SQL

- Join
  - Inner
  - Outer
  - Cross
- Union
- Progetto di riferimento
  - https://github.com/egalli64/jd – folder mySql *(modulo 2)*

JOIN

- Selezione di dati provenienti da due tabelle
- INNER **JOIN** – viene creata una riga nel risultato per ogni regola di join soddisfatta
  - “INNER” è il default per una JOIN
- **LEFT/RIGHT** OUTER **JOIN** – come INNER, ma la tabella left/right
  - preserva i valori delle righe anche dove la regola non è soddisfatta
  - “OUTER” è implicito se una JOIN è LEFT/RIGHT
- FULL OUTER JOIN, combina LEFT e RIGHT, non è implementata da MySQL
  - può essere emulata via UNION
- self JOIN – left e right nella JOIN sono la stessa tabella
- non-equi JOIN – usano operatori diversi da “=”

INNER JOIN

- Selezione dati correlati su diverse tabelle

```
select region_name from regions where region_id = 1;
select country_name from countries where region_id = 1;
-- region_id = 1 .. 4
```

- Equi-join “classica” sulla relazione PK→FK

```
select region_name, country_name
from regions, countries
where regions.region_id = countries.region_id;
```

Alias per tabelle

- Si possono definire nel FROM alias per tabelle validi solo per la query corrente

```
select r.region_name, c.country_name
from regions r, countries c
where r.region_id = c.region_id;
```

JOIN – USING vs NATURAL JOIN

- INNER JOIN standard SQL/92

```
select region_name, country_name
from regions join countries -- join è “inner” per default
using (region_id);
```

- Se la relazione è “naturale” → NATURAL JOIN

```
select region_name, country_name
from regions natural join countries;
```

JOIN – ON

- NATURAL JOIN e JOIN – USING implicano una relazione equi-join per PK e FK con lo stesso nome
- JOIN – ON ci permette una maggior libertà

```
select region_name, country_name
from regions join countries
on regions.region_id = countries.region_id;
```

JOIN – WHERE

- JOIN – ON

```
select region_name, country_name
from regions r join countries c
on r.region_id = c.region_id
where r.region_id = 1;
```

- JOIN – USING

```
select region_name, country_name
from regions join countries
using (region_id)
where region_id = 1;
```

- NATURAL JOIN

```
select region_name, country_name
from regions natural join countries
where region_id = 1;
```

- query classica equivalente

```
select region_name, country_name
from regions r, countries c
where r.region_id = c.region_id
and r.region_id = 1;
```

Prodotto Cartesiano

- Se manca la condizione in una JOIN, ogni riga della prima tabella viene abbinata con tutte le righe della seconda

```
select region_name, country_name
from regions, countries;
```

- SQL/92 CROSS JOIN, richiede che sia esplicito

```
select region_name, country_name
from regions cross join countries;
```

- Ma MySQL interpreta JOIN senza ON o USING come CROSS

Self JOIN

- La FK si riferisce alla PK della stessa tabella

```
select e.last_name as employee, m.last_name as manager
from employees e join employees m
on e.manager_id = m.employee_id;
```

- Versione “classica”

```
select e.last_name as employee, m.last_name as manager
from employees e, employees m
where e.manager_id = m.employee_id;
```

JOIN su più di due tabelle

- JOIN – ha solo una tabella left e una right → 2 JOIN per 3 tabelle

```
select employee_id, city, department_name
from employees join departments using (department_id)
join locations using (location_id);
```

- Versione “classica” → 2 condizioni nel WHERE per 3 tabelle

```
select employee_id, city, department_name
from employees e, departments d, locations l
where d.department_id = e.department_id and d.location_id = l.location_id;
```

Non-equi JOIN

- JOIN basate su operatori diversi da “=”

```
select j.job_title, j.min_salary, j.max_salary
from employees e join jobs j
on e.salary between j.min_salary and j.max_salary and e.job_id != j.job_id
where e.employee_id = 107;
```

- Versione “classica”

```
select j.job_title, j.min_salary, j.max_salary
from employees e, jobs j
where e.salary between j.min_salary and j.max_salary
    and e.job_id != j.job_id and e.employee_id = 107;
```

LEFT OUTER JOIN

- Genera un risultato per ogni riga nella tabella left
  - anche se non ha una relazione con righe nella tabella right
- La mancanza di valori nella tabella right è resa con NULL

```
select first_name, department_name
from employees left outer join departments
using(department_id)
where last_name = 'Grant';
```

RIGHT OUTER JOIN

- Genera un risultato per ogni riga nella tabella right
  - anche se non ha una relazione con righe nella tabella left
- La mancanza di valori nella tabella left è resa con NULL

```
select first_name, last_name, department_name
from employees right outer join departments
using(department_id)
where department_id between 110 and 120;
```

UNION

- Combina più SELECT statement generando un singolo result set
  - Ogni SELECT deve ritornare lo stesso numero di colonne
  - Colonne corrispondenti nelle SELECT devono avere un tipo di dato compatibile
  - Il nome della colonna nel result set finale è determinato dalla prima SELECT
  - Il result set generato può essere ordinato via ORDER BY

UNION – esempio

- Cerca Tim… nelle tabelle employees e coders

```
select first_name, last_name, 'coder' as role
from coders where first_name like 'Tim%'
union
select first_name, last_name, 'employee'
from employees where first_name like 'Tim%';
```

Esercizi

- Nome degli employees e del loro department
- Nome degli employees e job title (da JOBS)
- Nome degli employees che hanno il salario minimo o massimo previsto per il loro job title
- Nome degli employees basati in UK (LOCATIONS)
- Nome dei departments e manager associato

Esercizi /2

- Nome di ogni department e, se esiste, del relativo manager
- Nome dei department che non hanno un manager associato
- Nome degli employees e del loro manager