Park North-Italy s.r.l.

ABSTRACT

Park North-Italy s.r.l. è un'azienda che gestisce quattro parchi naturalistici nell'Italia Settentrionale. L'azienda utilizza un sistema informativo con lo scopo di tenere traccia e utilizzare le informazioni riguardanti i parchi.

Vengono raccolti dati riguardanti i clienti e i biglietti da loro acquistati, i dipendenti, le aree del parco, la flora e la fauna che le popolano e i vari servizi offerti dall'azienda.

Il cliente che vuole accedere al parco può acquistare un biglietto di tipo Evento o di tipo Tour guidato inserendo i suoi dati personali, con la possibilità di pagare un prezzo ridotto per i ragazzi e le ragazze di età inferiore ai 18 anni.

Il programma che gestisce il database dell'azienda fornisce la possibilità di visualizzare le informazioni più rilevanti delle varie entità sopracitate e di ricercare dati peculiari che potrebbero essere d'interesse all'azienda stessa.

ANALISI DEI REQUISITI

Requisiti delle entità:

Nel database sono presenti i dati suddivisi per entità:

Di ogni **parco** si conosce:

- Nome (univoco)
- Paese (Luogo in cui è localizzato)
- Dimensione del parco

Ogni parco è suddiviso in diverse **aree** di cui sono noti:

- ID (univoco)
- Nome
- Posizione
- Parco in cui si trova
- Dipendente responsabile dell'area

All'interno dei parchi vivono vari animali e piante tutti accomunati dall'avere:

- ID (univoco)
- Descrizione
- Età
- Area in cui si trova

In particolare della **pianta** si posseggono:

- Nome latino
- Fabbisogno Idrico (ovvero l'acqua necessaria giornaliera)
- Data in cui è stata piantata
- Tipo di foglie
- Locazione specifica nel parco

Dell'**animale**, inoltre, si vuole sapere:

- Nome dell'animale
- Data di nascita
- Tipo di alimentazione
- Classe animale di appartenenza
- Famiglia animale di appartenenza

Al parco, dei **clienti** che accedono si vuole conoscere:

- Codice fiscale (o Social Security Number)
- Nome
- Cognome
- Data di nascita

I clienti acquistano i **biglietti** che contengono:

- ID (univoco)
- Informazioni che specificano se è un biglietto ridotto o no
- Data di acquisto
- Prezzo di acquisto
- Codice fiscale dell'acquirente
- Parco alla quale si accede

Come anticipato i biglietti risultano essere del tipo ridotto se l'acquirente ha età inferiore ai 18 anni.

I biglietti poi sono suddivisi in **Biglietto Evento** e **Biglietto Tour** entrambi con un campo specifico Evento per i primi e Tour per i secondi dove viene specificato a che evento o a che tour si può partecipare.

Ogni **evento** ha:

- ID (univoco)
- Numero di Persone massime che possono partecipare
- Data
- Tipo di evento (Musicale o Naturalistico ecc.)
- Dipendente Organizzatore (unico)

Ogni **tour** invece:

- ID (univoco)
- Numero di Persone massime che possono partecipare
- Data
- Tipo di tour (Flora o Fauna)
- Dipendente Guida che spiega nel tour

Per finire, per ogni **dipendente** si memorizza:

- ID (univoco)
- Nome
- Cognome
- Data di nascita
- Guadagno
- Parco in cui lavora

Ci sono diversi tipi di dipendente: l'organizzatore che organizza gli eventi, il responsabile che controlla e gestisce una o più aree del parco, il ricercatore, la guida che accompagna i visitatori nei tour, la manutenzione che svolge lavori pratici per il mantenimento del parco e la sicurezza che mantiene al sicuro e in ordine il parco. Di ognuno si vuole sapere rispettivamente l'evento che organizza, le aree gestite, le specializzazioni conseguite, le lingue parlate, i compiti specifici che svolge e i corsi di sicurezza frequentati.

