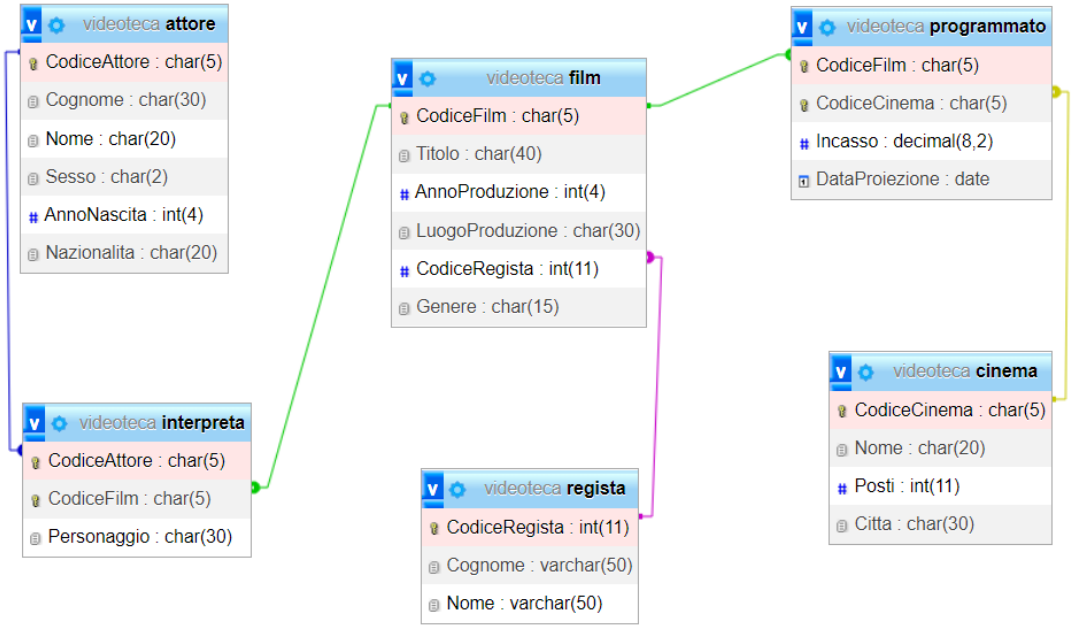
Linguaggio SQL
quinta parte

Unione, intersezione e differenza

Operazioni di unione , intersezione e differenza

Per ottenere tutti i registi e tutti gli attori scriveremo:

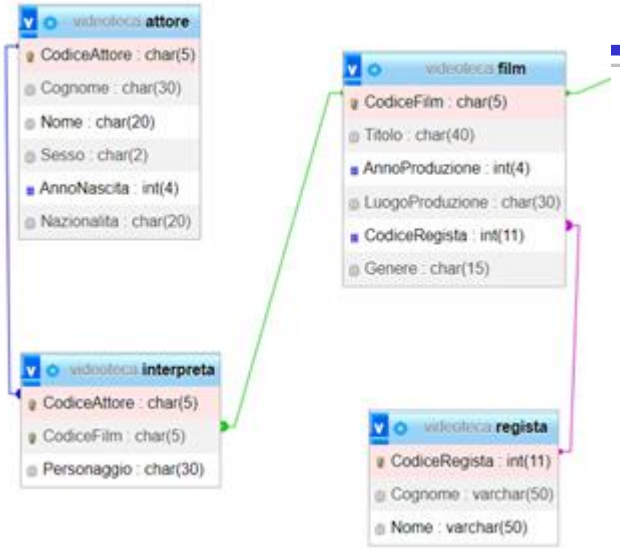
```
(SELECT Cognome, Nome FROM REGISTA)
UNION
(SELECT Cognome, Nome FROM ATTORE);
```



ESEMPI

Considerando il database in figura, creiamo due query:

- **Una query per ottenere l'elenco completo degli attori e dei registri**
- **Una query per ottenere l'elenco dei registri che non non sono attori**



Rifletti

Nei tre casi precedenti le due relazioni risultato parziale della SELECT devono essere compatibili per poter essere utilizzate come operandi degli operatori di intersezione, unione e differenza simmetrica.

Per ottenere i registi che sono stati anche attori scriveremo:

```
(SELECT Cognome, Nome FROM REGISTA)
INTERSECT
(SELECT Cognome, Nome FROM ATTORE);
```

Le parentesi sono obbligatorie.

