INGEGNERIA E SCIENZE INFORMATICHE

UNIBO, Cesena (FC)

Museums Manager

Gestionale per l’amministrazione interna di musei e famiglie di musei

*Esame di Basi di Dati 2020-21*

**Candidati:**

Farneti Andrea(mat. 0000832033)

Pantieri Andrea(mat. 0000825053)

Sommario

# **ANALISI DEI REQUISITI**

Intervista al committente (manager di diversi musei sparsi per l’Italia):

*“I nostri musei necessitano di un sistema gestionale per gli amministratori che permetta l’inserimento e la gestione di ognuno di essi, e l’archivio di ogni informazione ritenuta utile ed importante.*

*Le funzionalità principali che riteniamo essenziali sono la gestione di musei e dipendenti, l’elenco di ogni oggetto esposto all’interno del museo e relative informazioni, ed il calcolo di saldo e flusso per ogni museo.*

*È necessario che il sistema permetta anche la creazione e modifica di biglietti d’ingresso per un museo.”*

Individuiamo le parole chiave da questa prima intervista:

*“I nostri musei necessitano di un sistema* ***gestionale*** *per gli amministratori che permetta* ***l’inserimento*** *e la gestione di ognuno di essi, e* ***l’archivio*** *di* ***ogni informazione*** *ritenuta utile ed importante.*

*Le funzionalità principali che riteniamo essenziali sono la* ***gestione di musei e dipendenti****, l’elenco di* ***ogni oggetto esposto*** *all’interno del museo* ***e relative informazioni*** *ed il* ***calcolo di saldo e flusso*** *per ogni museo.*

*È necessario che il sistema permetta anche la* ***creazione e modifica*** *di biglietti d’ingresso per ogni museo.”*

La prima intervista è stata troppo vaga e ci ha lasciato con diverse domande:

* Quali sono i criteri secondo i quali un’informazione è ritenuta importante?
* Cosa si intende per gestione di musei e dipendenti nello specifico?
* Di quali informazioni dobbiamo prendere nota per ogni oggetto esposto?
* Ogni quanto è necessario calcolare saldi e flussi dei musei?
* In che livello di dettaglio bisogna scendere per musei, ed oggetti?
* Come funzionano i biglietti nello specifico?

Abbiamo richiesto una seconda intervista al committente per delucidazioni:

*“Il servizio di gestione di ogni museo e dipendente deve permettere l’aggiunta, la modifica e l’eliminazione di ognuno di essi.*

*Informazioni ritenute importanti per noi sono le varie statistiche prodotte da ogni museo, da archiviare sia mensilmente che annualmente. Per ogni oggetto esposto all’interno di un museo è tassativamente necessario conoscerne provenienza, periodo storico ed autore/i. Un oggetto può essere composto da più parti, per cui potrebbe essere interessante disporre di tali informazioni anche per le singole parti di cui si compone un oggetto. Per quanto riguarda i musei, essi potrebbero contenere delle sezioni e, a loro volta, delle sottosezioni. I dipendenti, invece, devono essere registrati in base al loro ruolo ed in base all’orario di arrivo ed uscita dal lavoro. Ogni biglietto dovrà avere una data di validità e un proprio prezzo, oltre che una tipologia (sconto studente, bambini...). Inoltre, voglio che i clienti possano vedere le date in cui un museo è chiuso o aperto specialmente, in modo che possano comprare i biglietti di conseguenza. Il numero di biglietti totale potrebbe anche variare se è un giorno di apertura speciale. Un museo può offrire ai visitatori diversa scelta riguardo ai biglietti d’entrata.”*

Individuiamo ora nuovamente le parole chiave:

*“Il servizio di gestione di ogni museo e dipendente deve permettere* ***l’aggiunta, la modifica e l’eliminazione*** *di ognuno di essi.*

*Informazioni ritenute importanti per noi sono* ***le varie statistiche prodotte da ogni museo****, da archiviare sia* ***mensilmente che annualmente****. Per ogni oggetto esposto all’interno di un museo è tassativamente necessario conoscerne* ***provenienza, periodo storico ed autore/i****. Un oggetto può essere composto da più* ***parti****, per cui potrebbe essere interessante disporre di tali informazioni anche per le singole parti di cui si compone un oggetto. Per quanto riguarda i musei, essi potrebbero contenere delle* ***sezioni*** *e, a loro volta, delle* ***sottosezioni****. I dipendenti, invece, devono essere registrati in base al loro* ***ruolo*** *ed in base* ***all’orario di arrivo ed uscita*** *dal lavoro. Ogni biglietto dovrà avere una* ***data di validità*** *e un proprio* ***prezzo****, oltre che una* ***tipologia*** *(sconto studente, bambini...). Inoltre, voglio che i clienti possano vedere le* ***date*** *in cui un museo è* ***chiuso*** *o* ***aperto specialmente****, in modo che possano comprare i biglietti di conseguenza. Il numero di biglietti totale potrebbe anche* ***variare*** *se è un giorno di apertura speciale. Un museo può offrire ai visitatori* ***diversa scelta*** *riguardo ai biglietti d’entrata.”*

[Si ipotizza sia avvenuta anche una terza ed ultima intervista chiarificativa col committente nella quale egli ha specificato nel dettaglio di quali caratteristiche è composta ogni “entità” definita nella seconda intervista. Ad esempio: “per ogni museo bisognerà inserire gli orari di apertura/chiusura, il numero di biglietti massimo...”)]

Una volta individuati ed estratti i concetti principali dall’intervista col committente stiliamone un documento dettagliato riassuntivo che includa tutti i requisiti di cui sarà composto l’applicativo e che dovranno necessariamente essere implementati (*specifica dei requisiti*):

L’applicativo da implementare è un gestionale per musei rivolto unicamente al personale di amministrazione, in altre parole ai dirigenti/capi dei musei stessi. Esso deve permettere la realizzazione delle seguenti funzionalità (per le quali è necessario un database):

* + Creazione, modifica ed eliminazione di un museo (con relative informazioni di base quali nome, luogo, orario di apertura/chiusura e numero di biglietti massimo)
  + Creazione di nuove famiglie di musei (gruppi di musei correlati tra loro per i quali si vogliono ottenere statistiche di gruppo; Es. Isola dei musei di Berlino)
  + Creazione, modifica ed eliminazione di specifiche sezioni (ed eventuali sottosezioni) di cui è composto un museo. Esempio: la macro-sezione di Storia di un museo potrebbe essere composta dalle sottosezioni di storia romana, storia etrusca ecc...
  + Aggiunta, modifica ed eliminazione di contenuti all’interno di un museo (e degli eventuali sottocontenuti di cui è composto), posizionati all’interno di una sezione del museo. Ogni contenuto disporrà di diverse informazioni quali nome, descrizione, data di ritrovamento e di arrivo al museo, provenienza, creatore e periodo storico. Esempio: il modellino di una bomba atomica contenuto nella sezione di Scienza di un museo è formato a sua volta dalla scocca, il detonatore, l’esplosivo... Anche per ognuna di queste sottoparti si vogliono avere delle informazioni.
  + Deve essere possibile cercare i contenuti di un museo per provenienza, tipo, creatore e periodo storico.
  + Aggiunta, modifica ed eliminazione degli autori che hanno realizzato determinati contenuti (nome, cognome, anno di nascita, descrizione).
  + Aggiunta, modifica ed eliminazione del personale che lavora all’interno di un museo (bisogna conoscerne nome, cognome, numero di cellulare/indirizzo mail e lo stipendio orario)
  + Funzionalità di registro presenze che permetta di registrare la data di entrata/uscita dei dipendenti dal lavoro.
  + Funzionalità di registro manutenzioni che permetta di tenere conto di tutte le manutenzioni che sono state fatte per un determinato museo (la data della manutenzione, ciò che è stato fatto ed il costo)
  + Creazione, modifica ed eliminazione di biglietti per l’accesso ad un museo. Ogni biglietto avrà una data di validità, un prezzo e potrà essere di un determinato tipo (esempio: ingresso studente, ingresso bambino...)
  + Aggiunta ed eliminazione dei giorni di chiusura.
  + Aggiunta, modifica ed eliminazione dei giorni di apertura speciali ed eventuali orari speciali di apertura/chiusura.
  + Calcolo di tutte le statistiche mensili ed annuali di ogni museo e di ogni famiglia di musei (spese totali, fatturato, numero di biglietti venduti, numero di manutenzioni svolte, numero di nuovi contenuti aggiunti e numero di giorni di chiusura totali).

Ciò di cui non deve occuparsi questo gestionale riguarda:

* Come il personale compila il registro presenze (si ipotizza che esista un altro applicativo per dipendenti che utilizza la stessa sezione di database e tramite il quale essi compiono giornalmente check-in e check-out)
* Come i clienti acquistano i biglietti (si ipotizza che vengano stampati e comprati direttamente dentro il museo, oltreché esista un applicativo/sito web che interagisca con la stessa sezione di database e permetta di comprare/prenotare biglietti online registrandone le vendite).

**PROGETTAZIONE CONCETTUALE**

A seguito della fase di analisi dei requisiti vengono progettati e proposti i seguenti schemi scheletro per definire la struttura dell’applicativo:

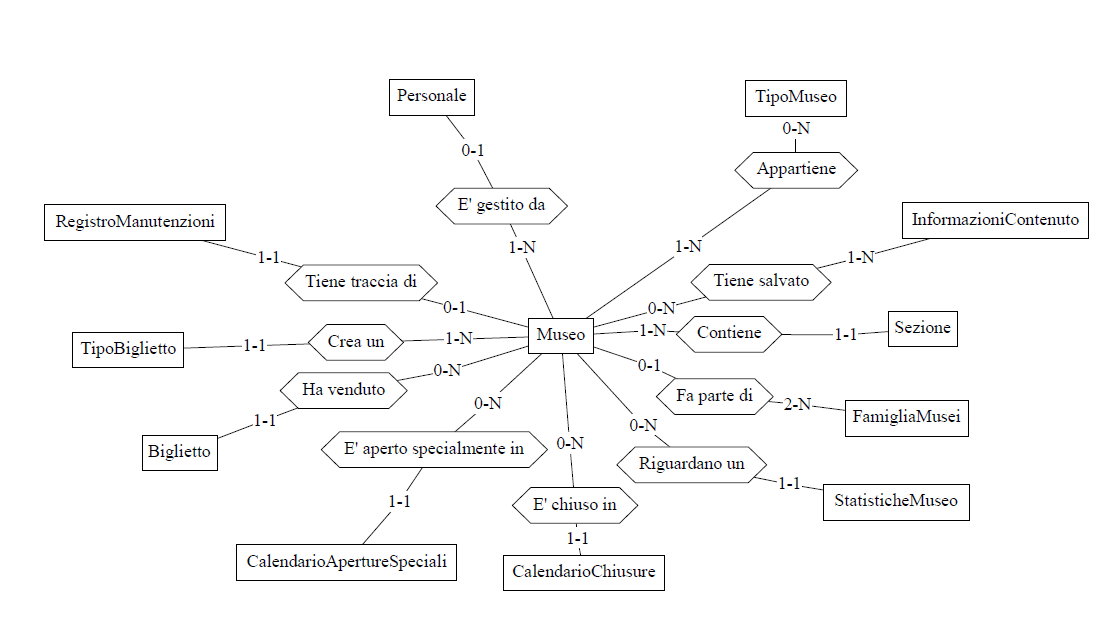
*Museo:*

L’entità “**Museo**”rappresenta il fulcro dell’intero schema E/R ed è l’entità principale sulla quale si basa il progetto. Ogni entità associata ad essa rappresenta, nella pratica, una funzionalità che il museo svolge e che l’applicativo deve permettere di realizzare.

È possibile suddividere queste funzionalità in due macro-argomenti:

* la gestione del personale, dei clienti e delle statistiche
* la gestione dei contenuti interni al museo

In seguito sarà necessario convertire la relazione “*Appartiene”* associata a “**Museo**”e “**TipoMuseo**”in un’entità, in quanto un museo può essere di più tipi e più tipi di museo potrebbero ripetersi fra più musei (di conseguenza per individuare univocamente un museo ed il suo tipo (senza duplicare i record all’interno di TipoMuseo che creerebbe ridondanza) è necessario l’utilizzo di una terza entità). Anche la relazione “*Tiene salvato*” fra “**Museo**” e “**InformazioniContenuto**” dovrà essere convertita in un’entità, in quanto più musei potrebbero utilizzare le stesse informazioni di un contenuto (ad esempio la provenienza “Italia” non dovrà essere creata due volte).



*Personale:*

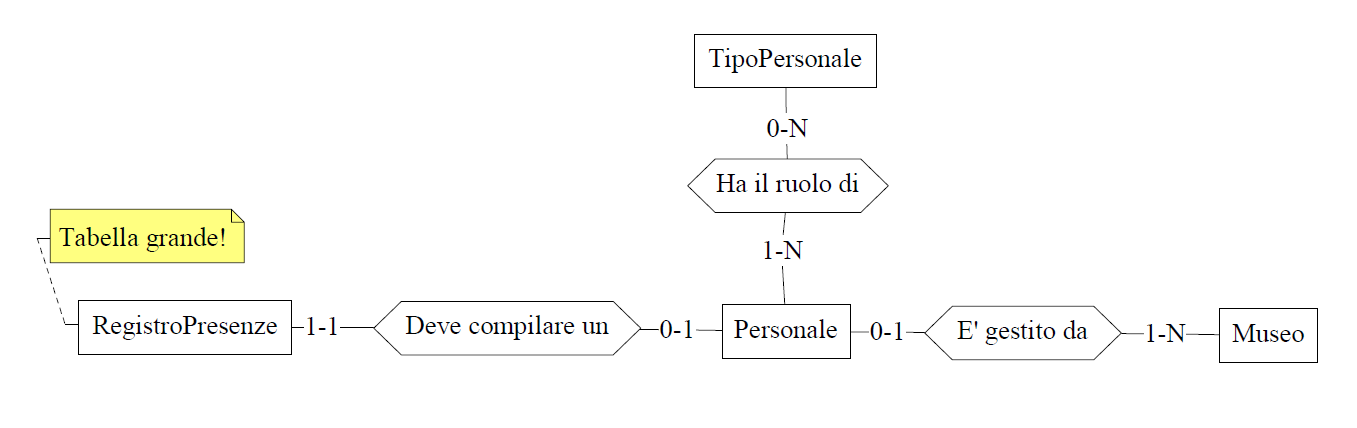
Dato che l'applicativo che ci è stato commissionato non necessita di definire entità specifiche di personale con caratteristiche e ruoli univoci (ad esempio manager, dipendente, guida...) non è stato necessario realizzare una generalizzazione gerarchica con delle entità specificative, ma è stato sufficiente definire l’entità “**TipoPersonale**”, associata a “**Personale**” dalla relazione “*Ha ruolo di*”.

Adottando questa soluzione però, si nota che sarà necessario trasformare quest’ultima relazione in un’entità, in quanto più ruoli di lavoro potrebbero ripetersi fra più persone e una persona potrebbe avere più ruoli.

Chiaramente il personale varierà a seconda del museo in cui lavora e ognuno di essi dovrà effettuare giornalmente l’operazione di check-in e check-out a lavoro, che verranno salvate nell’entità “**RegistroPresenze**”.

Dato che l’applicativo da realizzare è rivolto solamente agli amministratori del museo e non ai lavoratori, “RegistroPresenze” è utilizzato all’interno dell’applicativo solo in lettura. Esso non si occupa di come la futura tabella verrà compilata (si ipotizza esista un sistema o un applicativo esterno dedicato al personale tramite il quale vengono registrate le presenze sul luogo di lavoro e salvate come record in “RegistroPresenze”).

Si noti che ”RegistroPresenze” risulterà essere una tabella di grandi dimensioni, in quanto dovrà ricevere giornalmente nuovi record per ogni personale presente a lavoro in ogni museo; sarà quindi necessario provvedere nella fase di progettazione logica ad ottimizzarne la sua gestione tramite diverse tecniche, a seconda delle operazioni che dovrà svolgere (select, insert into..).



