Università degli Studi di Cagliari
Corso di Laurea Magistrale in Ing. per l'Ambiente e il Territorio

Laboratorio d'Informatica
A.A. 2019/2020
Docente: Giorgio Fumera

Basi di dati – Esercizi

Gli esercizi riportati di seguito riguardano la progettazione concettuale e logica di basi di dati a partire dalla descrizione sintetica dei requisiti applicativi, e la formulazione di interrogazioni. Questi esercizi sono analoghi a quelli che verranno proposti nella prova d'esame.

1 Progettazione di basi di dati

Gli esercizi di questa sezione richiedono la progettazione concettuale e logica di una base di dati a partire dalla descrizione sintetica dei requisiti applicativi. In particolare si richiede:

- la definizione dello schema concettuale Entità-Relazione, indicando chiaramente il significato di ogni suo elemento, e in particolare la motivazione delle cardinalità delle relazioni e il dominio di ciascun attributo (per semplicità nei requisiti applicativi non viene riportata la dimensione massima delle sequenze di caratteri che compongono nomi di persone e di luoghi, indirizzi, ecc.: in questi casi si scelga un valore opportuno);
- la definizione dello schema logico relazionale, indicando in particolare l'origine di ciascuna tabella in riferimento allo schema Entità-Relazione;
- la scrittura dei comandi SQL per la creazione delle tabelle definite nello schema logico.

Si consiglia poi di creare le varie basi di dati con il DBMS PostgreSQL (tramite il comando `CREATE TABLE`), di inserire alcuni dati nelle varie tabelle (tramite l'interfaccia grafica fornita dal programma pgAdmin III, oppure con il comando SQL `INSERT INTO VALUES`), e di eseguire su di esse alcune interrogazioni.

I primi due esercizi corrispondono alle basi di dati *Segreteria studenti* e *Circolo nautico* usate come esempi durante le lezioni.

1. Un certo corso di laurea offre diversi insegnamenti che hanno un nome (per esempio, "Analisi Matematica 1"), un codice univoco di cinque caratteri assegnato dalla segreteria studenti, e un numero di crediti (uno o più). Al corso di laurea possono essere iscritti diversi studenti. Per ogni iscritto si vogliono memorizzare nome, cognome, codice fiscale (una sequenza di 16 caratteri), data di nascita e numero di matricola (una sequenza di cinque caratteri che identifica univocamente ogni studente, assegnata dalla segreteria studenti). Per ogni esame superato da ogni studente si vogliono conoscere il voto (un intero tra 18 e 30, oppure 30 e lode) e la data; un esame superato non può essere ripetuto.

2. Un certo circolo nautico ha diversi soci e possiede diverse barche che possono essere prenotate dai soci. Ogni velista socio del circolo ha un nome, un'età (un numero frazionario) e un livello di esperienza rappresentato da un numero intero tra 1 (esperienza minima) e 10 (esperienza massima); ogni socio deve essere maggiorenne (età minima 18 anni) ed è identificato univocamente da un numero di tessera (un intero) assegnato dalla direzione del circolo. Ogni barca ha un codice identificativo (un numero intero), un nome e un colore. Ogni velista può usare le barche messe a disposizione dal circolo nautico: ogni barca può essere prenotata per un'intera giornata da un solo velista, ma non può essere prenotata più volte in uno stesso giorno, neanche da parte di velisti diversi.

3. Si consideri ancora l'esercizio 1, e si modifichi il progetto della base di dati assumendo che si voglia memorizzare in essa due informazioni aggiuntive: il piano di studi personale di ciascuno studente (cioè l'insieme degli esami che devono essere sostenuti), e gli esami non superati (oltre a quelli

superati). Per gli esami non superati la valutazione non è costituita da un voto numerico, ma dal giudizio "insufficiente". Si assuma che uno studente non possa sostenere nello stesso giorno più esami per lo stesso insegnamento.

4. Si vuole progettare una base di dati per memorizzare le seguenti informazioni su ciascuno degli studenti iscritti ai corsi di studi di una certa facoltà universitaria istituita nel 1995: nome, cognome, data di nascita, codice fiscale, numero di matricola (una sequenza di nove caratteri che identifica univocamente ogni studente), e corso di studi al quale lo studente è iscritto; di ogni corso si vogliono memorizzare in particolare il nome, il livello (triennale, magistrale, a ciclo unico) la durata (numero di anni, compreso tra 3 e 6) e l'anno di istituzione. Ogni studente è iscritto a uno e un solo corso di studi, mentre in un certo istante un corso di studi può non avere iscritti; non possono esistere due corsi di studi dello stesso livello con lo stesso nome. L'anno di istituzione di un corso di studi non può essere precedente a quello della facoltà.

5. Si vuole progettare una base di dati per memorizzare le seguenti informazioni su ciascuno degli studenti iscritti agli atenei italiani: nome, cognome, data di nascita, numero di matricola (una sequenza di nove caratteri che identifica univocamente ogni studente all'interno di un dato ateneo), e ateneo al quale lo studente è iscritto; di ogni ateneo si vogliono memorizzare in particolare il nome (per esempio, "Università degli Studi di Cagliari"), la città in cui ha sede e l'anno di istituzione (si assume che l'ateneo italiano più antico sia quello di Bologna, la cui data di fondazione è per convenzione il 1088). Ogni studente è iscritto a uno e un solo ateneo, mentre in un certo istante un ateneo può non avere iscritti; non possono esistere due atenei con lo stesso nome.

