

LABORATORIO DI BASI DI DATI: ESERCITAZIONE 1

ESERCIZIO 1 ✓

- 1.a) Si definisca in PostgreSQL la base di dati per la memorizzazione di relazioni parentali costituita dalle tabelle **persona** e **genitore** vista a lezione.
- 1.b) Si utilizzi il comando \d per visualizzare le tabelle create.
- 1.c) Si utilizzi il comando \d <nome_tabella> per visualizzare la descrizione di ciascuna tabella creata.
- ~~1.d)~~ Si utilizzi il comando DROP TABLE <nome_tabella> per cancellare le tabelle di prova create.

implicit index? ←

ESERCIZIO 2 ✓

- 2.a) Si utilizzino i comandi del DDL di SQL visti a lezione per definire in PostgreSQL uno schema **didattica** corrispondente allo schema relazionale descritto di seguito, e soggetto ai vincoli specificati.

~~1.a)~~ **persona** (id_persona, codice_fiscale, nome, cognome, data_nascita)

dove:

- * id_persona e' un intero e costituisce la chiave primaria della tabella;
- * codice_fiscale e' un campo di 11 cifre e costituisce **una chiave consolidata della tabella. → unigul**
- * nome e cognome sono dati obbligatori mentre la data di nascita e' opzionale.

didattica . persona ...

OR

set search path

to didattica

~~1.b)~~ **corso** (id_corso, id_insegnante, sigla, crediti, descrizione)

dove:

- * id_corso e' un intero e costituisce la chiave primaria della tabella;
- * id_insegnante e' chiave esterna su persona;
- * sigla e' una stringa di 7 caratteri e la sua inserzione e' obbligatoria.
- * crediti e' un intero positivo oppure NULL

~~1.c)~~ **frequenza** (id_studente, id_corso, voto)

dove:

- * id_studente e' chiave esterna su persona;
- * id_corso e' chiave esterna su corso;
- * voto e' un intero positivo e minore o uguale a 30;
- * la chiave primaria e' costituita dall'insieme di attributi (id_studente, id_corso);

- 2.b) Si utilizzino i comandi DROP TABLE <nome_tabella>, DROP SCHEMA <nome_schema> per cancellare lo schema e le tabelle definite.

.sql salvare su nome
esecuzione li nome file .sql

LABORATORIO DI BASI DI DATI: ESERCITAZIONE 2

ESERCIZIO 1

E2 e 1 .sql

- 1.a) Si utilizzino i comandi del DDL di SQL visti a lezione per definire in PostgreSQL le tabelle dello schema relazionale definito di seguito:

- *persona* (id_persona, nome, cognome)
- *film* (id_film, id_regista, titolo, genere, anno): dove id_regista è una chiave esterna che fa riferimento a *persona*;
- *partecipazione* (id_attore, id_film, ruolo): dove id_attore ed id_film sono chiavi esterne che fanno riferimento rispettivamente a *persona* e *film*;
- *cinema* (id_cinema, nome, indirizzo)
- *proiezione* (id_cinema, id_film, giorno): dove id_cinema e id_film sono chiavi esterne che fanno riferimento rispettivamente a *cinema* e *film*.
tenendo presenti le specifiche, i vincoli e le politiche di reazione illustrate nel seguito:
- Il nome ed il cognome di una persona sono campi obbligatori;
- L'anno di produzione di un film è un intero positivo oppure NULL;
- E' necessario specificare il nome di un cinema;
- Quando un individuo viene cancellato dalla tabella *persona*, il riferimento alla sua identità nella colonna id_regista della tabella *film* deve automaticamente prendere il valore NULL;
- L'eliminazione di una persona, un cinema, oppure un film devono automaticamente innescare la soppressione delle tuple associate nelle tabelle *partecipazione* e *proiezione*.

external key

Δ SET NULL

Δ CASCADE

- 1.b) Utilizzare il comando INSERT INTO del DML di SQL per popolare la tabella *cinema* utilizzando l'informazione specificata di seguito:

id_cinema	nome	indirizzo
02	S. Angelo	Via Lucida 6 Perugia
01	Zenith	Via Bonfigli 11 Perugia
03	Multisala Clarici	Corso Cavour 84 Foligno
04	Multiplex Giometti	Strada Centova Perugia