PROGETTAZIONE CONCETTUALE

Lista delle entità:

Se non specificato l'attributo è NOT NULL

- Parco:
    - nome: VARCHAR(255) PRIMARY KEY
    - luogo: VARCHAR(255)
    - dimensione: DOUBLE PRECISION, può essere NULL

- Area:
    - id: SERIAL PRIMARY KEY
    - nome: VARCHAR(255)
    - posizione: VARCHAR(255)
    - parco: VARCHAR(255)
    - responsabile: INT

- EsseriViventi:
    - id: VARCHAR(255) PRIMARY KEY
    - descrizione: VARCHAR(255), può essere NULL
    - nomeLatino: VARCHAR(255) età: SMALL INT
    - età: SMALL INT
    - area: INT

  EsseriViventi si suddivide in:
    - Pianta
        - acquaNecessaria: DOUBLE PRECISION
        - locazione: VARCHAR(255)
        - tipoFoglie: enumFoglie
        - dataTrapianto: TIMESTAMP

      enumFoglie = { 'Filiformi', 'Aghiformi', 'Oblunghe', 'Ovali', 'Seghettate' }

    - Animale:
        - alimentazione: enumAlimentazione
        - dataNascita: TIMESTAMP
        - classe: enumClasseAnimale
        - famiglia: VARCHAR(255)

      enumAlimentazione = { ‘Carnivoro’, ‘Erbivoro’, ‘Onnivoro’ }

      enumClasseAnimale = { ‘Mammifero’, ‘Pesce’, ‘Uccello’, ‘Rettile’, ‘Anfibio’, ‘Porifero’, ‘Celenterato’, ‘Artropode’, ‘Mollusco’, ‘Verme’, ‘Echinoderma’ }

- Cliente:
    - codiceFiscale: VARCHAR(255) PRIMARY KEY
    - nome: VARCHAR(255)
    - cognome: VARCHAR(255)
    - dataNascita: TIMESTAMP, può essere NULL

- Biglietto:
    - id: SERIAL PRIMARY KEY
    - ridotto: BOOLEAN
    - data: TIMESTAMP
    - prezzo: DOUBLE PRECISION, > 0
    - codiceFiscaleCliente: VARCHAR(255)
    - parco: VARCHAR(255)

  Biglietto è generalizzazione di:
    - BigliettoEvento:
        - evento: VARCHAR(255)
    - BigliettoTour:
        - tour: VARCHAR(255)

- Evento:
    - id: VARCHAR(255) PRIMARY KEY
    - numeroPersone: INT
    - data: TIMESTAMP
    - tipoEvento: enumEvento
    - organizzatore: INT

  enumEvento = { ‘Musicale’, ‘Artistico’, ‘Naturalistico’ }

- TourGuidato:
    - id: VARCHAR(255) PRIMARY KEY
    - numeroPersone: INT
    - data: TIMESTAMP
    - tipoTour: enumTour
    - guida: INT

  enumTour = { ‘Flora’, ‘Fauna’ }

- Dipendente:
    - id: SERIAL PRIMARY KEY
    - nome: VARCHAR(255)
    - cognome: VARCHAR(255)
    - dataNascita: TIMESTAMP
    - guadagno: DOUBLE PRECISION
    - tipoDipendente: enumDipendente

  enumDipendente = { ‘Responsabile’, ‘Ricercatore’, ‘Guida’, ‘Manutenzione’, ‘Sicurezza’, ‘Organizzatore’ }

  Dipendente è generalizzazione di:
    - Responsabile
    - Ricercatore
        - specializzazione: VARCHAR(255)
    - Guida
        - linguaParlate: VARCHAR(255)
    - Manutenzione
        - compitoSpecifico: VARCHAR(255)
    - Sicurezza
        - corsiFrequentati: VARCHAR(255)
    - Organizzatore

Tabella delle relazioni:

RELAZIONE	ENTITÀ COINVOLTE	DESCRIZIONE
Suddiviso	Parco (1,N) Area (1,1)	Ogni parco è suddiviso in più aree (almeno una) mentre ogni area appartiene ad uno e un solo parco.
Lavora	Parco (1,N) Dipendente (1,1)	In ogni parco ci lavorano tanti dipendenti (almeno uno) mentre ogni dipendente lavora in uno e un solo parco.
Accedi	Parco (1,N) Biglietto (1,1)	Si può accedere ad ogni parco con diversi biglietti (almeno uno) mentre un biglietto permette l’accesso ad uno e un solo parco.
Vivono	Area (1,N) EsseriViventi (1,1)	In ogni area vivono numerosi esseri viventi (almeno uno) mentre un essere vivente vive in una e una sola area.
Gestita	Area (1,1) Dipendente (1,1)	Ogni area è gestita da uno e un solo dipendente responsabile e allo stesso modo ogni responsabile gestisce una e una sola area.