6. Una certa banca ha diverse filiali distribuite nel territorio. Ciascuna filiale ha un indirizzo (città, via, numero civico e CAP, quest'ultimo formato da cinque cifre) e un certo numero di clienti titolari di conti correnti. In una stessa città possono esserci più filiali, ma in questo caso ognuna ha un indirizzo diverso. Ogni cliente ha nome, cognome, codice fiscale e luogo di residenza (città, via, numero civico e CAP), ed è titolare di uno o più conti correnti, in una o più filiali. Ogni conto corrente è caratterizzato da un codice IBAN (consistente in Italia in una sequenza di 27 caratteri) che lo distingue univocamente, una data di apertura e un saldo (in Euro), e può avere un solo titolare.

7. Una certa casa discografica fondata nel 1970 produce dischi aventi un titolo, un codice identificativo univoco (tredici caratteri), e un anno d'incisione. Ogni disco contiene più brani, ognuno dei quali ha un titolo e una durata (in minuti e secondi); uno stesso disco non può contenere diversi brani con lo stesso titolo. Ogni disco può essere inciso da uno o più autori, ognuno dei quali ha un nome, un cognome, una data di nascita e un nome d'arte; si assuma che non esistano autori diversi con lo stesso nome d'arte.

8. Un'azienda ha diversi dipendenti ed è suddivisa in diversi dipartimenti. Ogni dipartimento ha un nome (diverso dagli altri) e può avere una o più sedi con un loro indirizzo (città, via, numero civico e CAP); ogni sede di uno stesso dipartimento si trova in una diversa città. Ciascuna sede di ogni dipartimento è diretta da uno dei dipendenti dell'azienda, a partire da una certa data; uno stesso dipendente può dirigere al più una sede di dipartimento. L'azienda gestisce inoltre un insieme di progetti: ogni progetto ha un nome e un codice numerico univoco assegnati dall'azienda, e viene coordinato da una delle sedi di un singolo dipartimento; su ogni progetto lavorano uno o più dipendenti, e ogni sede di ogni dipartimento gestisce almeno un progetto. Ogni dipendente ha nome, cognome, sesso, data di nascita, codice fiscale, indirizzo di residenza (città, via, numero civico e CAP), uno stipendio annuale (espresso in Euro), e afferisce a uno e un solo dipartimento. Ogni dipendente, eccetto i direttori dei dipartimenti, ha un supervisore (un altro dipendente); uno stesso dipendente può essere supervisore di più dipendenti. Ogni dipendente lavora su uno o più progetti, a ciascuno dei quali dedica un certo numero di ore settimanali.

2 Formulazione di interrogazioni

Le interrogazioni riportate in questa sezione sono riferite alle basi di dati degli esercizi 1 (*Segreteria studenti*) e 2 (*Circolo nautico*) della sezione 1. Si suggerisce di eseguire le interrogazioni sul DBMS PostgreSQL su istanze opportune delle due basi di dati, come quelle disponibili nel sito web del laboratorio.

1. Con riferimento alla base di dati dell'esercizio 1 della sezione 1 (*Segreteria studenti*), formulare e scrivere in linguaggio SQL le seguenti interrogazioni:

   (a) Trovare nomi e cognomi degli studenti nati nel 1991

   (b) Trovare le matricole degli studenti che hanno conseguito la votazione di 30 in qualche esame (estrarre una sola occorrenza per ciascuno di tali numeri di matricola)

   (c) Trovare i cognomi degli studenti che hanno sostenuto l'esame avente codice `AMI01`

   (d) Trovare i cognomi degli studenti che hanno sostenuto un esame denominato "Analisi I"

   (e) Trovare i cognomi degli studenti che hanno sostenuto l'esame `AMI01`, mediante interrogazioni nidificate

   (f) Trovare i nomi degli esami in cui qualche studente abbia conseguito la votazione di 30, mediante interrogazioni nidificate

   (g) Trovare i cognomi degli studenti più giovani, mediante operatori insiemistici e interrogazioni nidificate

   (h) Trovare le matricole degli studenti che non hanno ottenuto il voto più basso tra quelli conseguiti negli esami dell'insegnamento `AMI01`, mediante operatori insiemistici e interrogazioni nidificate (in altre parole, si è interessati agli studenti che hanno ottenuto nell'esame `AMI01` un voto maggiore rispetto a quello conseguito da qualche altro studente nello stesso esame)

   (i) Trovare i cognomi degli studenti più giovani, mediante operatori di aggregazione

   (j) Trovare le matricole degli studenti che non hanno ottenuto il voto più basso tra quelli conseguiti negli esami dell'insegnamento `AMI01`, mediante operatori di aggregazione

   (k) Trovare le matricole degli studenti che hanno ottenuto il voto più alto nell'esame `AMI01`, usando interrogazioni nidificate e: (1) operatori insiemistici; (2) operatori di aggregazione

2. Con riferimento alla base di dati dell'esercizio 2 della sezione 1 (*Circolo nautico*), formulare e scrivere in linguaggio SQL le seguenti interrogazioni:

   (a) Trovare il codice e il nome di tutti i velisti

   (b) Estrarre i dati di tutte le barche

   (c) Trovare i nomi dei velisti con più di 40 anni

   (d) Trovare i nomi dei velisti con più di 40 anni, oppure con esperienza maggiore di 7

   (e) Trovare i codici delle barche prenotate l'11 dicembre 2018

   (f) Trovare i colori di tutte le barche (estrarre una sola occorrenza di ciascun colore)

   (g) Trovare i nomi di tutte le barche che non siano rosse

   (h) Trovare i nomi dei velisti che hanno prenotato la barca numero 103

   (i) Trovare i colori delle barche prenotate da qualche velista di nome Rusty

   (j) Trovare i codici dei velisti che hanno prenotato qualche barca blu, oppure qualche barca verde