- 1.c) Utilizzare il comando

SELECT * FROM <nome_tabella>

per verificare l'avvenuto popolamento della tabella.



d) Utilizzare il comando di PostgreSQL:

[\copy <nome_tabella> FROM <nome_file>

per popolare le tabelle *partecipazione*, *film*, *proiezione* e *persona*, previa la definizione di opportuni files di testo *partecipazione.txt*, *film.txt*, *proiezione.txt* e *persona.txt*, reperibili su:

<http://www.dmi.unipg.it/raffaella.gentilini/cinema/persona.txt>

<http://www.dmi.unipg.it/raffaella.gentilini/cinema/film.txt>

<http://www.dmi.unipg.it/raffaella.gentilini/cinema/partecipazione.txt>

<http://www.dmi.unipg.it/raffaella.gentilini/cinema/proiezione.txt>



Si vuole eliminare l'attore John Travolta dalla base di dati. Quali operazioni e' necessario effettuare per mettere a punto tale soppressione?



Si definisca uno script BDcinema.sql per la generazione ed il popolamento della base di dati vista nei punti precedenti. In particolare tale script deve prevedere:

- l'eliminazione di ciascuna delle tabelle che si vuole creare, se gia' presente nella BD
- la creazione di ciascuna tabella, come specificato al punto (1.a)
- il popolamento di ciascuna delle tabelle



Si verifichi la funzionalita' dello script creato mediante il comando di PostgreSQL \i.

ESERCIZIO 2



E2e2.sql

Si consideri il seguente schema relazionale relativo ad una porzione di un ipotetico sistema di gestione del calendario degli *Europei di Calcio 2012*:



stadio(nome, *citta*, *capienza*)



squadra(nazione, *confederazione*, *data_qualifica*, *sponsor*)



partita(*stadio*, *data*, *squadra1*, *squadra2*, *goal1*, *goal2*, *spettatori*), dove l'attributo *stadio* e' chiave esterna sulla relazione *stadio*, e l'attributo *squadra1* (resp. *squadra2*) e' chiave esterna sulla relazione *squadra*.

Si definisca uno script SQL per la generazione e la popolazione di uno schema *Europei2012* che implementa lo schema relazionale proposto. Tale script dovra' essere composto da 3 parti principali:

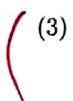


La prima, per cancellare schemi e tabelle omonime eventualmente presenti nella base di dati



la seconda per generare lo schema definendo vincoli opportuni ed in particolare:

- Non si possono cancellare le informazioni su uno stadio (resp. una squadra), se il calendario prevede una partita in tale stadio (resp. di tale squadra);
- Se si aggiorna uno stadio (resp. una squadra), le partite giocate in tale stadio (resp. da tale squadra) vanno aggiornate in cascata;
- Per ogni partita, e' obbligatorio inserire i dati relativi alle due squadre sfidanti.



(3) la terza, per popolare opportunamente lo schema generato utilizzando i dati reperibili sul sito WEB:

http://it.wikipedia.org/wiki/Campionato_europeo_di_calcio_2012

LABORATORIO DI BASI DI DATI: ESERCITAZIONI 3-4

ESERCIZIO 1

Con riferimento al database realizzato in PostgreSQL nel laboratorio precedente, relativo allo schema relazionale:

- *persona* (id_persona, nome, cognome)
- *film* (id_film, id_regista, titolo, genere, anno): dove *id_regista* è una chiave esterna che fa riferimento a *persona*;
- *partecipazione* (id_attore, id_film, ruolo): dove *id_attore* ed *id_film* sono chiavi esterne che fanno riferimento rispettivamente a *persona* e *film*;
- *cinema* (id_cinema, nome, indirizzo)
- *proiezione* (id_cinema, id_film, giorno): dove *id_cinema* e *id_film* sono chiavi esterne che fanno riferimento rispettivamente a *cinema* e *film*.