Per ottenere i registi che non sono mai stati attori scriveremo:

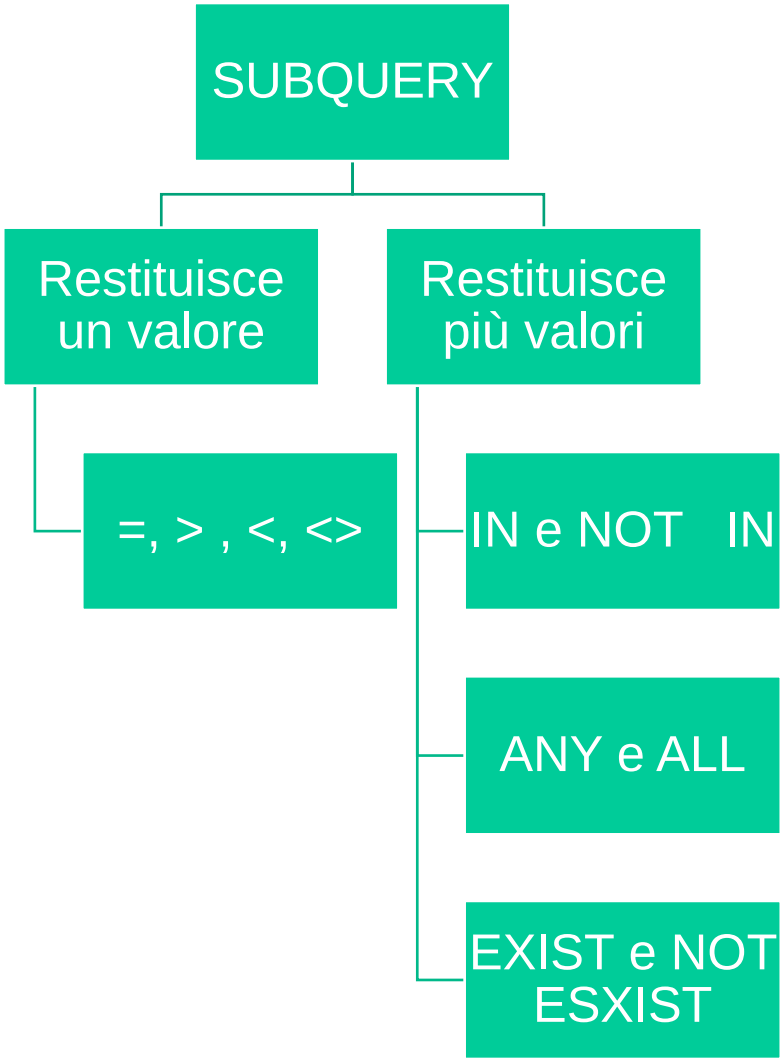
```
(SELECT Cognome, Nome FROM REGISTA)
MINUS
(SELECT Cognome, Nome FROM ATTORE);
```

IN MYSQL SI PUO' UTILIZZARE EXCEPT AL POSTO DI MINUS

Query e subquery annidate

SUBQUERY

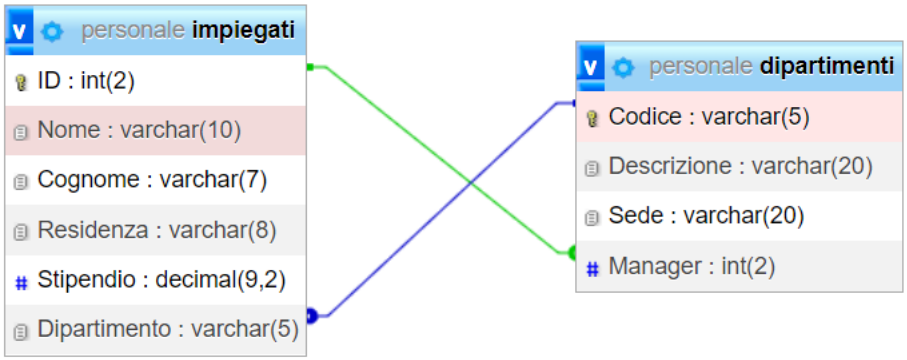
- **E’ possibile creare query complesse, query che contengono al loro interno altre query**
- **In query complesse abbiamo una query esterna (individuata dal primo SELECT) e una o più subquery**
- **La query interna passa i risultati alla query esternache li verifica nella condizione che segue la clausola WHERE**
- **Una subquery può generare:**
  - Un solo valore
  - Una sola riga (ma più colonne)
  - Più righe e più colonne


SUBQUERY
Restituisce un valore
Restituisce più valori
=, > , <, <>
IN e NOT IN
ANY e ALL
EXIST e NOT ESXIST

```
SELECT Attributi
FROM Tabella1
WHERE Attributo { = | > | < | >= | <= | <> }
      ( SELECT { Funzione | Attributo }
      FROM Tabella2
      WHERE Condizione );
```