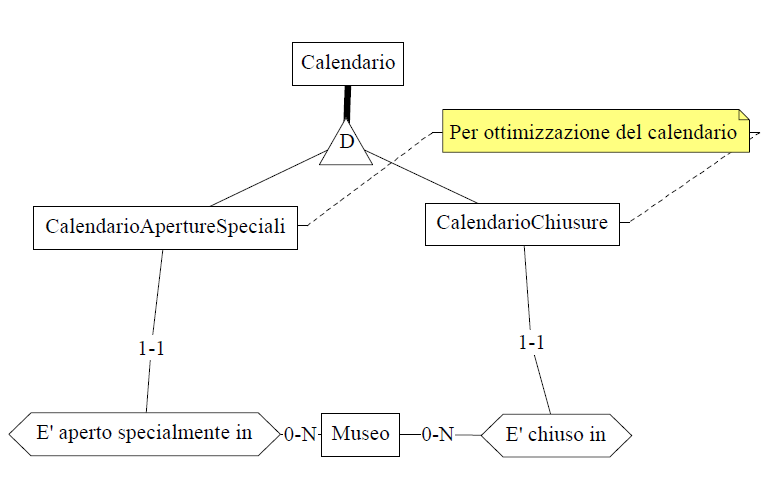
*Calendario:*

Un’altra funzionalità richiesta per questo applicativo è la gestione delle giornate di apertura/chiusura e dei relativi orari di ogni museo. Questa funzionalità risulta essere inoltre necessaria per permettere ai clienti (tramite un applicativo esterno) di visionare le date di apertura di un museo, permettendo quindi l’acquisto di biglietti. Per realizzare tali operazioni sarà quindi necessaria la creazione di un sistema di calendario (variabile, ovviamente, per ogni museo).

Per stabilire sia i giorni di apertura che quelli di chiusura abbiamo quindi creato ed organizzato una gerarchia per “**Calendario”** in modo tale da ottimizzarlo, rimuovendo la necessità di salvare ed aggiornare periodicamente lo stato di tutti i giorni annuali (questa affermazione è valida finché ogni museo mantiene un numero maggiore di giorni di apertura rispetto a quelli di chiusura).

Questa gerarchia sarà composta da due figli:

* **CalendarioApertureSpeciali:** indentifica i giorni di apertura speciale, come i giorni festivi oppure giornate con orari speciali
* **CalendarioChiusure:** indica tutti i giorni di chiusura.



*Biglietto:*

“**Biglietto**” non è altro che un’entità che ha l'obbiettivo di archiviare gli acquisti effettuati dai clienti a fini statistici. Dato che l’applicativo che ci è stato commissionato è rivolto solamente al personale di amministrazione, esso non dovrà occuparsi di fornire un’interfaccia ai clienti per l’acquisto di biglietti, ma il database deve comunque essere realizzato in modo tale da permettere anche tale funzione (perché si presuppone che esista un applicativo esterno che si connetterà e utilizzerà la stessa porzione di database, salvando gli acquisti in “Biglietto”).

Il nostro applicativo utilizzerà quindi “Biglietto” in sola modalità “lettura”.

Specifichiamo ora tre limitazioni che abbiamo deciso di imporre a carattere generale ai biglietti e al calendario per semplificare il dominio del problema:

* Un biglietto è valido per una sola persona.
* Un biglietto è valido per una sola giornata.
* Il calendario non prevede come impostare un periodo di chiusura per un museo (è solo possibile impostare ogni giorno di chiusura singolarmente)

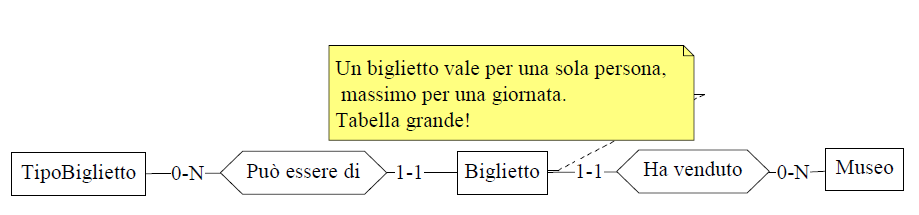
Abbiamo deciso di imporre tali limitazioni onde evitare in “Calendario” anche la definizione di un sistema di “range temporale” per gestire solamente questi ultimi due casi specifici (ovvero biglietti validi per più giornate ed il poter impostare periodi di chiusura): il tutto sarebbe risultato ridondante per la data ed eccessivo come carico di lavoro.

È stato necessario anche modellare come gestire più tipologie di biglietti; Uno studente, ad esempio, potrebbe avere diritto ad uno sconto sul prezzo del biglietto rispetto ad un adulto.

Di seguito ne verranno indicate solo alcune delle possibili tipologie esistenti:

* Studente (prezzo 70%);
* Biglietto normale (over 25, under 65);
* Under 25, over 15 (prezzo 75%);
* Under 15 (prezzo 50%);
* Over 65 (prezzo 50%).
* ...

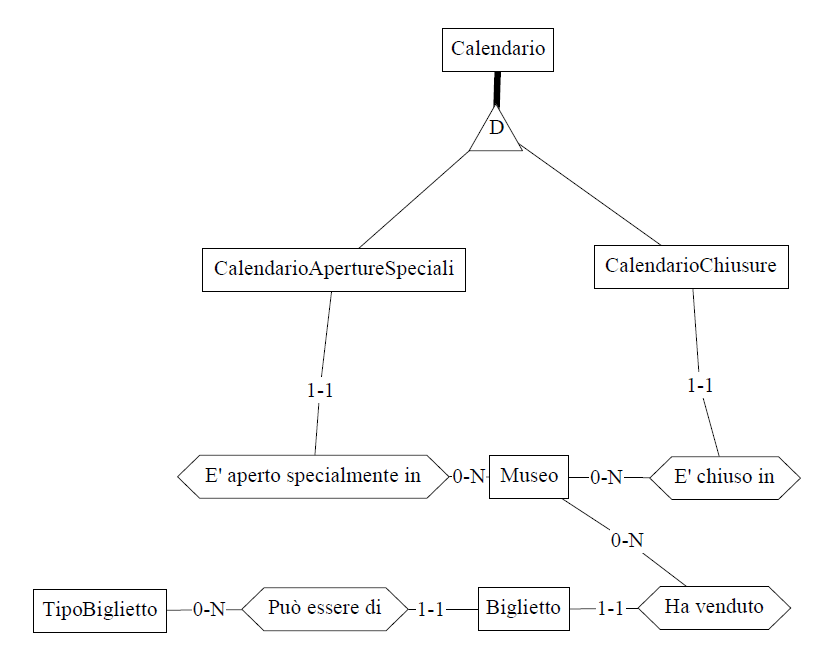
Abbiamo deciso quindi di non utilizzare una gerarchia per i tipi di biglietti affinché non ne sia limitato il numero massimo di varietà.



*Relazione fra Calendario e Biglietto:*

Il seguente schema scheletro mostra l’interazione che esiste fra la funzionalità di calendario e la funzionalità di vendita biglietti. È già stato spiegato come sia l’acquisto di biglietti che la popolazione della (futura) relativa tabella “**Biglietto**” non vengano gestiti all'interno di questo applicativo ma esternamente, tuttavia, il database è stato progettato in modo tale da permettere il corretto funzionamento anche di tali funzionalità (supponendo quindi che il servizio esterno si colleghi a questo database); proprio per questo è importante la relazione fra “**Biglietto**” e “**Calendario**”: tramite quest'ultimo è possibile risalire alle giornate di apertura di un museo, mentre tramite “**TipoBiglietto**” è possibile selezionare il biglietto desiderato. Con queste informazioni è poi possibile per il cliente procedere con l’acquisto, che verrà infine salvato in “**Biglietto**” con data di validità selezionata (ricordando che un biglietto per un museo ha validità di un solo giorno).

Inoltre, tramite questo tipo di relazione è possibile specificare anche il numero di biglietti massimi giornalieri per ogni circostanza, ovvero a seconda che sia per un museo un giorno di apertura normale che un giorno di apertura speciale.



*Statistiche:*

L’elaborazione delle statistiche rappresenta forse la parte più significativa ed importante che deve svolgere l’applicativo: esse rappresentano l’output del programma e permettono di ottenere dati di importanza economica riguardo a tutte le informazioni che gli amministratori inseriscono in input.

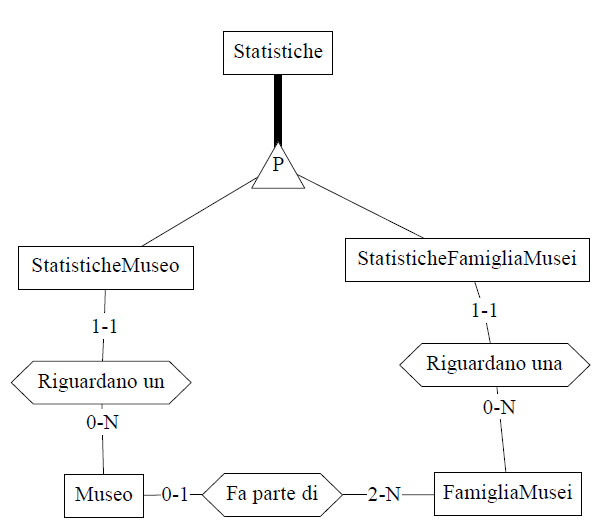
L'applicativo deve permettere di fornire dati mensili e annuali periodici per ogni museo, oltre che per ogni famiglia di musei.

Le statistiche fra un museo e una famiglia di musei sono le stesse, la differenza sta solamente nel fatto che ogni singola statistica di una famiglia di musei è ottenuta dalla somma totale della stessa di ogni singolo museo, per ogni museo che compone la famiglia.

Ad esempio: la statistica “fatturato” della famiglia di musei “A” (composta da 3 musei) sarà data dalla somma della statistica “fatturato” del museo 1, del museo 2 e del museo 3.

Deve però anche essere possibile visionare le statistiche di ogni singolo museo che compone la famiglia singolarmente.

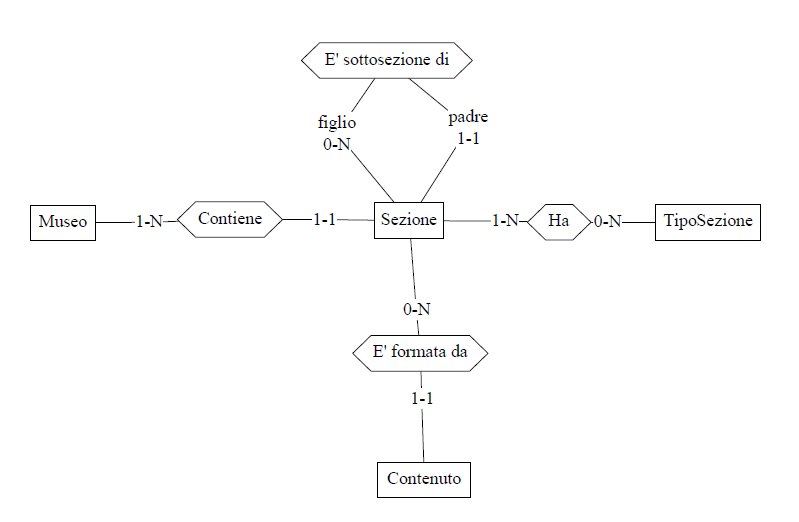
Per ottenere ciò abbiamo realizzato il seguente schema scheletro:



*Sezione:*

Ciascun museo può essere diviso in varie sezioni (“**Sezione**”), da quella principale ad altre di diverso tipo (storia, cultura, arte...). L’elaborazione di queste è stata gestita nella maniera più generale possibile in modo da includere diverse possibilità e combinazioni; Ciascuna sezione del museo potrà contenere, infatti, diverse sottosezioni (“**Sottosezione**”)(gestito tramite l’associazione *“E’ sottosezione di”*) e così via continuando in modo ricorsivo. In aggiunta, ognuna di queste apparterranno a più tipi di sezioni (“**TipoSezione**”), ad esempio “Sezione di storia e arte” sarà sia di tipo storia che di tipo arte, mentre le due singole sottosezioni di cui è formata saranno relativamente di tipo storia, e di tipo arte.

L’applicativo deve infine permettere di catalogare all’interno di ogni “Sezione” i contenuti (“**Contenuto**”) sparsi all’interno di tutte le aree del museo.



*Contenuto:*

Ultimo aspetto da trattare nello specifico è quello del contenuto.

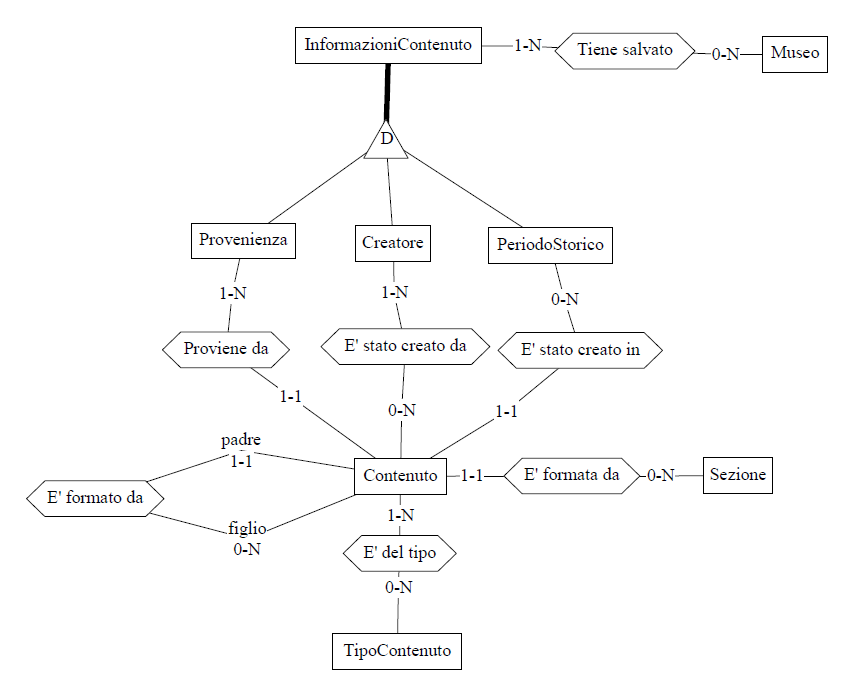
Come richiesto dal committente, un contenuto può essere formato da più parti e, a loro volta, ognuna di esse può essere formata da altre e così via: per l’implementazione di tale funzionalità è stata sufficiente una relazione unaria a “**Contenuto**” con cardinalità 1-1 per il padre e 0-N per il figlio: in questo modo, se uno più figli esistono (0-N, ovvero uno o più sottocontenuti) essi avranno necessariamente uno e uno solo padre (1-1, ovvero il contenuto del livello immediatamente sovrastante).

Inoltre, come nel caso di “**Sezione**” e “**TipoSezione**”, per le stesse ragioni e per il fatto che all’interno dell’applicativo dovrà essere possibile eseguire ricerche per tipo, ogni “Contenuto” avrà un “**TipoContenuto**”.