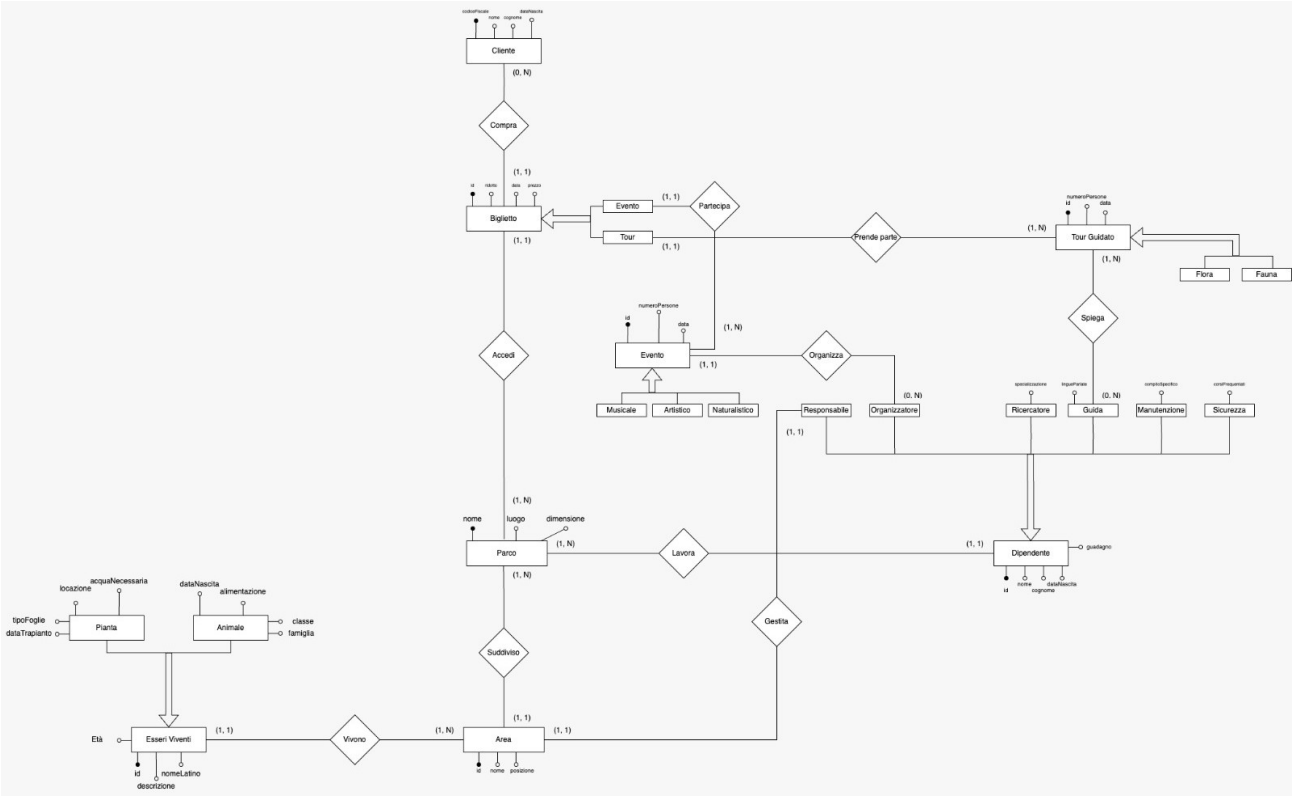
Compra	Cliente (0,N) Biglietto (1,1)	Ogni cliente può comprare dei biglietti (anche zero) mentre un biglietto può essere comprato da uno e un solo cliente.
Partecipa	Biglietto (1,1) Evento (1,N)	Si può partecipare ad ogni evento attraverso più biglietti (almeno uno) mentre un biglietto permette di partecipare ad uno e un solo evento.
Prende parte	Biglietto (1,1) TourGuidato (1,N)	Si può partecipare ad ogni tour guidato attraverso più biglietti (almeno uno) mentre un biglietto permette di partecipare ad uno e un solo tour guidato.
Spiega	TourGuidato (1,N) Dipendente (0,N)	Un tour guidato può essere spiegato da diverse guide (almeno una) e allo stesso modo una guida può spiegare diversi tour guidati (anche zero).
Organizza	Evento (1,1) Dipendente (0,N)	Un dipendente organizzatore può organizzare diversi eventi (anche zero) mentre un evento è organizzato da uno e un solo dipendente.

