



Capitolo 6

Scrittura di query avanzate

- 6.1 **Funzioni SQL avanzate**
- 6.2 **Sfruttamento delle viste**
- 6.3 **Utilizzo di SQL per la generazione di SQL**
- 6.4 **L'espressione CASE**
- 6.5 **Quiz**

Prima di passare a Data Manipulation Language nel Capitolo 7, il presente capitolo intende approfondire le conoscenze sulle query SQL descrivendo alcuni argomenti avanzati accennati nei Capitoli 4 e 5. Sono trattati i seguenti argomenti:

- funzioni SQL avanzate, tra cui funzioni carattere, matematiche e data/ora;
- una descrizione dello sfruttamento delle viste;
- una descrizione dell'utilizzo di SQL per la generazione di dichiarazioni SQL;
- informazioni sull'espressione SQL CASE e sul suo utilizzo per la creazione di dichiarazioni contenenti parti eseguite solo in determinate condizioni.

6.1 Funzioni SQL avanzate

Le funzioni SQL sono state presentate nel Capitolo 4. Gli argomenti di seguito descrivono utili funzioni non trattate nel Capitolo 4. Oltre alle funzioni carattere e matematiche, sono comprese alcune funzioni per data e ora. Ricordare che tutte le funzioni SQL hanno una caratteristica in comune: restituiscono un singolo valore, quindi sono utili in numerose dichiarazioni SQL, tra cui l'elenco di colonne della dichiarazione SELECT e la clausola WHERE. Si ricordi inoltre che esistono molte altre funzioni specifiche per ciascuna implementazione offerte dai diversi produttori di DBMS, quindi consultare sempre la documentazione fornita dal produttore.

Funzioni carattere

Le *funzioni carattere* operano su dati carattere. L'argomento presenta alcune funzioni molto utilizzate, oltre a quelle descritte nel Capitolo 4. Se non indi-



cato diversamente, la funzione è supportata dalle implementazioni SQL più diffuse, tra cui Microsoft SQL Server, Oracle, DB2 e MySQL.

REPLACE

La funzione REPLACE cerca una stringa di caratteri e sostituisce i caratteri trovati nella stringa di ricerca con i caratteri elencati in un stringa di sostituzione. Di seguito la sintassi generale:

```
REPLACE(stringa_caratteri, stringa_ricerca, stringa_sostituzione)
```

- **stringa_caratteri:** è la stringa in cui effettuare la ricerca e molto spesso è un nome di colonna di tabella, ma può essere qualsiasi espressione che generi una stringa di caratteri;
- **stringa_ricerca:** è la stringa di uno o più caratteri da trovare in stringa_caratteri;
- **stringa_sostituzione:** è la stringa che sostituisce tutte le occorrenze di stringa_ricerca trovate in stringa_caratteri.

Di seguito un esempio che sostituisce con punti tutti i trattini trovati nel numero di telefono di una persona (sono mostrate solo le prime due righe del set di risultati):

```
SELECT PERSON_PHONE,
       REPLACE(PERSON_PHONE, '-', '.') AS DISPLAY_PHONE
FROM PERSON;
```

PERSON_PHONE	DISPLAY_PHONE
230-229-8976	230.229.8976
401-617-7297	401.617.7297

LTRIM

La funzione LTRIM elimina gli spazi iniziali (a sinistra) in una stringa di caratteri. Si noti che vengono eliminati solo gli spazi iniziali: gli spazi interni e gli spazi finali vengono lasciati nella stringa. Non esistono dati con spazi iniziali e finali nel database della videoteca, quindi viene mostrato un esempio generale:

```
LTRIM (' String with spaces ')
Restituisce questa stringa: 'String with spaces '
```

RTRIM

La funzione RTRIM funziona come LTRIM, ma elimina gli spazi finali. Se è necessario eliminare gli spazi iniziali e finali, è possibile annidare LTRIM ed RTRIM, come di seguito:

```
RTRIM(LTRIM (' String with spaces '))
Restituisce questa stringa: 'String with spaces'
```

NOTA Oracle offre un'utile funzione chiamata TRIM che elimina gli spazi iniziali e finali. Per altre implementazioni, è sempre possibile annidare le funzioni LTRIM ed RTRIM e ottenere il medesimo risultato.