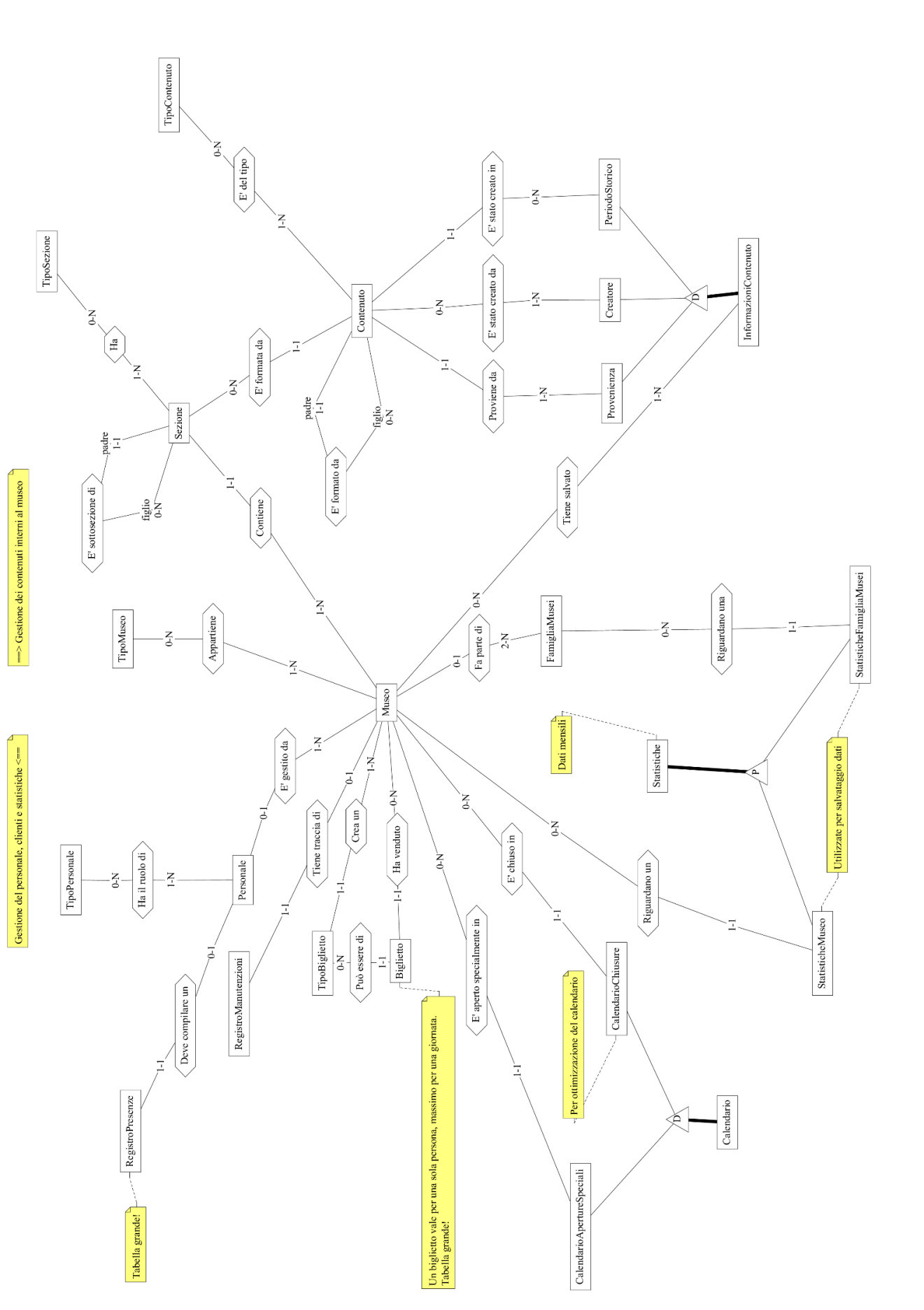
Per finire, oltre agli attributi di base che un contenuto può avere, il committente ha richiesto 3 specifiche informazioni complesse che ogni contenuto deve possedere e tramite cui si deve poter eseguire ricerche di contenuti all’interno del programma:

* “**Provenienza**”**:** definisce il luogo di origine del contenuto.
* “**Creatore**”**:** indica chi ha creato il contenuto; Nota che un contenuto potrebbe avere più autori (0-N, 0 se non se ne conosce l’autore).
* “**PeriodoStorico**”**:** rappresenta il periodo storico del contenuto; Esempio: Medioevo, 500 d.C. - 1000 d.C.

Per la realizzazione di questa specifica abbiamo utilizzato una gerarchia per ottenere un livello di astrazione che ottimizzasse le informazioni evitando ridondanze, dato che avevamo bisogno di tre specifiche entità dedicate le quali hanno anche diverse caratteristiche in comune, e non di un particolare meccanismo di estendibilità per aggiungere nuove informazioni.



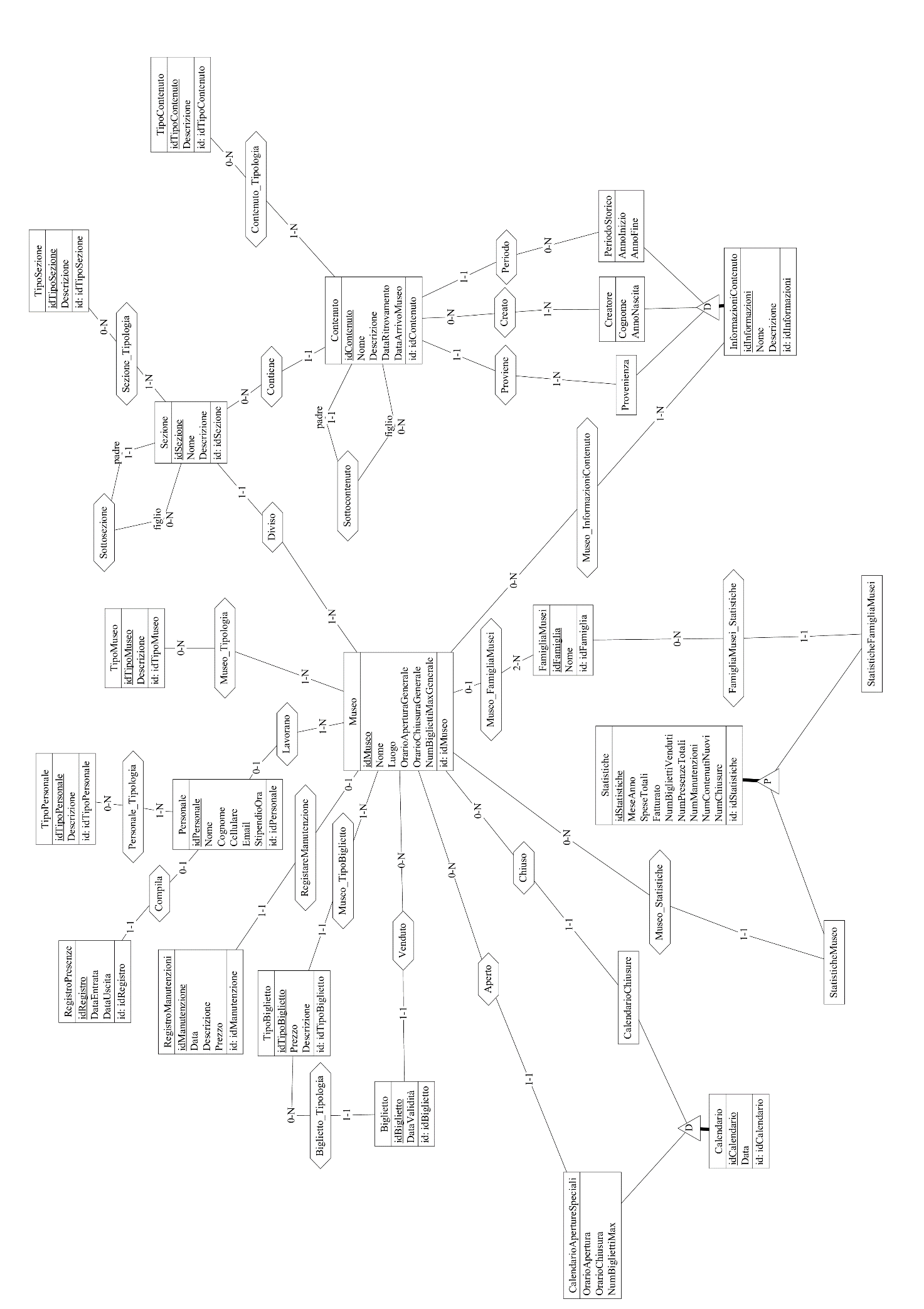
*Schema scheletro completo (generico):*



*Schema scheletro completo (finale):*

NOTA: i nomi delle associazioni per lo schema finale sono stati sostituiti per maggiore leggibilità.

Dopo aver analizzato il dominio del problema ed aver definito la struttura di specifiche soluzioni (spiegate precedentemente in dettaglio) per ogni funzionalità definita nella specifica dei requisiti, prendendo anche già nota di possibili problemi di ottimizzazione che sono sorti legati al futuro (tramite le note in giallo, dei quali ce ne occuperemo nel loro contesto ovvero nella progettazione logica), abbiamo concluso la progettazione concettuale con il seguente schema E/R, aggiungendo ogni attributo necessario al corretto funzionamento di base dell’applicativo:



**PROGETTAZIONE LOGICA**

La fase di progettazione logica prevede il passaggio dallo schema concettuale allo schema logico, ricevendo in input il diagramma E/R e fornendo in output lo schema logico relazionale; tramite quest’ultimo si andrà infine a realizzare la progettazione fisica e lo schema fisico.

Stima del volume dei dati:

NOTA: la stima del volume dei dati si basa a livello regionale italiano, da un punto di vista annuale.

|  |  |  |
| --- | --- | --- |
| **Concetto** | **Costrutto** | **Volume** |
| Museo | E | 50 |
| Diviso | R | 200 |
| Sezione | E | 200 |
| Sottosezione | R | 100 |
| Sezione\_Tipologia | R | 300 |
| TipoSezione | E | 200 |
| Contiene | R | 50.000 |
| Contenuto | E | 50.000 |
| Sottocontenuto | R | 10.000 |
| Contenuto\_Tipologia | R | 75.000 |
| TipoContenuto | E | 50.000 |
| Provenienza | E | 20.000 |
| Proviene | R | 50.000 |
| Creatore | E | 40.000 |
| Creato | R | 75.000 |
| PeriodoStorico | E | 30.000\* |
| Periodo | R | 45.000 |
| Museo\_Tipologia | R | 100 |
| TipoMuseo | E | 50 |
| Lavorano | R | 2500 |
| Personale | E | 2500 |
| Personale\_Tipologia | R | 3000 |
| TipoPersonale | E | 50 |
| Compila | R | 550.000 |
| RegistroPresenze | E | 550.000\*\* |
| RegistrareManutenzione | R | 50 |
| RegistroManutenzioni | E | 50 |
| Venduto | R | 24.000.000 |
| Biglietto | E | 24.000.000\*\*\* |
| Biglietto\_Tipologia | R | 24.000.000 |
| TipoBiglietto | E | 500 |
| Museo\_TipoBiglietto | R | 500 |
| Museo\_InformazioniContenuto | R | 90.000\*\*\*\* |
| Aperto | R | 250 |
| CalendarioApertureSpeciali | E | 250 |
| Chiuso | R | 5000 |
| CalendarioChiusure | E | 5000 |
| Statistiche | E | 636\*\*\*\*\* |
| StatisticheMuseo | E | 600 |
| Museo\_Statistiche | R | 600 |
| StatisticheFamigliaMusei | E | 36 |
| FamigliaMusei\_Statistiche | R | 36 |
| Museo\_FamigliaMusei | R | 20 |
| FamigliaMusei | E | 3 |

\*= numero comprensivo dei casi nei quali le date rappresentano un periodo storico definito (es. Medioevo) e dei casi nei quali rappresentano uno specifico anno (es. 1935-1935)

\*\*= 2500 (n. tot. Personale) \* 5 (giorni medi lavoro settimanale) \* 4 (settimane di lavoro mensili) \* 11 (mesi di lavoro annuali) (considerando in media un mese di ferie).

\*\*\*= 2000 (n. tot. Biglietti venduti giornalmente per museo) \* 6 (giorni medi apertura settimanale) \* 4 (settimane di apertura mensili) \* 10 (mesi di apertura a pagamento annuali) (considerando un mese di apertura libera (gratuita) e un mese di chiusura) \* 50 (n. medio di musei).

\*\*\*\*= somma delle informazioni dei contenuti (“Provenienza”, “Creatore”, “PeriodoStorico”).

\*\*\*\*\*= somma delle statistiche totali di StatisticheMuseo (600) e le statistiche totali di StatisticheFamigliaMusei (36).

Descrizione delle operazioni principali e stima della loro frequenza:

|  |  |  |
| --- | --- | --- |
| **Codice** | **Operazione** | **Frequenza** |
| 1 | Creazione di un museo | 5 volte all’anno |
| 2 | Modifica di un museo | 2 volte all’anno |
| 3 | Eliminazione di un museo | 1 volta all’anno |
| 4 | Creazione di una famiglia di musei | 1 volta all’anno |
| 5 | Creazione di una sezione | 3 volte all’anno |
| 6 | Modifica di una sezione | 10 volte all’anno |
| 7 | Eliminazione di una sezione | 1-2 volte all’anno |
| 8 | Aggiunta di un contenuto | 5 volte al mese |
| 9 | Modifica di un contenuto | 1 volta al mese |
| 10 | Eliminazione di un contenuto | 2 volte al mese |
| 11 | Ricerca di contenuti per provenienza | 30 volte al mese |
| 12 | Ricerca di contenuti per creatore | 30 volte al mese |
| 13 | Ricerca di contenuti per tipo | 30 volte al mese |
| 14 | Ricerca di contenuti per periodo storico | 30 volte al mese |
| 15 | Aggiunta di un creatore | 2 volte al mese |
| 16 | Modifica di un creatore | 1 volta all’anno |
| 17 | Eliminazione di un creatore | 10 volte all’anno |
| 18 | Aggiunta del personale | 2-3 volte all’anno |
| 19 | Modifica del personale | 1-2 volte all’anno |
| 20 | Eliminazione del personale | 2-3 volte all’anno |
| 21 | Aggiunta delle presenze giornaliere del personale | 20 volte al mese |
| 22 | Eliminazione delle presenze nel registro | 1 volta all’anno |
| 23 | Aggiunta di una manutenzione al registro | 2-3 volte al mese |
| 24 | Eliminazione delle manutenzioni nel registro | 1 volta all’anno |
| 25 | Creazione di un tipo di biglietto | 1 volta all’anno |
| 26 | Modifica di un tipo di biglietto | 1 volta all’anno |
| 27 | Eliminazione di tutti i biglietti acquistati | 1 volta all’anno |
| 28 | Aggiunta di un giorno di chiusura | 1 volta a settimana |
| 29 | Eliminazione di un giorno di chiusura | 1 volta al mese |
| 30 | Creazione di un giorno di apertura speciale | 1 volta al mese |
| 31 | Modifica di un giorno di apertura speciale | 1 volta al mese |
| 32 | Eliminazione di un giorno di apertura speciale | 1 volta al mese |
| 33 | Calcolo delle statistiche mensili | 1 volta al mese |
| 33.5 | Calcolo delle statistiche mensili di “FamigliaMusei” | 0-1 volte al mese |
| 34 | Calcolo delle statistiche annuali | 1 volta all’anno |

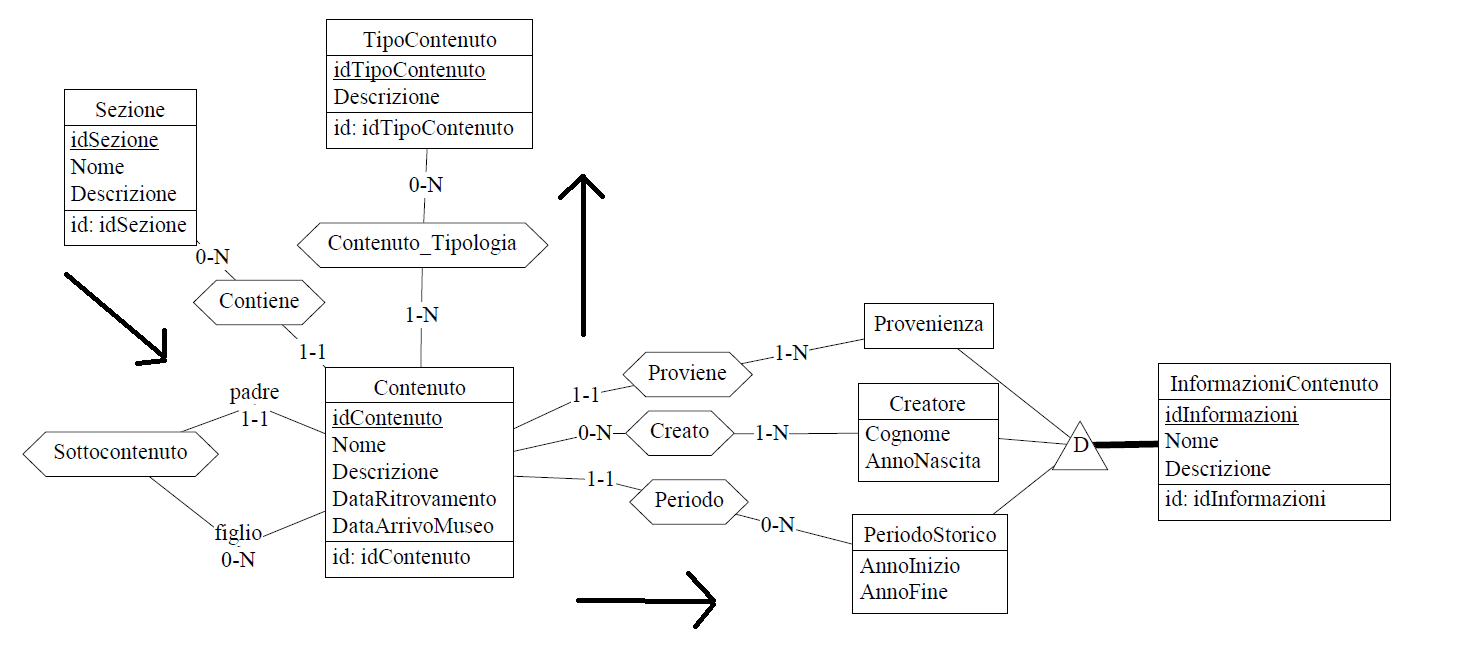
Schemi di navigazione e tabelle degli accessi:

Dopo aver determinato il volume dei dati ed aver associato a ciascuna operazione principale richiesta la propria frequenza di esecuzione, si procede determinando lo schema di navigazione di riferimento per le principali operazioni richieste e si associa ad ognuna di essa anche la relativa tavola degli accessi. Le operazioni per cui questo passaggio risulta essere banale o per cui risulta essere equivalente a quello di una operazione già vista, non verranno considerate.