Vincoli:

Assumiamo come vincolo non esprimibile tramite SQL che:

- Un biglietto è di tipo ridotto se il cliente che lo compra ha un'età inferiore ai 18 anni;
- Il dipendente referenziato nell'attributo "responsabile" nella tabella Area sia effettivamente un dipendente di tipo Responsabile;
- Il dipendente referenziato nell'attributo "dipendente" nella tabella Spiega sia un dipendente di tipo Guida;
- Il dipendente referenziato nell'attributo "organizzatore" nella tabella Evento sia effettivamente un dipendente di tipo Organizzatore;
- La data di acquisto di un biglietto sia antecedente alla data dello svolgimento dell'evento o del tour per la quale è stato comprato;
- L'età di un animale corrisponda esattamente alla differenza tra la data attuale e la data di nascita dell'animale;
- Il numero di biglietti acquistati per un evento o per un tour sia inferiore al numero massimo di persone ammesse a quell'evento o a quel tour;

Diagramma ER:



PROGETTAZIONE LOGICA

Schema ER Ristrutturato:

È necessario ora ristrutturare il diagramma ER in modo tale che non ci siano generalizzazioni e che le entità possano essere utilizzate al meglio per le query che si andranno a creare.

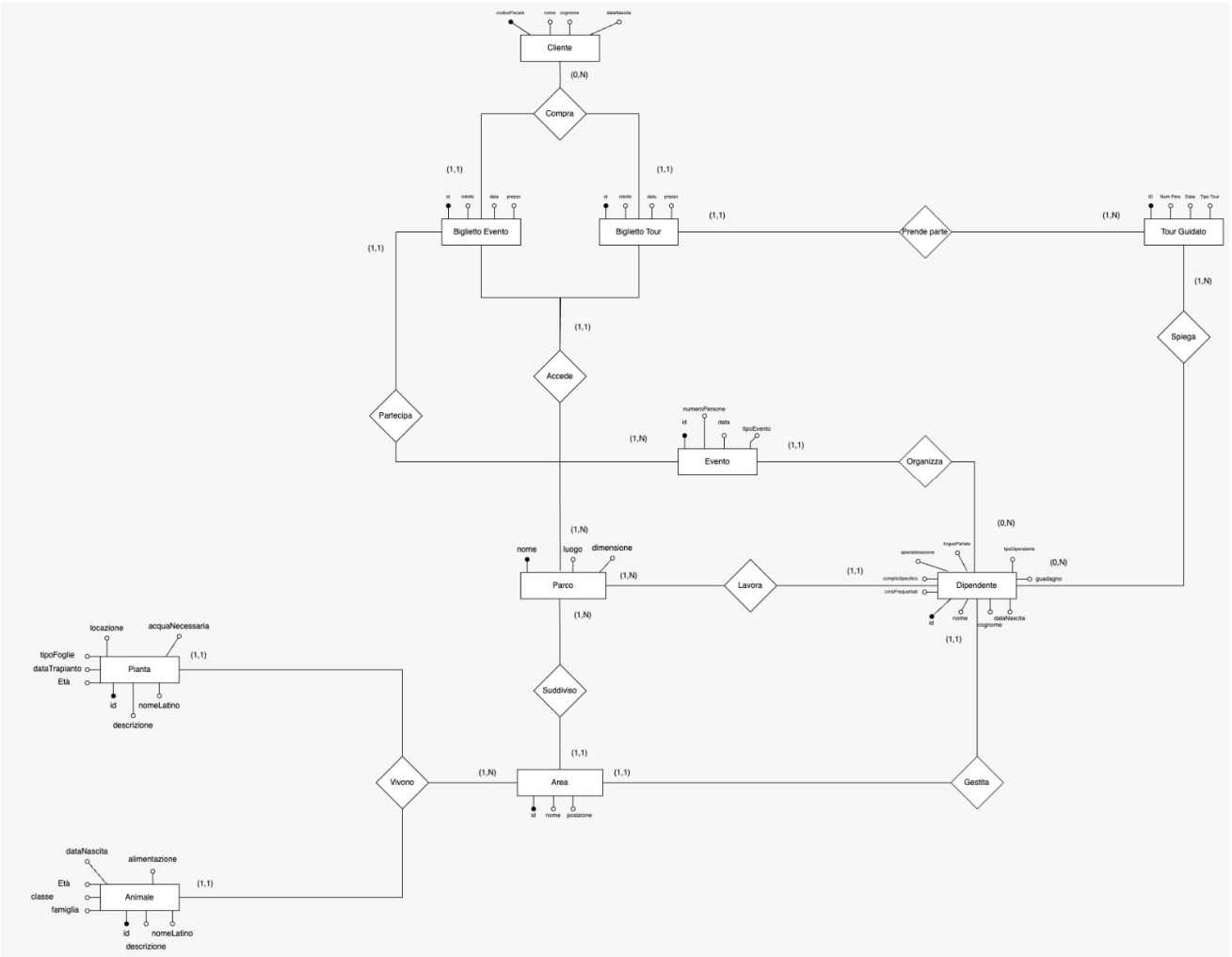
Essendo che nella generazione di query (riguardanti ad esempio i biglietti per gli eventi, i biglietti per i tour, i vari tipi di dipendenti, gli animali e le piante) ci sarebbero numerose ripetizioni dovuta alla ridondanza degli attributi, è una buona soluzione quella di creare entità indipendenti che possano gestire al meglio le ridondanze.