si scrivano in SQL le seguenti interrogazioni

- (1) Qual'è il contenuto della tabella *persona*? ✓
- (2) Selezionare i cognomi delle persone nella BD, senza eliminare i doppi. ✓
- (3) Selezionare i cognomi delle persone ordinati in ordine alfabetico, senza eliminare i doppi. ✓
- (4) Definire l'insieme dei cognomi delle persone mantenute nella BD. ✓
- (5) Selezionare le persone di nome John mantenute nella BD. ✓
- (6) Selezionare i cognomi delle persone di nome John, mantenute nella BD. ✓
- (7) Selezionare le persone mantenute nella BD che sono attori. ✓
- (8) Definire la lista di tutte le interpretazioni, precisando il nome dell'attore, il cognome dell'attore, il ruolo interpretato ed il titolo del film. Il risultato dovrà essere nella forma: ✓


cognome	nome	ruolo	titolo
Nicole	Kidman	Grace	Dogville
Paul	Bettany	Tom Edison	Dogville
...

- non in query-sql → (9) Definire la lista di tutte le interpretazioni, precisando il nome dell'attore, il cognome dell'attore, il ruolo interpretato ed il titolo del film. Il risultato dovrà essere nella forma: ✓

Nicole Kidman ha interpretato il ruolo di Grace nel film Dogville
Paul Bettany ha interpretato il ruolo di Tom Edison nel film Dogville
...

- (10) Quali sono i titoli dei film di genere drammatico? ✓
- (11) Quali film (titoli) sono stati proiettati nel 2002? ✓
- (12) Elencare i titoli dei film diretti da Lars von Trier. ✓
- (13) Elencare i film (titoli) proiettati al cinema *Le Fontanelle*. ✓

TROLL! ✓
(lolol)

-  Elencare i nomi ed i cognomi dei registi. ✓
- (15) Selezionare i nomi ed i cognomi degli attori. ✓
- (16) Selezionare i nomi d i cognomi degli attori che sono anche registi. ✓
- (17) Quali attori hanno preso parte a film proiettati al cinema *Le Fontanelle* dopo l'anno 2000? ✓
- ? (18) Quali sono i titoli dei film interpretati da Nicole Kidman che non sono stati proiettati al cinema *Le Fontanelle*? ~o ≠ (non: --) ✓
- (19) Quali sono i cognomi delle persone che contengono la lettera *s*?
- (20) Elencare i titoli dei film che contengono almeno tre *e*.

LABORATORIO DI BASI DI DATI: ESERCITAZIONE 5

ESERCIZIO 1 (facile)

Con riferimento al database realizzato in PostgreSQL, nelle lezioni di laboratorio precedenti, relativo allo schema relazionale:

- *persona* (*id_persona*, *nome*, *cognome*)
- *film* (*id_film*, *id_regista*, *titolo*, *genere*, *anno*): dove *id_regista* è una chiave esterna che fa riferimento a *persona*;
- *partecipazione* (*id_attore*, *id_film*, *ruolo*): dove *id_attore* ed *id_film* sono chiavi esterne che fanno riferimento rispettivamente a *persona* e *film*;
- *cinema* (*id_cinema*, *nome*, *indirizzo*)
- *proiezione* (*id_cinema*, *id_film*, *giorno*): dove *id_cinema* e *id_film* sono chiavi esterne che fanno riferimento rispettivamente a *cinema* e *film*.

si scrivano in SQL le seguenti interrogazioni

1. Definire la lista degli attori (nome e cognome) specificando per ognuno il numero di film interpretati. Il risultato deve essere presentato in ordine alfabetico rispetto ai cognomi degli attori.
2. Modificare la query precedente in modo da aggiungere al risultato due colonne per precisare:
 - 1. l'anno di produzione del primo film interpretato da ogni attore elencato nel risultato della query
 - 2. l'anno di produzione dell'ultimo film interpretato da ogni attore elencato nel risultato della query
3. Modificare il risultato della query precedente per tenere conto solo degli attori che hanno interpretato almeno due film.
4. Elencare gli attori (nome e cognome) che hanno interpretato film drammatici, e per ognuno di tali attori precisare il numero dei film drammatici interpretati.
5. Quanti film ha diretto ciascun regista nella BD?

