

BASE DATI

Catena multisala

Facoltà di Ingegneria Informatica, Federico II
Anno accademico 2016-2107
Corso di Basi di Dati tenuto dal prof. Vincenzo Moscato
Studenti: Carmine Cesarano, Michele De Sena, Enza D'Angelo

Questo lavoro si basa sulla traccia assegnata al corso di Basi di Dati I dell'attuale anno accademico, riguardante la creazione di un database per la gestione di una serie di servizi web forniti da una catena di cinema multisala.

Dapprima verrà introdotta la traccia, poi si passerà all'analisi del problema e alla strutturazione della soluzione, andando dalla descrizione in ER alla implementazione SQL. In seguito sarà analizzata la costruzione della GUI in PHP/HTML.

1 Traccia

Una multinazionale di informatica ha recentemente acquisito una serie di società che gestiscono una serie di servizi web per la prenotazione di biglietti a cinema collocati sul territorio svizzero (in particolare ogni società gestisce la prenotazione per i cinema di un singolo cantone).

Come parte della nuova strategia gestionale, la suddetta società intende fornire ai propri utenti un servizio centralizzato per:

1. la consultazione della programmazione delle proiezioni di film presso i vari cinema;
2. la prenotazione ed acquisto on-line di biglietti.

La società prevede che i servizi sopraelencati siano accessibili dai propri utenti mediante interfacce web.

Ovviamente le informazioni riguardo i cinema di un dato cantone, la relativa programmazione, gli orari, i prezzi sono di competenza delle singole società locali, ciascuna delle quali ha nel tempo sviluppato un proprio sistema informativo. Mentre le informazioni sui film devono essere recuperate da siti specializzati gestiti da terze parti (e.g. imdb). Infine, il pagamento con carta di credito deve essere effettuato in modalità sicura sfruttando web service offerti da istituti di credito certificati.

2 Descrizione del problema

Il database è incentrato sugli elementi che appartengono alla realtà di interesse “consultazione, prenotazione e gestione del calendario”, quali Cinema, Sale, Film, Programmazione, Prenotazioni, Utenti e Biglietti. Pertanto sono considerate solo quelle classi in stretta correlazione con la traccia, ovvero non sono implementate altre funzionalità standard che potrebbero essere implementate in un sistema gestionale (amministrazione personale ad esempio).

3 Progettazione Concettuale

In fase di progettazione è stata scelta e adottata una **strategia top-down**. Questo perché le specifiche utente son chiare fin dall’inizio. Sono stati quindi individuati i macro concetti e a partire da questi si è proceduto con raffinamenti successivi, esplicitando di volta in volta nuovi dettagli. Analizzando le specifiche del paragrafo 1 possiamo stilare il seguente glossario.

3.1 Glossario

Termine	Descrizione	Collegamenti
CINEMA	Luogo dove sono presenti sale cinematografiche	Sala
SALA	La sala dove si proietta un film	Cinema, Poltrona,
POSTO	La poltrona fisica del cinema	Sala, Prenotazione
FILM	La pellicola proiettata	Programmazione, Sala
UTENTI	L’utente che prenota un film	Prenotazione
PROGRAMMAZIONE	Lo svolgersi di una proiezione di un film	Film, Sala
PRENOTAZIONI	Effettuate dagli utenti per riservare un posto per una programmazione	Sala, Programmazione, Posto

3.2 Diagramma E/R

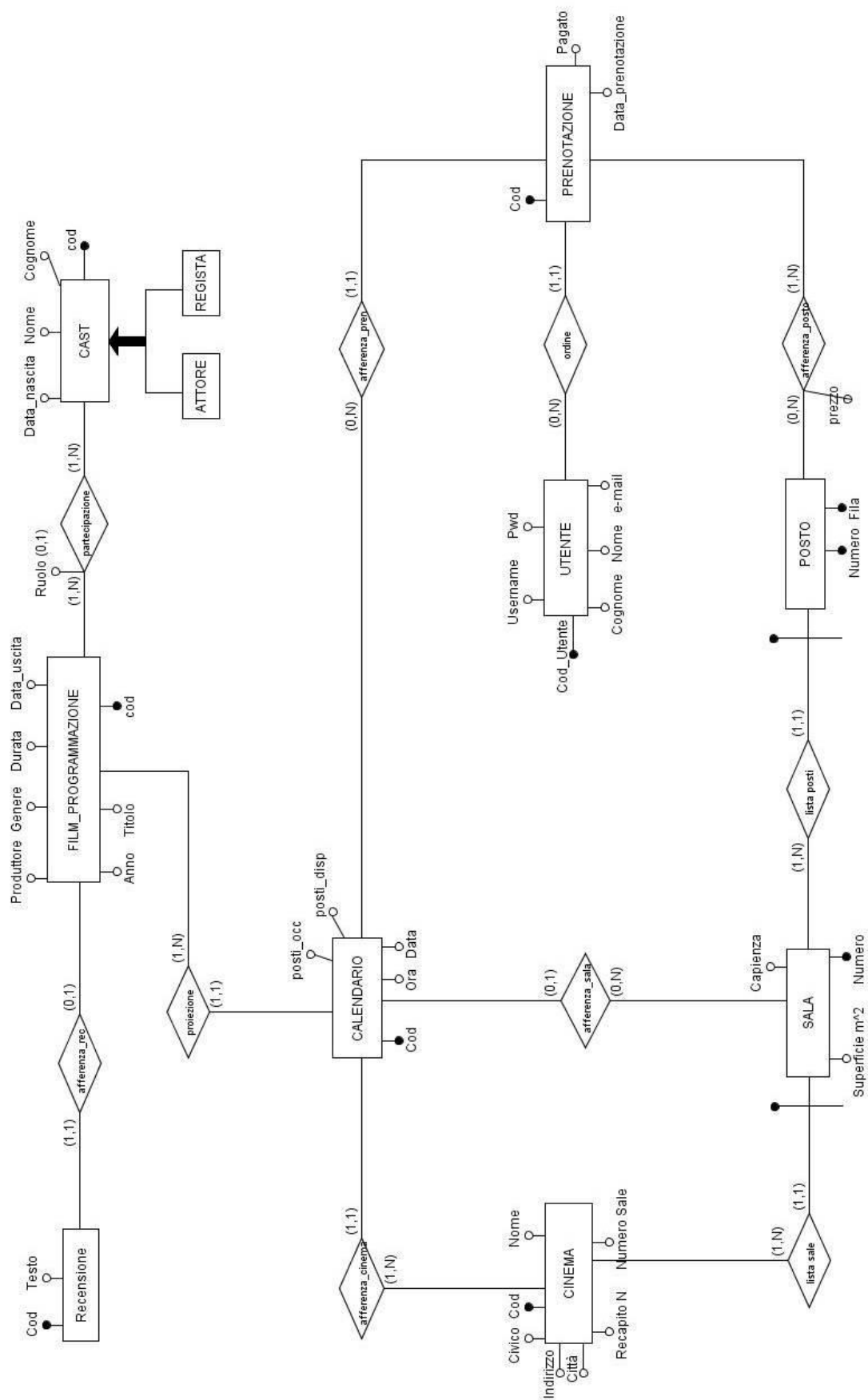


Figura 1: diagramma E/R del sistema cinema

3.3 Dizionario Dati

3.3.1 Entità

Classe	Descrizione	Attributi	Identificatori
UTENTE	Utenti registrati alla piattaforma	Cod_utente, Cognome, Nome, Username, Pwd, Email	Cod_utente
FILM	Stora tutti i film mandati in programmazione fino ad ora	Cod, titolo, anno, Produttore, Genere, durata, data_uscita	Cod
RECENSIONE	Contiene le recensioni dei soli film in programmazione	Cod, testo	Cod
CAST	Gruppo persone che lavora nella produzione (generalizza le entità attore e regista)	Cod, nome, cognome, data_nascita	Cod
CALENDARIO	Comprende tutte le istanze di proiezione, identificate da un codice univoco	Cod, Data, Ora, posti_occ, posti_disp	Cod
CINEMA	Insieme dei cinema multisala	Cod, Nome, Numero_Sale, Città, indirizzo, Civico, Recapito	Cod
SALA	Insieme delle sale	Numero, Capienza, Superficie	Numero
POSTO	Contiene tutti i posti, di tutte le sale, di tutti i cinema	Numero, Fila	Numero, Fila
PRENOTAZIONE	Contiene tutte le istanze di prenotazione utente	Cod, data_prenotazione, pagato	Cod