   (k) Trovare i codici dei velisti che hanno prenotato sia qualche barca blu che qualche barca verde

   (l) Trovare i codici dei velisti che hanno prenotato qualche barca blu, ma non barche verdi

   (m) Trovare i codici delle barche che siano state prenotate da qualche velista con esperienza pari almeno a 8, e da nessun velista con età maggiore di 35 anni

   (n) Trovare i nomi dei velisti che hanno prenotato la barca numero 103, mediante interrogazioni nidificate

   (o) Trovare i nomi dei velisti che **non** hanno prenotato la barca numero 103, mediante interrogazioni nidificate

   (p) Estrarre tutti i dati dei velisti la cui esperienza sia maggiore di quella di qualche velista di nome Lubber, mediante operatori insiemistici e interrogazioni nidificate

   (q) Trovare i codici dei velisti più esperti, mediante operatori insiemistici e interrogazioni nidificate

   (r) Trovare nome ed età dei velisti più anziani, mediante operatori insiemistici e interrogazioni nidificate

(s) Trovare il numero dei velisti soci del circolo nautico

(t) Trovare l'età media di tutti i velisti

(u) Trovare il numero di valori distinti di esperienza dei velisti

(v) Trovare l'età media dei velisti con esperienza pari a 10

(w) Trovare nome ed età dei velisti più anziani, mediante interrogazioni nidificate scritte usando: (1) operatori insiemistici; (2) operatori di aggregazione

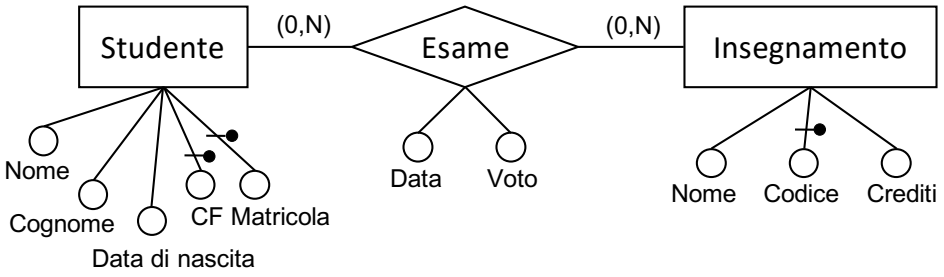
(x) Trovare i nomi dei velisti più esperti, usando operatori di aggregazione

Soluzioni

1 Progettazione di basi di dati

La descrizione degli schemi concettuale e logico viene riportata per esteso solo per il primo esercizio. Per tutti gli altri esercizi vengono riportati solo i dettagli più rilevanti.

1. Lo **schema Entità-Relazione** è il seguente:



Le entità `Studente` e `Insegnamento` rappresentano rispettivamente l'insieme di studenti iscritti al corso di studi e l'insieme di insegnamenti attivi, in un certo istante di tempo. I domini degli attributi sono: un intero positivo per il numero di crediti di un insegnamento, una data per la data di nascita di ciascuno studente, e una stringa di un numero opportuno di caratteri per tutti gli altri attributi (in particolare, 16 caratteri per il codice fiscale e 5 per il numero di matricola degli studenti e il codice degli insegnamenti).

Dai requisiti applicativi si evince che il numero di matricola è un identificatore dell'entità `Studente`, e il codice dell'insegnamento è un identificatore di `Insegnamento`. L'entità `Studente` ha anche un altro identificatore, il codice fiscale, poiché non possono esistere due persone con lo stesso codice fiscale (vincolo generale, e per questo non riportato esplicitamente nei requisiti applicativi).

Le informazioni sugli esami fanno riferimento a entrambe le entità, dato che ogni esame è sostenuto da un certo studente per un certo insegnamento (in una certa data e con un certo voto). Inoltre, poiché si vogliono memorizzare nella base di dati solo gli esami superati, e nessuno studente può ripetere l'esame di uno stesso insegnamento (se già sostenuto con esito positivo), le informazioni sugli esami possono essere rappresentate come una relazione (`Esame`) tra le due entità `Studente` e `Insegnamento`: un dato studente è in relazione con un dato insegnamento se ne ha superato l'esame. La data e il voto dell'esame saranno quindi attributi della relazione `Esame`; il loro dominio sarà rispettivamente un intero tra 18 e 30 oppure "30 e lode", e una data.

La cardinalità della relazione `Esame` è $(0, N)$ rispetto a entrambe le entità coinvolte, poiché in un certo istante uno studente può aver superato nessuno, uno o più esami, e l'esame di un certo insegnamento può essere stato superato da nessuno, uno o più studenti.

In base alle regole per la definizione dello **schema logico relazionale** a partire dallo schema Entità-Relazione, e in particolare tenendo conto che la relazione `Esame` è di tipo molti-a-molti, si ricava facilmente che lo schema logico sarà composto da tre tabelle corrispondenti alle due entità e alla relazione; a tali tabelle si possono assegnare nomi corrispondenti a quelli dell'entità o relazione da cui derivano (per convenzione il nome delle tabelle è al plurale): `Studenti`, `Insegnamenti`, `Esami`.

Come chiave primaria della tabella `Studenti`, tra le colonne `CF` e `Matricola` (corrispondenti ai due attributi identificatori dell'entità `Studente`) si può scegliere la seconda, tenendo conto del ruolo del numero di matricola in un corso di studi. È allora opportuno stabilire un vincolo di unicità sulla colonna `CF`. La chiave primaria di `Insegnamenti` corrisponderà invece all'unico identificatore dell'entità `Insegnamento`, cioè il codice dell'esame.

