UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO DI MATEMATICA

LAUREA TRIENNALE IN INFORMATICA

PROGETTO DI BASI DI DATI

Basi di Dati di un Sistema Informativo per la Gestione di Crociere Organizzate

Ghiraldin Mirco, Stevanin Michele

1 Abstract

In questo elaborato viene presentato lo sviluppo di una base di dati relazionale pensata per **gestire le attività delle compagnie di crociere**. L'obiettivo è costruire un sistema informativo in grado di organizzare in modo chiaro ed efficiente tutte le informazioni legate alla gestione delle navi, del personale e delle attività a bordo. Il progetto copre l'intero ciclo di vita di una crociera: si parte dall'identificazione delle compagnie organizzatrici (tramite partita IVA), per arrivare alla gestione delle singole crociere, ciascuna con il proprio itinerario, i porti di partenza e arrivo, le date, le tappe intermedie e altri dettagli tecnici. A bordo interagiscono diverse figure: passeggeri, membri dell'equipaggio e animatori, rappresentati nel modello tramite un sistema di generalizzazioni e specializzazioni che permette di descrivere con flessibilità i diversi ruoli. Il modello affronta anche aspetti legati alle prestazioni del sistema, come ad esempio l'analisi della ridondanza relativa al numero di passeggeri prenotati. Il progetto è stato sviluppato seguendo un approccio modulare, ispirato alle metodologie viste durante il corso, con l'obiettivo di garantire integrità, coerenza e adattabilità in contesti reali di gestione delle crociere.

2 Analisi dei Requisiti

Compagnie Marittime

Ogni compagnia è identificata da una partita IVA univoca e include le seguenti informazioni:

- **Partita IVA** (PK)
- **Nome**
- **Sede**
- **Recapito**

Crociere

Ogni crociera è identificata dal codice IMO e registra:

- **Codice IMO** (PK)
- **Nome della nave**
- **Porto di partenza**
- **Porto di arrivo**
- **Data e ora di partenza**
- **Durata** (in giorni)
- **Numero minimo di membri dell'equipaggio**
- **Numero massimo di passeggeri**
- **Tipologia** (es. mediterranea, fluviale, transatlantica)

Ogni crociera:

- È di proprietà di una **compagnia**
- Parte da un **porto** e arriva in un altro e può prevedere **tappe** intermedie
- Ha a bordo **ospiti** e **membri dell'equipaggio**
- Include **eventi** gestiti da **animatori**

Porti

Ogni porto è identificato dalla città in cui si trova:

- **Nome della città** (PK)
- **Numero massimo di navi**

Una crociera può prevedere soste in più porti (tappe).

Tappe

Le tappe rappresentano le fermate della crociera nei porti e includono:

- **Data e ora di arrivo**
- **Data e ora di partenza**

Ogni tappa è collegata a un porto e a una specifica crociera.

Persone

Tutti gli individui a bordo (sia personale che ospiti) sono entità del tipo "Persona", identificata tramite:

- **Codice Fiscale** (PK)
- **Nome**
- **Cognome**
- **Sesso**

Una persona può essere un ospite o un membro dell'equipaggio.

Ospiti (Passeggeri)

Gli ospiti sono persone che hanno prenotato una crociera:

- **Codice Fiscale** (PK, FK da Persona)
- **Costo della crociera**
- **Crociera** (IMO) (FK)

Equipaggio

L'equipaggio rappresenta il personale operativo a bordo. Ogni membro è associato a:

- **Codice Fiscale** (PK, FK da Persona)
- **ID equipaggio**
- **Stipendio**
- **Anni di servizio**
- **Lingue parlate**

Un membro dell'equipaggio può essere specializzato in animatore.

Animatori

Gli animatori sono una specializzazione dell'equipaggio e dispongono di una o più abilità:

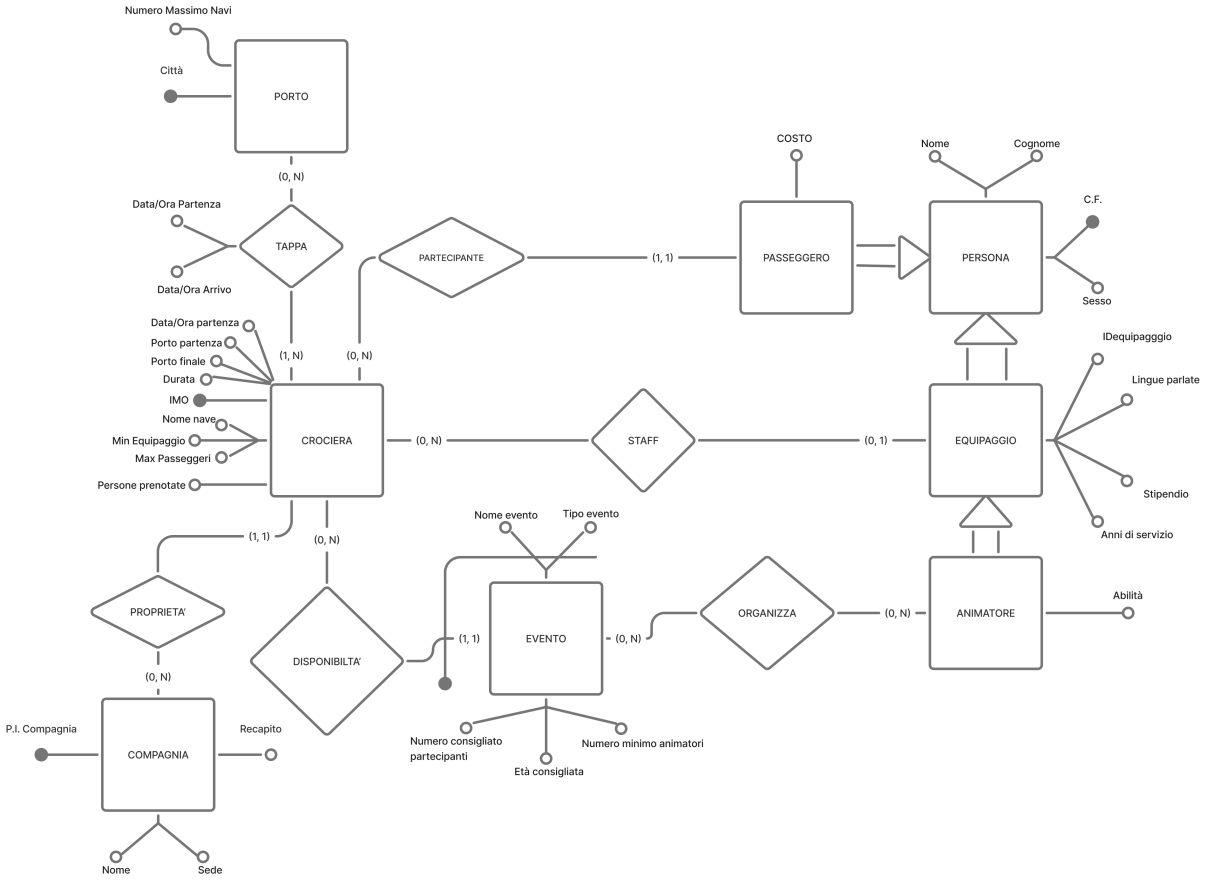
- **Codice Fiscale** (PK, FK da Equipaggio)
- **Abilità specifiche** (es. ballo, canto, giochi)

Eventi

Durante le crociere vengono organizzati eventi, ciascuno identificato da:

- **Nome dell'evento** (PK)
- **Tipo di evento** (PK)
- **Età consigliata**
- **Numero minimo di animatori**
- **Numero consigliato di partecipanti**

Ogni evento è gestito da uno o più animatori e può essere frequentato da più ospiti.



(Grafico1)