3.3.2 Associazioni

Associazione	Descrizione	Classi coinvolte
AFFERENZA_REC	Mette in relazione i film con un'unica recensione	FILM (0,1) RECENSIONE (1,1)
PARTECIPAZIONE	Mette in relazione i film con il relativo cast, specificando il RUOLO (opzionale) degli attori	FILM (1,N) CAST (1,N)
PROIEZIONE	Relaziona i film in programmazione con un calendario di tutte le proiezioni	FILM (1,N) CALENDARIO (1,1)
AFFERENZA_CIN	Associa un'istanza del calendario ad un cinema	CALENDARIO (1,1) CINEMA (1,N)
LISTA_SALE	Mette in relazione ogni cinema con le relative sale	CINEMA (1,N) SALA (1,1)
LISTA_POSTI	Mette in relazione ogni sala con i relativi posti	SALA (1,N) POSTO (1,1)
AFFERENZA_SALA	Associa una proiezione (istanza di calendario) ad una sala	CALENDARIO (0,1) SALA (0,N)
AFFERENZA_POSTO	Mette in relazione le prenotazioni con i posti, specificando il PREZZO per ogni afferenza.	PRENOTAZIONE (1,N) POSTO (0,N)
ORDINE	Associa un'istanza di prenotazione con l'utente che l'ha effettuata.	UTENTE (0,N) PRENOTAZIONE (1,1)
AFFERENZA_PREN	Mette in relazione le prenotazioni con le relative istanze di calendario	CALENDARIO (0,N) PRENOTAZIONE (1,1)

NOTE:

La cardinalità di partecipazione di CALENDARIO all'associazione AFFERENZA_SALA è opzionale perché un cinema può o meno decidere di introdurre, al momento della pubblicazione, il numero della sala nel calendario.

4 Progettazione Logico-Relazionale

Prima ancora di tradurre lo schema concettuale in quello logico è opportuno analizzare lo schema ER, al fine di apportare delle modifiche che rendano più efficiente l'implementazione finale.

4.1 Attributi Multivalore

E' stato rimosso l'attributo "recapito", presente CINEMA, per essere sostituito dall'entità "RECAPITO_CINEMA" opportunamente associata al cinema da una relazione 1-N, effettuando di fatto un *partizionamento verticale*.

4.2 Analisi delle ridondanze

L'unica ridondanza è dovuta agli attributi `posti_occ` e `posti_disp`, calcolabili, data una occorrenza p di PROGRAMMA, contando le occorrenze di AFFERENZA_PREN in cui è presente p nel primo caso e sottraendo al numero di posti della sala, il numero di posti occupati, nel secondo caso.

4.3 Eliminazione delle generalizzazioni

CAST:

La specializzazione in ATTORE e REGISTA è del tipo **totale e disgiunta**.

Questa può essere tradotta accorpendo le entità sottoclassi nella superclasse ed includendo nella superclasse l'attributo "tipo" che indica la sottoclasse a cui appartiene ciascuna tupla. Questa soluzione è attuabile solo per le spec. totali/disgiunte.

Il fatto che le sottoclassi non specificano alcun attributo la superclasse garantisce la non presenza di valori nulli in CAST e quindi il massimo grado di efficienza per questa soluzione.

4.4 Eliminazione attributi composti

Era stato pensato, in una fase iniziale della progettazione, di implementare nello schema ER gli indirizzi come attributi composti. Questa funzionalità non è resa però disponibile dal tool utilizzato (JDER), dunque questa fase non è necessaria.

4.5 Scelta identificatori

Quasi per tutte le entità è stato preferibile introdurre l'attributo `codice` come identificatore. Osserviamo infatti che quasi tutte le entità referenziano chiavi esterne, quindi il codice, rispetto alle altre possibili alternative, garantisce un risparmio sia in termini di byte che di tempo durante i join tra le tabelle. Solo per l'entità POSTO si è deciso di identificare le tuple con gli attributi numero e fila.

4.6 Diagramma E/R ristrutturato

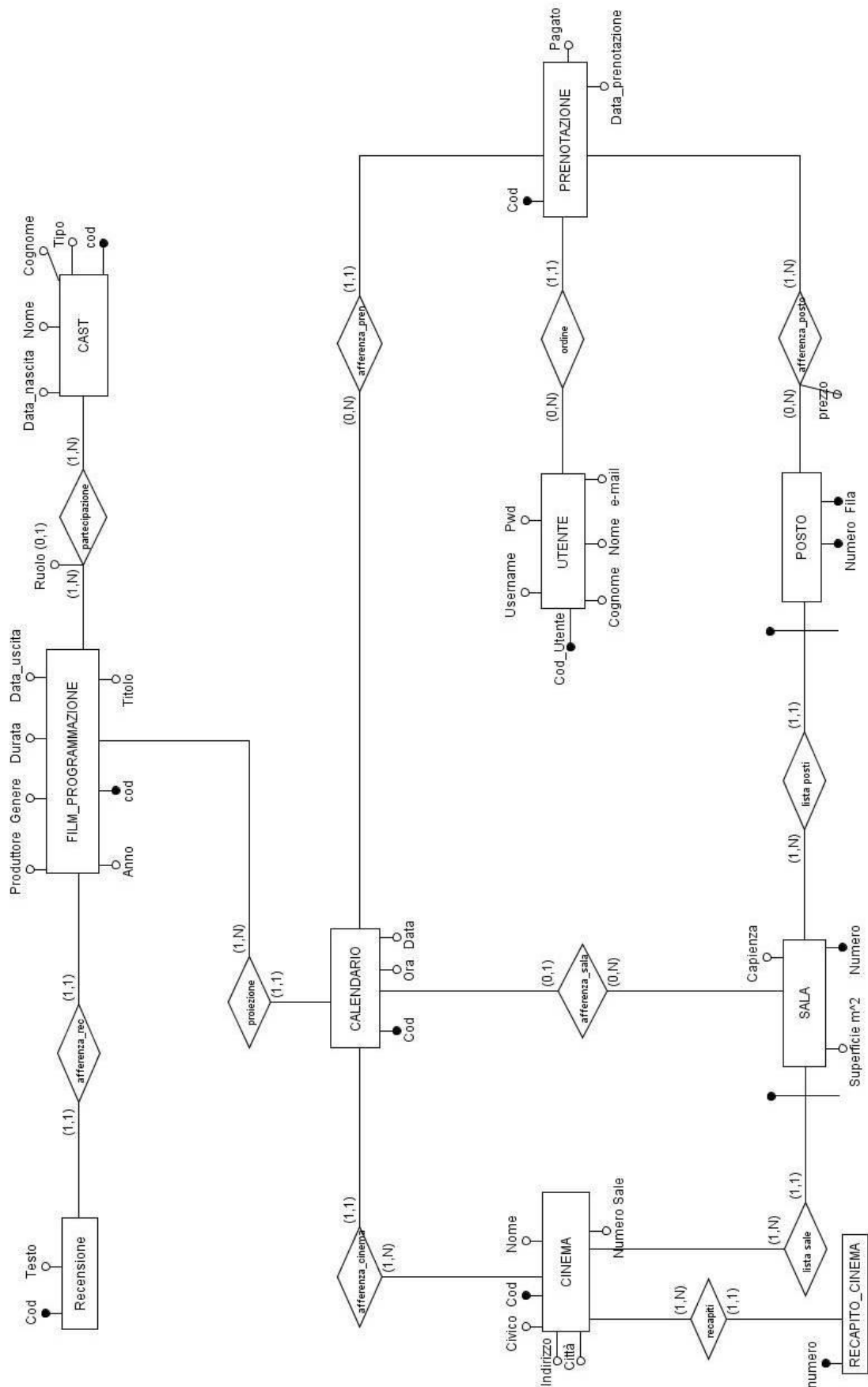


Figura 2: Diagramma E/R ristrutturato

4.7 Approcci di conversione da E/R a relazionale

Per rendere più comprensibile il significato dello schema è stato opportuno rinominare gli identificatori delle entità coinvolte assegnando loro un nome più comprensibile rispetto a “codice”.

