by e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME
order by 4 desc)
where rownum<=1;

WITH imp1 AS (SELECT LASTNAME, FIRSTNAME, count(ORDERID) as NumOrdini
FROM NW.EMPLOYEES E join  NW.ORDERS o on (e.EMPLOYEEID=o.EMPLOYEEID)
group by e.EMPLOYEEID, e.LASTNAME, e.FIRSTNAME
order by 4 desc)
SELECT LASTNAME, FIRSTNAME, NumOrdini
FROM imp1
where rownum<=1;
```

17

Select Top K in Oracle

- **TOP K** (MSSql) e **Limit** (MySQL) non esistono in Oracle

```
select a, b from tabella where rownum<K order by b;
--non funziona perchè rownum è valutato prima di ordinare
```

- 2 Soluzioni (con Table expression):

```
select a, b
from (select a, b from tabella order by b)
where rownum<K;

select a, b
from (select a, b, rank() over (order by b) r from tabella)
where r<K;
```

- La funzione **RANK** restituisce l'ordine in un gruppo di valori (vedi anche **DENSE_RANK**)

18

Select Top K

- Selezionare i 10 ordini con valore più alto

```
select ORDERID,  Totale from
            (select ORDERID, sum (UNITPRICE*QUANTITY*(1-DISCOUNT)) as Totale
            from NW.ORDERDETAILS
            group by ORDERID
            order by 2 desc)
where rownum<=10;
```

13.1 Selezionare i 5 clienti che hanno effettuato il maggior numero di ordini

13.2 Selezionare i 4 fornitori che forniscono il maggior numero di prodotti

19

CASE Statement

- Nella SELECT LIST:

```
CASE [ expression ]
  WHEN condition_1 THEN result_1
  ...
  WHEN condition_n THEN result_n
  ELSE result
END
```

- Raggruppare i prodotti in fasce in base alle quantità venduta: fascia alta (>1000), fascia bassa (<500), fascia media

```
with PView as (SELECT P.PRODUCTID, sum (o.quantity) as QtaVenduta
FROM NW.ORDERDETAILS O
group by P.PRODUCTID)
select PRODUCID, case
            when QtaVenduta>1000 then 'fascia alta'
            when QtaVenduta<500 then 'fascia bassa'
            else 'fascia media'
            end  fascia, QtaVenduta
            from PView;
```

14.1 Modificare le soglie in (s*2/3), (s*1/3), dove s è la quantità venduta dal prodotto più venduto.

20