1, 4, 5, 8, 10

ESERCIZIO 1

Si consideri il seguente schema relazionale relativo ad una porzione di un ipotetico sistema di gestione di una biblioteca:

- ~~scrittore~~(nome, sesso, nazione)
- ~~libro~~(ISBN, titolo, autore, genere), dove l'attributo autore (resp. genere) e' chiave esterna sulla relazione scrittore (resp. generi)
- ~~socio~~(id, ~~nome~~, sesso, eta)
- ~~ha_letto~~(ISBN, socio), dove l'attributo socio (resp. ISBN) e' chiave esterna su socio (resp. libro).
- ~~generi~~(nome, sala)

Si definisca uno script SQL per la generazione di uno schema biblioteca che implementa lo schema relazionale proposto. Tale script dovra' essere composto da 2 parti principali: La prima, per cancellare le tabelle omonime eventualmente presenti nella base di dati, la seconda per generare lo schema definendo vincoli opportuni ed in particolare:

- Non si possono cancellare le informazioni su uno scrittore, se e' presente un libro dello stesso.
- Se si cancella/aggiorna un libro, le consultazioni ad esso relative vanno cancellate/aggiornate in cascata
- Se un socio della sala di lettura viene aggiornato/cancellato, nelle consultazioni ad esso relative l'attributo socio viene aggiornato/cancellato in cascata

- (2) Si popoli opportunamente lo schema generato utilizzando i dati nei file di testo allegati all'esercizio.
- (3) Si definiscano in SQL le seguenti interrogazioni:

- (a) Ottenere i nomi dei soci di sesso femminile che hanno letto qualche libro
- (b) Determinare i titoli dei libri nella sala A
- (c) Ottenere i titoli dei libri e la sala in cui sono collocati
- (d) Ottenere i titoli dei libri e la sala in cui sono collocati, inclusando le opere di cui non e' possibile reperire la collocazione → Finzioni
- (e) Identificare i soci (i.e. elencarne gli id) della biblioteca che hanno letto almeno un libro nella sala A
- (f) Determinare i nomi delle coppie di soci della biblioteca che hanno letto uno stesso libro
- (g) Elencare i nomi dei soci della biblioteca che hanno letto almeno un libro nella sala A
- (h) Determinare gli autori dei libri letti da almeno una donna
- (i) Determinare i soci della biblioteca che non hanno mai letto un libro situato nella sala B

rivedere a monte lucide

delirio

→ nome scrittore.txt

FF66 → 66A1Fier CC33 → Elcaci Padem

Uopdi al ← 77
Mondal ← 55
Poe ← 33

{ 8822
~~FF66~~ dame
FF66

CA F
11 88 33
33

11
33
88

CC33
AA11
FF66

LABORATORIO DI BASI DI DATI: ESERCITAZIONE 5

ESERCIZIO 1

Con riferimento al database realizzato in PostgreSQL nelle lezioni di laboratorio precedenti, relativo allo schema relazionale:

- *giocatore* (nome, nazionalita')
- *apertura* (eco, nome, variante)
- *partita* (bianco,nero,luogo,anno,round, eco, risultato, num_mosse)

si scrivano in SQL le seguenti interrogazioni

- ~~(1)~~ Ottenere i nomi dei giocatori della stessa nazionalita' di Kramnik
 - ~~(2)~~ Determinare i nomi e le varianti delle aperture il cui nome segue, nell'ordinamento lessicografico, il nome dell'apertura con codice E86
 - ~~(3)~~ Determinare le partite aperte con la variante '1 e4 c5; 2 c3' NOT IN
 - ~~(4)~~ Determinare i risultati delle partite le cui aperture sono diverse da quelle usate da Kasparov con il Bianco (attenzione ai valori nulli!)
 - ~~(5)~~ Ottenere i nomi delle aperture di partite il cui numero di mosse e' inferiore al numero di mosse di qualche partita giocata da Deep Blue
 - ~~(6)~~ Ottenere la nazionalita' dei giocatori che hanno vinto qualche partita con Bianco
 - ~~(7)~~ Ottenere i nomi dei giocatori che, con il Nero, hanno vinto tutte le partite, ossia tali che non esista alcuna partita che non abbiano vinto (attenzione ai valori nulli!) ALL
- Eno' {
- (8) Ottenere i nomi delle aperture giocate da tutti i giocatori di cui non e' nota la nazionalita'. Si presti attenzione alla possibilita' che esistano giocatori di cui non e' nota la nazionalita' e che non hanno giocato alcuna partita
 - (9) Ottenere il numero di vittorie con il Nero per ciascun giocatore che abbia giocato qualche partita