NOTA: Nel calcolo degli accessi si stima come doppio il peso degli accessi in scrittura, rispetto a quelli in lettura. Nei calcoli per cui il numero di accessi è indentificato come “1-N” verrà preso in considerazione il valore “N”, ovvero il totale massimo di accessi.   
Per i casi indicati come L/S, prenderemo come peso il corrispettivo 50% di ognuno di loro (50% L, 50% S).

Le stime del totale si basano tutte su frequenza annuale.

*Aggiunta, modifica ed eliminazione di un contenuto (8-9-10)*



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Codice | Concetto | Costrutto | Accessi | Lettura/Scrittura |
| 8 | Sezione | E | 1 | L |
| 8 | Contiene | R | 1 | L |
| 8 | Contenuto | E | 1-5 | S |
| 8 | Sottocontenuto | R | 0-4 | S |
| 8 | Contenuto\_Tipologia | R | 1-10 | S |
| 8 | TipoContenuto | E | 1-10 | L/S |
| 8 | Proviene | R | 1 | L/S |
| 8 | Creato | R | 1-3 | L/S |
| 8 | Periodo | R | 1 | L/S |
| 8 | Provenienza | E | 1 | L/S |
| 8 | Creatore | E | 1-3 | L/S |
| 8 | PeriodoStorico | E | 1 | L/S |
| 9 | Sezione | E | 1 | L |
| 9 | Contiene | R | 1 | L |
| 9 | Contenuto | E | 1-5 | S |
| 9 | Sottocontenuto | R | 0-4 | S |
| 9 | Contenuto\_Tipologia | R | 1-10 | S |
| 9 | TipoContenuto | E | 1-10 | S |
| 9 | Proviene | R | 1 | S |
| 9 | Creato | R | 1-3 | S |
| 9 | Periodo | R | 1 | S |
| 9 | Provenienza | E | 1 | S |
| 9 | Creatore | E | 1-3 | S |
| 9 | PeriodoStorico | E | 1 | S |
| 10 | Sezione | E | 1 | L |
| 10 | Contiene | R | 1 | L |
| 10 | Contenuto | E | 1-5 | S |
| 10 | Sottocontenuto | R | 0-4 | S |
| 10 | Contenuto\_Tipologia | R | 1-10 | S |
| 10 | TipoContenuto | E | 1-10 | L |
| 10 | Proviene | R | 1 | S |
| 10 | Creato | R | 1-3 | S |
| 10 | Periodo | R | 1 | S |
| 10 | Provenienza | E | 1 | L |
| 10 | Creatore | E | 1-3 | L |
| 10 | PeriodoStorico | E | 1 | L |

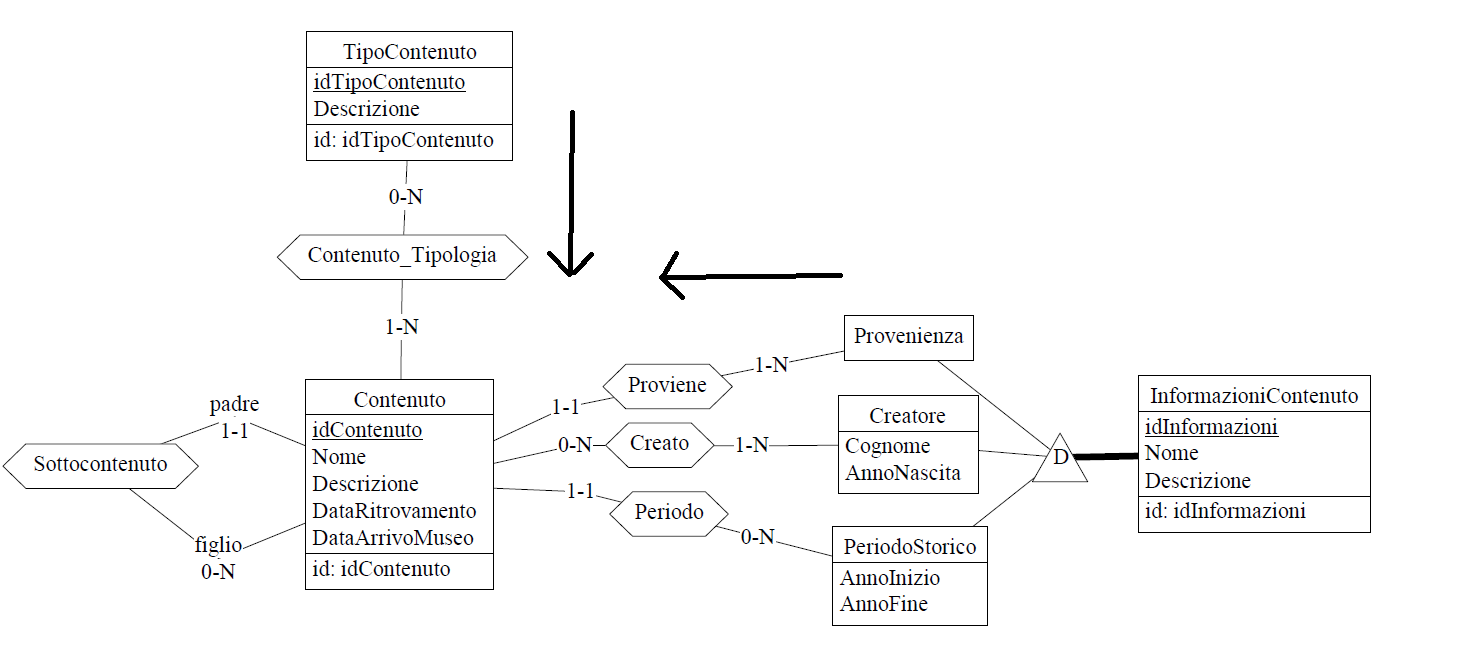
Totale op. 8: (2L + 19S + 20L/S) \* 60 (freq. annuale) = (12 + 58) \* 60 = **4200**

Totale op. 9: (2L + 39S) \* 12 (freq. annuale) = (2 + 78) \* 12 = **960**

Totale op. 9: (17L + 24S) \* 24 (freq. annuale) = (17 + 48) \* 24 = **1560**

**Totale:** 4200 + 960 + 1560 = **4080** (numero di accessi annuali max all’area dei contenuti per singolo museo)

*Ricerca di contenuti per provenienza, creatore, tipo e periodo storico (11-12-13-14)*



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Codice | Concetto | Costrutto | Accessi | Lettura/Scrittura |
| 11 | Contenuto | E | 1-5 | L |
| 11 | Sottocontenuto | R | 0-4 | L |
| 11 | Contenuto\_Tipologia | R | 1-10 | L |
| 11 | TipoContenuto | E | 1-10 | L |
| 11 | Proviene | R | 1 | L |
| 11 | Creato | R | 1-3 | L |
| 11 | Periodo | R | 1 | L |
| 11 | Provenienza | E | 1 | L |
| 11 | Creatore | E | 1-3 | L |
| 11 | PeriodoStorico | E | 1 | L |
| 12 | Contenuto | E | 1-5 | L |
| 12 | Sottocontenuto | R | 0-4 | L |
| 12 | Contenuto\_Tipologia | R | 1-10 | L |
| 12 | TipoContenuto | E | 1-10 | L |
| 12 | Proviene | R | 1 | L |
| 12 | Creato | R | 1-3 | L |
| 12 | Periodo | R | 1 | L |
| 12 | Provenienza | E | 1 | L |
| 12 | Creatore | E | 1-3 | L |
| 12 | PeriodoStorico | E | 1 | L |

NOTA: le ricerche per tipo e periodo storico (op.13-14) sono equivalenti alle altre.

Totale op. 11: 39L \* 360 (frequenza annuale) = 39 \* 360 = **14040**

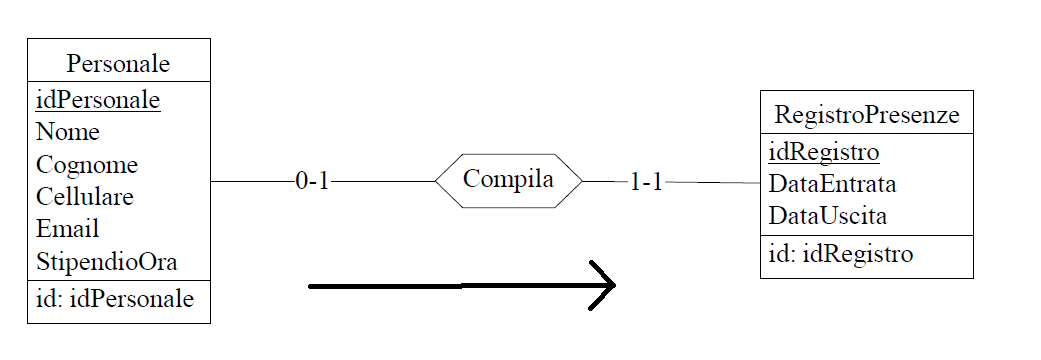
Totale op. 12: 39L \* 360 (frequenza annuale) =39 \* 360 = **14040**

Totale op. 13 = **14040**

Totale op. 14 = **14040**

**Totale:** 14040 \* 4 = **56160** (numero di accessi annuali max per singolo museo per le operazioni di ricerca)

*Aggiunta ed eliminazione delle presenze nel registro (21-22)*

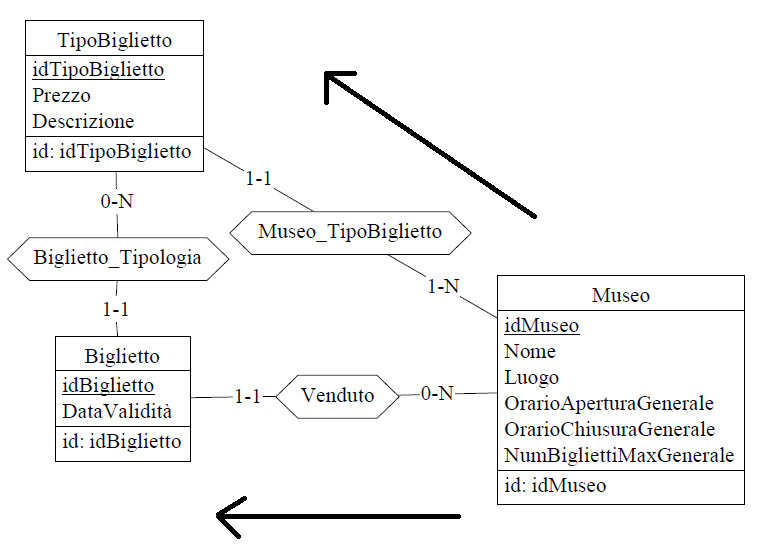


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Codice | Concetto | Costrutto | Accessi | Lettura/Scrittura |
| 21 | Personale | E | 1 | L |
| 21 | Compila | R | 1 | L |
| 21 | RegistroPresenze | E | 1 | S |
| 22 | Personale | E | 1 | L |
| 22 | Compila | R | 1 | L |
| 22 | RegistroPresenze | E | 1 | S |

Totale op. 21 = (2L + S) \* 240 = 4 \* 240 = **960** (numero di accessi annuali per singolo museo per l’aggiunta del personale al registro presenze)

Totale op. 22 = (2L + S) \* 1 = **4** (numero di accessi annuali per singolo museo per l’eliminazione del registro presenze)

*Aggiunta e modifica di un tipo di biglietto ed eliminazione di tutti i biglietti acquistati (25-26-27)*



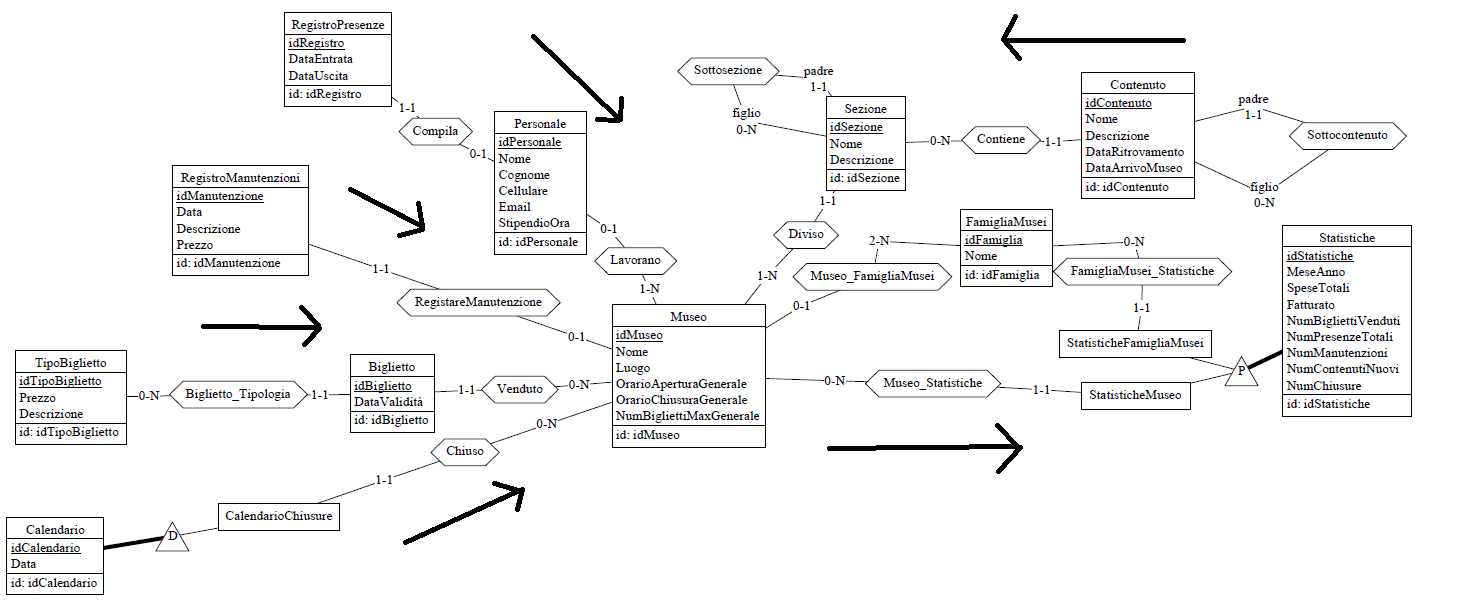
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Codice | Concetto | Costrutto | Accessi | Lettura/Scrittura |
| 25 | Museo | E | 1 | L |
| 25 | Venduto | R | 1 | L |
| 25 | Biglietto | E | 1 | S |
| 25 | Biglietto\_Tipologia | R | 1 | L/S |
| 25 | TipoBiglietto | E | 1 | L/S |
| 26 | Museo | E | 1 | L |
| 26 | Museo\_TipoBiglietto | R | 1 | L |
| 26 | TipoBiglietto | E | 1 | S |
| 27 | Museo | E | 1 | L |
| 27 | Venduto | R | 0-480.000 | L |
| 27 | Biglietto | E | 0- 480.000 | S |
| 27 | Biglietto\_Tipologia | R | 0- 480.000 | S |
| 27 | TipoBiglietto | E | 0- 480.000 | L |

Totale op. 25: (2L + 2S) \* 1 = 2 + 4= **6** (aggiunta di un biglietto)

Totale op. 26: (2L + S) \* 1 = 2 + 2= **4** (modifica di un tipo di biglietto)

Totale op. 27: = (960.001L + 960.000S) \* 1 = 960.001 + 1.920.000 = **2.880.001** (eliminazione di tutti i biglietti acquistati per un museo)