3 Progettazione Concettuale

La Figura 1 riporta il diagramma Entità–Relazione (E–R) che riassume i requisiti descritti nella sezione 2. Nel modello la progettazione concettuale si è basata su una visione gerarchica del personale di bordo, dove la generalizzazione rappresenta il fatto che tutti i membri dell'equipaggio e gli ospiti condividono un insieme comune di attributi base, ma possono essere specializzati in ruoli specifici. Ogni crociera è organizzata da una compagnia marittima e ha una propria identità definita dal codice IMO. Le crociere sono associate a uno o più porti attraverso una relazione Tappa, che rappresenta gli scali effettuati durante il viaggio. Ogni tappa è caratterizzata da una data e ora di arrivo e partenza, ed è sempre riferita a un porto e a una crociera specifici. Le persone imbarcate sono suddivise in Equipaggio mediante una generalizzazione parziale. Il personale di bordo può ricoprire ruoli specializzati: ad esempio gli Animatori, sottoinsieme dell'equipaggio, sono incaricati dell'organizzazione e conduzione degli eventi. Gli Eventi sono attività previste a bordo della crociera. Ogni evento può essere gestito da uno o più animatori (tramite la relazione Gestione). Gli eventi sono identificati in modo univoco dalla combinazione tra nome e tipologia e presentano attributi come età consigliata e limiti sulla partecipazione. La generalizzazione tra Equipaggio e la specializzazione Animatore è modellata come parziale: non tutti i membri dell'equipaggio sono necessariamente animatori. Tabella 1 riassume le entità e relazioni individuate nella progettazione concettuale, riportando per ciascuna gli attributi rilevanti e l'identificatore scelto. Per le entità derivate da generalizzazione viene anche specificato il tipo di specializzazione utilizzato. Il presente schema E-R non permette di rappresentare direttamente il seguente vincolo: se una persona pe è animatore in due crociere cr' e cr'', allora cr' e cr'' appartengono alla stessa Classe:

```
(cr', pe) ∈ Animatore ∧ (cr'', pe) ∈ Animatore ⇒ (cr'.Classe = cr''.Classe)
```

4 Progettazione Logica

In questa sezione viene illustrato il processo di "traduzione" dello schema concettuale in uno schema logico, con l'obiettivo di rappresentare i dati in modo preciso ed efficiente. Il primo passo consiste nell'analizzare le eventuali ridondanze nel modello, al fine di ottimizzare la struttura complessiva. Successivamente, si procede con l'eliminazione delle due generalizzazioni. Infine, viene presentato il diagramma ristrutturato, con una descrizione delle modifiche apportate.

Entità	Descrizione	Attributi	Identificatore
Crociera	Mezzo di trasporto marittimo	IMO, Nome nave, Porto partenza, Porto finale, Data/ora partenza, Durata, Min equipaggio, Max passeggeri, Persone prenotate	IMO
Porto	Località in cui la crociera può attraccare	Città, Numero massimo navi	Città
Persona	Persone partecipanti alle crociere	Codice fiscale (CF), Nome, Cognome, Sesso	CF

Entità	Descrizione	Attributi	Identificatore
Passeggero	Passeggeri ospiti della crociera	Codice fiscale (CF), Costo, IMO crociera	CF
Equipaggio	Persone membri dell'equipaggio	Codice fiscale (CF), Id equipaggio, Lingue parlate, Stipendio, Anni di servizio, IMO crociera	CF
Animatore	Membri dell'equipaggio che si occupano degli eventi	Codice fiscale (CF), Abilità	CF
Compagnia	Società che possiedono le navi da crociera	Partita IVA (P.I. compagnia), Nome, Sede, Recapito	P.I. compagnia
Evento	Eventi ricreativi svolti a bordo delle crociere	Nome evento, Tipo evento, Età consigliata, Numero minimo animatori, Numero consigliato partecipanti, IMO crociera	Nome evento, Tipo evento, IMO crociera

Tabella 1 (Entità)

Relazione	Descrizione	Componenti	Attributi
Tappa	Tappe portuali fatte da una nave	Porto, Crociera	Data/ora arrivo, Data/ora partenza
Partecipante	Assegnazione di una persona (ospite) a una crociera	Crociera, Passeggero (ospite)	–
Staff	Membri dell'equipaggio assegnati a una crociera	Crociera, Equipaggio	–
Proprietà	Navi da crociera possedute da una compagnia	Crociera, Compagnia	–
Disponibilità	Eventi programmati per una crociera	Crociera, Evento	–
Organizza	Assegnazione di uno o più animatori a un evento	Evento, Animatore	–

Tabella 2 (Relazioni)

4.1 Analisi delle ridondanze

L'attributo Persone_Prenotate in CROCIERA, che memorizza il numero di persone prenotate in quella crociera, presenta una ridondanza. Questo valore può essere infatti ottenuto contando il numero di passeggeri attivi per quella crociera tramite la relazione PARTECIPANTE. Questo attributo viene modificato ogni volta che si aggiunge una nuova persona alla crociera (circa 400 persone nuove al giorno tra tutte le crociere) e viene visualizzato ogni ora del giorno per monitorare il numero di posti rimanenti. Questo si riassume nelle seguenti due operazioni:

- Operazione 1 (400 al giorno): memorizza una nuova prenotazione in relativa crociera.
- Operazione 2 (24 al giorno): visualizza il numero di prenotazioni attuali in una crociera.