La tabella `Esami` conterrà, oltre alla data e al voto di un esame, anche le colonne corrispondenti alle chiavi primarie di `Studenti` e `Insegnamenti`, che ne costituiranno anche la chiave primaria (non possono esistere più occorrenze della relazione tra uno stesso studente e uno stesso insegnamento, corrispondenti a esami superati). Ciascuna di tali colonne dovrà avere lo stesso dominio della corrispondente colonna di `Studenti` e `Insegnamenti`, e un vincolo d'integrità referenziale con essa.

Riguardo ai domini delle colonne di ciascuna tabella, per le stringhe di caratteri si è scelta una lunghezza ragionevole quando non specificata dai requisiti applicativi. Per il numero di crediti di un insegnamento, al vincolo di dominio (numero intero) si è aggiunto un vincolo di dominio esteso (il valore deve essere positivo). Nel caso del voto si è scelto di usare due colonne: una per il valore numerico (anche in questo caso al vincolo di dominio si è aggiunto un vincolo di dominio esteso, per garantire che il valore sia compreso tra 18 e 30) e una per la presenza o assenza della lode; per quest'ultima si è scelto il dominio dei valori logici (booleani) "vero" e "falso". Queste scelte comportano la necessità di un vincolo di *tupla* che coinvolga le colonne del voto e della lode, per garantire che la lode non possa essere presente se il voto è inferiore a 30.

Lo schema logico si può descrivere in modo informale come segue (la chiave primaria di ciascuna tabella corrisponde alle colonne il cui nome è sottolineato):

`Studenti` (Matricola: stringa di 5 caratteri, CF: stringa di 16 caratteri, Cognome: stringa di 50 caratteri, Nome: stringa di 50 caratteri, Data di nascita: data)

`Insegnamenti` (Nome: stringa di 50 caratteri, Codice: stringa di 5 caratteri, Crediti: intero)

`Esami` (Data: data, Voto: intero, Lode: booleano, Studente: stringa di 5 caratteri, Esame: stringa di 5 caratteri)

Vincoli d'integrità referenziale (chiavi esterne):

- Studente (tabella Esami) verso Matricola (tabella Studenti)
- Esame (tabella Esami) verso Codice (tabella Insegnamento)

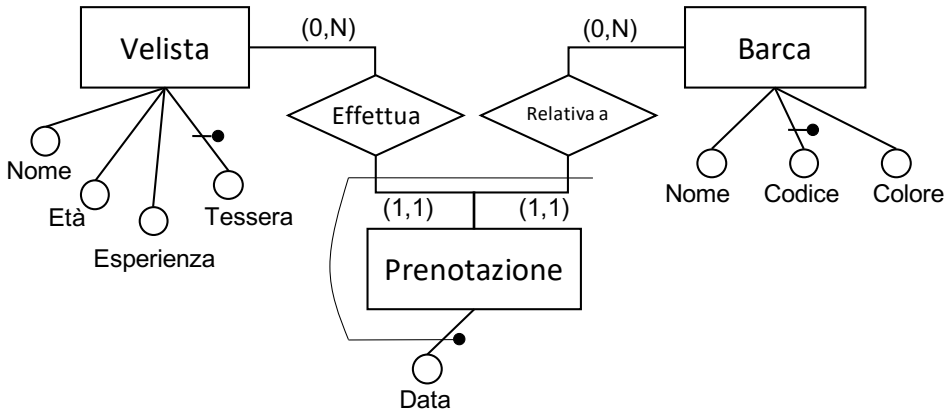
Vincoli di dominio esteso:

- il numero di crediti di ogni insegnamento deve essere positivo;
- nella tabella Esami il voto deve essere compreso tra 18 e 30 (estremi inclusi).

Vincolo di tupla: nella tabella Esami il voto non può essere minore di 30 se è presente la lode.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato: `SQL_segreteria_studenti.sql`.

2. Lo **schema Entità-Relazione** è il seguente:



Le informazioni sulle prenotazioni non possono essere rappresentate come una relazione tra le entità `Velista` e `Barca`, poiché uno stesso velista può prenotare più volte (in giorni diversi) una stessa barca: tale informazione non costituisce quindi un'associazione *binaria* tra velisti e barche. In altre parole, l'informazione che si richiede di rappresentare non consiste nel fatto che una certa barca sia stata prenotata o meno da un certo velista (informazione a due soli valori, "vero" e "falso"): essa consite invece in ogni istanza di una prenotazione di una barca da parte di un velista. Tali informazioni devono quindi essere rappresentate da un'entità (`Prenotazione`), che indica l'insieme di tutte le prenotazioni effettuate; più precisamente, ogni elemento di tale entità rappresenta una prenotazione eseguita da un certo velista in una certa data per una certa barca. Si noti che `Prenotazione` è un'entità debole, dato che il suo unico attributo (la data della prenotazione) non è ovviamente in grado di distinguere univocamente i suoi elementi. A questo scopo è necessario

affiancare alla data della prenotazione sia il velista che l'ha effettuata che la barca coinvolta, tramite gli identificatori delle entità corrispondenti.

Per quanto detto sopra sull'entità `Prenotazione`, le due relazioni hanno cardinalità $(1,1)$ rispetto a tale entità; la loro cardinalità rispetto a entrambe le entità `Velista` e `Barca` è invece $(0,N)$, dato che in un certo istante di tempo un velista potrebbe aver effettuato nessuna, una o più prenotazioni, e una barca potrebbe essere stata prenotata nessuna, una o più volte.