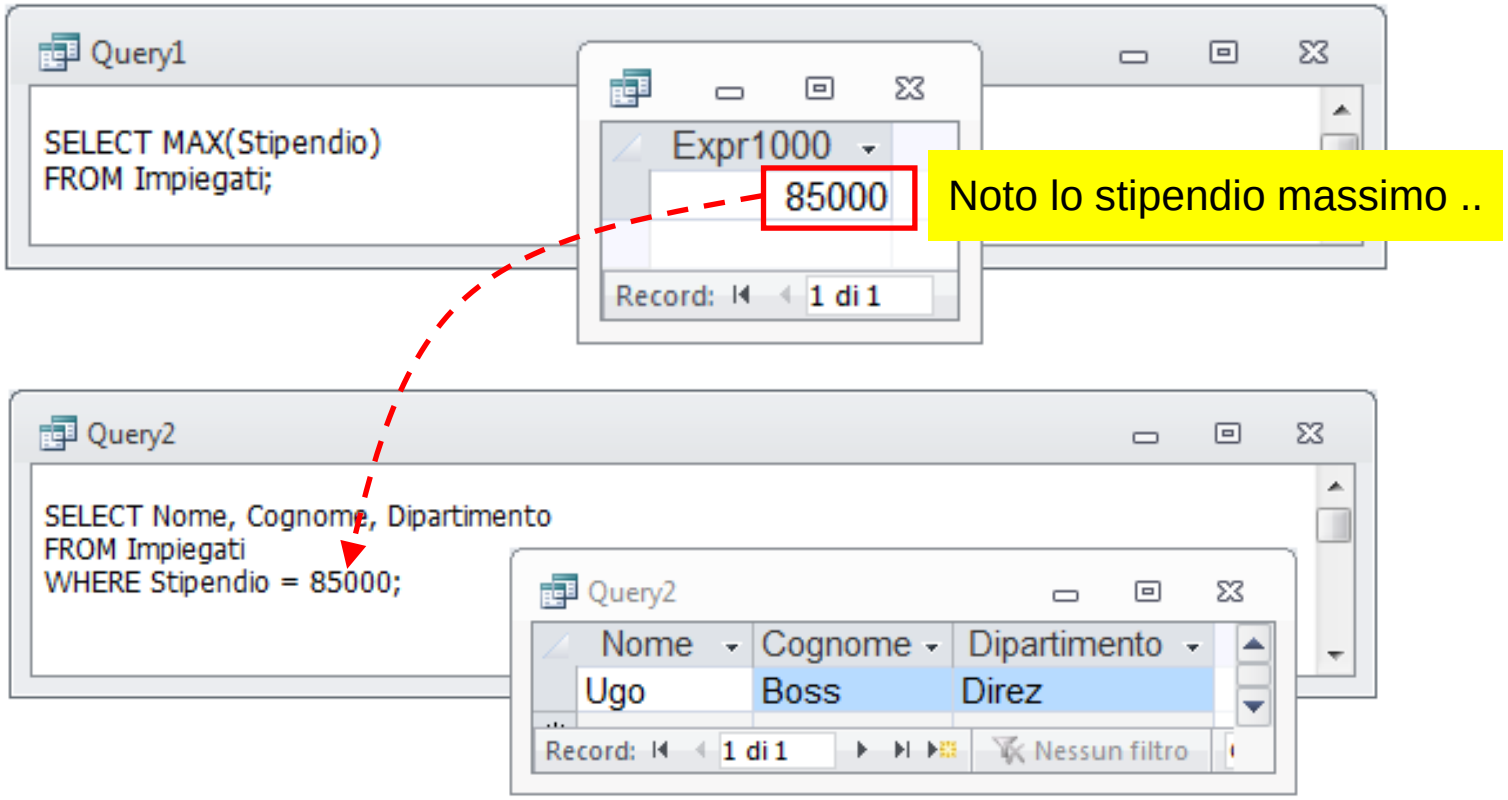
La query interna viene sempre eseguita prima della query esterna.

Il database esempio



SUBQUERY CHE RESTITUISCONO UN VALORE

- *Nome*, *Cognome* e *Dipartimento* del dipendente con lo stipendio massimo



- Perché non cortocircuitare le due interrogazioni?

Query annidate che restituiscono un solo valore e che utilizzano MAX

Una costante in una clausola **WHERE** può essere rimpiazzata con l'interrogazione che genera tale costante

```
SELECT Nome, Cognome, Dipartimento
FROM Impiegati
WHERE Stipendio = ( SELECT MAX(Stipendio)
                    FROM Impiegati );
```

Sottointerrogazione: deve essere racchiusa da parentesi.

L'interrogazione è una sola e c'è un solo ; finale.

