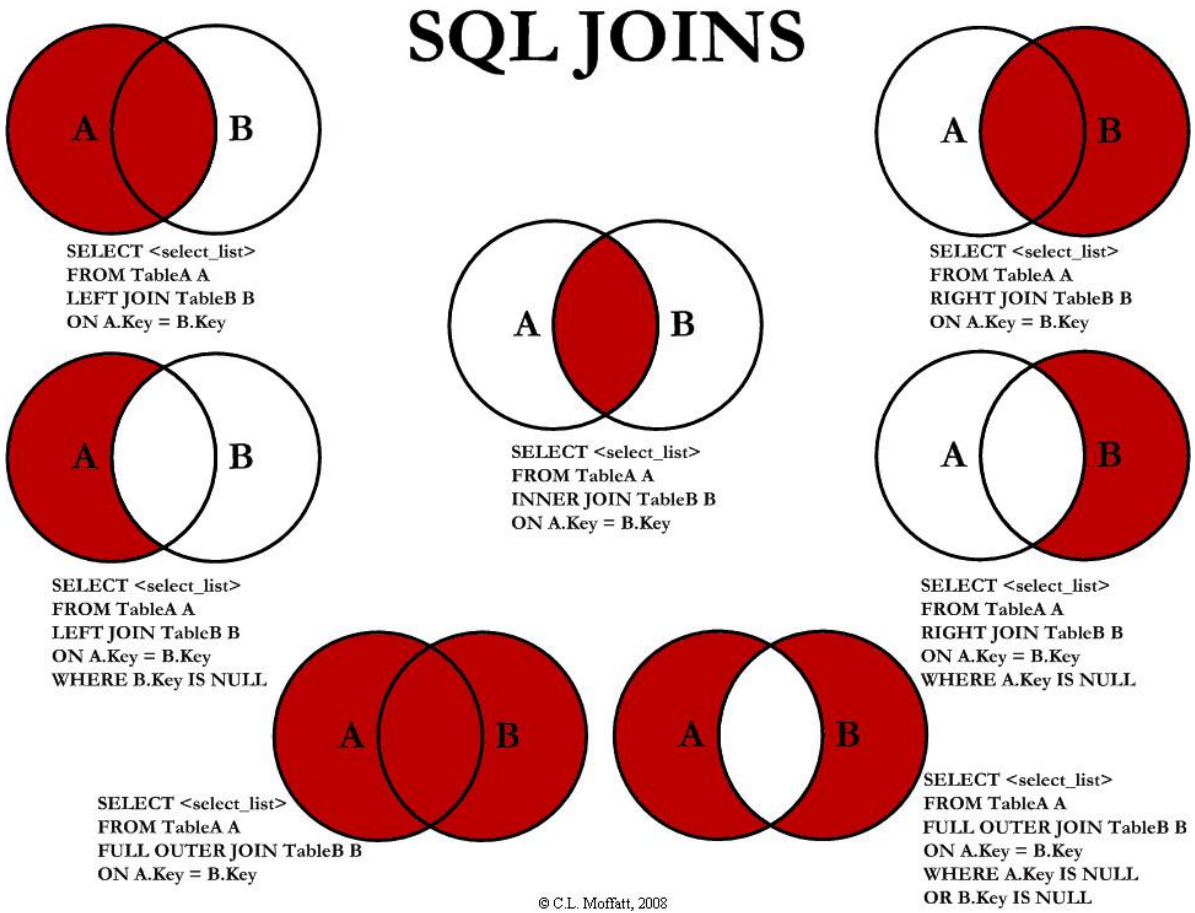
Progettazione Fisica e SQL

Ing. Alessandro Pellegrini, PhD
pellegrini@diag.uniroma1.it

Query SQL

Le operazioni di Join



Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Le città con un aeroporto di cui non è noto il numero di piste.

```
select Città
from AEROPORTO
where NumPiste is NULL
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Le nazioni da cui parte e arriva il volo AZ274.

```
select A1.Nazione, A2.Nazione
from AEROPORTO as A1 join VOLO on A1.Citta = CittaArr
join AEROPORTO as A2 on CittaPart = A2.Citta
where IdVolo='AZ274'
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - I tipi di aereo usati nei voli che partono da Torino.

```
select TipoAereo
from VOLO
where CittaPart = 'Torino'
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - I tipi di aereo e il corrispondente numero di passeggeri per i tipi di aereo usati nei voli che partono da Torino. Se la descrizione dell'aereo non è disponibile, visualizzare solamente il tipo.

```
select VOLO.TipoAereo, NumPasseggeri
from VOLO left join AEREO on VOLO.TipoAereo=AEREO.TipoAereo
where CittaPart = 'Torino'
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Le città da cui partono voli internazionali

```
select CittàPart
from AEROPORTO as A1 join VOLO on CittaPart=A1.Citta
join AEROPORTO as A2 on CittaArr=A2.Citta
where A1.Nazione <> A2.Nazione
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Le città da cui partono voli diretti a Bologna, ordinate alfabeticamente

```
select CittaPart
from VOLO
where CittaArr = 'Bologna'
order by CittaPart
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Il numero di voli internazionali che partono il giovedì da Napoli

```
select count(*)
from VOLO join AEROPORTO on CittaArr=Citta
where Nazione<>'Italia' and CittaPart='Napoli' and
DAYOFWEEK(GiornoSett)=5 -- Domenica è 1
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Il numero di voli internazionali che partono ogni settimana da città italiane