*Calcolo delle statistiche mensili ed annuali (33-33.5-34)*



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Codice | Concetto | Costrutto | Accessi | Lettura/Scrittura |
| 33 | Museo | E | 1 | L |
| 33 | Chiuso | R | 0-5 | L |
| 33 | CalendarioChiusure | E | 0-5 | L |
| 33 | Venduto | R | 40.000 | L |
| 33 | Biglietto | E | 40.000 | L |
| 33 | Biglietto\_Tipologia | R | 40.000 | L |
| 33 | TipoBiglietto | E | 40.000 | L |
| 33 | RegistrareManutenzione | R | 1 | L |
| 33 | RegistroManutenzione | E | 1 | L |
| 33 | Lavorano | R | 1-50 | L |
| 33 | Personale | E | 1-50 | L |
| 33 | Compila | R | 1-50 | L |
| 33 | RegistroPresenze | E | 1-50 | L |
| 33 | Diviso | R | 1-4 | L |
| 33 | Sezione | E | 1-4 | L |
| 33 | Sottosezione | R | 0-2 | L |
| 33 | Contiene | R | 1-1000 | L |
| 33 | Contenuto | E | 1-200 | L |
| 33 | Sottocontenuto | R | 0-2 | L |
| 33 | Museo\_Statistiche | R | 1 | S |
| 33 | StatisticheMuseo | E | 1 | S |
| 33 | Statistiche | E | 1 | S |
| 33.5 | FamigliaMusei | E | 1 | L |
| 33.5 | Museo\_FamigliaMusei | R | 2-6 | L |
| 33.5 | Museo | E | 2-6 | L |
| 33.5 | FamigliaMusei\_Statistiche | R | 1 | S |
| 33.5 | StatisticheFamigliaMusei | E | 1 | S |
| 33.5 | Museo\_Statistiche | R | 2-6 | L |
| 33.5 | StatisticheMuseo | E | 2-6 | L |
| 33.5 | Statistiche | E | 3-7 | S |
| 34 | StatisticheFamigliaMusei | E | 12 | L |
| 34 | StatisticheMuseo | E | 12 | L |
| 34 | Statistiche | E | 12 | L |

Totale op. 33: (161.425L + 3S) \* 12 = **1.937.172** (n. accessi annuali max per museo per il calcolo delle statistiche mensili)

Totale op. 33.5: (25L + 9S) \* 6 = **258** (n. accessi annuali max per famiglia di musei per il calcolo delle statistiche mensili)

*NOTA (33.5): essendo la frequenza mensile 0 oppure 1, per tradurla in una annuale l’abbiamo considerata come 0.5 e quindi moltiplicata per 12, ottenendo 6.*

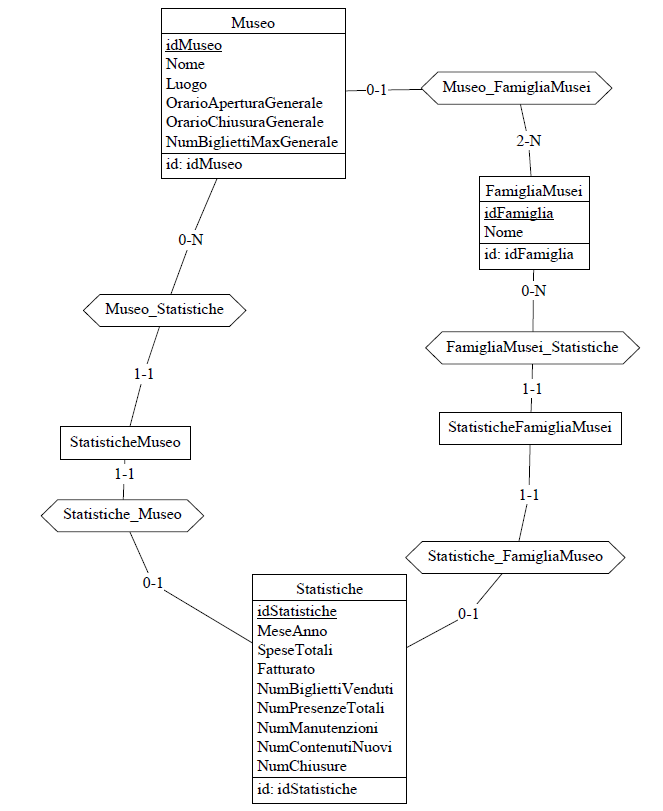
Totale op. 34: = 36L \* 1 = **36** (n. accessi annuali max per singolo museo e singola famiglia di musei per il calcolo delle statistiche annuali)

Raffinamento dello schema:

*Eliminazione delle gerarchie:*

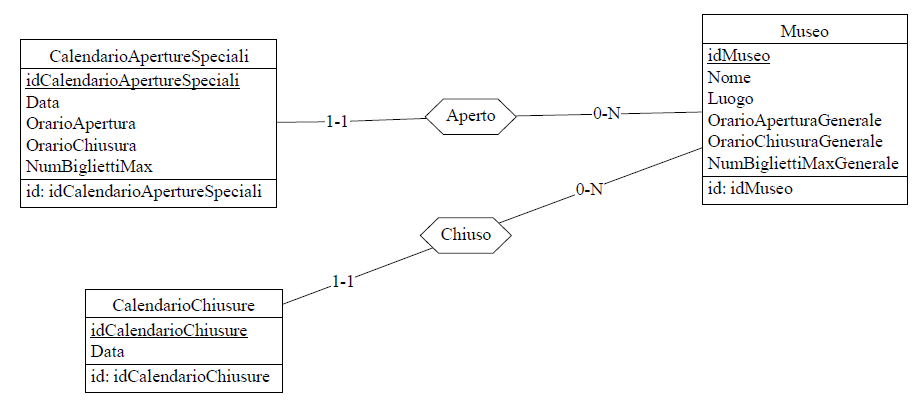
Nello schema E/R compaiono in tutto tre gerarchie da eliminare.

Per la gerarchia “**Statistiche**”(t, e) si decide come soluzione di creare nuove associazioni da collegare all’entità padre “**Statistiche**”. Questa scelta è stata fatta poiché il numero di attributi del padre è abbastanza grande e quindi non converrebbe copiarli nei figli (quindi no collasso verso il basso). Per quanto riguarda un collasso verso l’alto abbiamo ritenuto inefficiente collegare al padre tutte le associazioni dei figli.

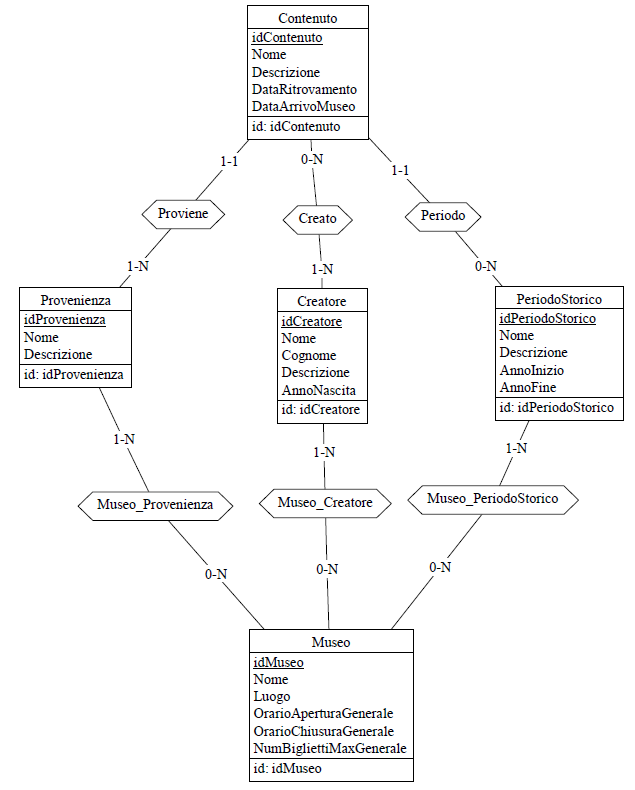


Per quanto riguarda le altre due gerarchie, entrambe solo esclusive e considerato il basso numero di attributi dei rispettivi padri, abbiamo deciso di applicare il collasso verso il basso.

La prima è la gerarchia “**Museo**”:



La seconda è “**InformazioniContenuto**”:



*Eliminazione degli attributi composti:*

Nello schema E/R non è stato individuato **nessun** tipo di attributo composto, quindi non vi sarà necessaria l’eliminazione.

*Scelta delle chiavi primarie:*

Nello schema E/R proposto sono già evidenziate tutte le chiave primarie. Da notare che sono state scelte tutte come identificatori numeri per facilitare la ricerca ed anche l’ordinamento del database.

*Chiavi esterne:*

Laddove necessario verranno utilizzate, per semplicità, chiavi esterne con lo stesso nome della chiave primaria a cui fanno riferimento (ad eccezione dei pochi casi in cui si potrebbe creare fraintendimento).

La scelta delle chiavi esterne viene specificata in dettaglio nella fase di “Traduzione di entità e associazioni in relazioni”.

Analisi delle ridondanze:

Lo schema logico relazionale così ottenuto presenta un’unica ridondanza: la tabella “**Statistiche**”. L'utilizzo di questa ridondanza permette il salvataggio delle statistiche mensili, in modo tale da non doverle ricalcolare per il calcolo delle statistiche annuali, oltre a svolgere una funzione utilitaria di archivio dati, nel caso in cui si vogliano revisionare le statistiche mensili passate. Inoltre, questo sistema permette la cancellazione di tutti i record nelle tabelle “**Biglietto**” e “**RegistroPresenze**” annualmente in maniera sicura.

Tabella degli accessi con la ridondanza:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Codice | Concetto | Costrutto | Accessi | Lettura/Scrittura |
| 34 | StatisticheFamigliaMusei | E | 12 | L |
| 34 | StatisticheMuseo | E | 12 | L |
| 34 | Statistiche | E | 12 | L |

Tabella degli accessi senza la ridondanza:

\*Per ottenere il risultato dell’operazione 34 sarà necessario calcolare le operazioni 33 e 33.5 per 12 volte.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Codice | Concetto | Costrutto | Accessi | Lettura/Scrittura |
| 33 | Museo | E | 12 | L |
| 33 | Chiuso | R | 0-60 | L |
| 33 | CalendarioChiusure | E | 0-60 | L |
| 33 | Venduto | R | 480.000 | L |
| 33 | Biglietto | E | 480.000 | L |
| 33 | Biglietto\_Tipologia | R | 480.000 | L |
| 33 | TipoBiglietto | E | 480.000 | L |
| 33 | RegistrareManutenzione | R | 12 | L |
| 33 | RegistroManutenzione | E | 12 | L |
| 33 | Lavorano | R | 12-600 | L |
| 33 | Personale | E | 12-600 | L |
| 33 | Compila | R | 12-600 | L |
| 33 | RegistroPresenze | E | 12-600 | L |
| 33 | Diviso | R | 12-48 | L |
| 33 | Sezione | E | 12-48 | L |
| 33 | Sottosezione | R | 0-24 | L |
| 33 | Contiene | R | 1-12000 | L |
| 33 | Contenuto | E | 1-2400 | L |
| 33 | Sottocontenuto | R | 0-24 | L |
| 33 | Museo\_Statistiche | R | 12 | S |
| 33 | StatisticheMuseo | E | 12 | S |
| 33 | Statistiche | E | 12 | S |
| 33.5 | FamigliaMusei | E | 12 | L |
| 33.5 | Museo\_FamigliaMusei | R | 24-72 | L |
| 33.5 | Museo | E | 24-72 | L |
| 33.5 | FamigliaMusei\_Statistiche | R | 12 | S |
| 33.5 | StatisticheFamigliaMusei | E | 12 | S |
| 33.5 | Museo\_Statistiche | R | 24-72 | L |
| 33.5 | StatisticheMuseo | E | 24-72 | L |
| 33.5 | Statistiche | E | 36-84 | S |

Si può notare come in entrambi i casi la ridondanza garantisca un notevole vantaggio in termini di accessi e calcoli risparmiati, per questo verrà mantenuta.

Traduzione di entità e associazioni in relazioni:

//Museo

**Museo**(idMuseo, Nome, Luogo, OrarioAperturaGenerale, OrarioChiusuraGenerale, NumBigliettiMaxGenerale, idFamiglia[0-1])

FK: idFamiglia REFERENCES **FamigliaMusei**

//Museo\_Tipologia

**Museo\_Tipologia**(idMuseo, idTipoMuseo)

FK: idMuseo REFERENCES **Museo**

FK: idTipoMuseo REFERENCES **TipoMuseo**

//TipoMuseo

**TipoMuseo**(idTipoMuseo, Descrizione)

//FamigliaMusei

**FamigliaMusei**(idFamiglia, Nome)

//Personale

**Personale**(idPersonale, Nome, Cognome, Cellulare, Email, StipendioOra, idMuseo[0-1])

FK: idMuseo REFERENCES **Museo**

//Personale\_Tipologia

**Personale\_Tipologia**(idPersonale, idTipoPersonale)

FK: idPersonale REFERENCES **Personale**

FK: idTipoPersonale REFERENCES **TipoPersonale**

//TipoPersonale

**TipoPersonale**(idTipoPersonale, Descrizione)

//RegistroPresenze

**RegistroPresenze**(idRegistro, DataEntrata, DataUscita, idPersonale)

FK: idPersonale REFERENCES **Personale**

//RegistroManutenzioni

**RegistroManutenzioni**(idManutenzione, Data, Descrizione, Prezzo, idMuseo)

FK: idMuseo REFERENCES **Museo**

//Biglietto

**Biglietto**(idBiglietto, DataVadilita, PrezzoAcquisto, idMuseo, idTipoBiglietto)

FK: idMuseo REFERENCES **Museo**

FK: idTipoBiglietto REFERENCES **TipoBiglietto**

//TipoBiglietto

**TipoBiglietto**(idTipoBiglietto, Prezzo, Descrizione, idMuseo)

FK: idMuseo REFERENCES **Museo**

//Calendario Aperture Speciali

**CalendarioApertureSpeciali**(idCalendarioApertureSpeciali, Data, OrarioApertura, OrarioChiusura, NumBigliettiMax, idMuseo)

FK: idMuseo REFERENCES **Museo**

//Calendario Chiusure

**CalendarioChiusure**(idCalendarioChiusure, Data, idMuseo)

FK: idMuseo REFERENCES **Museo**

//Sezione

**Sezione**(idSezione, Nome, Descrizione, idSezionePadre)

FK: idSezionePadre REFERENCES **Sezione**

//Sezione\_Tipologia

**Sezione\_Tipologia**(idSezione, idTipoSezione)

FK: idTipoSezione REFERENCES **TipoSezione**

FK: idSezione REFERENCES **Sezione**

//TipoSezione

**TipoSezione**(idTipoSezione, Descrizione)

//Contenuto

**Contenuto**(idContenuto, Nome, Descrizione, DataRitrovamento, DataArrivoMuseo, idContenutoPadre, idProvenienza, idPeriodoStorico, idSezione)

FK: idContenutoPadre REFERENCES **Contenuto**

FK: idProvenienza REFERENCES **Provenienza**

FK: idPeriodoStorico REFERENCES **PeriodoStorico**

FK: idSezione REFERENCES **Sezione**

//Contenuto\_Tipologia

**Contenuto\_Tipologia**(idContenuto, idTipoContenuto)

FK: idTipoContenuto REFERENCES **TipoContenuto**

FK: idContenuto REFERENCES **Contenuto**

//TipoContenuto

**TipoContenuto**(idTipoContenuto, Descrizione)

//Provenienza

**Provenienza**(idProvenienza, Nome, Descrizione)

//Museo\_Provenienza

**Museo\_Provenienza**(idProvenienza, idMuseo)

FK: idProvenienza REFERENCES **Provenienza**

FK: idMuseo REFERENCES **Museo**

//Creato

**Creato**(idContenuto, idCreatore)

FK: idContenuto REFERENCES **Contenuto**

FK: idCreatore REFERENCES **Creatore**

//Creatore

**Creatore**(idCreatore, Nome, Cognome, Descrizione, AnnoNascita)

//Museo\_Creatore

**Museo\_Creatore**(idCreatore, idMuseo)

FK: idCreatore REFERENCES **Creatore**

FK: idMuseo REFERENCES **Museo**

//PeriodoStorico

**PeriodoStorico**(idPeriodoStorico, Nome, Descrizione, AnnoInizio, AnnoFine)

//Museo\_PeriodoStorico

**Museo\_PeriodoStorico**(idPeriodoStorico, idMuseo)

FK: idPeriodoStorico REFERENCES **PeriodoStorico**

FK: idMuseo REFERENCES **Museo**

//Statistiche

**Statistiche**(idStatistiche, MeseAnno, SpeseTotali, Fatturato, NumBigliettiVenduti, NumPresenzeTotali, NumManutenzioni, NumContenutiNuovi, NumChiusure)