Query annidate che restituiscono un solo valore e che utilizzano AVG

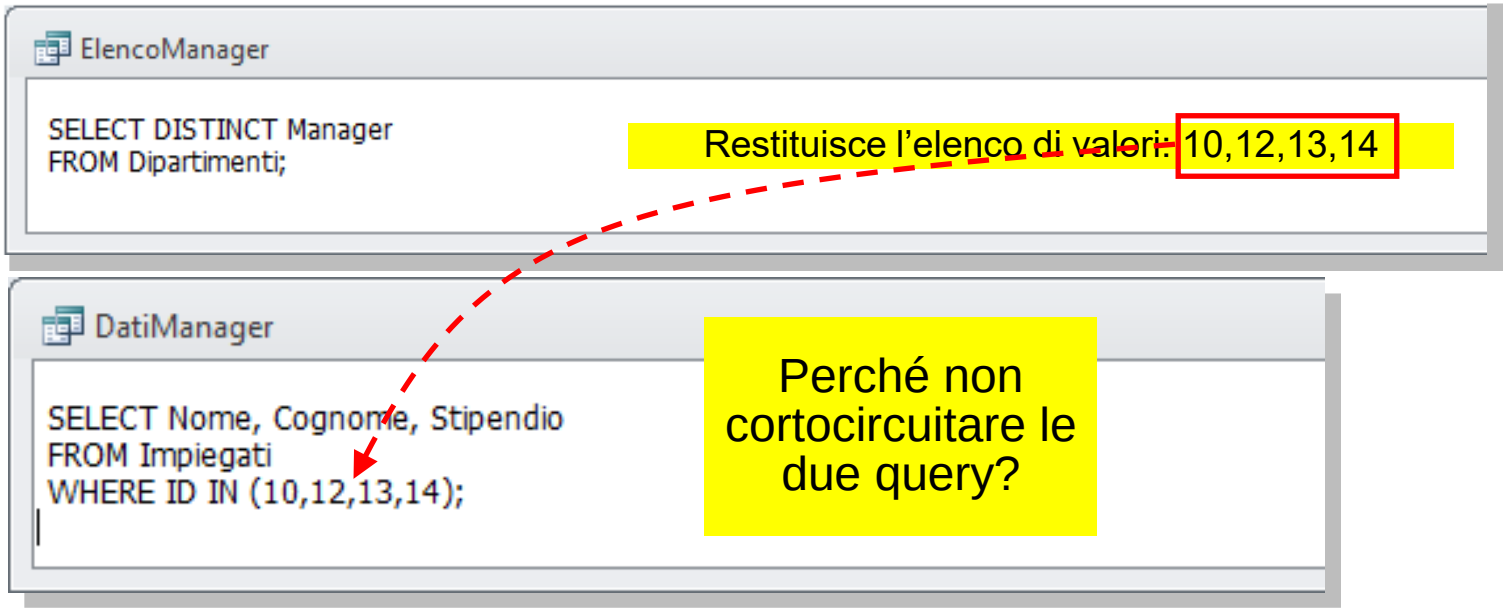
- *Cognome, Nome e Stipendio* degli impiegati con stipendio inferiore alla media degli stipendi degli altri impiegati

Cognome	Nome	Stipendio
Rossi	Mario	32000
Viola	Marco	28300
Mori	Enrico	25000
Magenta	Fabrizio	41000
Gregis	Elisabetta	29000
Bianco	Anita	39000

```
SELECT Cognome, Nome, Stipendio
FROM Impiegati
WHERE Stipendio < ( SELECT AVG(Stipendio)
                    FROM Impiegati );
```

SUBQUERY CHE RESTITUISCONO PIU' VALORI
UTILIZZO DI «IN» E «NOT IN»

- Sottointerrogazioni che restituiscono un **elenco** di valori: *Nome*, *Cognome* e *Stipendio* dei dipendenti che sono manager.

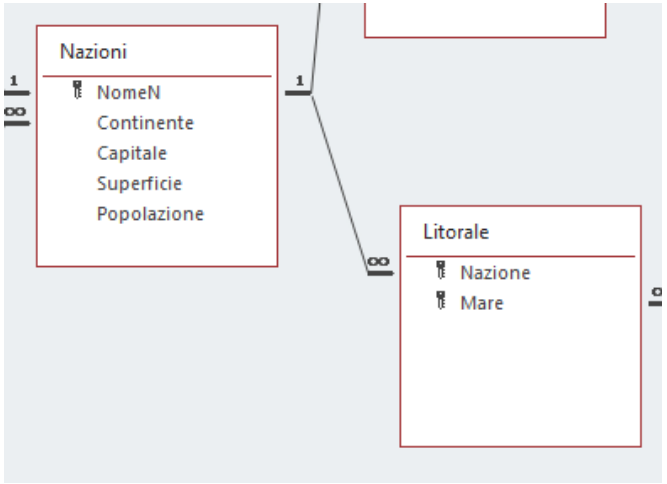


L'elenco di valori che compare nel predicato **IN**, in una clausola **WHERE**, può essere sostituito dalla sottointerrogazione che lo genera.

ESEMPIO DI UTILIZZO DI SUBQUERY CON «IN»

Elenco delle nazioni che sono bagnate da almeno due mari, con i nomi dei mari che le bagnano.

```
SELECT L.Nazione, L.Mare
FROM Litorale AS L
WHERE L.Nazione IN
    ( SELECT Nazione
      FROM Litorale
      GROUP BY Nazione
      HAVING COUNT(*) > 1
    )
ORDER BY L.Nazione;
```



Una query di questo tipo può essere utile in generale per sapere se un valore è presente più di una volta in una colonna e in tal caso in corrispondenza di quali righe.

Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Città (IdCittà, NomeC, Nazione, Popolazione)
Mari (NomeM, Oceano)
Fiumi (NomeF, Lunghezza)
Laghi (NomeL, Superficie)
Litorale (Nazione, Mare)
Attraversamento (Nazione, Fiume)

SUBQUERY CHE RESTITUISCONO PIU' VALORI
UTILIZZO DI «ANY» E «ALL»

Si possono usare **sottoquery che producono un elenco di valori** inserendo nella condizione, dopo l'operatore relazionale di confronto, uno dei predicati **ANY** o **ALL**.

```
SELECT Attributi
FROM Tabella1
WHERE Attributo { = | > | < | >= | <= | <> } { ANY | ALL }
    ( SELECT Attributo
    FROM Tabella2
    WHERE Condizione );
```

ESEMPIO DI UTILIZZO DI ANY

I predicati ANY e ALL permettono di effettuare un confronto tra un attributo e un insieme di valori:

- **ANY** dà come risultato vero se il confronto è vero per almeno uno dei valori dell'elenco, falso se la sottoquery restituisce un insieme vuoto o se il confronto è falso per tutti i valori dell'elenco;

ESEMPIO 3.39

Nome delle città della Francia con popolazione superiore a quella di almeno una città della Germania.

```
SELECT NomeC
FROM Citta
WHERE Nazione = 'Francia' AND Popolazione > ANY
   ( SELECT Popolazione
     FROM Citta
     WHERE Nazione = 'Germania'
);
```

Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Città (IdCittà, NomeC, Nazione, Popolazione)
Mari (NomeM, Oceano)
Fiumi (NomeF, Lunghezza)
Laghi (NomeL, Superficie)
Litorale (Nazione, Mare)
Attraversamento (Nazione, Fiume)

ESEMPIO DI UTILIZZO DI ALL

- **ALL** dà come risultato vero se il confronto è vero per tutti i valori dell'elenco, falso se il confronto è falso per almeno uno dei valori dell'elenco.

ESEMPIO 3.40

Nome delle città della Francia con popolazione superiore a quella di tutte le città della Germania.

```
SELECT NomeC
FROM Citta
WHERE Nazione = 'Francia' AND Popolazione > ALL
   ( SELECT Popolazione
     FROM Citta
     WHERE Nazione = 'Germania'
);
```

Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Città (IdCittà, NomeC, Nazione, Popolazione)
Mari (NomeM, Oceano)
Fiumi (NomeF, Lunghezza)
Laghi (NomeL, Superficie)
Litorale (Nazione, Mare)
Attraversamento (Nazione, Fiume)

ANY: ESEMPIO DEL LIBRO PAG 619

Per rispondere alla domanda: "Quali sono i dipendenti che hanno lo stipendio superiore *ad almeno uno* degli stipendi medi delle nazioni europee?" scriveremo:

```
SELECT Nome, Cognome
FROM DIPENDENTE
WHERE StipendioNetto >ANY (SELECT DISTINCT StipendioMedio
                           FROM STIPENDIO
                           WHERE Continente = "Europa");
```

la condizione indica di estrarre i dipendenti il cui stipendio netto sia superiore ad almeno uno degli elementi dell'insieme che è il risultato della query interna

Restituisce l'elenco senza ripetizioni degli stipendi medi europei

ALL: ESEMPIO DEL LIBRO PAG 620

Per rispondere, invece, alla domanda: "Quali sono i dipendenti che hanno lo stipendio superiore *a tutti* gli stipendi medi delle nazioni europee?" scriveremo:

```
SELECT Nome, Cognome
FROM DIPENDENTE
WHERE StipendioNetto > ALL (SELECT DISTINCT StipendioMedio
                            FROM STIPENDIO
                            WHERE Continente = "Europa");
```

dipendenti il cui stipendio netto sia superiore a tutti gli elementi dell'insieme che è il risultato della query interna

Elenco senza ripetizioni degli stipendi medi europei

QUERY CORRELATE

QUERY CORRELATE

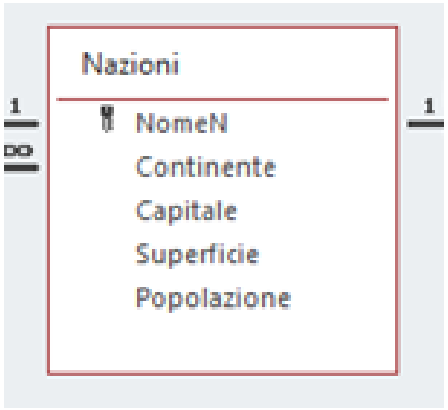
Si dicono query correlate le query in cui la sottoquery agisce su una tabella presente nella query esterna. Se la sottoquery indica nella clausola FROM la tabella presente nella query esterna, risulta necessario utilizzare un alias.

```
SELECT Attributi
FROM Tabella1
WHERE Attributo { = | > | < | <= | <= | <> } [ ANY | ALL ]
    ( SELECT Attributo
    FROM Tabella2
    WHERE Tabella2.Attributo { = | > | < | <= | <= | <> } Tabella1.Attributo );
```