ESERCIZIO 2

Con riferimento al database realizzato in PostgreSQL nelle lezioni di laboratorio precedenti, relativo allo schema relazionale:

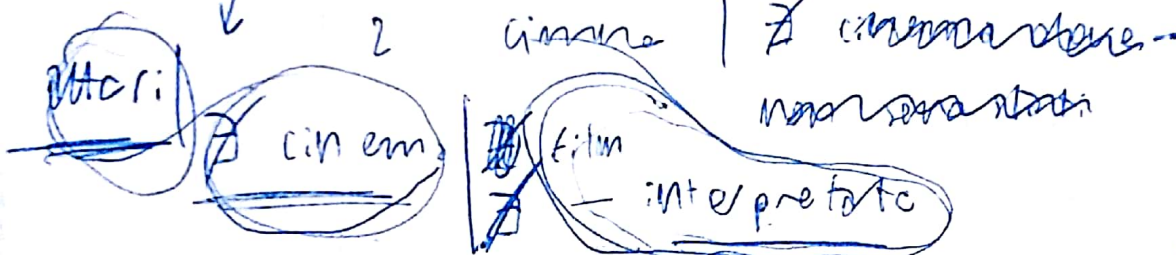
- *persona* (id_persona, nome, cognome)
- *film* (id_film, id_regista, titolo, genere, anno): dove id_regista e' una chiave esterna che fa riferimento a persona;
- *partecipazione* (id_attore, id_film, ruolo): dove id_attore ed id_film sono chiavi esterne che fanno riferimento rispettivamente a persona e film;
- *cinema* (id_cinema, nome, indirizzo)
- *proiezione* (id_cinema, id_film, giorno): dove id_cinema e id_film sono chiavi esterne che fanno riferimento rispettivamente a cinema e film.

si scrivano in SQL le seguenti interrogazioni

LABORATORIO DI BASI DI DATI ESERCITAZIONE 5

- (1) Definire la lista degli attori (nome e cognome) che hanno interpretato film drammatici precisandone il numero di film (drammatici e non) interpretati;
- (2) Quali cinema hanno proiettato tutti i film nella BD? *divisione*
- (3) Quali attori hanno interpretato, per ogni cinema nella BD, almeno un film proiettato in tale cinema? *no*
- (4) Definire nome, cognome e film realizzati dai registi che hanno diretto almeno un film dello stesso genere di uno dei film diretti da David Cronenberg.
- (5) Chi sono gli attori che hanno interpretato tutti i film di Lars von Trier? *divisione*

*cinema
mai non
esiste un
film
non
proiettato*



Un popolare sito WEB a tema scacchistico fornisce agli utenti informazioni relative a famose partite di scacchi. Una semplificazione dello schema di dati della base di dati usata dal sito e' il seguente:

*DA
FINIRE*

- *giocatore* (*nome*: stringa, *nazionalita'*: stringa)
 - *apertura* (*eco*: dom.eco, *nome*: stringa, *variante*: stringa): dove *nome* e' un attributo obbligatorio e *variante* e' una superchiave;
 - *partita* (*bianco*: stringa, *nero*: stringa, *luogo*: stringa, *anno*: N, *round*: N, *eco*: dom.eco, *risultato*: dom.risultato, *num_mosse*: N): dove *bianco* e *nero* sono chiavi esterne che fanno riferimento a *giocatore* ed *eco* e' una chiave esterna che fa riferimento ad *apertura*;
- Si tenga presente le osservazioni elencate nel seguito:
- Per *apertura* si intende la fase iniziale di una partita. Le aperture sono convenzionalmente identificate attraverso il cosiddetto codice ECO, che e' un codice di tre lettere;
 - Il risultato di una partita e' espresso mediante una stringa di tre lettere, che puo' essere 1-0 (vittoria di Bianco), 0-1 (vittoria di Nero), 1/2 (pareggio).