Le **entità** sono state tradotte con una tabella che ha il nome dell’entità stessa, ma al plurale.

Le **associazioni 1-N** possono essere tradotte con la “*soluzione 2*”. Questa risulta essere la più efficiente perché permette di ridurre notevolmente il numero di tabelle (quindi la dimensione fisica della base) e la ridondanza dati.

Secondo questa soluzione l’associazione non compare nella traduzione e i suoi attributi insieme all’identificatore lato N vengono inclusi come attributi nella tabella lato 1.

L’unico inconveniente di questa soluzione, è la presenza di molti valori null in corrispondenza di partecipazione opzionale dell’entità lato 1. Si preferisce in questi casi considerare una terza tabella che traduce la relazione che ha per chiave l’identificatore lato 1 e include come attributo l’identificatore lato N. `AFFERENZA_SALA` è stata tradotta con questa strategia.

Le **associazioni 1-1** possono essere tradotte anch’esse secondo la “*soluzione 2*”. La chiave dell’entità lato opzionale diventa attributo della tabella che traduce l’entità lato non opzionale.

OSS. La traduzione tramite la soluzione 1 (l’associazione diventa una tabella) si rende necessaria solo nel caso in cui entrambe le entità partecipano con cardinalità opzionale.

Nelle nostre associazioni 1-1 non si presenta questo caso.

Le **associazioni N-N** sono tradotte in una tabella la cui chiave primaria è la combinazione degli identificatori delle due entità collegate.

4.8 Correttezza dello schema, verifica della normalità

Da un’attenta analisi, che tralasciamo, risulta che tutti gli schemi relazionali R_i della base rispettano la BCNF pertanto è garantita una decomposizione senza perdita ($R=R1 \text{ join } R2$).

4.9 Traduzione dello schema in relazionale

Risultato della traduzione finale, dopo aver ottimizzato lo schema con gli step precedenti.

RECENSIONI (CodRecensione, Film, Testo)

FILM_PROGRAMMAZIONE (CodFilm, Titolo, Anno, Produttore, Genere, Durata, DataUscita)

CAST (IdCast, Tipo, Nome, Cognome, DataNascita)

PARTECIPAZIONE (Film, Cast, Ruolo)

CALENDARI (CodProgramma, Ora, Data, Film, Cinema)

CINEMA (CodCinema, Nome, NumeroSale, Citta, Indirizzo, Civico)

RECAPITI_CINEMA (Numero, Cinema)

SALE (NumSala, Cinema, Superficie, Capienza)

AFFERENZE_SALA (Programma, Sala, Cinema)

POSTI (Numero, Fila, Sala, Cinema)

PRENOTAZIONI (IdPrenotazione, DataPrenotazione, Pagato, Utente, Programma)

UTENTI (CodUtente, Username, Pwd, Cognome, Nome, E-mail)

AFFERENZE_POSTO(Prenotazione, NumPosto, NumFila, Sala, Cinema, Prezzo)