ESEMPIO

Nome delle nazioni che hanno superficie superiore alla superficie media delle nazioni del continente a cui appartengono.

```
SELECT NomeN
FROM Nazioni AS N1
WHERE Superficie >
    ( SELECT AVG(Superficie)
      FROM Nazioni AS N2
      WHERE N2.Continente = N1.Continente);
```



SUBQUERY CHE RESTITUISCONO PIU' VALORI «EXIST» E «NOT EXIST»

Il significato è il seguente:

- `EXISTS`: la condizione della clausola `WHERE` è vera se la subquery produce una tabella non vuota;
- `NOT EXISTS`: la condizione della clausola `WHERE` è vera se il risultato della subquery è una tabella vuota, senza alcuna riga.

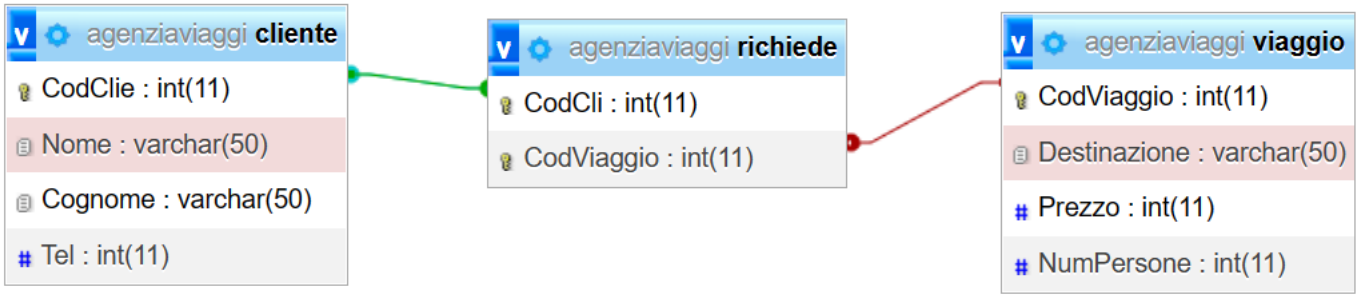
L'operatore logico **EXISTS** dà come risultato vero se la sottoquery restituisce almeno un valore, falso se non restituisce nessun valore (insieme vuoto).

```
SELECT Attributi
FROM Tabella1
WHERE [ NOT ] EXISTS
    ( SELECT *
    FROM Tabella2
    WHERE Tabella2.Attributo { = | > | < | <= | <= | <> } Tabella1.Attributo );
```

L'operatore EXISTS è utilizzabile in modo significativo solo nelle query correlate. **Nota**

Gli attributi specificati nella sottoquery usata col predicato EXISTS sono irrilevanti, quindi di solito si usa l'asterisco. **Nota**

EXIST: ESEMPIO DEL LIBRO pag. 621

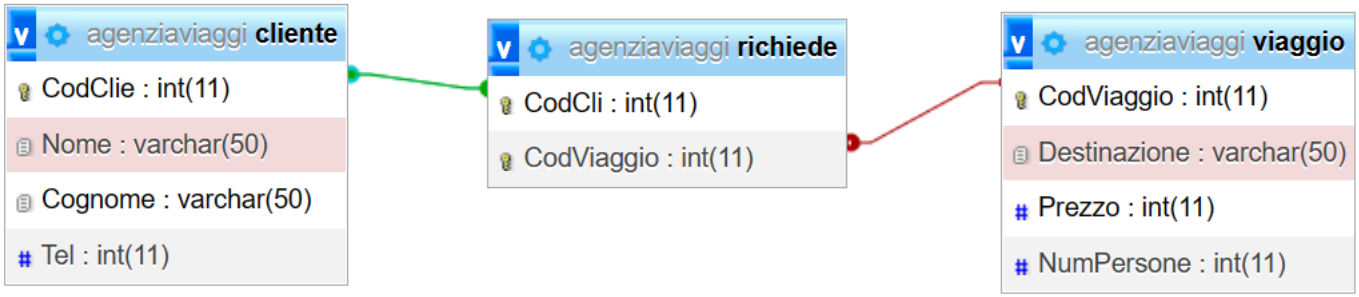


Quali sono i clienti che hanno richiesto di viaggiare?

```
SELECT *
FROM CLIENTE C
WHERE EXISTS (SELECT * FROM RICHIEDE H WHERE C.CodCli = H.CodCli);
```

Da notare il differente significato dell'impiego dell'asterisco "*": nella query principale serve per visualizzare tutti gli attributi di CLIENTE, mentre nella subquery non ha importanza quali siano gli attributi contenuti nella subquery, ciò che conta è la cardinalità della tabella derivata, non il suo specifico contenuto.