Dato che non sono presenti relazioni di tipo molti-a-molti, lo schema logico relazionale conterrà solo le tabelle corrispondenti alle tre entità.

Lo schema logico relazionale è il seguente (i nomi delle colonne sono quelli della base di dati usata come esempio a lezione):

`Velisti` (vid: intero, età: reale, esperienza: intero, vnome: stringa di 20 caratteri)

`Barche` (nome: stringa di 25 caratteri, bid: intero, colore: stringa di 10 caratteri)

`Prenotazioni` (data: data, vid: intero, bid: intero)

Vincoli d'integrità referenziale (chiavi esterne):

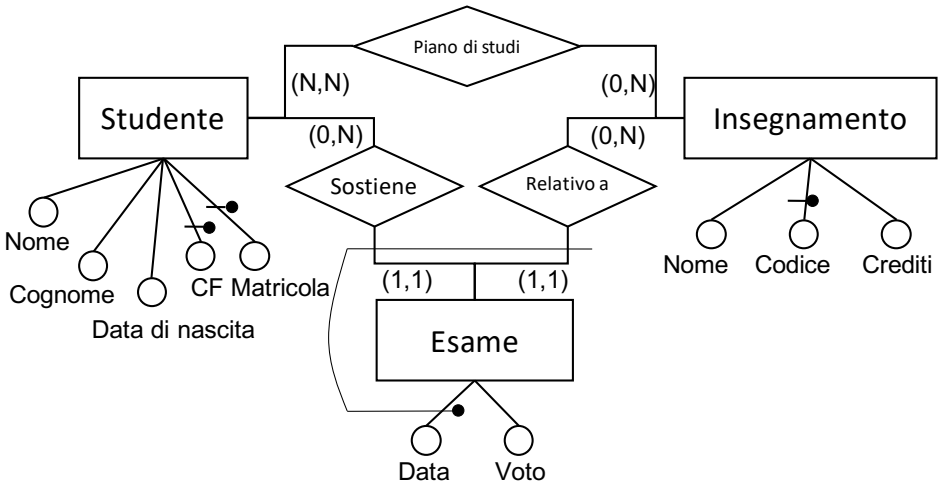
- vid (tabella Prenotazioni) verso vid (tabella Velisti)
- bid (tabella Prenotazioni) verso bid (tabella Barche)

Vincoli di dominio esteso:

- l'età di ogni velista deve essere non inferiore a 18;
- il numero di tessera di ogni velista deve essere positivo;
- l'esperienza di ogni velista deve essere compresa tra 1 e 10 (estremi inclusi);
- il codice di ogni barca deve essere positivo.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato `SQL_circolo_nautico.sql`.

3. Lo **schema Entità-Relazione** è il seguente:

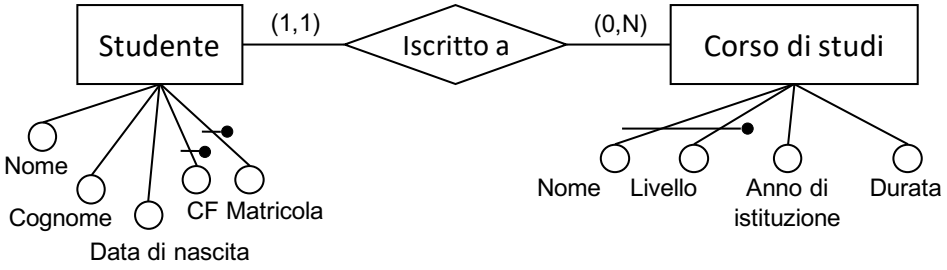


Le informazioni sul piano di studi possono essere rappresentate come una relazione tra le entità `Studente` e `Insegnamento`, dato che sono di natura binaria (un certo insegnamento può trovarsi o meno nel piano di studi di un certo studente). Quelle sugli esami sostenuti non possono invece essere rappresentate da una relazione tra le stesse entità, poiché in questo caso può esserci più di un'associazione tra uno stesso studente e uno stesso insegnamento (nel caso in cui uno stesso esame venga ripetuto più volte da uno stesso studente). È necessario dunque introdurre l'entità `Esame` (analoga all'entità `Prenotazione` dell'esercizio precedente): ogni suo elemento corrisponde a una prova d'esame per un certo insegnamento sostenuta da un certo studente in un certa data con un certo esito.

L'unica relazione molti-a-molti è `Piano di studi`: lo schema logico relazionale conterrà quindi quattro tabelle corrispondenti a tale relazione e alle tre entità.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato: `SQL_segreteria_studenti_2.sql`.

4. Lo **schema Entità-Relazione** è il seguente:

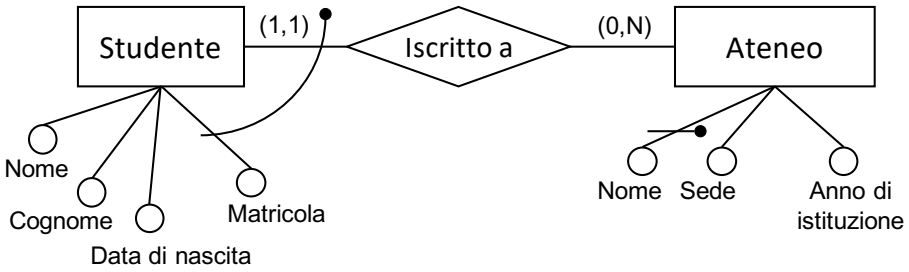


Si noti che l'identificatore dell'entità `Corso di studi` è composto dalla coppia di attributi `Nome` e `Livello`, poiché il solo nome di un corso non è sufficiente per distinguerlo univocamente dagli altri (si pensi ai corsi di studi triennale e magistrale in Ingegneria per l'Ambiente e il Territorio).