Nel caso degli esseri viventi la soluzione migliore è quella di creare due entità separate, ‘Animale’ e ‘Pianta’, che condividono gli attributi che ereditano da ‘EsseriViventi’ ma si differenziano con attributi specifici.

Anche nel caso dei biglietti la soluzione ottimale è quella di creare le entità ‘BigliettoEvento’ e ‘BigliettoTour’ che si differenziano referenziando l’ID del rispettivo evento o tour guidato.

Nel caso dei dipendenti, invece, una soluzione può essere il far collassare tutte le differenziazioni dei dipendenti in un’unica entità ‘Dipendente’ sotto forma di attributi. Ottengo un’entità con motlti attributi che mutualmente e in base al tipo sono NULL. Ad esempio, il Ricercatore ha un campo ‘specializzazione’ mentre tutti gli altri campi aggiunti sono NULL. La Guida, invece, ha un campo ‘linguеParlate’ mentre gli altri campi aggiunti, come il sopracitato ‘specializzazione’, sono NULL.

Il risultato è il seguente:



Schema relazionale:

Ne risulta quindi uno schema relazionale che dati gli attributi precedentemente elencati è organizzato nel seguente modo.

Parco (<u>nome</u>, luogo, dimensione)

Area (<u>id</u>, nome, posizione, parco, responsabile)

Area (parco) → Parco (nome)

Area (responsabile) → Dipendente (id)

Pianta (<u>id</u>, descrizione, nomeLatino, acquaNecessaria, locazione, tipoFoglie, dataTrapianto, età, area)

Pianta (area) → Area (id)

Animale (<u>id</u>, descrizione, nomeLatino, alimentazione, dataNascita, età, classe, famiglia, area)

Animale (area) → Area (id)

Cliente (<u>codiceFiscale</u>, nome, cognome, dataNascita)

BigliettoTour (id, ridotto, data, prezzo, codiceFiscaleCliente, tour, parco)

BigliettoTour(codiceFiscaleCliente) → Cliente (codiceFiscale)

BigliettoTour(tour) → TourGuidato (id)

BigliettoTour(parco) → Parco (id)