NOT EXIST: ESEMPIO DEL LIBRO PAG. 622



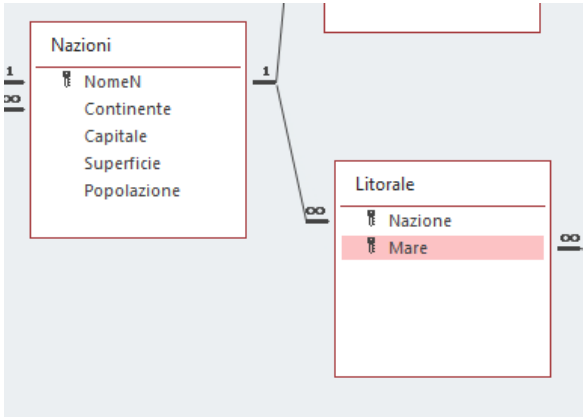
Quali sono i clienti che NON hanno richiesto di viaggiare?

```
SELECT *
FROM cliente C
WHERE NON EXIST (SELECT * FROM RICHIEDE H
                 WHERE C.CodCli = H.CodCli)
```

ESEMPIO

Nome e continente delle nazioni che non sono bagnate da nessun mare.

```
SELECT NomeN, NomeCon
    FROM Nazioni JOIN Continenti
    ON Continente = Sigla
    WHERE NOT EXISTS
    ( SELECT *
      FROM Litorale
      WHERE Nazione = NomeN
    );
```



Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Città (IdCittà, NomeC, Nazione, Popolazione)
Mari (NomeM, Oceano)
Fiumi (NomeF, Lunghezza)
Laghi (NomeL, Superficie)
Litorale (Nazione, Mare)
Attraversamento (Nazione, Fiume)

PRECISAZIONE

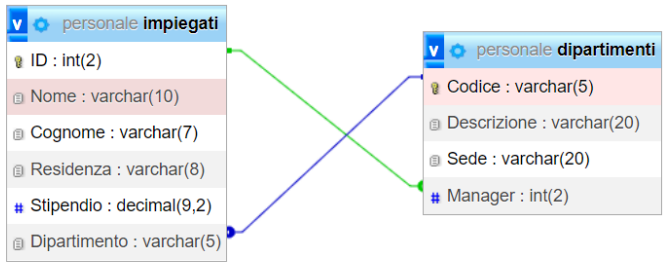
Tabella1 INTERSECT *Tabella2*;

equivale a:

```
SELECT *
FROM Tabella1
WHERE Chiave IN
      (SELECT Chiave
      FROM Tabella2);
```

SUBQUERY NELLA FROM

- Da quante differenti città di residenza provengono i dipendenti del dipartimento *Produzione*?




```
SELECT COUNT(DISTINCT Residenza)
FROM Impiegati
WHERE Dipartimento = 'Prod';
```

Sintassi non ammessa nell'SQL di Access

- Bisogna ricorrere a una query annidata nella clausola **FROM**

```
SELECT COUNT(*)
FROM ( SELECT DISTINCT Residenza
       FROM Impiegati
       WHERE Dipartimento = 'Prod');
```

QUERY PARAMETRICHE

Pur non supportate dallo standard ANSI, e quindi assenti nella maggior parte delle versioni di SQL, le interrogazioni parametriche sono molto utili quando si utilizza questo linguaggio in modalità stand alone.
Riferiamoci sempre al nostro database sulle proiezioni cinematografiche e consideriamo la seguente query:

```
SELECT *
FROM FILM
WHERE CognomeRegista = "Ozpetek" AND AnnoProduzione = "2008";
```

Per visualizzare i dati relativi al regista "Verdone" dovremmo riscrivere la query. Alcuni DBMS, come ad esempio Microsoft Access, permettono di parametrizzare una query in modo da scriverla una sola volta e sfruttarla per diversi valori del parametro. Per fare questo, è necessario racchiudere il parametro tra parentesi quadre. Prima di eseguire l'interrogazione, verrà chiesto di inserire il valore per i parametri specificati. L'esempio precedente può allora essere riscritto nel seguente modo:

```
SELECT *
FROM FILM
WHERE CognomeRegista = [Inserire il cognome] AND AnnoProduzione =
                                              [Inserire l'anno];
```