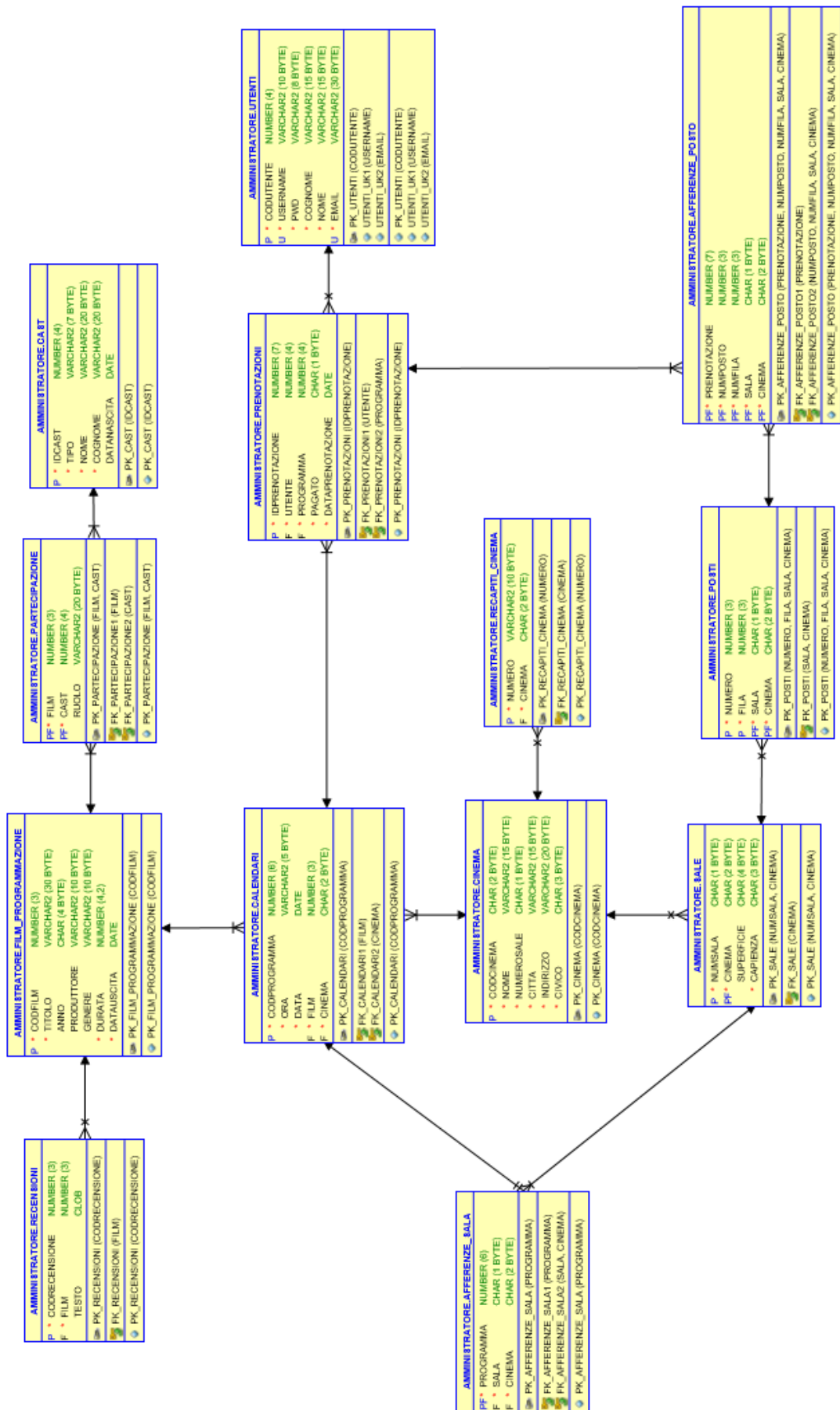


Figura 3: Schema E/R SQL Data Modeler tradotto

4.9.1 Vincoli di integrità referenziale

Chiave Esterna	Chiave Primaria (referenziata)
PARTECIPAZIONE (Film)	FILM_STORICO (CodFilm)
PARTECIPAZIONE (Cast)	CAST (IdCast)
SALE (Cinema)	CINEMA (CodCinema)
POSTI (Sala)	SALE (NumSala)
POSTI (Cinema)	CINEMA (CodCinema)
AFFERENZE_POSTO (Prenotazione)	PRENOTAZIONE (CodPrenotazione)
AFFERENZE_POSTO (NumPosto)	POSTI (Numero)
AFFERENZE_POSTO (NumFila)	POSTI (Fila)
AFFERENZE_POSTO (Sala)	POSTI (Sala)
AFFERENZE_POSTO (Cinema)	POSTI (Cinema)
PRENOTAZIONI (Utente)	UTENTI (CodUtente)
PRENOTAZIONI (Programma)	CALENDARI (CodProgramma)
RECAPITI_CINEMA (Cinema)	CINEMA (CodCinema)
CALENDARI (Film)	FILM_PROGRAMMAZIONE (CodFilm)
CALENDARI (Cinema)	CINEMA (CodCinema)
AFFERENZE_SALA (Programma)	CALENDARI (CodProgramma)
AFFERENZE_SALA (Sala)	SALE (NumSala)
AFFERENZE_SALA (Cinema)	SALE (Cinema)
RECENSIONI (Film)	FILM_PROGRAMMAZIONE (CodFilm)

5 Business Rules

- BR01. Le prenotazioni sono spostate nello storico alla fine di ogni anno (Stored Procedure)
- BR02. I film sono tenuti in programmazione per 7 giorni (gestito a livello applicativo)
- BR03. Un utente può prenotare un film fino ad un'ora prima della sua programmazione (gestito a livello applicativo)
- BR04. Le chiavi delle tabelle sono incrementate automaticamente.
- BR05. Le chiavi possono essere aggiornate senza perdita di dati riferiti.
- BR06. Numero massimo di utenti registrabili 2500.

6 Implementazione fisica

6.1 Dimensionamento e specifiche

Per effettuare delle considerazioni sul comparto fisico della base, si presenta una Tavola dei volumi nella quale si riportano le cardinalità delle entità e delle associazioni.

Le specifiche utente saranno integrate con opportune ipotesi.

I volumi verranno aggiornati dopo 2 anni di servizio. Pertanto lo storage è calcolato a partire da quest'anno (anno inizio attività) e considerando l'anno seguente.

- La società gestisce **50 cinema** con un totale di **150 sale**. Si suppone inoltre che ogni sala abbia mediamente 80 posti, per un totale di **12000 posti**.
- Come da specifiche, sono registrati alla piattaforma circa **1000 utenti**.
- Si suppone che ogni cinema abbia al massimo 2 recapiti telefonici.
- Si suppone che ogni sala effettui in media 3 proiezioni al giorno, per un totale di 450 proiezioni giornaliere (**164.250 proiezioni** all'anno).
- Supposto che un utente possa prenotare più posti con un'istanza di prenotazione, e che per ogni proiezione in media la sala è piena al 40%, il numero giornaliero di posti prenotati $450 \text{ proiezioni} \times 80 \text{ posti} \times 0,4 = 14.400$ (**5.256.800 afferenze_posto** all'anno)
- Supposto che ogni prenotazione sia effettuata in media su tre posti contiamo circa $5.256.800 / 3 =$ **1.752.266 prenotazioni** all'anno
- Ogni anno sono proiettati circa **70 film**.
- Per ogni film si suppone che il regista sia unico e che recitino in media 10 attori principali. Ogni attore ha recitato in media in 3 film. Ogni occorrenza di film partecipa all'associazione PARTECIPAZIONE 11 volte. L'associazione partecipazione ha un numero complessivo di $70 \times 3 \times 11 =$ **2310 occorrenze di partecipazione**
- Se ogni attore ha recitato in 3 film, allora devono esserci circa $2310 / 3 =$ **770 attori e registi** registrati.

6.2 Tavola dei Volumi

Tipo	Costrutto	Volume
E	Recensioni	70
E	Film_Programmazione	70
E	Cast	770
E	Calendari	164.250
E	Cinema	50
E	Recapiti_cinema	100
E	Sale	150
E	Posti	12000
E	Prenotazioni	1.752.266
A	Afferenze_sala	164.250
A	Afferenza_posto	5.256.800
A	Partecipazione	2310

6.2.1 RECENSIONI

Attributo	Tipo	Byte	Initial	Next
			Occorrenze: 70	Occorrenze: 70
CodRecensione	NUMBER(3)	3	210	210
Film	NUMBER(3)	3	210	210
Testo	CLOB	4	280	280
TOT			700	700

Storage	Initial (Kb)	Next (Kb)	Initial	Next
	0,68	0,68	701	701

6.2.2 FILM_PROGRAMMAZIONE

Attributo	Tipo	Byte	Initial	Next
			Occorrenze: 70	Occorrenze: 70
CodFilm	NUMBER(3)	3	210	210
Titolo	VARCHAR2(30)	30	2100	2100
Anno	CHAR(4)	4	280	280
Produttore	VARCHAR2(10)	10	700	700
Genere	VARCAHR2(10)	10	700	700
Durata	NUMBER(4,2)	4	280	280
DataUscita	DATE	7	490	490
TOT			4760	4760

Storage	Initial (Kb)	Next (Kb)	Initial	Next
	4,65	4,65	4761	4761

Si presume che l'anno seguente escano ancora 70 film.

6.2.3 CAST

Attributo	Tipo	Byte	Initial	Next
			Occorrenze: 770	-
IdCast	NUMBER(4)	4	3.080	-
Tipo	VARCHAR2(7)	7	5.390	-
Nome	VARCHAR2(20)	20	15.400	-
Cognome	VARCHAR2(20)	20	15.400	-
DataNascita	DATE	7	5.390	-
TOT			40.040	-
Storage	Initial (Kb)	Next (Kb)	Initial	Next
	43,61	-	44.661	-

6.2.4 PARTECIPAZIONE

Attributo	Tipo	Byte	Initial	Next		
			Occ:	2.310	Occ:	1.000
Film	NUMBER(3)	3		6.930		3.000
Cast	NUMBER(4)	4		9.240		4.000
Ruolo	VARCHAR2(20)	20		46.200		20.000
TOT				62.370		27.000
Storage	Initial (Kb)	Next (Kb)		Initial		Next
	60,90	26,37		62.371		27.001

6.2.5 CALENDARI

Attributo	Tipo	Byte	Initial		Next	
			Occ:	164.250	Occ:	82.125
CodProgramma	NUMBER(6)	5		821.250		410.625
Ora	NUMBER(4,2)	4		657.000		328.500
Data	DATE	7		1.149.750		574.875
Film	NUMBER(3)	3		492.750		246.375
Cinema	CHAR(2)	2		328.500		164.250
TOT				3.449.250		1.724.625
Storage	Initial (Kb)	Next (Kb)	Initial		Next	
	3368,40	1684,20		3.449.251		1.724.626

Si suppongono per l'anno successivo 2 proiezioni per sala al giorno

6.2.6 CINEMA

Attributo	Tipo	Byte	Initial	Next
			Occorrenze:	50
CodCinema	CHAR(2)	2	100	-
Nome	VARCHAR2(15)	15	750	-
NumeroSale	CHAR(1)	1	50	-
Città	VARCHAR2(15)	15	750	-
Indirizzo	VARCHAR2(20)	20	1.000	-
Civico	CHAR(3)	3	150	-
TOT			2.800	-
Storage	Initial (Kb)	Next (Kb)	Initial	Next
	2,73	-	2.801	-

6.2.7 RECAPITI_CINEMA

Attributo	Tipo	Byte	Initial	Next
			Occorrenze:	100
Numero	VARCHAR2(10)	10	1000	-
Cinema	CHAR(2)	2	200	-
TOT			1200	-
Storage	Initial (Kb)	Next (Kb)	Initial	Next
	1,17	-	1.201	-