Assumendo i seguenti volumi nella base di dati:

Concetto	Costrutto	Volume
CROCIERA	E	30
PARTECIPANTE	R	150000
PASSEGGERO	E	150000

La seguente analisi serve per stabilire se sia utile o meno tenere l'attributo ridondante Persone_Prenotate in CROCIERA.

CON RIDONDANZA

Analizziamo prima il costo totale con ridondanza.

- Operazione 1:

Concetto	Costrutto	Accessi	Tipo	Ripetizioni
PASSEGGERO	E	1	S	× 400
PARTECIPANTE	R	1	S	× 400
CROCIERA	E	1	L	× 400
CROCIERA	E	1	S	× 400

- Operazione 2:

Concetto	Costrutto	Accessi	Tipo	Ripetizioni
Crociera	E	1	L	× 24

Assumendo costo doppio per gli accessi in scrittura:

```
Costo Totale = 400 * 3 * 2 + 400 + 24 = 2824
```

SENZA RIDONDANZA

Analizziamo il costo totale senza ridondanza.

- Operazione 1:

Concetto	Costrutto	Accessi	Tipo	Ripetizioni
PASSEGGERO	E	1	S	× 400
PARTECIPANTE	R	1	S	× 400

- Operazione 2 (con circa 150000/30 = 5000 passeggeri al giorno):

Concetto	Costrutto	Accessi	Tipo	Ripetizioni
CROCIERA	E	1	L	× 24
PARTECIPANTE	R	5000	L	× 24

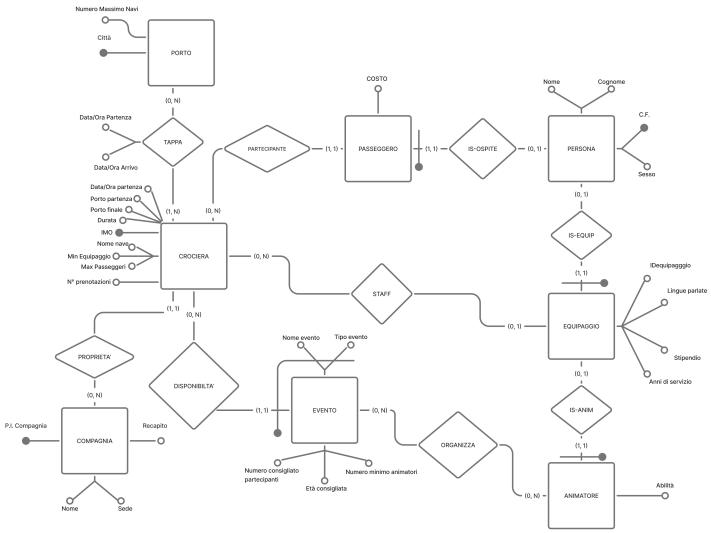
Assumendo costo doppio per gli accessi in scrittura:

```
Costo Totale = 400 * 2 * 2 + 5001 * 24 = 121624
```

L'analisi suggerisce quindi di tenere l'attributo ridondante, ottimizzando così il numero di accessi.

4.2 Eliminazioni delle Generalizzazioni

Le generalizzazioni descritte in Sezione 3 vengono eliminate attraverso una ristrutturazione dello schema concettuale, con l'obiettivo di semplificare la successiva implementazione del modello relazionale e ridurre la presenza di valori nulli. Le modifiche vengono applicate come segue:



(Grafico2)

PERSONA. La generalizzazione parziale PERSONA viene sostituita con le relazioni **IS-OSPITE** e **IS-EQUIP** (vedi Figura 2), che collega alcuni individui alla relativa specializzazione: EQUIPAGGIO o PASSEGGIERO. Tale scelta consente di evitare la presenza di valori nulli che si verificherebbero mantenendo un'unica entità PERSONA con tutti gli attributi specifici delle due categorie (ad esempio, Stipendio, Anni_Di_Servizio, Lingue_Parlate per EQUIPAGGIO, Costo per PASSEGGERO). Separando le informazioni tramite relazioni specializzate, si garantisce che ciascuna entità contenga soltanto gli attributi rilevanti per il proprio ruolo. In linea con la metodologia adottata a lezione, l'identificatore di EQUIPAGGIO e di PASSEGGERO coincide con quello della rispettiva PERSONA. Poiché la generalizzazione è parziale da entrambe le parti (non tutte le persone sono membri dell'equipaggio né passeggeri), l'eliminazione dell'entità padre PERSONA non è corretta. Essa viene mantenuta per rappresentare tutte le informazioni comuni (come Nome, Cognome, Data_Nascita), mentre le informazioni specifiche sono distribuite nelle entità figlie.