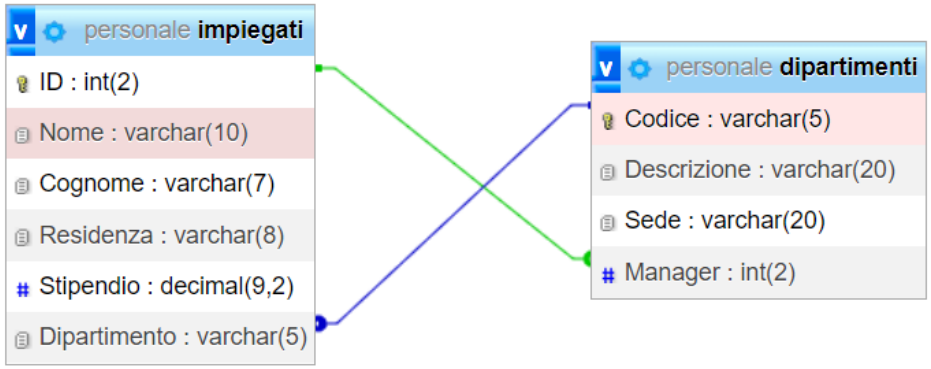
VISTE LOGICHE

In SQL è possibile definire un'altra classe di tabelle, chiamate **viste**, che non sono fisicamente memorizzate nella base di dati (sono infatti costruite nella memoria RAM), ma che possono essere definite solo logicamente.

Le viste solitamente sono utilizzare per:

- **PROTEGGERE I DATI: tramite le viste diamo la possibilità di fare visualizzare agli utenti solo di dati che voliamo rendere disponibili**
- **CONVERTIRE LE UNITA' DI MISURA: se ad esempio abbiamo un campo Importo che contiene il valore in dollari, tramite una vista si possono rendere convertiti in euro**
- **SEMPLIFICARE LA COSTRUZIONE DI QUERY COMPLESSE: tramite una vista si rendono disponibili in modo semplici dati che derivano da query complesse**

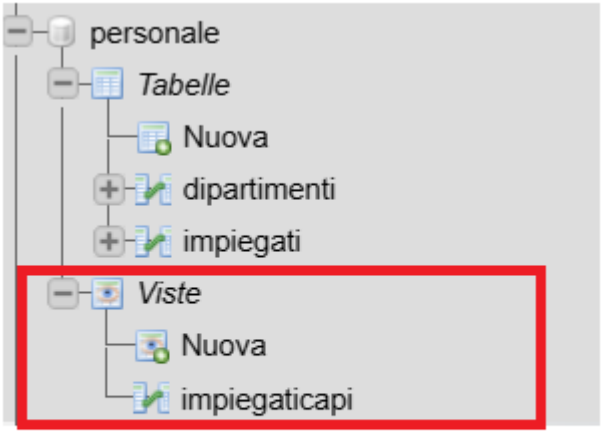
ESEMPIO



Utilizzando il database in figura si può creare la seguente vista

```
CREATE VIEW ImpiegatiCapi AS
SELECT ID, Nome, Cognome, Dipartimento
FROM Impiegati
WHERE ID IN (10, 12, 13,14)
```

ESEMPIO



Nel database è stata inserita la vista.

Sulla vista possiamo fare una select come su una tabella.

```
1 SELECT *
2 FROM ImpiegatiCapi;
```

Abilita i controlli sulle chiavi esterne

Esegui | Annulla

Profiling [Modifica inline] [Modifica] [Spiega SQL] [Crea il codice PHP] [Aggiorna]

Mostra tutti | Numero di righe: 25 | Filtra righe: Cerca nella tabella

Opzioni extra

	ID	Nome	Cognome	Dipartimento
Modifica Copia Elimina	10	Margherita	Colombi	Prod
Modifica Copia Elimina	12	Franco	Volpi	Amm
Modifica Copia Elimina	13	Ugo	Boss	Direz
Modifica Copia Elimina	14	Mario	Gatti	R&S

PER ELIMINARE UNA VISTA

PER ELIMINARE LA VISTA

DROP VIEW ImpiegatiCapi;

VISTA: ESEMPIO DEL LIBRO A PAG 625

Data la seguente tabella

CodiceAccess	Descrizione	PrezzoAcquisto	PrezzoVendita	Quantita
A01	Batteria	100,00	120,00	3
A02	Tergicristalli	80,00	100,00	0
A03	Specchietto Dx	50,00	75,00	1
A04	Specchietto Sx	50,00	75,00	0

E' possibile creare la seguente vista in modo da poter visualizzare solo gli accessori che hanno la quantità a 0

```
CREATE VIEW DAORDINARE AS
SELECT *
FROM ACCESSORIO
WHERE Quantita = 0;
```

CodiceAccess	Descrizione	PrezzoAcquisto	PrezzoVendita	Quantita
A02	Tergicristalli	80,00	100,00	0
A04	Specchetto Sx	50,00	75,00	0

VISTA: ESEMPIO DEL LIBRO A PAG 625

Data la seguente tabella

CodiceAccess	Descrizione	PrezzoAcquisto	PrezzoVendita	Quantita
A01	Batteria	100,00	120,00	3
A02	Tergicristalli	80,00	100,00	0
A03	Specchietto Dx	50,00	75,00	1
A04	Specchietto Sx	50,00	75,00	0

E' possibile creare la seguente vista in modo da poter visualizzare alcuni dei campi della tabella

```
CREATE VIEW VENDITA AS
SELECT CodiceAccess, Descrizione, PrezzoVendita, Quantita
FROM ACCESSORIO;
```