//StatisticheMuseo

**StatisticheMuseo**(idStatistiche, idMuseo)

FK: idStatistiche REFERENCES **Statistiche**

FK: idMuseo REFERENCES **Museo**

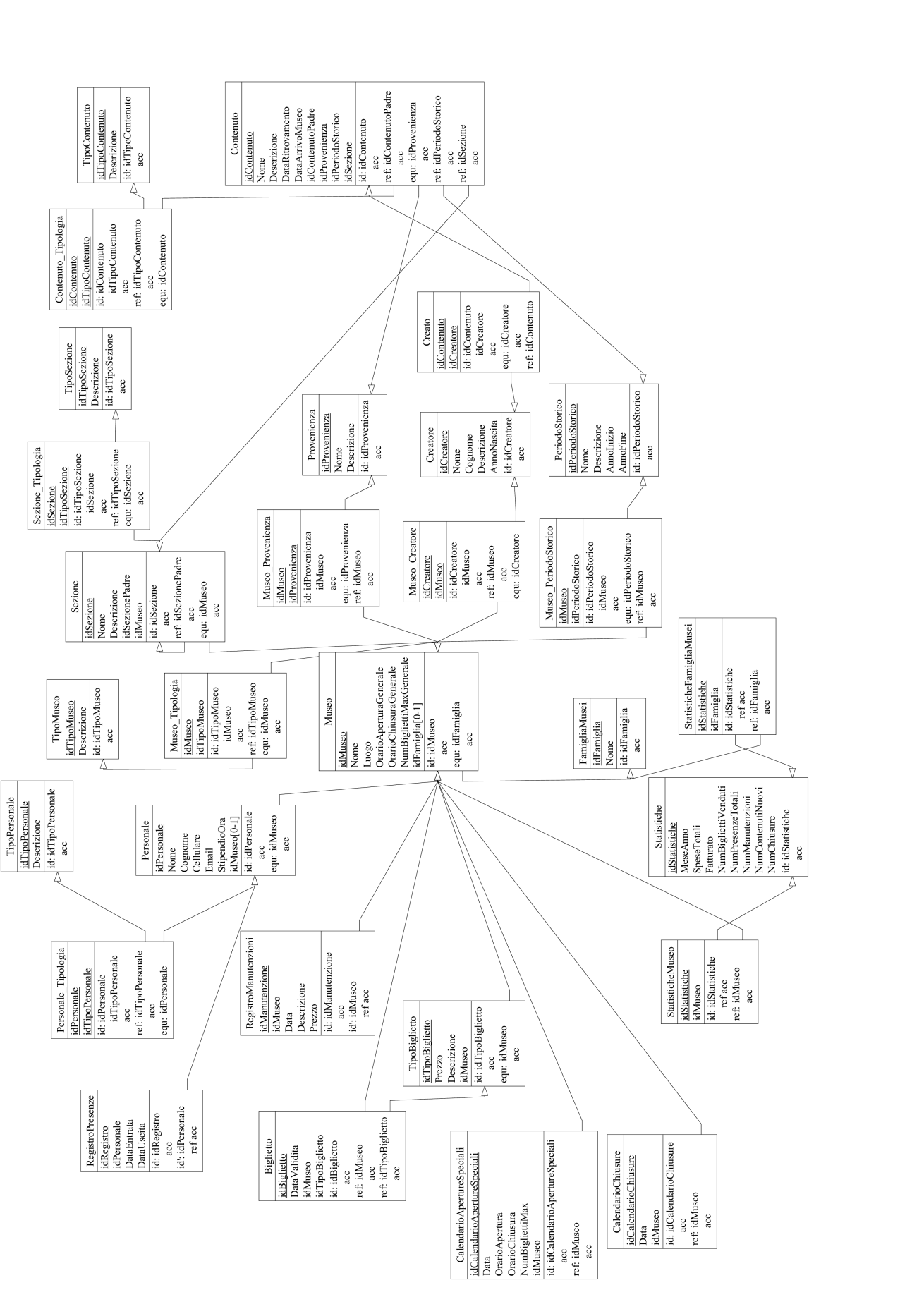
//StatisticheFamigliaMusei

**StatisticheFamigliaMusei**(idStatistiche, idFamiglia)

FK: idStatistiche REFERENCES **Statistiche**

FK: idFamiglia REFERENCES **FamigliaMusei**

Schema relazionale finale:



Costruzione delle tabelle in linguaggio SQL:

**//Biglietto**

create table Biglietto (

idBiglietto int not null primary key identity(1,1),

DataValidita date not null,

PrezzoAcquisto float not null,

idMuseo int not null,

idTipoBiglietto int not null);

**//Calendario aperture speciali**

create table CalendarioApertureSpeciali (

idCalendarioApertureSpeciali int not null primary key identity(1,1),

Data date not null,

OrarioApertura time not null,

OrarioChiusura time not null,

NumBigliettiMax int,

idMuseo int not null,

unique(Data, idMuseo));

**//Calendario chiusure**

create table CalendarioChiusure (

idCalendarioChiusure int not null primary key identity(1,1),

Data date not null,

idMuseo int not null,

unique(Data, idMuseo));

**//Contenuto**

create table Contenuto (

idContenuto int not null primary key identity(1,1),

Nome nvarchar(255) not null,

Descrizione nvarchar(512) not null,

DataRitrovamento date not null,

DataArrivoMuseo date not null,

idContenutoPadre int,

idProvenienza int not null,

idPeriodoStorico int not null,

idSezione int not null);

**//Contenuto\_Tipologia**

create table Contenuto\_Tipologia (

idContenuto int not null,

idTipoContenuto int not null,

primary key (idContenuto, idTipoContenuto));

**//Creato**

create table Creato (

idContenuto int not null,

idCreatore int not null,

primary key (idContenuto, idCreatore));

**//Creatore**

create table Creatore (

idCreatore int not null primary key identity(1,1),

Nome nvarchar(255) not null,

Cognome nvarchar(255) not null,

Descrizione nvarchar(512) not null,

AnnoNascita int not null);

**//FamigliaMusei**

create table FamigliaMusei (

idFamiglia int not null primary key identity(1,1),

Nome nvarchar(255) not null);

**//Museo**

create table Museo (

idMuseo int not null primary key identity(1,1),

Nome nvarchar(255) not null,

Luogo nvarchar(255) not null,

OrarioAperturaGenerale time not null,

OrarioChiusuraGenerale time not null,

NumBigliettiMaxGenerale int,

idFamiglia int);

**//Museo\_Creatore**

create table Museo\_Creatore (

idCreatore int not null,

idMuseo int not null,

primary key (idCreatore, idMuseo));

**//Museo\_PeriodoStorico**

create table Museo\_PeriodoStorico (

idMuseo int not null,

idPeriodoStorico int not null,

primary key (idPeriodoStorico, idMuseo));

**//Provenienza**

create table Museo\_Provenienza (

idMuseo int not null,

idProvenienza int not null,

primary key (idProvenienza, idMuseo));

**//Museo\_Tipologia**

create table Museo\_Tipologia (

idMuseo int not null,

idTipoMuseo int not null,

primary key (idTipoMuseo, idMuseo));

**//PeriodoStorico**

create table PeriodoStorico (

idPeriodoStorico int not null primary key identity(1,1),

Nome nvarchar(255) not null,

Descrizione nvarchar(512) not null,

AnnoInizio int not null,

AnnoFine int not null);

**//Personale**

create table Personale (

idPersonale int not null primary key identity(1,1),

Nome nvarchar(255) not null,

Cognome nvarchar(255) not null,

Cellulare nvarchar(255) not null,

Email nvarchar(255) not null,

StipendioOra float not null,

idMuseo int);

**//Personale\_Tipologia**

create table Personale\_Tipologia (

idPersonale int not null,

idTipoPersonale int not null,

primary key (idPersonale, idTipoPersonale));

**//Provenienza**

create table Provenienza (

idProvenienza int not null primary key identity(1,1),

Nome nvarchar(255) not null,

Descrizione nvarchar(512) not null);

**//RegistroManutenzioni**

create table RegistroManutenzioni (

idManutenzione int not null primary key identity(1,1),

idMuseo int not null,

Data date not null,

Descrizione nvarchar(512) not null,

Prezzo float not null);

**//RegistroPresenze**

create table RegistroPresenze (

idRegistro int not null primary key identity(1,1),

idPersonale int not null,

DataEntrata datetime2 not null,

DataUscita datetime2 not null);

**//Sezione**

create table Sezione (

idSezione int not null primary key identity(1,1),

Nome nvarchar(255) not null,

Descrizione nvarchar(512) not null,

idSezionePadre int,

idMuseo int not null);

**//Sezione\_Tipologia**

create table Sezione\_Tipologia (

idSezione int not null,

idTipoSezione int not null,

primary key (idTipoSezione, idSezione));

**//Statistiche**

create table Statistiche (

idStatistiche int not null primary key identity(1,1),

MeseAnno date not null,

SpeseTotali float not null,

Fatturato float not null,

NumBigliettiVenduti int not null,

NumPresenzeTotali int not null,

NumManutenzioni int not null,

NumContenutiNuovi int not null,

NumChiusure int not null);

**//StatisticheFamigliaMusei**

create table StatisticheFamigliaMusei (

idStatistiche int not null,

idFamiglia int not null,

primary key (idStatistiche));

**//StatisticheMuseo**

create table StatisticheMuseo (

idStatistiche int not null,

idMuseo int not null,

primary key (idStatistiche));

**//TipoBiglietto**

create table TipoBiglietto (

idTipoBiglietto int not null primary key identity(1,1),

Nome nvarchar(255) not null,

Prezzo float not null,

Descrizione nvarchar(512) not null,

idMuseo int not null);

**//TipoContenuto**

create table TipoContenuto (

idTipoContenuto int not null primary key identity(1,1),

Descrizione nvarchar(512) not null unique);

**//TipoMuseo**

create table TipoMuseo (

idTipoMuseo int not null primary key identity(1,1),

Descrizione nvarchar(512) not null unique);

**//TipoPersonale**

create table TipoPersonale (

idTipoPersonale int not null primary key identity(1,1),

Descrizione nvarchar(512) not null unique);

**//TipoSezione**

create table TipoSezione (

idTipoSezione int not null primary key identity(1,1),

Descrizione nvarchar(512) not null unique);

**//Inserimento chiavi esterne (Alter Table)**

alter table Biglietto add constraint REF\_Bigli\_Museo\_FK

foreign key (idMuseo)

references Museo (idMuseo);

alter table Biglietto add constraint REF\_Bigli\_TipoB\_FK

foreign key (idTipoBiglietto)

references TipoBiglietto (idTipoBiglietto) on delete cascade;

alter table CalendarioApertureSpeciali add constraint REF\_Calen\_Museo\_1\_FK

foreign key (idMuseo)

references Museo (idMuseo) on delete cascade;

alter table CalendarioChiusure add constraint REF\_Calen\_Museo\_FK

foreign key (idMuseo)

references Museo (idMuseo) on delete cascade;

alter table Contenuto add constraint EQU\_Conte\_Prove\_FK

foreign key (idProvenienza)

references Provenienza (idProvenienza);

alter table Contenuto add constraint REF\_Conte\_Perio\_FK

foreign key (idPeriodoStorico)

references PeriodoStorico (idPeriodoStorico);

alter table Contenuto add constraint REF\_Conte\_Sezio\_FK

foreign key (idSezione)

references Sezione (idSezione) on delete cascade;

alter table Contenuto\_Tipologia add constraint REF\_Conte\_TipoC\_FK

foreign key (idTipoContenuto)

references TipoContenuto (idTipoContenuto) on delete cascade;

alter table Contenuto\_Tipologia add constraint EQU\_Conte\_Conte

foreign key (idContenuto)

references Contenuto (idContenuto) on delete cascade;

alter table Creato add constraint EQU\_Creat\_Creat\_FK

foreign key (idCreatore)

references Creatore (idCreatore) on delete cascade;

alter table Creato add constraint REF\_Creat\_Conte

foreign key (idContenuto)

references Contenuto (idContenuto);

alter table Museo add constraint EQU\_Museo\_Famig\_FK

foreign key (idFamiglia)

references FamigliaMusei (idFamiglia) on delete cascade;

alter table Museo\_Creatore add constraint REF\_Museo\_Museo\_2\_FK

foreign key (idMuseo)

references Museo (idMuseo) on delete cascade;

alter table Museo\_Creatore add constraint EQU\_Museo\_Creat

foreign key (idCreatore)

references Creatore (idCreatore) on delete cascade;

alter table Museo\_PeriodoStorico add constraint EQU\_Museo\_Perio

foreign key (idPeriodoStorico)

references PeriodoStorico (idPeriodoStorico) on delete cascade;

alter table Museo\_PeriodoStorico add constraint REF\_Museo\_Museo\_1\_FK

foreign key (idMuseo)

references Museo (idMuseo) on delete cascade;

alter table Museo\_Provenienza add constraint EQU\_Museo\_Prove

foreign key (idProvenienza)

references Provenienza (idProvenienza) on delete cascade;

alter table Museo\_Provenienza add constraint REF\_Museo\_Museo\_FK

foreign key (idMuseo)

references Museo (idMuseo) on delete cascade;

alter table Museo\_Tipologia add constraint REF\_Museo\_TipoM

foreign key (idTipoMuseo)

references TipoMuseo (idTipoMuseo) on delete cascade;

alter table Museo\_Tipologia add constraint EQU\_Museo\_Museo\_FK

foreign key (idMuseo)

references Museo (idMuseo) on delete cascade;

alter table Personale add constraint EQU\_Perso\_Museo\_FK

foreign key (idMuseo)

references Museo (idMuseo) on delete cascade;

alter table Personale\_Tipologia add constraint REF\_Perso\_TipoP\_FK

foreign key (idTipoPersonale)

references TipoPersonale (idTipoPersonale) on delete cascade;

alter table Personale\_Tipologia add constraint EQU\_Perso\_Perso

foreign key (idPersonale)

references Personale (idPersonale) on delete cascade;

alter table RegistroManutenzioni add constraint SID\_Regis\_Museo\_FK

foreign key (idMuseo)

references Museo (idMuseo) on delete cascade;

alter table RegistroPresenze add constraint SID\_Regis\_Perso\_FK

foreign key (idPersonale)

references Personale (idPersonale) on delete cascade;

alter table Sezione add constraint REF\_Sezio\_Sezio\_FK

foreign key (idSezionePadre)

references Sezione (idSezione);

alter table Sezione add constraint EQU\_Sezio\_Museo\_FK

foreign key (idMuseo)

references Museo (idMuseo) on delete cascade;

alter table Sezione\_Tipologia add constraint REF\_Sezio\_TipoS

foreign key (idTipoSezione)

references TipoSezione (idTipoSezione) on delete cascade;

alter table Sezione\_Tipologia add constraint EQU\_Sezio\_Sezio\_FK

foreign key (idSezione)

references Sezione (idSezione) on delete cascade;

alter table StatisticheFamigliaMusei add constraint ID\_Stati\_Stati\_1\_FK

foreign key (idStatistiche)

references Statistiche (idStatistiche) on delete cascade;

alter table StatisticheFamigliaMusei add constraint REF\_Stati\_Famig\_FK

foreign key (idFamiglia)

references FamigliaMusei (idFamiglia) on delete cascade;

alter table StatisticheMuseo add constraint ID\_Stati\_Stati\_FK

foreign key (idStatistiche)

references Statistiche (idStatistiche) on delete cascade;

alter table StatisticheMuseo add constraint REF\_Stati\_Museo\_FK

foreign key (idMuseo)

references Museo (idMuseo) on delete cascade;

alter table TipoBiglietto add constraint EQU\_TipoB\_Museo\_FK

foreign key (idMuseo)

references Museo (idMuseo) on delete cascade;

Traduzione delle operazioni in query SQL:

*Aggiunta, modifica ed eliminazione di un contenuto (8-9-10)*

8)

INSERT INTO Contenuto(Nome, Descrizione, DataRitrovamento, DataArrivoMuseo, idContenutoPadre, idProvenienza, idPeriodoStorico, idSezione) VALUES(…);

9)

UPDATE Contenuto  
SET …  
WHERE idContenuto = value;

10)

Per la delete del contenuto si usa la procedura DeleteContenuto per eliminarne anche i sotto contenuti

ALTER PROCEDURE [dbo].[DeleteContenuto] (@idContenuto INT)  
AS  
BEGIN  
DECLARE @ids TABLE (idContenuto INT)  
DECLARE @tmp TABLE (idContenuto INT)  
DELETE Contenuto WHERE idContenuto = @idContenuto;  
INSERT INTO @tmp(idContenuto)  
SELECT idContenuto   
FROM Contenuto   
WHERE idContenutoPadre = @idContenuto;  
INSERT INTO @ids  
SELECT idContenuto  
FROM @tmp;  
WHILE (SELECT COUNT(idContenuto) FROM @tmp) > 0  
BEGIN  
DELETE @tmp;  
DELETE Contenuto WHERE idContenuto IN (SELECT idContenuto FROM @ids);  
INSERT INTO @tmp(idContenuto)  
SELECT idContenuto   
FROM Contenuto   
WHERE idContenutoPadre IN (SELECT idContenuto FROM @ids);  
DELETE @ids;  
INSERT INTO @ids  
SELECT idContenuto  
FROM @tmp;  
END  
END

*Ricerca di contenuti per provenienza, creatore, tipo e periodo storico (11-12-13-14)*

SELECT Contenuto.\*  
FROM Contenuto INNER JOIN Sezione ON Contenuto.idSezione = Sezione.idSezione  
INNER JOIN Creato ON Contenuto.idContenuto = Creato.idContenuto  
INNER JOIN Contenuto\_Tipologia ON Contenuto.idContenuto = Contenuto\_Tipologia.idContenuto   
WHERE Contenuto.idSezione = value  
AND idProvenienza = value  
AND idCreatore = value  
AND idPeriodoStorico = value  
AND idTipoContenuto = value  
AND idMuseo = value;

*Aggiunta e modifica di tipo di biglietto ed eliminazione di tutti i biglietti acquistati (25-26-27)*

25)

INSERT INTO TipoBiglietto(Nome, Prezzo, Descrizione, idMuseo)  
VALUES (...)