6.2.8 SALE

Attributo	Tipo	Byte	Initial	Next
			Occorrenze:	150
NumSala	CHAR(1)	1	150	-
Cinema	CHAR(2)	2	300	-
Superficie	CHAR(4)	4	600	-
Capienza	CHAR(3)	3	450	-
TOT			1500	-
Storage	Initial (Kb)	Next (Kb)	Initial	Next
	1,46	-	1501	-

6.2.9 AFFERENZE_SALA

Attributo	Tipo	Byte	Initial		Next	
			Occ:	164.250	Occ:	82.125
Programma	NUMBER(6)	5		821.250		410.625
Sala	CHAR(1)	1		164.250		82.125
Cinema	CHAR(2)	2		328.500		164.250
TOT				1.314.000		657.000
Storage	Initial (Kb)	Next (Kb)	Initial		Next	
	1283,20	641,60	1.314.001		657.001	

Nel secondo anno si suppongono 2 proiezioni al giorno

6.2.10 POSTI

Attributo	Tipo	Byte	Initial	Next	
			Occ:	12.000	-
Numero	NUMBER(3)	3		36.000	-
Fila	NUMBER(3)	3		36.000	-
Sala	CHAR(1)	1		12.000	-
Cinema	CHAR(2)	2		24.000	-
TOT				108.000	-
Storage	Initial (Kb)	Next (Kb)		Initial	Next
	105,46	-		108.001	

6.2.11 PRENOTAZIONI

Attributo	Tipo	Byte	Initial	Next
			Occ: 1.752.266	-
IdPrenotazione	NUMBER(7)	5	8.761.330	-
Utente	NUMBER(4)	4	7.009.064	-
Programma	NUMBER(4)	4	7.009.064	-
Pagato	CHAR(1)	1	1.752.266	-
DataPrenotazione	DATE	7	12.265.862	-
TOT			36.797.586	-
Storage	Initial (Kb)	Next (Kb)	Initial	Next
	35.935,14	-	36.797.587	-

Le prenotazioni sono spostate nello storico alla fine di ogni anno.

6.2.12 UTENTI

Attributo	Tipo	Byte	Initial	Next
			Occ: 1.000	Occ: 200
CodUtente	NUMBER(4)	4	4.000	800
Username	VARCHAR2(10)	10	10.000	2.000
Pwd	VARCHAR2(8)	8	8.000	1.600
Cognome	VARCHAR2(15)	15	15.000	3.000
Nome	VARCHAR2(15)	15	15.000	3.000
EMail	VARCHAR2(30)	30	30.000	6.000
TOT			82.000	16.400
Storage	Initial (Kb)	Next (Kb)	Initial	Next
	80,08	16,01	82.001	16.401

6.2.13 AFFERENZE_POSTO

Attributo	Tipo	Byte	Initial	Next
			Occ: 5.256.800	-
Prenotazione	NUMBER(7)	5	26.284.000	-
NumPosto	NUMBER(3)	3	15.770.400	-
NumFila	NUMBER(3)	3	15.770.400	-
Sala	CHAR(1)	1	5.256.800	-
Cinema	CHAR(2)	2	10.513.600	-
Prezzo	NUMBER(4,2)	4	21.027.200	-
TOT			94.622.400	-
Storage	Initial (Kb)	Next (Kb)	Initial	Next
	92.404,68	-	94.622.400	-

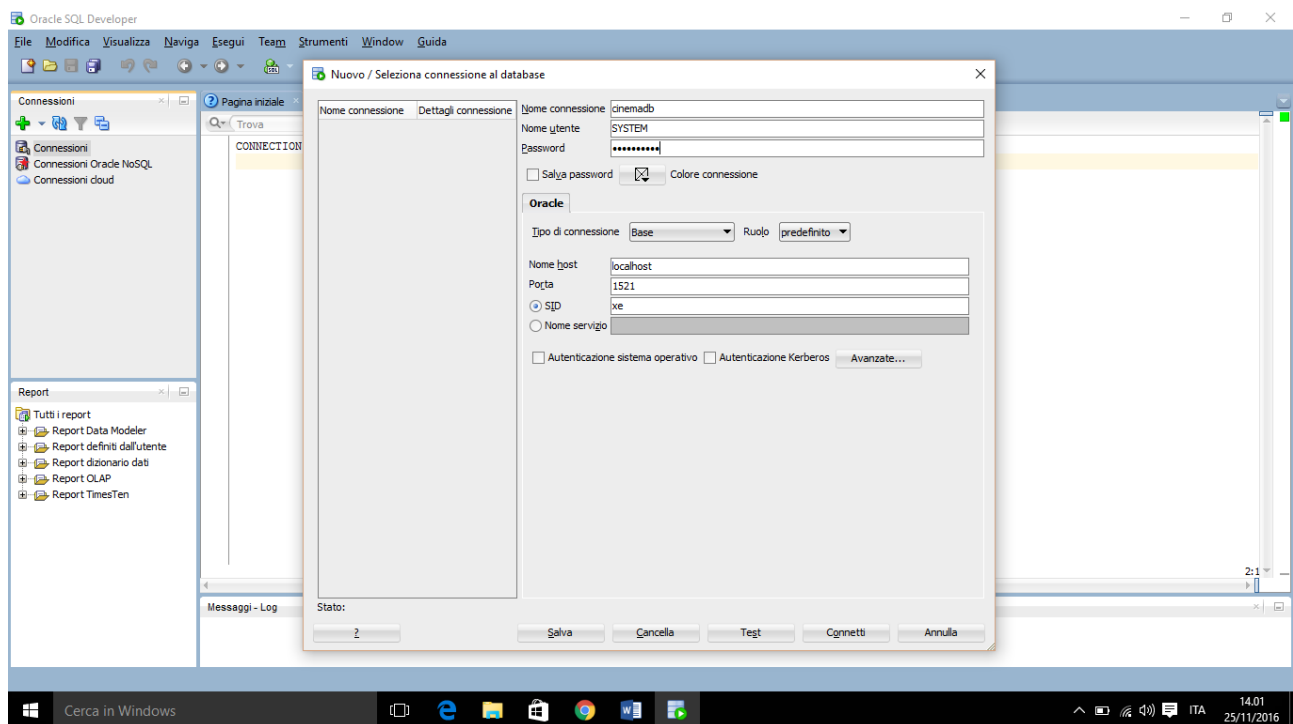
6.2.14 STORAGE TOTALE

	Initial(Mb)	Next(Mb)	Initial(Kb)	Next(Kb)
RECENSIONI			0,68	0,68
FILM_PROGRAMMAZIONE			4,65	4,65
CAST			43,61	-
PARTECIPAZIONE			60,90	26,37
CALENDARI			3368,40	1684,20
CINEMA			2,73	-
RECAPITI_CINEMA			1,17	-
SALE			1,46	-
AFFERENZE_SALA			1283,20	641,60
POSTI			105,46	-
PRENOTAZIONI			35.935,14	-
UTENTI			80,08	16,01
AFFERENZE_POSTO			92.404,68	-
TOT	Mb 130,18	Mb 2,32	Kb 133.292,16	Kb 2.373,51

6.3 Implementazione in SQL

6.3.1 Creazione del database

La creazione del Database è un'operazione preliminare all'implementazione dei Tablespace. Questa fase però esula dalle nostre possibilità dato che la distribuzione utilizzata (Oracle XE 11g) permette di gestire un unico Database già preconfigurato e pronto per essere utilizzato. È necessario unicamente il settaggio di una nuova connessione.



6.3.2 Tablespace

Le operazioni di creazione tablespace sono effettuate tramite le credenziali di SYSTEM, che dispone di completa libertà operativa sul DB.

Il tablespace "ts_cinema" conterrà il patrimonio di informazioni presenti nel nostro Database.

```
CREATE TABLESPACE ts_cinema
DATAFILE 'D:\CINEMA\ts_cinema.dbf'
SIZE 150M
AUTOEXTEND ON NEXT 3M
MAXSIZE UNLIMITED;
```

I valori di SIZE e AUTOEXTEND sono stati ricavati, rispettivamente, dalla somma di tutte le dimensioni initial e next calcolate nel paragrafo "Volumi Totali".