Lo schema logico relazionale comprenderà solo le tabelle associate alle due entità, dato che la relazione `Iscritto a` non è di tipo molti-a-molti.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato `SQL_facolta.sql`.

5. Lo **schema Entità-Relazione** è il seguente:

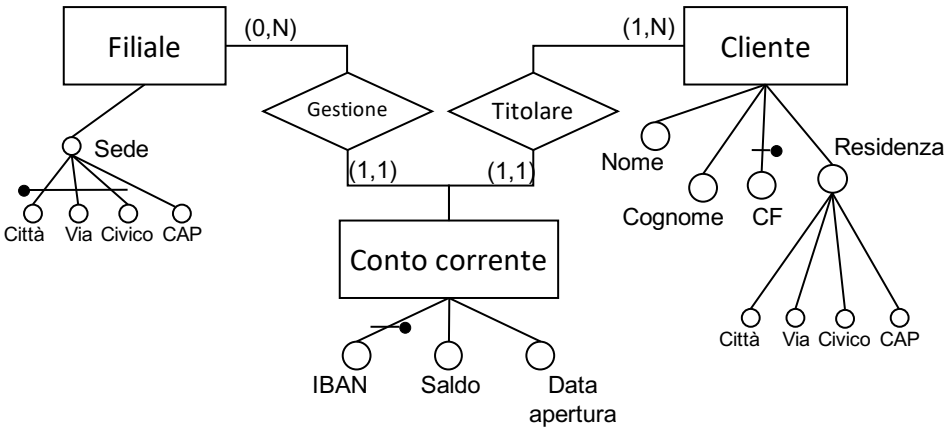


Si noti che in questo caso l'entità `Studente` è debole, dato che non si può escludere che studenti di atenei diversi abbiano numero di matricola identico. L'identificatore di `Studente` dovrà essere composto sia dal numero di matricola che dall'identificatore dell'entità `Ateneo`, poiché i numeri di matricola assegnati da un dato ateneo non possono ripetersi.

Si è scelto di codificare il livello dei corsi con le stringhe `L` (laurea), `LM` (laurea magistrale) e `CU` (corso a ciclo unico): si è quindi definito il dominio dell'attributo corrispondente come una stringa di due caratteri, e si è previsto un vincolo di dominio esteso per assicurare che gli unici valori ammessi siano le tre stringhe indicate sopra.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato `SQL_atenei.sql`.

6. Lo **schema Entità-Relazione** è il seguente:



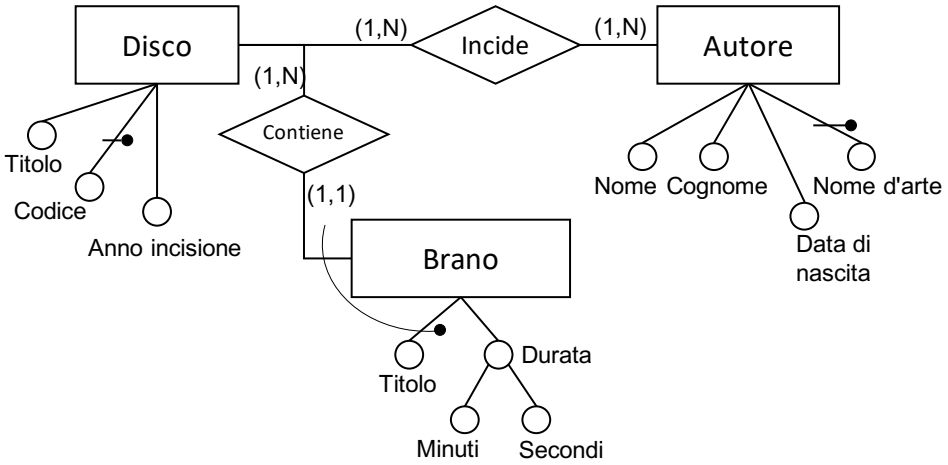
Si noti che non è necessario rappresentare esplicitamente le filiali presso le quali ogni cliente possiede conti correnti (questo si potrebbe fare attraverso una relazione tra le entità corrispondenti), dato che questa informazione può essere ricavata in modo indiretto attraverso le relazioni `Titolare` e `Gestione` e l'entità `Conto corrente`. Per esempio, come esercizio si scriva in linguaggio SQL la seguente interrogazione: trovare le sedi di tutte le filiali di cui sia cliente una certa persona avente codice fiscale `ABCXYZ12A34B567C`.

Gli attributi `Sede` e `Residenza` delle entità `Filiale` e `Cliente` sono composti. Nello schema logico compariranno solo le colonne corrispondenti agli attributi che li compongono.

Per il CAP si è scelto come dominio una stringa di cinque caratteri invece che un numero intero, per tener conto della presenza della cifra `0` come primo carattere. Si dovrebbe anche prevedere un vincolo di dominio esteso che assicuri che i cinque caratteri corrispondano a quelli dei CAP esistenti in Italia: per semplicità si trascura questo vincolo.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato `SQL_banca.sql`.