EQUIPAGGIO. Analogamente, la generalizzazione parziale EQUIPAGGIO viene sostituita con **IS-ANIM** (vedi Figura 2). Anche in questo caso, la ristrutturazione consente di evitare valori nulli, in quanto non tutti i membri dell'equipaggio svolgono il ruolo di Animatore. Essendo la generalizzazione parziale, l'eliminazione dell'entità padre EQUIPAGGIO risulterebbe nuovamente scorretta.

Il diagramma E–R ristrutturato, riportato in **Figura 2** riflette tali modifiche rispetto alla versione originale presentata in **Figura 1**.

4.3 Schema Relazionale

Lo schema ristrutturato in Figura 2 contiene solamente costrutti mappabili in corrispettivi dello schema relazionale, detto anche schema logico. Lo schema logico è rappresentato a seguire, dove l'asterisco dopo il nome degli attributi indica quelli che ammettono valori nulli.

- **Crociera**(IMO, Nome_Nave, Min_Equipaggio, Max_Passeggeri, Num_Prenotazioni, Porto_Partenza*, Porto_Finale*, Data_Ora_Partenza*, Durata*, PI_Compagnia)
  - Crociera.PI_Compagnia -> Compagnia.PI
  - Crociera.Porto_Partenza -> Porto.Città
  - Crociera.Porto_Finale -> Porto.Città
- **Porto**(Città, Numero_Massimo_Navi*)
- **Tappa**(IMO, Città, Data_Ora_Partenza, Data_Ora_Arrivo)
  - Tappa.IMO -> Crociera.IMO
  - Tappa.Città -> Porto.Città
- **Compagnia**(PI, Nome, Sede, Recapito_Telefonico*)
- **Persona**(CF, Nome, Cognome, Sesso)
- **Ospite**(CF, Costo*, IMO_Crociera)
  - Ospite.CF -> Persona.CF
  - IMO_Crociera -> Crociera.IMO
- **Equipaggio**(CF, IDequipaggio, Lingue_Parlate*, Stipendio*, Anni_Servizio*, IMO_Crociera*)
  - Equipaggio.CF -> Persona.CF
  - IMO_Crociera -> Crociera.IMO
- **Animatore**(CF, Abilità*)
  - Animatore.CF -> Equipaggio.CF
- **Evento**(Nome, Tipologia, IMO_Crociera, Num_Cons_Partecipanti*, Num_Min_Animatori*, Età_Consigliata*)
  - Evento.IMO_Crociera -> Crociera.IMO

- **ORGANIZZA**(CF_Animatore, Nome_Evento, Tipologia_Evento, IMO_Crociera_Evento)
  - Organizza.CF_Animatore -> Animatore.CF
  - Organizza.Nome_Evento -> Evento.Nome
  - Organizza.Tipologia_evento -> Evento.Tipologia
  - Organizza.IMO_Crociera_Evento -> Evento.IMO_Crociera

5 Implementazione in PostgreSQL e Definizione delle Query

Il file Crociere.sql contiene il codice SQL necessario per la creazione e il popolamento delle tabelle del database. Questo file include inoltre una serie di query per l'estrazione dei dati e un indice creato specificamente per migliorare le prestazioni di una di queste interrogazioni.

5.1 Definizione delle Query

Di seguito vengono presentate e descritte le query con i relativi output generati e viene motivato l'utilizzo dell'indice proposto.