BigliettoEvento (id, ridotto, data, prezzo, codiceFiscaleCliente, evento, parco)

BigliettoEvento(codiceFiscaleCliente) → Cliente (codiceFiscale)

BigliettoEvento(Evento) → Evento (id)

BigliettoEvento(parco) → Parco (id)

Evento (id, numeroPersone, data, tipoEvento, organizzatore)

Evento (organizzatore) → Dipendente (id)

TourGuidato (id, numeroPersone, data, tipoTour)

Spiega (dipendente, tour)

Spiega (dipendente) → Dipendente (id)

Spiega (tour) → TourGuidato (id)

Dipendente (id, nome, cognome, dataNascita, guadagno, tipoDipendente, linguaParlate, specializzazione,

compitoSpecifico, corsiFrequentati, parco)

Dipendente (parco) → Parco (id)

QUERY E INDICI

Query:

Le query possibili presenti nel codice eseguibile sono:

1) Elenco delle informazioni principali delle entità principali
   Nell'esempio le informazioni principali di tutti i parchi registrati:

```
SELECT nome, luogo
FROM Parco
```

```
nome                                luogo

Parco Naturale del Gran Paradiso    Piemonte
Parco Nazionale di Camponogara      Venezia
Riserva Naturale Misquilese         Mussolente
Parco Nazionale dello Stelvio       Lombardia
```

2) Conteggio del popolamento animale suddiviso per età:

```
SELECT età, COUNT(età)
FROM Animale
GROUP BY età
ORDER BY count ASC
```

età	count
16	1
2	1
30	1
14	2
5	2
9	2
20	2
22	2
23	3
4	3
3	3
10	4
19	4
21	4
11	4
7	4
6	4
8	5
12	5
1	6
17	6
15	6
18	6

3) Dieta di un animale del quale si è fornito l'ID
Nell'esempio l'alimentazione dell'animale con ID 'A16A4' cioè un Cacatua:

```
SELECT id, nomeLatino, alimentazione
FROM Animale
WHERE (id= $1::varchar) //IN QUESTO CASO 'A16A4'
```

id	nomelatino	alimentazione
A16A4	Cacatua tenuirostris	Onnivoro

4) Fabbisogno idrico di una pianta della quale si è fornito l'ID
Nell'esempio il fabbisogno idrico della pianta con ID 'A20P4':

```
SELECT id, nomeLatino, acquaNecessaria
FROM Pianta
WHERE (id= $1::varchar) //IN QUESTO CASO 'A20P4'
```

id	nomelatino	acquanecessaria
A20P4	Bacopa repens	7.7

5) Posizione, area, parco in cui si trova una pianta della quale si è fornito l'ID
Nell'esempio la posizione della pianta con ID 'A20P4':

```
SELECT p.id, p.nomeLatino, p.locazione, p.area, a.parco
FROM Pianta AS p, Area AS a
WHERE (p.area = a.id AND p.id = $1::varchar) //IN QUESTO CASO 'A20P4'
```

id	nomelatino	locazione	area	parco
A20P4	Bacopa repens	Laces	20	Parco Nazionale dello Stelvio

6) Conteggio delle piante con un dato tipo di foglie
   Nell'esempio le piante con foglie 'Oblunghe':

```
SELECT tipoFoglie, COUNT(id) AS Numero_Piante
FROM Pianta
WHERE (tipoFoglie = $1::enumFoglie) //IN QUESTO CASO 'Oblunghe'
GROUP BY tipoFoglie
```

tipofoglie	numero_piante
Oblunghe	10

7) Lista dei clienti che partecipano ad un evento dato l'ID del dipendente che lo organizza
   Nell'esempio i clienti che partecipano all'evento organizzato dal dipendente '7':

```
SELECT c.nome, c.cognome
FROM Cliente AS c, BigliettoEvento AS b, Evento AS e
WHERE (c.codiceFiscale = b.codiceFiscaleCliente AND b.evento = e.id AND
e.organizzatore = $1::int) //IN QUESTO CASO '7'
```

nome	cognome
Flynn	Hrihorovich
Dalton	Harewood
Isaac	Dogerty
Gage	Jeaffreson
Marguerite	McGirl
Joshuah	Dolman
Alexander	Cawley
Rolland	Matthew