7. Lo **schema Entità-Relazione** è il seguente:

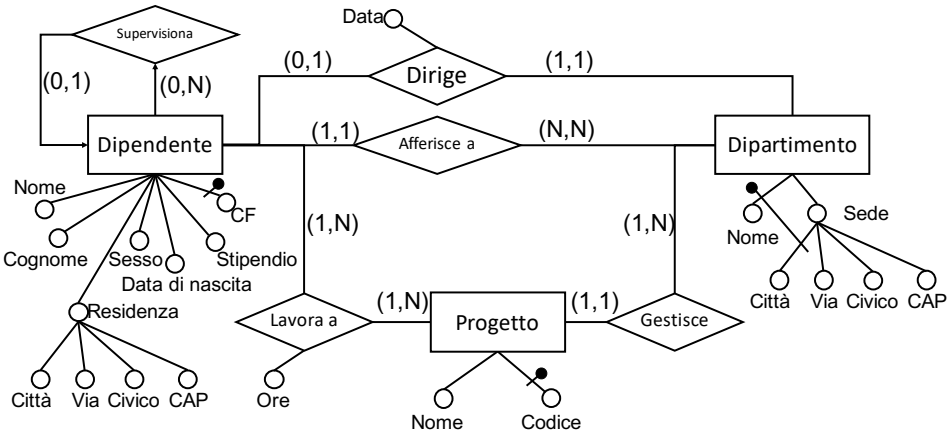


L'entità `Brano` è debole, e il suo identificatore può essere definito associando al titolo di un brano il disco (ovvero il codice del disco) a cui appartiene.

Si notino i vincoli di dominio esteso sulle colonne `Minuti` e `Secondi` della tabella `Brani` (per garantire che i loro valori siano compresi tra 0 e 59) e il vincolo di *tupla* su entrambe tali colonne per garantire che la durata di un brano non sia nulla.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato: `SQL_casa_discografica.sql`.

8. Lo **schema Entità-Relazione** è il seguente:



Si noti che la relazione `Supervisiona` è definita su coppie di elementi di una *stessa* entità (`Dipendente`). Le frecce sono state introdotte per chiarire, insieme al nome della relazione, il significato della cardinalità: un dipendente *supervisiona* nessuno, uno o più dipendenti (cardinalità $(0, N)$), e può *essere supervisionato* da nessuno o da un altro dipendente. Trattandosi di una relazione uno-a-molti, nello schema logico relazionale l'informazione da essa rappresentata (chi è il supervisore di un dato dipendente) sarà inserita nella tabella `Dipendenti` come una colonna aggiuntiva (`Supervisore`) che conterrà il codice fiscale del supervisore; su tale colonna sarà definito un vincolo d'integrità referenziale con la colonna `CF` della *stessa* tabella. Se un dipendente non ha un supervisore, il valore della colonna `Supervisore` sarà `NULL`.

Nello schema logico si è usato il nome `Dipendenti-Progetti` per la tabella corrispondente alla relazione molti-a-molti `Lavora a`.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato `SQL_azienda.sql`.

2 Formulazione di interrogazioni

1. Interrogazioni sulla base di dati dell'esercizio 1 della sezione 1 (*Segreteria studenti*):

   (a)
   ```
   SELECT "Nome", "Cognome" FROM "Studenti"
   WHERE "Data di nascita" >= '1991-01-01' AND "Data di nascita" <= '1991-12-31';
   ```

   (b)
   ```
   SELECT DISTINCT "Studente" FROM "Esami" WHERE "Voto" = 30;
   ```

   (c)
   ```
   SELECT "Studenti"."Cognome" FROM "Studenti", "Esami"
   WHERE "Esami"."Esame" = 'AMI01' AND "Studenti"."Matricola" = "Esami"."Studente";
   ```
   La stessa interrogazione scritta con variabili di *range*:
   ```
   SELECT S."Cognome" FROM "Studenti" S, "Esami" E
   WHERE E."Esame" = 'AMI01' AND S."Matricola" = E."Studente";
   ```

   (d)
   ```
   SELECT S."Cognome" FROM "Studenti" S, "Insegnamenti" I, "Esami" E
   WHERE I."Nome" = 'Analisi I' AND S."Matricola" = E."Studente"
         AND I."Codice" = E."Esame";
   ```

   (e)
   ```
   SELECT "Cognome" FROM "Studenti" WHERE "Matricola" IN
      (SELECT "Studente" FROM "Esami" WHERE "Esame" = 'AMI01');
   ```

   (f)
   ```
   SELECT "Nome" FROM "Insegnamenti" WHERE "Codice" IN
      (SELECT "Esame" FROM "Esami" WHERE "Voto" = 30);
   ```

   (g)
   ```
   SELECT "Cognome" FROM "Studenti" WHERE "Data di nascita" >= ALL
      (SELECT "Data di nascita" FROM "Studenti");
   ```

   (h)
   ```
   SELECT "Studente" FROM "Esami" WHERE "Esame"='AMI01' AND E."Voto" > ANY
      (SELECT "Voto" FROM "Esami" WHERE "Esame" = 'AMI01');
   ```

   (i)
   ```
   SELECT "Cognome" FROM "Studenti" WHERE "Data di nascita" =
      (SELECT MAX ("Data di nascita") FROM "Studenti");
   ```

   (j)
   ```
   SELECT "Studente" FROM "Esami" WHERE "Esame" = 'AMI01' AND "Voto" >
      (SELECT MIN ("Voto") FROM "Esami" WHERE "Esame" = 'AMI01');
   ```

   (k) Usando operatori insiemistici:
   ```
   SELECT "Studente" FROM "Esami" WHERE "Esame" = 'AMI01' AND "Voto" >= ALL
      (SELECT "Voto" FROM "Esami" WHERE "Esame" = 'AMI01');
   ```
   Usando operatori di aggregazione:
   ```
   SELECT "Studente" FROM "Esami" WHERE "Esame" = 'AMI01' AND "Voto" =
      (SELECT MAX ("Voto") FROM "Esami" WHERE "Esame" = 'AMI01');
   ```