Query 1 Trovare le crociere che toccano più di un numero di porti diversi indicato dall'utente e indicarne la città di partenza, di arrivo e il numero di tappe. Ordinate in modo decrescente dalla crociera con il maggior numero di tappe. Nel nostro caso abbiamo selezionato 2.

```
SELECT C.IMO, C.Nome_Nave, C.Porto_Partenza, C.Porto_Finale, COUNT(DISTINCT
T.Città) AS Numero_Tappe
FROM Crociera C
JOIN Tappa T ON C.IMO = T.IMO
GROUP BY C.IMO, C.Nome_Nave, C.Porto_Partenza, C.Porto_Finale
HAVING COUNT(DISTINCT T.Città) > '<NUMERO TAPPE>'
ORDER BY Numero_Tappe DESC;
```

Estratto dell'output:

	imo [PK] character (10)	nome_nave character varying (100)	porto_partenza character varying (100)	porto_finale character varying (100)	numero_tappe bigint
1	IMO0000008	Tramonto d Oriente	Brindisi	Atene	4
2	IMO0000004	Oceano Blu	Civitavecchia	Valencia	4
3	IMO0000015	Poseidon	Ajaccio	Civitavecchia	4
4	IMO0000003	Stella del Mare	Palermo	Marsiglia	4
5	IMO0000011	Mediterranea Star	Salerno	Palermo	4
6	IMO0000018	Onda Luminosa	Palermo	Napoli	4
7	IMO0000020	Croazia Blu Tour	Dubrovnik	Spalato	4
8	IMO0000001	Nave Azzurra	Genova	Atene	4
9	IMO0000013	Aurora	Cagliari	Ajaccio	3
10	IMO0000016	Atlantis	Barcellona	Marsiglia	3
11	IMO0000017	Europa Cruise	Marsiglia	Valencia	3
12	IMO0000021	Baltico Express	Tallinn	Stoccolma	3
13	IMO0000002	Sole Mediterraneo	Napoli	Barcellona	3
14	IMO0000005	Alba Serena	Venezia	Dubrovnik	3
15	IMO0000006	Costa Magica	Livorno	Marsiglia	3
16	IMO0000007	Mare Dorato	La Spezia	Barcellona	3
17	IMO0000009	Sirena Bianca	Bari	Corfù	3
18	IMO0000010	Regina del Mare	Ancona	Salonicco	3
19	IMO0000012	Vento del Sud	Reggio Calabria	Messina	3

(Query1)

QUERY 2 Visualizzare tutte le crociere in partenza da una citta' inserita dall'utente, ordinate in modo crescente per il numero IMO della crociera. Nella nostra query abbiamo usato la città di Genova.

```
SELECT c.IMO, c.Nome_Nave, c.Num_Prenotazioni
FROM Crociera c
WHERE c.Porto_Partenza = '<PORTO>';
```

Estratto dell'output:

	imo [PK] character (10)	nome_nave character varying (100)	num_prenotazioni integer
1	IMO0000001	Nave Azzurra	400
2	IMO0000035	Blue Harmony	490

(Query2)

Query 3 Trovare le crociere che hanno una media del costo dei biglietti superiore ad un importo indicato. Vengono mostrare in ordine decrescente dalla più costosa fino al valore selezionato dall'utente. Nel nostro esempio abbiamo selezionato 500 euro.

```
SELECT O.IMO_Crociera, C.Nome_Nave, AVG(O.Costo) AS Media_Costo
FROM Ospite O
JOIN Crociera C ON O.IMO_Crociera = C.IMO
GROUP BY O.IMO_Crociera, C.Nome_Nave
HAVING AVG(O.Costo) > '<PREZZO>'
ORDER BY Media_Costo DESC;
```

Estratto dell'output:

	imo_crociera character (10)	nome_nave character varying (100)	media_costo numeric
1	IMO0000032	Azzurra Light	978.0000000000000000
2	IMO0000024	Sirena Nera	950.0000000000000000
3	IMO0000018	Onda Luminosa	931.0000000000000000
4	IMO0000010	Regina del Mare	840.5000000000000000
5	IMO0000026	Sea Spirit	821.0000000000000000
6	IMO0000030	Stella Polare	801.0000000000000000
7	IMO0000014	Azzurra Dream	773.0000000000000000
8	IMO0000021	Baltico Express	747.0000000000000000
9	IMO0000003	Stella del Mare	726.5000000000000000
10	IMO0000035	Blue Harmony	704.0000000000000000
11	IMO0000002	Sole Mediterraneo	689.5000000000000000
12	IMO0000008	Tramonto d Oriente	677.0000000000000000
13	IMO0000007	Mare Dorato	665.5000000000000000
14	IMO0000016	Atlantis	660.0000000000000000
15	IMO0000011	Mediterranea Star	647.0000000000000000
16	IMO0000009	Sirena Bianca	643.5000000000000000
17	IMO0000029	Costa del Nord	639.0000000000000000
18	IMO0000034	Riviera Star	631.0000000000000000
19	IMO0000004	Oceano Blu	614.5000000000000000
20	IMO0000020	Croazia Blu Tour	597.0000000000000000
21	IMO0000006	Costa Magica	589.0000000000000000
22	IMO0000028	Marina di Sera	574.0000000000000000
23	IMO0000013	Aurora	560.0000000000000000