8) Dipendenti che fanno da guida di un dato tipo di tour guidato
   Nell'esempio le guide dei tour di tipo 'Fauna':

```
SELECT DISTINCT d.id, d.nome, d.cognome
FROM Dipendente AS d, Spiega AS s, TourGuidato AS t
WHERE (d.id = s.dipendente AND s.tour = t.id AND t.tipoTour = $1::enumTour)
//IN QUESTO CASO 'Fauna'
```

id	nome	tipodipendente	guadagno
18	Kerby	Responsabile	4886
37	Linoel	Responsabile	4882
16	Harvey	Guida	4527

9) Dipendenti i cui stipendi sono superiori ad un dato stipendio, in ordine decrescente
   Nell'esempio i dipendenti i cui stipendi sono superiori a '4500':

```
SELECT id, nome, tipoDipendente, guadagno
FROM Dipendente WHERE (guadagno > $1::int) //IN QUESTO CASO '4500'
ORDER BY guadagno DESC
```

ridotti
45

10) Conteggio dei biglietti (evento e tour) ridotti:

```
SELECT COUNT(*)AS Ridotti
FROM (SELECT id FROM BigliettoEvento
WHERE ridotto = true
UNION
SELECT id
FROM BigliettoTour
WHERE ridotto = true) AS r
```

id	nomelatino	locazione
A10P1	Hymenopappus carrizoanus	Biassa
A12P2	Acarospora smaragdula	Fiastra
A13P4	Actinidia polygama	Preci
A17P1	Lecidea sublimosa	Prato allo Stelvio
A19P1	Saxifraga hirculus	Vermiglio

11) Piante piantate precendentemente ad una specifica data fornita
Nell'esempio le piante piantate prima del '2001-01-01':

```
SELECT id, nomeLatino, locazione
FROM Pianta
WHERE (dataTrapianto < $1::timestamp) //IN QUESTO CASO '2001-01-01'
```

id	nome	cognome
2	Avram	McGarahan
5	Dyan	Itzkowicz
11	Jenica	Halladay
16	Harvey	Carlton
23	Jessamine	Meek
26	Hugibert	Batkin
31	Iggie	Stitt
33	Silas	Cleghorn

12) Nome latino di un'animale o di una pianta fornito un ID
Nell'esempio il nome latino dell'ID 'A16A4':

```
SELECT nomeLatino
FROM Animale
WHERE (Animale.id = $1::varchar) //IN QUESTO CASO 'A16A4' - VUOTO
UNION
SELECT nomeLatino
FROM Pianta
WHERE (Pianta.id = $1::varchar) //IN QUESTO CASO 'A16A4' - NON VUOTO
```

nomelatino
Cacatua tenuirostris

13) Media degli stipendi dei dipendenti di un dato parco
Nell'esempio la media degli stipendi del parco 'Parco Nazionale dello Stelvio':

```
SELECT parco, AVG(guadagno)
FROM Dipendente
GROUP BY parco
HAVING (parco = $1::varchar) //IN QUESTO CASO 'Parco Nazionale dello Stelvio'
```

parco	avg
Parco Nazionale dello Stelvio	2984.9

14) Totale dei soldi guadagnati dai biglietti suddivisi per tipo e somma totale guadagnata, dato un parco
Nell'esmepio i soldi guadagati dai biglietti suddivisi per tipo e somma totale guadagnata del parco 'Parco Nazionale dello Stelvio':

```
SELECT tour, SUM(prezzo) AS somma
FROM bigliettoTour
WHERE (parco = $1::varchar) //IN QUESTO CASO 'Parco Nazionale dello Stelvio'
GROUP BY prezzo, tour
ORDER BY somma DESC
```

```
SELECT evento, SUM(prezzo) AS somma
FROM bigliettoEvento
WHERE (parco = $1::varchar) //IN QUESTO CASO 'Parco Nazionale dello Stelvio'
GROUP BY prezzo, evento ORDER BY somma DESC
```

```
SELECT SUM(prezzo) AS totale
FROM (SELECT prezzo
      FROM BigliettoTour
      WHERE parco = $1::varchar //IN QUESTO CASO 'Parco Nazionale dello Stelvio'
      UNION ALL
      SELECT prezzo
      FROM BigliettoEvento
      WHERE parco = $1::varchar) AS u
```