2. Interrogazioni sulla base di dati dell'esercizio 2 della sezione 1 (*Circolo nautico*):

   (a)
   ```
   SELECT "vid", "vnome" FROM "Velisti";
   ```

   (b)
   ```
   SELECT * FROM "Barche";
   ```

   (c)
   ```
   SELECT "vnome" FROM "Velisti" WHERE "età" > 40;
   ```

   (d)
   ```
   SELECT "vnome" FROM "Velisti" WHERE "età" > 40 OR "esperienza" > 7;
   ```

(e)
```
SELECT "bid" FROM "Prenotazioni" WHERE "giorno" = '2018-12-11';
```

(f)
```
SELECT DISTINCT "colore" FROM "Barche";
```

(g)
```
SELECT "bnome" FROM "Barche" WHERE NOT ("colore" = 'rosso');
```

(h)
```
SELECT "Velisti"."vnome" FROM "Velisti", "Prenotazioni"
WHERE "Prenotazioni"."bid"= 103 " AND "Velisti"."vid" = "Prenotazioni"."vid";
```

La stessa interrogazione scritta con variabili di *range*:

```
SELECT V."vnome" FROM "Velisti" V, "Prenotazioni" P
WHERE P."bid" = 103 AND V."vid" = P."vid";
```

(i)
```
SELECT B."colore" FROM "Velisti" V, "Barche" B, "Prenotazioni" P
WHERE V."vnome" = 'Rusty' AND P."vid" = V."vid" AND P."bid" = B."bid";
```

(j) Questa interrogazione può essere scritta come segue:

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND (B."colore" = 'blu' OR B."colore" = 'verde');
```

Può anche essere scritta usando l'operatore insiemistico `UNION`:

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'blu'
UNION
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'verde';
```

(k) L'interrogazione seguente **non** corrisponde a quella richiesta (il suo significato reale è: trovare i codici dei velisti che hanno prenotato qualche barca che abbia colore blu e verde – ciò implica anche che tale interrogazione non potrà produrre nessun risultato):

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'blu' AND B."colore" = 'verde';
```

L'interrogazione richiesta può essere scritta correttamente mediante l'operatore insiemistico `INTERSECT`:

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'blu'
INTERSECT
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'verde';
```

(l) Un altro esempio di interrogazione SQL che **non** corrisponde a quella richiesta (il suo significato reale è: trovare i codici dei velisti che hanno prenotato qualche barca il cui colore sia blu e non sia verde):

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'blu' AND B."colore" != 'verde';
```

L'interrogazione richiesta può essere scritta correttamente mediante l'operatore insiemistico `EXCEPT`:

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'blu'
EXCEPT
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'verde';
```

(m)
```
SELECT P."bid" FROM "Prenotazioni" P, "Velisti" V
WHERE P."vid" = V."vid" AND V."esperienza" >= 8
EXCEPT
SELECT P."bid" FROM "Prenotazioni" P, "Velisti" V
WHERE P."vid" = V."vid" AND V."età" <= 35;
```

(n)
```
SELECT "vnome" FROM "Velisti" WHERE "vid" IN
   (SELECT "vid" FROM "Prenotazioni" WHERE "bid" = 103);
```

(o)
```
SELECT "vnome" FROM "Velisti" WHERE "vid" NOT IN
   (SELECT "vid" FROM "Prenotazioni" WHERE "bid" = 103);
```

Si noti che questa interrogazione **non** può essere scritta come segue:

```
SELECT "Velisti"."vnome" FROM "Velisti", "Prenotazioni"
WHERE "Prenotazioni"."bid" != 103 " AND "Velisti"."vid" = "Prenotazioni"."vid";
```

Infatti questa interrogazione SQL estrae i codici dei velisti che hanno prenotato *anche* qualche barca diversa dalla numero 103 (quindi non esclude i velisti che abbiano prenotato *anche* la 103, oltre ad altre barche), mentre non estrae i codici dei velisti che non abbiano mai prenotato *nessuna* barca (quindi neanche la 103).

(p)
```
SELECT * FROM "Velisti" WHERE "esperienza" > ANY
   (SELECT "esperienza" FROM "Velisti" WHERE "vnome" = 'Lubber');
```

(q)
```
SELECT "vid" FROM "Velisti" WHERE "esperienza" >= ALL
   (SELECT "esperienza" FROM "Velisti");
```

Si noti che scrivendo `"esperienza" > ALL` invece che `"esperienza" >= ALL` l'interrogazione non potrebbe produrre nessun risultato, poiché il risultato dell'interrogazione nidificata include il valore dell'esperienza maggiore.

(r)
```
SELECT "vnome", "età" FROM "Velisti" WHERE "età" >= ALL
   (SELECT "età" FROM "Velisti");
```

(s)
```
SELECT COUNT (*) FROM "Velisti";
```

(t)
```
SELECT AVG ("età") FROM "Velisti";
```

(u)
```
SELECT COUNT (DISTINCT "esperienza") FROM "Velisti";
```

(v)
```
SELECT AVG ("età") FROM "Velisti" WHERE "esperienza" = 10;
```

(w) Usando operatori insiemistici:

```
SELECT "vnome", "età" FROM "Velisti" WHERE "età" >= ALL
   (SELECT "età" FROM "Velisti");
```

Usando operatori di aggregazione:

```
SELECT "vnome", "età" FROM "Velisti" WHERE "età" =
   (SELECT MAX ("età") FROM "Velisti");
```

(x)
```
SELECT "vnome" FROM "Velisti" WHERE "esperienza" =
   (SELECT MAX ("esperienza") FROM "Velisti");
```