(Query3)

Query 4 Visualizzare il numero di eventi organizzati per crociera e la media degli eventi a cui ogni animatore deve partecipare, ordinati in modo decrescente dalla crociera con la più alta media di eventi che deve fare ogni animatore.

```
SELECT
    C.Nome_Nave,
    C.IMO,
    COUNT(DISTINCT A.CF) AS Num_Animatori,
    ROUND(CAST(COUNT(O.CF_Animatore) AS DECIMAL) / NULLIF(COUNT(DISTINCT A.CF),
0), 2) AS Media_Eventi_Per_Animatore
FROM Crociera C
JOIN Evento E ON C.IMO = E.IMO_Crociera
JOIN ORGANIZZA O ON E.Nome = O.Nome_Evento
                AND E.Tipologia = O.Tipologia_Evento
                AND E.IMO_Crociera = O.IMO_Crociera_Evento
JOIN Animatore A ON A.CF = O.CF_Animatore
GROUP BY C.Nome_Nave, C.IMO
ORDER BY Media_Eventi_Per_Animatore DESC;
```

Estratto dell'output:

	nome_nave character varying (100)	imo [PK] character (10)	num_animatori bigint	media_eventi_per_animatore numeric
1	Nave Azzurra	IMO0000001	2	2.00
2	Sole Mediterraneo	IMO0000002	1	2.00
3	Stella del Mare	IMO0000003	2	2.00
4	Alba Serena	IMO0000005	2	2.00
5	Mare Dorato	IMO0000007	1	2.00
6	Tramonto d Oriente	IMO0000008	1	2.00
7	Sirena Bianca	IMO0000009	1	2.00
8	Poseidon	IMO0000015	1	2.00
9	Atlantis	IMO0000016	1	2.00
10	Onda Luminosa	IMO0000018	1	2.00
11	Grecia Explorer	IMO0000019	1	2.00
12	Croazia Blu Tour	IMO0000020	1	2.00

(Query4)

Query 5 Trovare, per ogni crociera, la percentuale di occupazione rispetto alla capacità massima (Num_Prenotazioni / Max_Passeggeri) e visualizzarle in ordine decrescente.

```
SELECT IMO, Nome_Nave,
    ROUND((Num_Prenotazioni * 100.0) / Max_Passeggeri, 2) AS
Percentuale_Occupazione
FROM Crociera
WHERE Max_Passeggeri > 0
ORDER BY Percentuale_Occupazione DESC;
```

Estratto dei primi 25 elemementi dell'output:

	imo [PK] character (10)	nome_nave character varying (100)	percentuale_occupazione numeric
1	IMO0000017	Europa Cruise	90.00
2	IMO0000025	Mediterraneo Lux	86.67
3	IMO0000002	Sole Mediterraneo	86.67
4	IMO0000007	Mare Dorato	86.67
5	IMO0000006	Costa Magica	86.21
6	IMO0000016	Atlantis	85.71
7	IMO0000008	Tramonto d Oriente	85.71
8	IMO0000009	Sirena Bianca	85.45
9	IMO0000033	Nave del Sole	85.29
10	IMO0000021	Baltico Express	83.87
11	IMO0000014	Azzurra Dream	83.33
12	IMO0000029	Costa del Nord	83.33
13	IMO0000024	Sirena Nera	82.35
14	IMO0000015	Poseidon	82.26
15	IMO0000022	Fjord Line	82.19
16	IMO0000028	Marina di Sera	81.82
17	IMO0000035	Blue Harmony	81.67
18	IMO0000023	Nordik Star	80.70
19	IMO0000027	Oceania Bellezza	80.39
20	IMO0000011	Mediterranea Star	80.33
21	IMO0000001	Nave Azzurra	80.00
22	IMO0000010	Regina del Mare	80.00
23	IMO0000019	Grecia Explorer	80.00
24	IMO0000034	Riviera Star	79.66
25	IMO0000031	Alba Marina	79.37