Tour Guidati:

tour	somma
A19BC	58.36
A19BC	51.4
A18BC	50
A19BC	20.62
A18BC	20.62
A19BC	19.15
A19BC	18.76
A18BC	18.64
A19BC	10

Eventi:

evento	somma
8	28.42
6	28.42
6	20
4	20
8	20
8	17.52
6	17.52
4	17.44
8	16.41
4	16.41
4	13.3
5	10

Totale:

totale
492.99000000000007

Indici:

```
CREATE UNIQUE INDEX indicePianta ON Pianta(id, descrizione, nomeLatino,
acquaNecessaria, locazione);

CREATE UNIQUE INDEX indiceAnimale ON Animale(id, descrizione, nomeLatino,
alimentazione, dataNascita);
```

L'indice "indicePianta" è un indice univoco creato sulla tabella "Pianta". Questo indice viene definito su cinque colonne: "id", "descrizione", "nomeLatino", "acquaNecessaria" e "locazione".

La scelta di utilizzare un indice univoco implica che i valori in queste colonne saranno univoci per ogni riga della tabella. Pertanto, non saranno consentiti duplicati di combinazioni di valori in queste colonne.

L'indice "indiceAnimale" è un indice univoco creato sulla tabella "Animale". Questo indice è stato definito su cinque colonne: "id", "descrizione", "nomeLatino", "alimentazione" e "dataNascita".

L'obiettivo principale di questo indice è garantire l'unicità dei dati nella tabella "Animale" in base alla combinazione di valori presenti in queste cinque colonne. Ciò significa che non saranno consentiti duplicati di combinazioni di valori in queste colonne.

L'utilità di questo indice è facilitare l'individuazione e la selezione rapida di animali specifici all'interno della tabella in base alle loro caratteristiche come la descrizione, il nome latino, l'alimentazione e la data di nascita.

C++

Descrizio e utilizzo del codice:

Il codice C++ per l'esecuzione delle query consiste in un file **main.cpp**, dove è contenuto il corpo principale del codice, un file header **progetto.h**, dove sono contenute le dichiarazioni delle funzioni utilizzabili e un file **progetto.cpp** dove sono presenti le vere e proprie funzioni e query. Il tutto va compilato attraverso il comando:

g++ main.cpp progetto.cpp -Idependencies/include -Ldependencies/lib -lpq -o codice

Prima di poter compilare il file è necessario copiare i file **libpq.dll** e **libpq.lib** in "./dependencies/lib" e copiare **libpq-fe.h**, **pg_consig_ext.h** e **postgres_ext.h** in "./dependencies/include", nella directory che contiene i file .cpp.

Per eseguire il codice, basta avviare l'eseguibile "codice".

Verrà mostrato a schermo la possibilità di scegliere se visualizzare la lista delle query possibili (1), eseguire una query (2) o uscire dal programma (3).

Per eseguire una query bisogna inserire da tastiera il numero della query scelta e, inoltre, alcune query richiedono l'inserimento di alcuni parametri da parte dell'utente.

Documentazione del codice:

Nel codice vengono utilizzate delle funzioni create appositamente per il funzionamento del codice e alcune funzioni presenti nelle librerie sopracitate come:

```
PGconn *connection(char *conninfo);
```
che stabilisce la connessione con il database attraverso la funzione **PQConnectdb**,

```
void printIntestazione(int campi, PGresult *res);
```
che stampa a schermo l'intestazione dei campi del risultato della query con **PQfname**,

```
void printValue(int tuple, int campi, PGresult *res);
```
che stampa i valori del risultato della query con **PQgetvalue**,

```
void checkResults(PGresult *res, const PGconn *conn);
```
che controlla che il risultato sia valido con **PQresultStatus**,

```
void checkPrint(const PGconn *conn, PGresult *res);
```
che unisce printIntestazione, printValue, checkResults e **PQclear** per controllare, stampare e resettare il risultato di una query.

Infine si termina la connessione con **PQfinish**.