```
select count(*)
from AEROPORTO as A1 join VOLO on A1.Citta=CittaPart
join AEROPORTO as A2 on A2.Citta=CittaArr
where A1.Nazione = 'Italia' and A2.Nazione <> 'Italia'
group by YEARWEEK(GiornoSett)
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Le città francesi da cui partono più di venti voli alla settimana diretti in Italia

```
select A1.Citta, YEARWEEK(GiornoSett) as Settimana,
count(A1.Citta)
from AEROPORTO as A1 join VOLO on A1.Citta=CittaPart
join AEROPORTO as A2 on A2.Citta=CittàArr
where A1.Nazione = 'Francia' and A2.Nazione = 'Italia'
group by YEARWEEK(GiornoSett), A1.Citta
having count(A1.Citta) > 20
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Gli aeroporti italiani che hanno solo voli interni, usando operatori insiemistici

```
select distinct CittaPart
from VOLO,AEROPORTO where CittaPart=Citta and Nazione='Italia'
except
select CittaPart
from AEROPORTO as A1 join VOLO on A1.Citta=CittaPart
join AEROPORTO as A2 on A2.Citta=CittaArr
where A1.Nazione = 'Italia' and A2.Nazione <> 'Italia'
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Gli aeroporti italiani che hanno solo voli interni, usando un'interrogazione nidificata con l'operatore not in

```
select distinct CittaPart
from VOLO,AEROPORTO where CittaPart=Citta
and CittaPart not in (select CittaPart
from AEROPORTO as A1 join VOLO on A1.Citta=CittaPart
join AEROPORTO as A2 on A2.Citta=CittaArr
where A1.Nazione = 'Italia' and A2.Nazione <> 'Italia')
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Gli aeroporti italiani che hanno solo voli interni, usando un'interrogazione nidificata con l'operatore not exists

```
select distinct CittaPart
from VOLO join AEROPORTO as A1 on CittaPart=A1.Citta
where Nazione='Italia' and not exists (select * from VOLO join
AEROPORTO as A2 on A2.Citta=CittaArr
where A1.Citta=CittaPart and A2.Nazione <> 'Italia' )
```

Esercizio 4.14

- Dato il seguente modello relazionale:
  AEROPORTO(Citta, Nazione, NumPiste*)
  VOLO(IdVolo, GiornoSett, CittaPart, OraPart, CittaArr, OraArr,TipoAereo)
  AEREO(TipoAereo, NumPasseggeri, QtaMerci)
- Scrivere le interrogazioni SQL che permettono di determinare:
  - Gli aeroporti italiani che hanno solo voli interni, usando una join ed un operatore di conteggio

```
select CittaPart
from AEROPORTO as A1 join VOLO on A1.Citta=CittaPart
left join AEROPORTO as A2 on CittaArr = A2.Citta
where A1.Nazione = 'Italia'
group by CittaPart
having count(case (A2.Nazione <> 'Italia') when true then 1 else
NULL end) = 0
```

SQL Avanzato

Utilizzo di Trigger per emulare le Asserzioni

- MySQL non supporta le asserzioni (in particolare il comando `STOP ACTION` non è supportato)
- Si possono utilizzare trigger “before update” per emulare il funzionamento delle asserzioni

```
create assertion AT_MOST_ONE_MANAGER as CHECK
((select count(*)  from `employees` E
   where E.ruolo = 'MANAGER') <= 1
)
```

Utilizzo di Trigger per emulare le Asserzioni

```
create trigger AT_MOST_ONE_MAGANGER
before insert on `employees` for each row
begin
	declare counter INT;
	select count(*) from `employees` E
	where E.ruolo = 'MANAGER' into counter;
	if counter > 1 then
		signal sqlstate '45000';
	end if;
end
```

Utilizzo di Trigger per emulare le Asserzioni

- Spesso è di interesse effettuare dei controlli sui valori che si stanno per inserire
  - In questo caso, la keyword `NEW` permette di accedere agli attributi della tupla che si sta inserendo

- Si può anche effettuare (nel caso di trigger di tipo `BEFORE UPDATE`) effettuare dei controlli sulla tupla che si sta per aggiornare
  - I vecchi valori sono accessibili mediante la keyword `OLD`
  - Questo può essere ad esempio utile per implementare meccanismi di storicizzazione

Eventi temporizzati

```
set global event_scheduler = on;


create event if not exists `cleanup`
    on schedule
      every 2 day
            on completion preserve
    comment 'Remove registrations which have expired'
    do
      delete from `assegnazione` where `confermato_il` is null and
            `richiesto_il` < (NOW() - interval 2 day)
```

Funzioni

```
delimiter !
create function `valid_email`(email varchar(45))
returns bool
deterministic
begin
        if email regexp '^[a-zA-Z0-9][a-zA-Z0-9._-]*[a-zA-Z0-9._-
]@[a-zA-Z0-9][a-zA-Z0-9._-]*[a-zA-Z0-9]\\.[a-zA-Z]{2,63}$' then
        return true;
    end if;
    return false;
end!
delimiter ;
```

Cursori

```
create procedure `itera_su_tabella`()
begin
   declare done int default false;
   declare varchar(45) var_nome;
   declare cur cursor for select nome from tabella;
   declare continue handler for not found set done = true;
   open cur;
   read_loop: loop
      fetch cur into var_nome;
      if done then
         leave read_loop;
      end if;
   end loop;
   close cur;
end
```