26)

UPDATE TipoBiglietto  
SET Nome = ‘’, Prezzo = ‘’, Descrizione = ‘’   
WHERE idTipoBiglietto = ‘’

**//quando viene modificato il prezzo di un TipoBiglietto non viene modificato il prezzo dei biglietti già acquistati in passato, in quanto anche Biglietto contiene un campo “PrezzoAcquisto” che assumerà il valore del prezzo in TipoBiglietto al momento dell’acquisto (funzionalità che ricordo essere gestita esternamente e non all’interno di questo applicativo).**

27)

DELETE Biglietto WHERE idMuseo = value;

*Calcolo delle statistiche mensili ed annuali (33-33.5-34)*

33)

Numero biglietti venduti:

SELECT COUNT(\*) AS numBiglietti   
FROM Biglietto   
WHERE idMuseo = 1 AND MONTH(DataValidita) = 12;

Numero manutenzioni:

SELECT COUNT(\*) AS numManutenzioni   
FROM RegistroManutenzioni   
WHERE idMuseo = 1 AND MONTH(Data) = 12;

Numero nuovi contenuti:

SELECT COUNT(\*) AS numNuoviContenuti   
FROM Contenuto INNER JOIN Sezione ON Contenuto.idSezione = Sezione.idSezione   
WHERE idMuseo = 1 AND MONTH(DataArrivoMuseo) = 12;

Numero giorni chiusura:

SELECT COUNT(\*) AS numGiorniChiusura   
FROM CalendarioChiusure   
WHERE idMuseo = 1 AND MONTH(Data) = 12;

Spese totali:

SELECT SUM(Spese) AS speseTotali   
FROM ((  
SELECT SUM(Prezzo) AS Spese   
FROM RegistroManutenzioni   
WHERE idMuseo = 1 AND MONTH(Data) = 12)   
UNION (  
SELECT SUM(StipendioOra) AS Spese   
FROM Personale INNER JOIN RegistroPresenze   
ON Personale.idPersonale = RegistroPresenze.idPersonale   
WHERE idMuseo = 1 AND (MONTH(DataEntrata) = 12 OR MONTH(DataUscita) = 12 ))) AS Spese;

Fatturato:

SELECT SUM(PrezzoAcquisto) AS fatturato   
FROM Biglietto   
WHERE idMuseo = 1 AND MONTH(DataValidita) = 12;

33.5) (in caso non siano state salvate di già le statistiche mensili per museo dentro alla tabella Statistiche)

Numero biglietti venduti:

SELECT COUNT(\*) AS numBiglietti   
FROM Biglietto   
WHERE idMuseo IN (  
SELECT idMuseo   
FROM Museo   
WHERE idFamiglia = 1 AND MONTH(DataValidita) = 12;

Numero manutenzioni:

SELECT COUNT(\*) AS numManutenzioni   
FROM RegistroManutenzioni   
WHERE idMuseo IN (  
SELECT idMuseo   
FROM Museo   
WHERE idFamiglia = 1 AND MONTH(Data) = 12;

Numero nuovi contenuti:

SELECT COUNT(\*) AS numNuoviContenuti   
FROM Contenuto INNER JOIN Sezione ON Contenuto.idSezione = Sezione.idSezione   
WHERE idMuseo IN (  
SELECT idMuseo   
FROM Museo   
WHERE idFamiglia = 1 AND MONTH(DataArrivoMuseo) = 12;

Numero giorni chiusura:

SELECT COUNT(\*) AS numGiorniChiusura   
FROM CalendarioChiusure   
WHERE idMuseo IN (  
SELECT idMuseo   
FROM Museo   
WHERE idFamiglia = 1 AND MONTH(Data) = 12;

Spese totali:

SELECT SUM(Spese) AS speseTotali   
FROM ((  
SELECT SUM(Prezzo) AS Spese   
FROM RegistroManutenzioni   
WHERE idMuseo IN (  
SELECT idMuseo   
FROM Museo   
WHERE idFamiglia = 1 AND MONTH(Data) = 12)   
UNION (  
SELECT SUM(StipendioOra) AS Spese   
FROM Personale INNER JOIN RegistroPresenze   
ON Personale.idPersonale = RegistroPresenze.idPersonale   
WHERE idMuseo IN (  
SELECT idMuseo   
FROM Museo   
WHERE idFamiglia = 1 AND (MONTH(DataEntrata) = 12 OR MONTH(DataUscita) = 12 ))) AS Spese;

Fatturato:

SELECT SUM(PrezzoAcquisto) AS fatturato   
FROM Biglietto   
WHERE idMuseo IN (  
SELECT idMuseo   
FROM Museo   
WHERE idFamiglia = 1 AND MONTH(DataValidita) = 12;

33.5) (in caso siano state salvate di già le statistiche mensili per museo dentro alla tabella Statistiche)

SELECT Statistiche.\*  
FROM Statistiche INNER JOIN StatisticheMuseo   
ON Statistiche.idStatistiche = StatisticheMuseo.idStatistiche INNER JOIN Museo ON  
Museo.idMuseo = StatisticheMuseo.idMuseo  
WHERE Museo.idFamiglia = value;

34)

SELECT Statistiche.\*   
FROM Statistiche   
INNER JOIN StatisticheFamigliaMusei   
ON Statistiche.idStatistiche = StatisticheFamigliaMusei.idStatistiche   
WHERE idFamiglia = ‘’ AND YEAR(MeseAnno) = '';

# **PROGETTAZIONE DELL’APPLICAZIONE**

Per la realizzazione vera e propria dell’applicativo si è deciso di utilizzare il linguaggio C# con librerie WPF (Windows Presentation Foundation), in quanto esse permettono la realizzazione di un’ottima interfaccia grafica in buone tempistiche temporali.

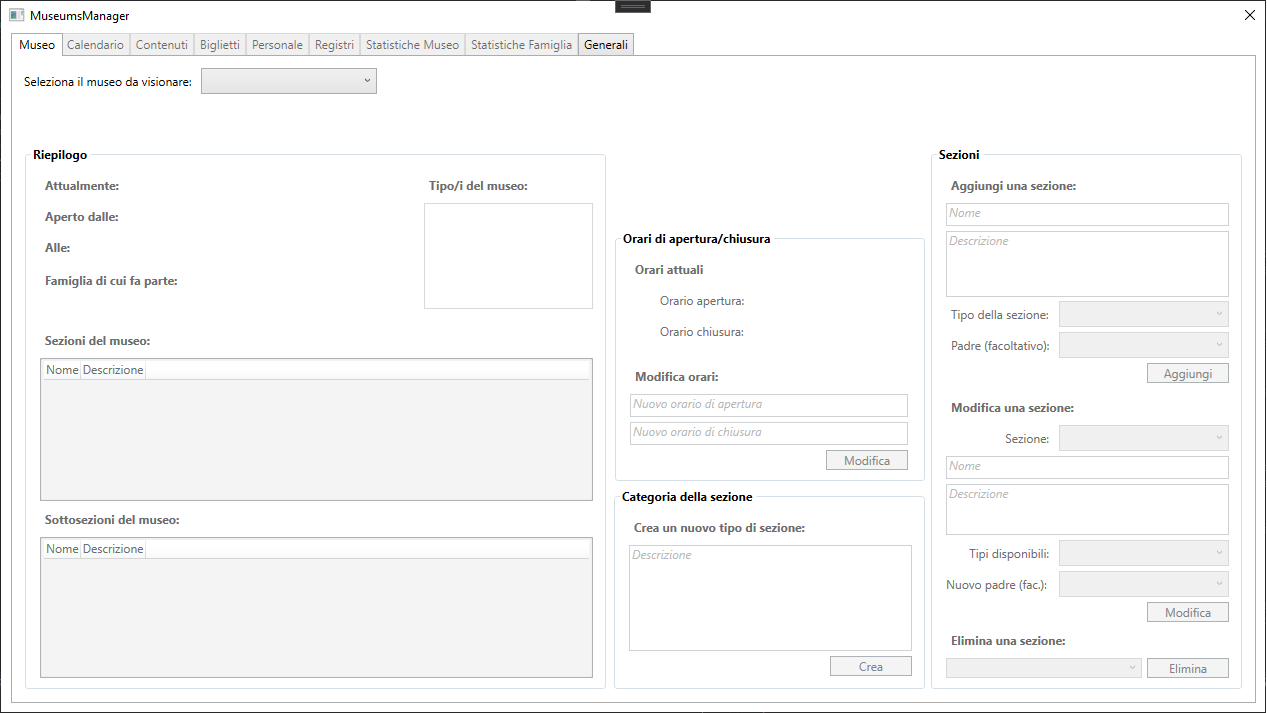
Il database è stato scritto e gestito tramite il DBMS “*Microsoft SQL Server Management Studio 18*”, quindi in linguaggio *TSql* (Sql Server Microsoft), in quanto esso è perfettamente integrato con l’IDE proprietario Microsoft (Visual Studio, C# WPF), sul quale si è deciso di basare l’intero progetto. Esso risiede in locale.

Il codice C# è stato scritto con un approccio MVC (pattern Model-View-Controller) semplificato.

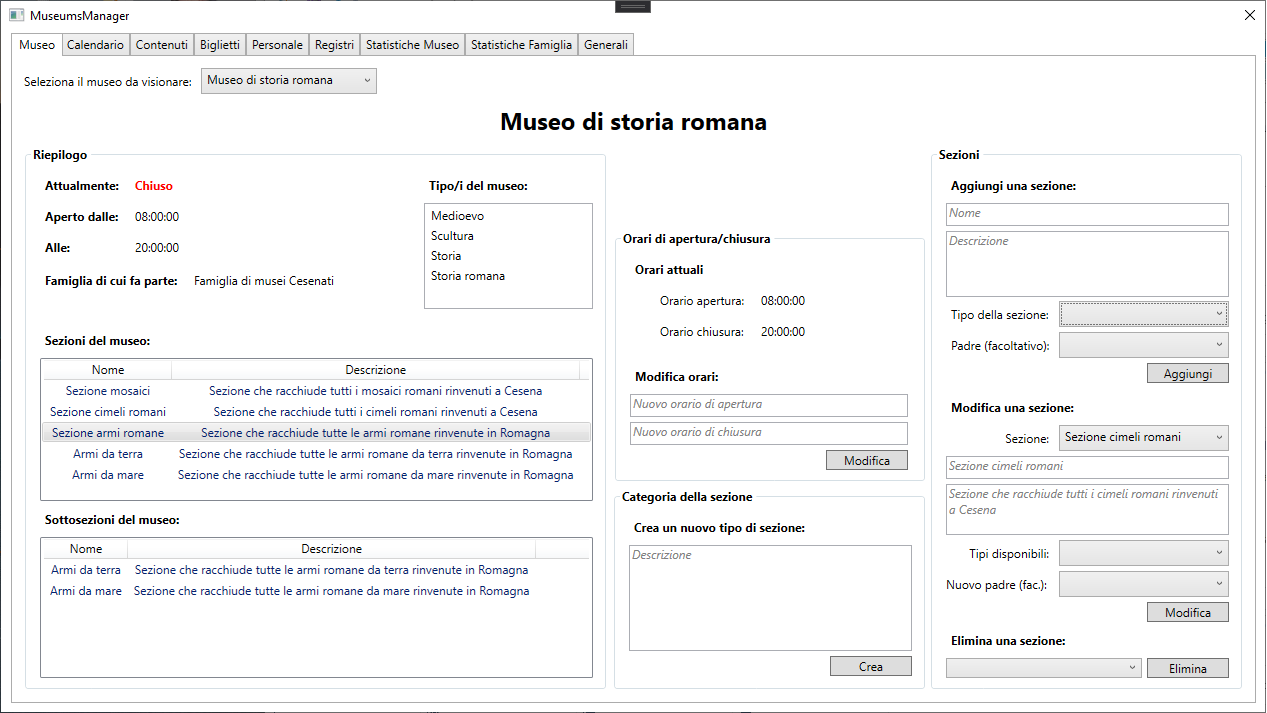
Per ogni tabella contenuta all'interno del database è stata creata una corrispettiva classe che la rappresenta all’interno del Model dell’applicazione. Oltretutto, è stato creato un meccanismo generico per la creazione delle query Select/Insert/Update/Delete, sia per relazioni [0-1]-[0-1-N] che per le relazioni N-N. Ogni classe è in grado di utilizzare questo meccanismo per una gestione delle query semplificata, a seconda del suo tipo.