Il valore sono stati volutamente aumentati per includere eventuali arrotondamenti nel dimensionamento.

Sull'entità PRENOTAZIONE si effettua un *partizionamento orizzontale*, cioè una suddivisione dell'insieme delle occorrenze, in due sottoinsiemi disgiunti. (prenotazione e storico_prenotazione) Questo perché la frequenza di consultazione delle prenotazioni a corto termine è molto più alta di quella che riguarda l'intero storico.

È sensato creare uno storico in cui verranno spostate le prenotazioni insieme alle istanze di afferenze_posto alla fine di ogni anno. Viene stanziato appositamente il tablespace "ts_storico".

```
CREATE TABLESPACE ts_storico
DATAFILE 'D:\CINEMA\ts_storico.dbf'
SIZE 130M
AUTOEXTEND ON NEXT 130M
MAXSIZE UNLIMITED;
```




Inoltre viene stanziato uno spazio logico dedicato alla memorizzazione dei tipi CLOB. Questo tablespace ospita gli effettivi contenuti dell'attributo "testo" della tabella RECENSIONI.

Si prevedono 70 film all'anno che moltiplicati per la media di 2000 caratteri di cui è composta una recensione, comportano una dimensione potenzialmente occupata di 14000 byte.

```
CREATE TABLESPACE ts_lob
DATAFILE 'D:\CINEMA\ts_lob.dbf'
SIZE 14K
AUTOEXTEND ON NEXT 14K
MAXSIZE UNLIMITED;
```

6.3.3 Utenti e ruoli

Considerando le operazioni previste sulla base e le applicazioni che si interfaceranno col DB, si forniscono i seguenti ruoli interagenti con la base:

-  **Amministratore:** gestore/creatore della base a cui si assegnano tutti i privilegi
-  **WebApp:** forniti privilegi di lettura e inserimento per effettuare consultazione e prenotazione
-  **DeskApp:** si occupa di gestire inserimenti, aggiornamenti, cancellazioni. Tale ruolo è associato alle applicazioni amministrative. (non implementato)

	<i>Amministratore</i>	<i>WEB_APP</i>	<i>DESK_APP</i>
<i>Recensioni</i>	All Privileges	Select	Select, Insert, Update, Delete
<i>Film_programmazione</i>	All Privileges	Select	Select, Insert, Update, Delete
<i>Cast</i>	All Privileges	Select	Select, Insert, Update, Delete
<i>Partecipazione</i>	All Privileges	Select	Select, Insert, Update, Delete
<i>Calendari</i>	All Privileges	Select	Select, Insert, Update, Delete
<i>Cinema</i>	All Privileges	Select	Select
<i>Recapiti_cinema</i>	All Privileges	Select	Select
<i>Sale</i>	All Privileges	Select	Select
<i>Afferenza_sala</i>	All Privileges	Select	Select
<i>Posti</i>	All Privileges	Select	Select
<i>Prenotazioni</i>	All Privileges	Select, Insert	Select
<i>Utenti</i>	All Privileges	Select, Insert	Select, Delete
<i>Afferenze_posto</i>	All Privileges	Select, Insert	Select
<i>PASSWORD</i>	<i>dba_cinema</i>	<i>web_app</i>	<i>desk_app</i>

```
CREATE USER Amministratore IDENTIFIED BY dba_cinema
DEFAULT TABLESPACE ts_cinema
TEMPORARY TABLESPACE temp;

GRANT dba TO Amministratore;

ALTER USER "AMMINISTRATORE" QUOTA UNLIMITED ON TS_CINEMA;
ALTER USER "AMMINISTRATORE" QUOTA UNLIMITED ON TS_STORICO;
```

Si è preferito utilizzare il ruolo di “dba” predefinito per assegnare i privilegi all’Amministratore.

Utilizzando i privilegi da Amministratore, vengono poi ampliate le politiche di sicurezza definendo gli altri ruoli.

Le seguenti istruzioni si presentano in questo paragrafo per motivi di ordine anche se sono definiti a valle della creazione delle tabelle, delle viste e dei trigger.

```
CREATE ROLE WebApp;
GRANT CONNECT TO WebApp;
GRANT SELECT ON Amministratore.CINEMA TO WebApp;
GRANT SELECT ON Amministratore.RECENSIONI TO WebApp;
GRANT SELECT ON Amministratore.FILM_PROGRAMMAZIONE TO WebApp;
GRANT SELECT ON Amministratore.CAST TO WebApp;
GRANT SELECT ON Amministratore.PARTECIPAZIONE TO WebApp;
GRANT SELECT ON Amministratore.CALENDARI TO WebApp;
GRANT SELECT ON Amministratore.RECAPITI_CINEMA TO WebApp;
GRANT SELECT ON Amministratore.SALE TO WebApp;
GRANT SELECT ON Amministratore.AFFERENZE_SALA TO WebApp;
GRANT SELECT ON Amministratore.POSTI TO WebApp;
GRANT SELECT, INSERT, UPDATE ON Amministratore.PRENOTAZIONI TO WebApp;
GRANT SELECT, INSERT ON Amministratore.UTENTI TO WebApp;
GRANT SELECT, INSERT ON Amministratore.AFFERENZE_POSTO TO WebApp;
```

```
CREATE ROLE DeskApp;
GRANT CONNECT TO DeskApp;
GRANT SELECT, INSERT, UPDATE, DELETE ON Amministratore.RECENSIONI TO
DeskApp;
GRANT SELECT, INSERT, UPDATE, DELETE ON
Amministratore.FILM_PROGRAMMAZIONE TO DeskApp;
GRANT SELECT, INSERT, UPDATE, DELETE ON Amministratore.CAST TO
DeskApp;
GRANT SELECT, INSERT, UPDATE, DELETE ON Amministratore.PARTECIPAZIONE
TO DeskApp;
GRANT SELECT, INSERT, UPDATE, DELETE ON Amministratore.CALENDARI TO
DeskApp;
GRANT SELECT ON Amministratore.CINEMA TO DeskApp;
GRANT SELECT ON Amministratore.RECAPITI_CINEMA TO DeskApp;
GRANT SELECT ON Amministratore.SALE TO DeskApp;
GRANT SELECT ON Amministratore.AFFERENZE_SALA TO DeskApp;
GRANT SELECT ON Amministratore.POSTI TO DeskApp;
GRANT SELECT ON Amministratore.PRENOTAZIONI TO DeskApp;
GRANT SELECT, DELETE ON Amministratore.UTENTI TO DeskApp;
GRANT SELECT ON Amministratore.RECAPITI_CINEMA TO DeskApp;
```

Dopo la creazione dei ruoli si procede alla creazione dei relativi utenti.

```
CREATE USER WEB_APP IDENTIFIED BY web_app
DEFAULT TABLESPACE ts_cinema;
GRANT WebApp to WEB_APP;

CREATE USER DESK_APP IDENTIFIED BY desk_app
DEFAULT TABLESPACE ts_cinema
GRANT DeskApp to DESK_APP;
```

6.3.4 Tabelle

Di seguito sono riportati gli statement SQL per la creazione delle tabelle e dei vincoli di integrità. Per ogni "CREATE TABLE" sono state specificate le clausole di storage che descrivono come lo spazio riservato per ognuna di esse viene utilizzato.

INITIAL: specifica la misura in byte dell'estensione iniziale della tabella.

NEXT: indica la misura per l'estensione successiva.

MINEXTENTS: specifica il numero minimo di extents allocati

MAXEXTENTS: specifica il numero massimo di extents allocati.

PCTINCREASE: indica la percentuale di incremento delle dimensioni delle extents dalla terza in poi relativa all'extets precedente.