(Query5)

5.2 Creazione degli indici

Si suppone di voler ottimizzare la Query 2, per la quale occorre considerare:

1. Condizione di Join: p.Città = c.Porto_Partenza
2. Group by sulla colonna p.Città e aggregazione COUNT(c.IMO), AVG(c.Num_Prenotazioni)
3. Ordinamento sul numero di crociere in partenza (alias Numero_Crociere)

Per il punto 1, è opportuno creare un indice sulla colonna Porto_Partenza della tabella Crociera, poiché questa viene utilizzata in un join basato sull'uguaglianza con Porto(Città) e successivamente per il raggruppamento dei risultati. Inoltre, Porto_Partenza non è una chiave primaria e non è automaticamente indicizzata dal sistema.

```
CREATE INDEX idx_porto_partenza ON Crociera (Porto_Partenza);
```

Questo indice consente:

1. Un accesso rapido alle tuple della tabella Crociera che partono da una certa città
2. Un'efficiente esecuzione del join con la tabella Porto usando l'equivalenza sulla città,
3. Una scansione ordinata utile all'ottimizzazione del GROUP BY, permettendo di aggregare le tuple con lo stesso valore di **Porto_Partenza** con complessità inferiore.

Per quanto riguarda il punto 3, l'ordinamento viene effettuato su un attributo derivato (**Numero_Crociere, alias di COUNT(c.IMO)**), quindi non può essere direttamente indicizzato. Tuttavia, un ordinamento efficace viene facilitato dalla pre-aggregazione ottimizzata tramite l'indice sul campo di raggruppamento (**Porto_Partenza**).

```
Nota: la colonna Città in Porto è chiave primaria, quindi PostgreSQL crea
automaticamente un indice B+ Tree su di essa. Non è necessario creare un ulteriore
indice su Porto(Città) per il join.
```

6 Applicazione Software

Il file **Query.c** implementa un programma in linguaggio C che consente di connettersi a un database PostgreSQL contenente i dati relativi alla gestione delle crociere. La connessione avviene in locale, e per semplificare la portabilità è stata configurata senza password, a condizione che il sistema sia impostato per l'autenticazione locale tramite modalità peer o trust. Lo scopo principale del programma è eseguire e visualizzare i risultati di diverse query SQL predefinite, come descritto nella Sezione 5 del progetto. All'avvio, il programma presenta un'interfaccia testuale interattiva con un menu numerato che elenca le interrogazioni disponibili. L'utente può selezionare la query desiderata digitando il numero corrispondente.

Un aspetto centrale del programma è la sua modularità e flessibilità. Le query non sono scritte direttamente nel codice sorgente, ma vengono lette da un file esterno (Crociere.sql) insieme ai rispettivi titoli. Questo consente di modificare, aggiungere o rimuovere query SQL semplicemente aggiornando il file .sql, senza dover ricompilare il programma: il menu si aggiornerà automaticamente. Il programma supporta anche le query parametriche, in cui i parametri vengono indicati nel file SQL con una sintassi come . Quando l'utente seleziona una query di questo tipo, il programma richiede interattivamente i valori da sostituire per ciascun parametro. Ogni query può contenere fino a un massimo di 25 parametri. Inoltre, Query.c gestisce anche l'inizializzazione del database: all'avvio viene eseguito uno script SQL che crea e popola automaticamente tutte le tabelle necessarie. In sintesi, le interrogazioni SQL sono completamente separate dal codice sorgente e gestite esternamente: è quindi possibile modificarle dinamicamente aggiornando il file Crociere.sql, con effetto immediato sul menu delle scelte del programma.

Note:

Per facilitare la portabilità del programma è stato aggiunto un file config.txt esterno in cui è richiesto di inserire nome utente e password di postgreSQL che verrà utilizzato dal codice c.

In alcuni computer si può compilare direttamente con:

```
- gcc -o query Query.c -lpq
```

In altri non riesce a linkare automaticamente il pacchetto di postgresql quindi serve compilare con il codice:

```
- gcc -o query Query.c -I/usr/include/postgresql -lpq
```

Questo dipende se le variabili di sistema contengono già il percorso della libreria specificata o meno.