17:30 - 19:30

www.dmi.unipg.it / roffoella.gentilini/scacchi/

*apertura.txt
giocatore.txt
partita.txt*

LABORATORIO DI BASI DI DATI: ESERCITAZIONE

Con riferimento al database realizzato in PostgreSQL, nelle lezioni di laboratorio precedenti, relativo allo schema relazionale:

- *persona* (*id_persona*, *nome*, *cognome*)
- *film* (*id_film*, *id_regista*, *titolo*, *genere*, *anno*): dove *id_regista* è una chiave esterna che fa riferimento a *persona*;
- *partecipazione* (*id_attore*, *id_film*, *ruolo*): dove *id_attore* ed *id_film* sono chiavi esterne che fanno riferimento rispettivamente a *persona* e *film*;
- *cinema* (*id_cinema*, *nome*, *indirizzo*)
- *proiezione* (*id_cinema*, *id_film*, *giorno*): dove *id_cinema* e *id_film* sono chiavi esterne che fanno riferimento rispettivamente a *cinema* e *film*.

Si svolgano i seguenti esercizi:

ESERCIZIO 1

- 1.a) Definire un trigger per implementare il seguente vincolo: La data di proiezione di un film deve essere posteriore all'anno di produzione del film.]
- 1.b) Verificare il funzionamento del trigger definito al punto precedente.

confronto date (anni)

EXTRACT('year' FROM data) > ...

ESERCIZIO 2

- 2.a) Definire un trigger per impedire la cancellazione di un film ogniqualvolta tale film è l'unica opera mantenuta nella BD del regista del suddetto film (i.e. certifica che l'autore del film è un regista). *115m30*
- 2.b) Verificare il funzionamento del trigger definito al punto precedente.

CREATE USER A UAG E
PGSQL

999.99

ESAME DI LABORATORIO DI BASI DI DATI

15/01/2018

(appellu @)

COMPITO A

Esercizio 1. Si consideri il seguente schema di base di dati relazionale:

- ASILONIDO(~~nome~~, num_posti, indirizzo, citta')
- BAMBINO(~~cod. fiscale~~, nome, data_nascita, punteggio)
- DOMANDA(~~asilo~~, bambino, data, num_preferenza)
- ISCRIZIONE(~~bambino~~, asilo, data, retta_mensile)

In particolare, l'attributo num_preferenza della relazione DOMANDA rappresenta l'ordine di preferenza (1 per l'asilo preferito dal bambino, 2 per l'eventuale secondo, 3 per la sua terza scelta ...). Un bambino puo' aver fatto domanda ad un asilo, ma non esserne iscritto. *le puo' en era iscritto a un solo asilo*

Si definisca uno script SQL per la generazione e la popolazione di uno schema Asili che implementa lo schema relazionale proposto. Tale script dovra' essere composto da 3 parti principali:

- (1) La prima, per cancellare schemi e tabelle omonime eventualmente presenti nella base di dati.
- (2) La seconda per generare lo schema *definendo vincoli opportuni*.
- (3) La terza, per popolare opportunamente lo schema, utilizzando i dati allegati al testo.

*anche
cambio
politica
mod.*

Esercizio 2. Si estenda lo script SQL creato al punto precedente al fine di eseguire le seguenti interrogazioni:

- (1) Trovare il codice fiscale dei bambini che hanno fatto domanda presso un asilo della provincia di Perugia, ma non hanno fatto domanda in alcun asilo della provincia di Terni.
- (2) Per ogni bambino che ha effettuato almeno due domande di iscrizione, determinare il numero totale delle suddette domande di iscrizione.
- (3) Determinare il codice fiscale dei bambini che hanno fatto domanda a tutti gli asili di Perugia. *o ogni*

*steno
.sql*

bambini

Esercizio 3. Si scriva un'applicazione Java che, caricando l'opportuno driver JDBC, si connette al DB di riferimento e: *→ schema preimpostato*

- (1) Aggiunge alla relazione ASILONIDO l'attributo NUMISCRITTI, relativo al numero di iscritti a tale asilo. *(ALTER TABLE)*

[7 (2+5)]

Esercizio 4. Si definisca un trigger per aggiornare automaticamente la colonna NUMISCRITTI nella relazione ASILONIDO. *(incremento)*

*conserva
cartella
.zip +*

(3) bambini: | 7 asilo | 7 domanda