I valori di storage sono stati arrondati cifra intera che segue. (1,32 ->2)

Il valore di NEXT deve essere NOT NULL. Dove non previsto è stato assegnato 1Kb.

Il parametro MAXEXTENTS è stato settato a 5.

Per le ipotesi di dimensionamento stabilite verrà allocato un nuovo extents circa ogni anno, garantendo quindi il funzionamento del sistema per 5 anni.

```
CREATE TABLE RECENSIONI (
    CodRecensione NUMBER(3),
    Film NUMBER(3) NOT NULL,
    Testo CLOB,
    CONSTRAINT PK_RECENSIONI PRIMARY KEY(CodRecensione)
)
LOB(Testo) STORE AS LOB_Recensioni (TABLESPACE ts_lob)
STORAGE(INITIAL 1K NEXT 1K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

CREATE TABLE FILM_PROGRAMMAZIONE (
    CodFilm NUMBER(3),
    Titolo VARCHAR2(30) NOT NULL,
    Anno CHAR(4),
    Produttore VARCHAR2(10),
    Genere VARCHAR2(10),
    Durata NUMBER(4,2) NOT NULL,
    DataUscita DATE NOT NULL,
    CONSTRAINT PK_FILM_PROGRAMMAZIONE PRIMARY KEY(CodFilm)
)
STORAGE(INITIAL 5K NEXT 5K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

CREATE TABLE CAST (
    IdCast NUMBER(4),
    Tipo VARCHAR2(7) NOT NULL,
    Nome VARCHAR2(20) NOT NULL,
    Cognome VARCHAR2(20) NOT NULL,
    DataNascita DATE,
    CONSTRAINT PK_CAST PRIMARY KEY(IdCast),
    CONSTRAINT CK_CAST CHECK(Tipo IN ('Attore', 'Regista'))
)
STORAGE(INITIAL 44K NEXT 5K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);
```



```

CREATE TABLE PARTECIPAZIONE(
    Film NUMBER(3),
    Cast NUMBER(4),
    Ruolo VARCHAR2(20),
    CONSTRAINT PK_PARTECIPAZIONE PRIMARY KEY(Film, Cast)
)
STORAGE(INITIAL 61K NEXT 27K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

CREATE TABLE CALENDARI(
    CodProgramma NUMBER(6),
    Ora NUMBER(4,2) NOT NULL,
    Data DATE NOT NULL,
    Film NUMBER(3) NOT NULL,
    Cinema CHAR(2) NOT NULL,
    CONSTRAINT PK_CALENDARI PRIMARY KEY(CodProgramma)
)
STORAGE(INITIAL 3367K NEXT 1685K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

CREATE TABLE CINEMA(
    CodCinema CHAR(2),
    Nome VARCHAR2(15) NOT NULL,
    NumeroSale CHAR(1) NOT NULL,
    Citta VARCHAR2(15) NOT NULL,
    Indirizzo VARCHAR2(20) NOT NULL,
    Civico CHAR(3) NOT NULL,
    CONSTRAINT PK_CINEMA PRIMARY KEY(CodCinema)
)
STORAGE(INITIAL 3K NEXT 1K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

CREATE TABLE RECAPITI_CINEMA(
    Numero VARCHAR2(10),
    Cinema CHAR(2) NOT NULL,
    CONSTRAINT PK_RECAPITI_CINEMA PRIMARY KEY(Numero),
    CONSTRAINT CK_RECAPITI CHECK(LENGTH(Numero)>=9)
)
STORAGE(INITIAL 2K NEXT 1K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

CREATE TABLE SALE(
    NumSala CHAR(1),
    Cinema CHAR(2),
    Superficie CHAR(4),
    Capienza CHAR(3) NOT NULL,
    CONSTRAINT PK_SALE PRIMARY KEY(NumSala, Cinema)
)
STORAGE(INITIAL 2K NEXT 1K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

CREATE TABLE AFFERENZE_SALA(
    Programma NUMBER(6),
    Sala CHAR(1) NOT NULL,
    Cinema CHAR(2) NOT NULL,
    CONSTRAINT PK_AFFERENZE_SALA PRIMARY KEY(Programma)
)
STORAGE(INITIAL 1284K NEXT 642K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

```

```

CREATE TABLE POSTI (
    Numero NUMBER(3),
    Fila NUMBER(3),
    Sala CHAR(1),
    Cinema CHAR(2),
    CONSTRAINT PK_POSTI PRIMARY KEY(Numero, Fila, Sala, Cinema)
)
STORAGE(INITIAL 106K NEXT 1K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

CREATE TABLE PRENOTAZIONI(
    IdPrenotazione NUMBER(7),
    Utente NUMBER(4) NOT NULL,
    Programma NUMBER(4) NOT NULL,
    Pagato CHAR(1) NOT NULL,
    DataPrenotazione DATE NOT NULL DEFAULT SYSDATE,
    CONSTRAINT PK_PRENOTAZIONI PRIMARY KEY(IdPrenotazione),
    CONSTRAINT CK_PRENOTAZIONI CHECK(Pagato IN ('Y', 'N'))
)
STORAGE(INITIAL 35936K NEXT 1K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

CREATE TABLE PRENOTAZIONI_STORICO(
    IdPrenotazione NUMBER(7),
    Utente NUMBER(4) NOT NULL,
    Programma NUMBER(4) NOT NULL,
    Pagato CHAR(1) NOT NULL,
    DataPrenotazione DATE NOT NULL,
    CONSTRAINT PK_PRENOTAZIONI_STORICO PRIMARY KEY(IdPrenotazione),
    CONSTRAINT CK_PRENOTAZIONI_STORICO CHECK(Pagato IN ('Y', 'N'))
)
TABLESPACE ts_storico
STORAGE(INITIAL 35936K NEXT 35936K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

CREATE TABLE UTENTI (
    CodUtente NUMBER(4),
    Username VARCHAR2(10) UNIQUE NOT NULL,
    Pwd VARCHAR2(8) NOT NULL,
    Cognome VARCHAR2(15) NOT NULL,
    Nome VARCHAR2(15) NOT NULL,
    Email VARCHAR2(30) UNIQUE NOT NULL,
    CONSTRAINT PK_UTENTI PRIMARY KEY(CodUtente),
    CONSTRAINT CK_UTENTI1 CHECK(LENGTH(Username)>5),
    CONSTRAINT CK_UTENTI2 CHECK(LENGTH(Pwd)>5),
    CONSTRAINT CK_UTENTI3 CHECK(Pwd!=Username);
)
STORAGE(INITIAL 81K NEXT 16K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

CREATE TABLE AFFERENZE_POSTO(
    Prenotazione NUMBER(7),
    NumPosto NUMBER(3),
    NumFila NUMBER(3),
    Sala CHAR(1),
    Cinema CHAR(2),
    Prezzo NUMBER(4,2) NOT NULL,
    CONSTRAINT PK_AFFERENZE_POSTO PRIMARY KEY(Prenotazione, NumPosto, NumFila, Sala, Cinema)
)
STORAGE(INITIAL 92405K NEXT 1K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE 0);

```

```

CREATE TABLE AFFERENZE_POSTO_STORICO(
    Prenotazione NUMBER(7),
    NumPosto NUMBER(3),
    NumFila NUMBER(3),
    Sala CHAR(1),
    Cinema CHAR(2),
    Prezzo NUMBER(4,2),
    CONSTRAINT PK_AFFERENZE_POSTO_STORICO PRIMARY KEY(Prenotazione,
NumPosto,
    NumFila, Sala, Cinema)
)
TABLESPACE ts_storico
STORAGE(INITIAL 92405K NEXT 1K MINEXTENTS 1 MAXEXTENTS 5 PCTINCREASE
0);

```

6.3.5 Vincoli aggiuntivi

Si è scelto di implementare in un secondo momento, mediante il comando ALTER TABLE, tutti i vincoli di integrità referenziale in modo da evitare di essere condizionati nell'ordine di creazione delle tabelle.