Di seguito mostriamo diverse immagini riguardanti l’UI dell’applicativo, in ogni sua sezione:

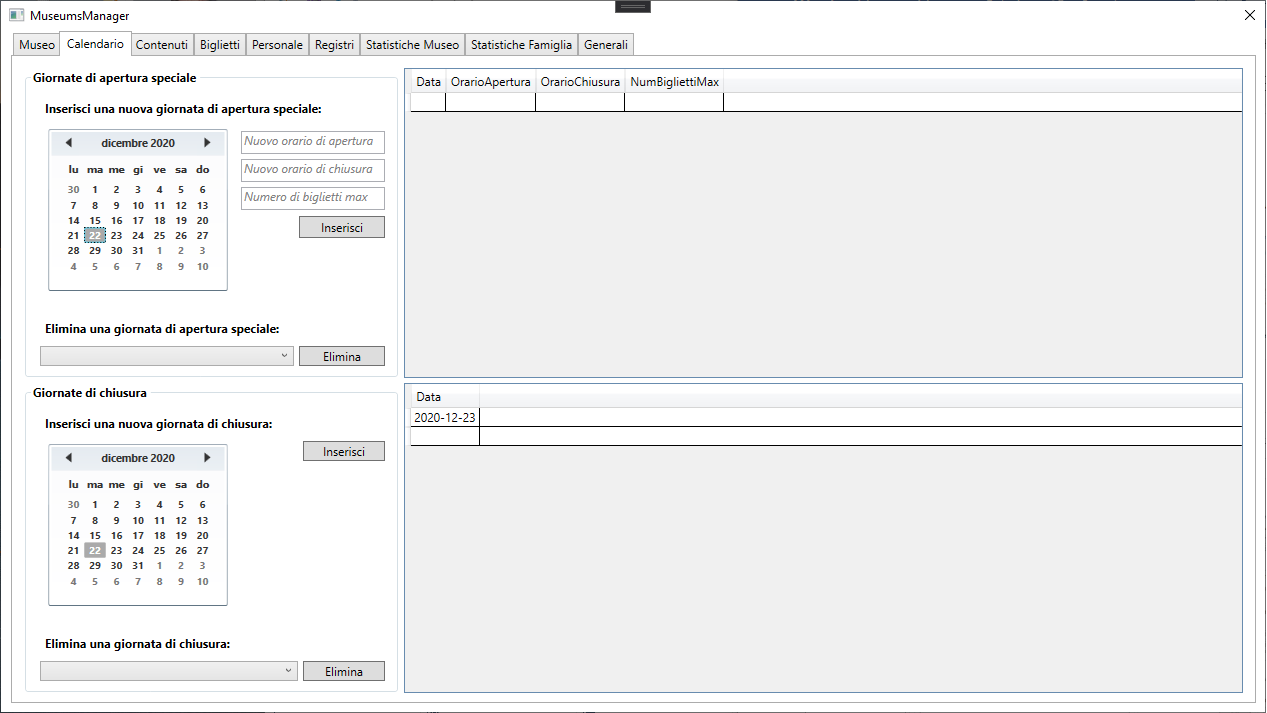
*Museo - Interfaccia principale (disabilitata):*



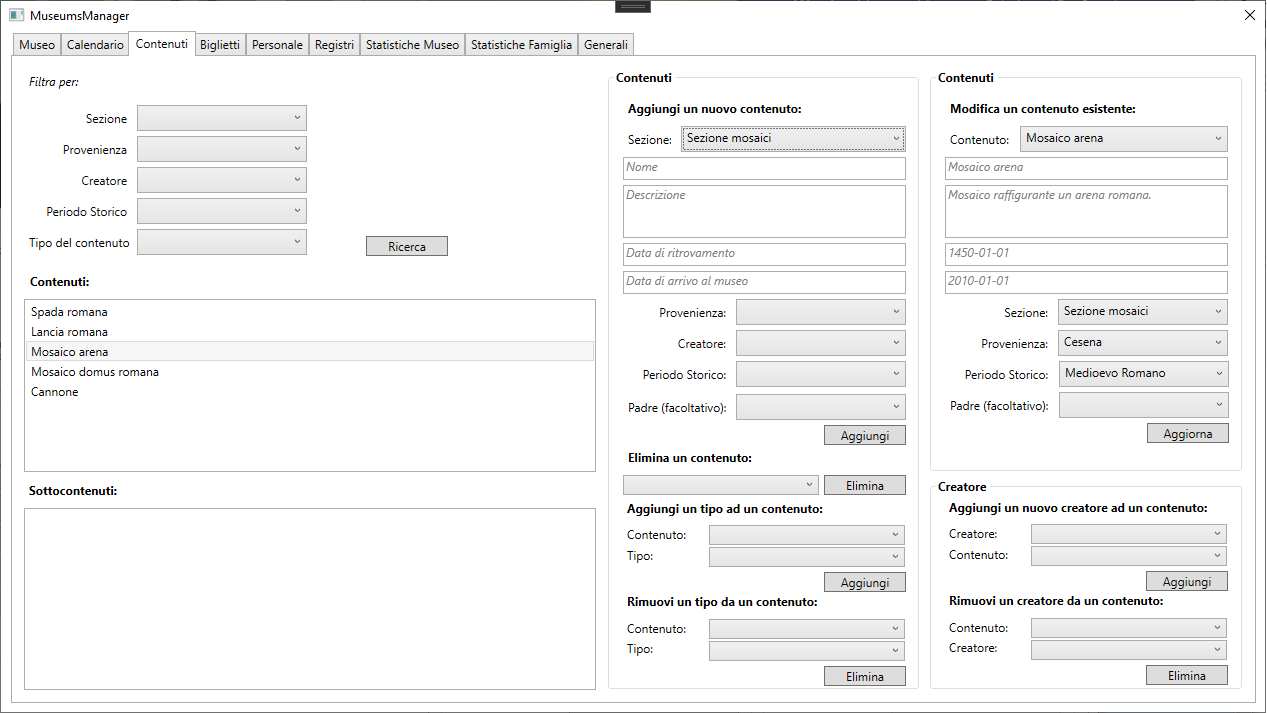
*Museo - Interfaccia principale (abilitata):*



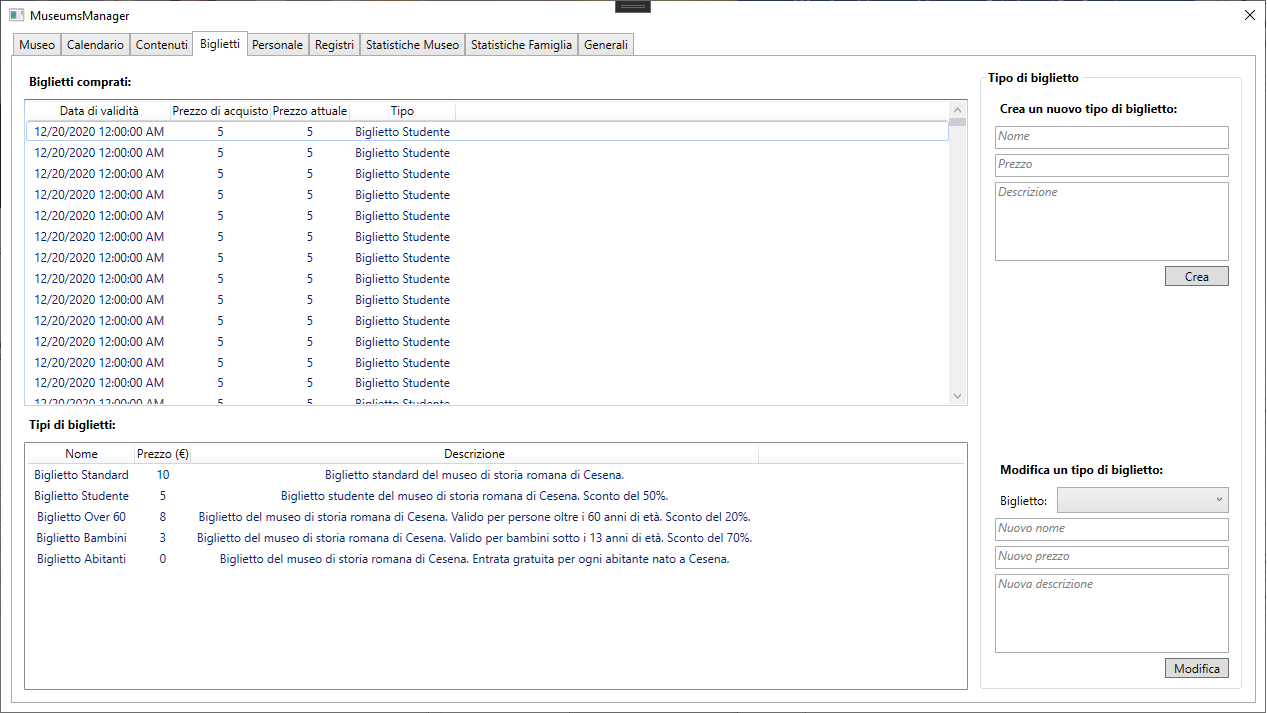
*Calendario - gestione giorni di chiusura e apertura speciale:*



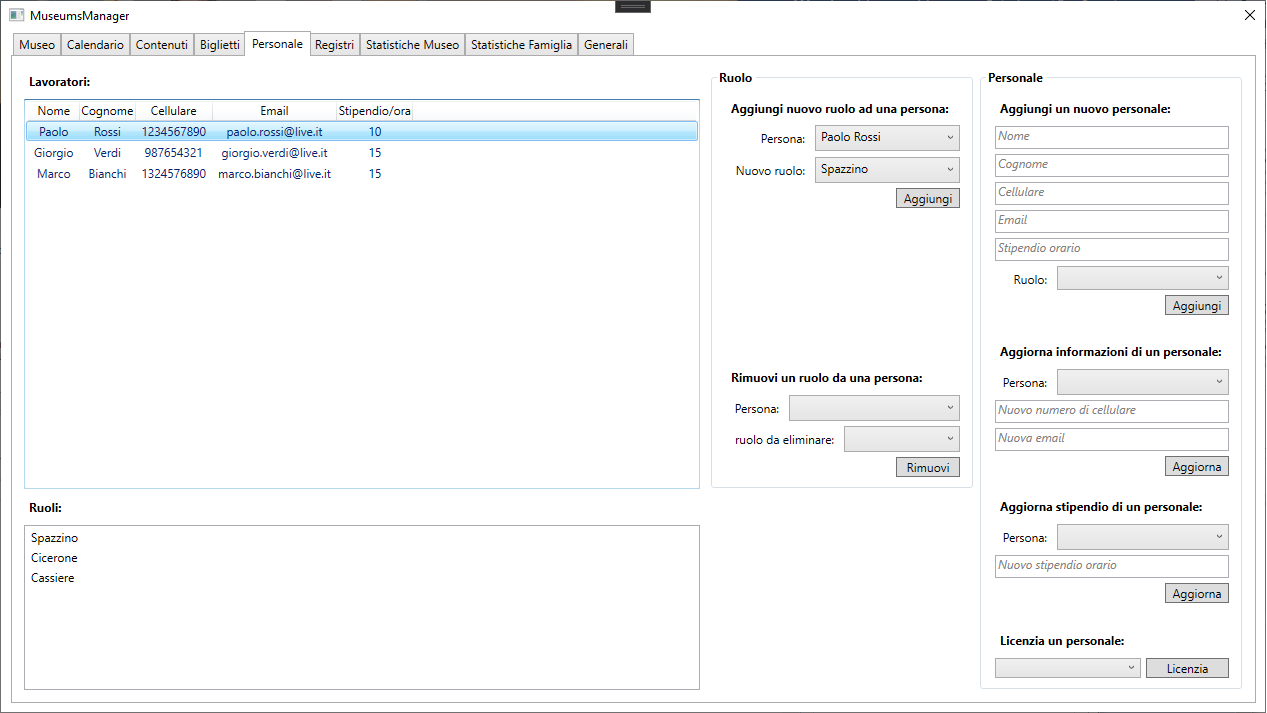
*Contenuti – ricerca e gestione:*



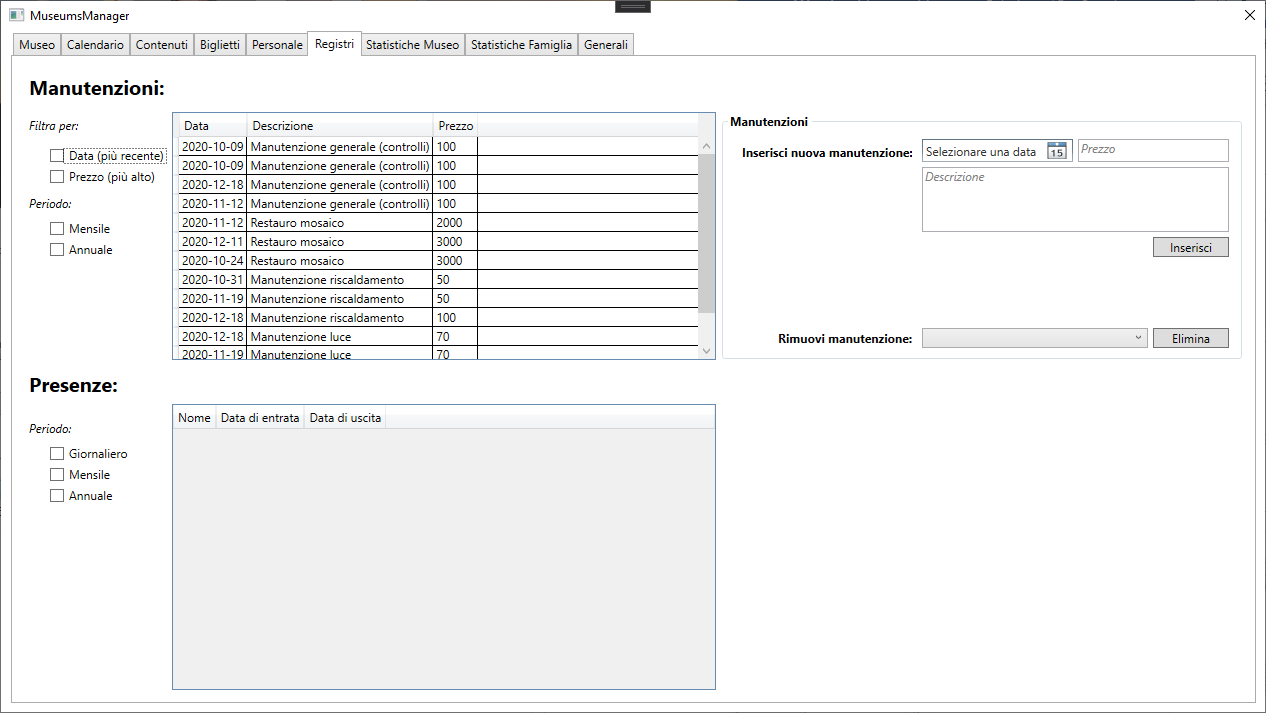
*Biglietti – gestione dei tipi e visualizzazione degli acquisti:*



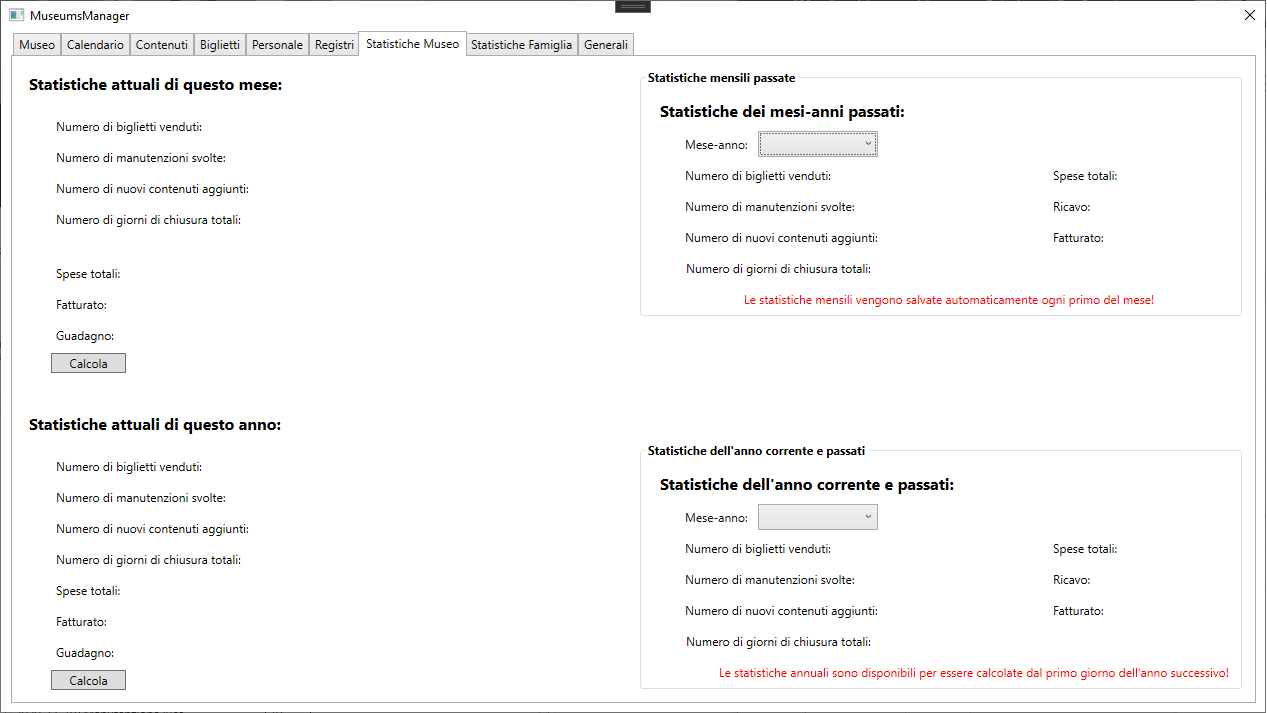
*Personale – riepilogo e gestione di informazioni e ruoli:*



*Registri – Visualizzazione di manutenzioni (+ inserimento) e presenze:*



*Statistiche - Interfaccia generale mesi-anni (analoga per le famiglie):*



*Generali - Operazioni generiche di base (esterne al museo selezionato):*