Si ricorda che dove non è stata specificata una differente politica di cancellazione, viene utilizzata la gestione di default di Oracle [ON DELETE NO ACTION] che permette l'eliminazione di una tupla solo se non vi sono altri elementi che fanno riferimento ad essa. Inoltre le politiche di gestione sull'UPDATE, non previste dal DBMS utilizzato, sono implementate in seguito tramite trigger.

```

ALTER TABLE SALE
ADD CONSTRAINT FK_SALE FOREIGN KEY (Cinema)
REFERENCES CINEMA(CodCinema)
ON DELETE CASCADE;

ALTER TABLE POSTI
ADD CONSTRAINT FK_POSTI FOREIGN KEY (Sala, Cinema)
REFERENCES SALE(NumSala, Cinema)
ON DELETE CASCADE;

ALTER TABLE PRENOTAZIONI
ADD CONSTRAINT FK_PRENOTAZIONI1 FOREIGN KEY (Utente)
REFERENCES UTENTI(CodUtente)
ON DELETE CASCADE;

ALTER TABLE PRENOTAZIONI
ADD CONSTRAINT FK_PRENOTAZIONI2 FOREIGN KEY (Programma)
REFERENCES CALENDARI(CodProgramma);

ALTER TABLE PRENOTAZIONI_STORICO
ADD CONSTRAINT FK_PRENOTAZIONI_STORICO1 FOREIGN KEY (Utente)
REFERENCES UTENTI(CodUtente);

ALTER TABLE PRENOTAZIONI_STORICO
ADD CONSTRAINT FK_PRENOTAZIONI_STORICO2 FOREIGN KEY (Programma)
REFERENCES CALENDARI(CodProgramma);

ALTER TABLE AFFERENZE_POSTO
ADD CONSTRAINT FK_AFFERENZE_POSTO1 FOREIGN KEY (Prenotazione)
REFERENCES PRENOTAZIONI(IdPrenotazione);

```

```

ALTER TABLE AFFERENZE_POSTO
ADD CONSTRAINT FK_AFFERENZE_POSTO2 FOREIGN KEY (NumFila, NumPosto,
Sala, Cinema)
REFERENCES POSTI (Fila, Numero , Sala, Cinema);

ALTER TABLE AFFERENZE_POSTO_STORICO
ADD CONSTRAINT FK_AFFERENZE_POSTO_STORICO1 FOREIGN KEY (Prenotazione)
REFERENCES PRENOTAZIONI_STORICO (IdPrenotazione);

ALTER TABLE AFFERENZE_POSTO_STORICO
ADD CONSTRAINT FK_AFFERENZE_POSTO_STORICO2 FOREIGN KEY (NumFila,
NumPosto, Sala, Cinema)
REFERENCES POSTI (Fila, Numero , Sala, Cinema);

ALTER TABLE CALENDARI
ADD CONSTRAINT FK_CALENDARI1 FOREIGN KEY (Film)
REFERENCES FILM_PROGRAMMAZIONE (CodFilm)
ON DELETE SET NULL;

ALTER TABLE CALENDARI
ADD CONSTRAINT FK_CALENDARI2 FOREIGN KEY (Cinema)
REFERENCES CINEMA (CodCinema)
ON DELETE SET NULL;

ALTER TABLE RECENSIONI
ADD CONSTRAINT FK_RECENSIONI FOREIGN KEY (Film)
REFERENCES FILM_PROGRAMMAZIONE (CodFilm)
ON DELETE CASCADE;

ALTER TABLE RECAPITI_CINEMA
ADD CONSTRAINT FK_RECAPITI_CINEMA FOREIGN KEY (Cinema)
REFERENCES CINEMA (CodCinema)
ON DELETE CASCADE;

ALTER TABLE AFFERENZE_SALA
ADD CONSTRAINT FK_AFFERENZE_SALA1 FOREIGN KEY (Programma)
REFERENCES CALENDARI (CodProgramma);

ALTER TABLE AFFERENZE_SALA
ADD CONSTRAINT FK_AFFERENZE_SALA2 FOREIGN KEY (Sala, Cinema)
REFERENCES SALE (NumSala, Cinema);

ALTER TABLE PARTECIPAZIONE
ADD CONSTRAINT FK PARTECIPAZIONE1 FOREIGN KEY (Film)
REFERENCES FILM_PROGRAMMAZIONE (CodFilm);

ALTER TABLE PARTECIPAZIONE
ADD CONSTRAINT FK PARTECIPAZIONE2 FOREIGN KEY (Cast)
REFERENCES CAST (IdCast);

```

1.1.1 Trigger

Sono stati attivati dei trigger sull' insert nelle tabelle utenti, prenotazioni, film_programmazione, cinema, cast, calendari per rendere automatico e sequenziale l'inserimento dei codici che identificano la chiave primaria (business rule BR04)

Esempio:

```
CREATE OR REPLACE TRIGGER PK_UTENTI
BEFORE INSERT ON UTENTI
FOR EACH ROW
BEGIN
    SELECT MAX(CODUTENTE)+1
    INTO :NEW.CODUTENTE
    FROM UTENTI;
END;
```

Altra tipologia di trigger implementata è quella che agisce sull'update delle chiavi di alcune tabelle. (BR05)

Esempio:

```
CREATE OR REPLACE TRIGGER UPDATE_CASCADE_CINEMA
BEFORE UPDATE ON CINEMA
FOR EACH ROW
BEGIN
    UPDATE SALE
    SET CINEMA=:NEW.CINEMA
    WHERE CINEMA=:OLD.CINEMA;
END;
```

Infine è stato implementato un trigger per impedire la registrazione di più di 2500 utenti (BR06).

```
CREATE OR REPLACE TRIGGER CHECK_NUM_UTENTI
BEFORE INSERT ON UTENTI
FOR EACH STATEMENT
BEGIN
    DECLARE
        ERRORE EXCEPTION;
        CONT INTEGER;
    BEGIN
        SELECT COUNT* INTO CONT FROM UTENTI
        IF (CONT > 2500) THEN RAISE ERRORE;
        END IF;
    WHEN ERRORE
    RAISE_APPLICATION_ERROR(-100, 'LIMITE UTENTI RAGGIUNTO');
END;
```

1.1.2 Stored Procedure

Per rispettare la business rule BR01 si rende necessaria l'implementazione di una stored procedure, che quando attivata sposta automaticamente tutte le prenotazioni dell'anno corrente nella tabella PRENOTAZIONI_STORICO.

```
CREATE OR REPLACE PROCEDURE PREN_STORICO AS

DECLARE
    CURSOR C IS
        SELECT IDPRENOTAZIONE, UTENTE, PROGRAMMA, PAGATO, DATAPRENOTAZIONE
        FROM PRENOTAZIONI
        WHERE DATAPRENOTAZIONE<SYSDATE;

    ID1 PRENOTAZIONI.IDPRENOTAZIONE %TYPE;
    UTENTE1 PRENOTAZIONI.UTENTE %TYPE;
    PROGRAMMA1 PRENOTAZIONI.PROGRAMMA %TYPE;
    PAGATO1 PRENOTAZIONI.PAGATO %TYPE;
    DATA1 PRENOTAZIONI.DATAPRENOTAZIONE %TYPE;

BEGIN
    OPEN C;
    LOOP

        FETCH C INTO ID1, UTENTE1, PROGRAMMA1, PAGATO1, DATA1;
        DELETE FROM PRENOTAZIONI
        WHERE IDPRENOTAZIONE = ID1;

        INSERT INTO PRENOTAZIONI_STORICO
        VALUES (ID1, UTENTE1, PROGRAMMA1, PAGATO1, DATA1);

    WHEN C% NOT NULL;
    END LOOP;
    CLOSE;
    COMMIT;
    END;

END PREN_STORICO;

- - Attivata con la sintassi SQL:

EXEC PREN_STORICO;
```