Funzione di valore null (NVL, ISNULL, IFNULL)

Oracle, Microsoft SQL Server e MySQL offrono una funzione che sostituisce i valori null con un valore selezionato. Sfortunatamente, ciascuna implementazione utilizza un nome diverso per la funzione, che si chiama NVL in Oracle, ISNULL in SQL Server e IFNULL in MySQL. Sembra che DB2 non abbia una funzione equivalente. Gli esempi di seguito selezionano LATE_OR_LOSS_FEE dalla tabella MOVIE_RENTAL con valori null sostituiti da 0. La transazione 9 è stata selezionata perché contiene due film, uno dei quali ha un valore null per LATE_OR_LOSS_FEE (un buon esempio che mostra che i valori null vengono trasformati, mentre i valori non null vengono lasciati inalterati).

Oracle:

```
SELECT NVL(LATE_OR_LOSS_FEE, 0) AS LATE_OR_LOSS_FEE
FROM MOVIE_RENTAL
WHERE TRANSACTION_ID=9;
```

```
LATE_OR_LOSS_FEE
-----
                0
                29.98
```

2 rows selected.

Microsoft SQL Server:

```
SELECT ISNULL(LATE_OR_LOSS_FEE, 0) AS LATE_OR_LOSS_FEE
FROM MOVIE_RENTAL
WHERE TRANSACTION_ID=9;
```

```
LATE_OR_LOSS_FEE
-----
                29.98
                .00
```

(2 rows affected)

MySQL:

```
SELECT IFNULL(LATE_OR_LOSS_FEE, 0) AS LATE_OR_LOSS_FEE
FROM MOVIE_RENTAL
WHERE TRANSACTION_ID=9;
```

```
+-----+
| LATE_OR_LOSS_FEE |
+-----+
|                29.98 |
```

```
|          0.00 |
+-----+
```

2 rows in set (0.16 sec)

Si noti la differenza nella formattazione dell'output per il client a riga di comando MySQL. Sebbene solitamente i client SQL di produttori diversi formattino i risultati in modo diverso, i dati sono uguali.

ASCII

La funzione ASCII restituisce il valore del set di caratteri ASCII (un numero tra 0 e 255) per una stringa di caratteri contenente un singolo carattere. Per esempio, il codice ASCII per lo spazio è 32, quindi ASCII(' ') restituisce un valore pari a 32.

CHAR (CHR)

La funzione CHAR (chiamata CHR in Oracle e DB2) restituisce il carattere associato a un valore ASCII (un numero tra 0 e 255). Per esempio, la funzione ASCII(44) restituisce una virgola, poiché il valore ASCII per la virgola è 44. Questa funzione è particolarmente utile per il concatenamento di caratteri che non possono essere visualizzati o sono difficili da gestire in SQL. Nella tabella di seguito sono elencati alcuni caratteri ASCII utilizzati solitamente con questa funzione. Se è necessario conoscere altri valori, è possibile utilizzare la funzione ASCII o una tabella di set di caratteri ASCII (facilmente reperibile su Internet).

VALORE	CARATTERE
9	Tabulazione
10	Interruzione di riga
13	Ritorno a capo
39	Virgoletta semplice

Di seguito alcuni esempi. Ricordare che gli operatori di concatenamento non sono uguali nelle diverse implementazioni di DBMS ('+' per Microsoft SQL Server, '||' per la maggior parte delle altre implementazioni).

- Trovare i titoli di film contenenti un carattere di tabulazione:

Microsoft SQL Server:

```
SELECT MOVIE_ID FROM MOVIE
WHERE MOVIE_TITLE LIKE '%'+CHAR(9)+'%';
```

```
MOVIE_ID
(0 rows affected)
```

Oracle e DB2:

```
SELECT MOVIE_ID FROM MOVIE
WHERE MOVIE_TITLE LIKE '%' || CHR(9) || '%';
```

no rows selected

NOTA È possibile modificare la query Tab in modo da trovare film con titoli contenenti virgolette semplici modificando 9 in 39. Il risultato è Movie ID 3 (Something's Gotta Give).

Funzioni matematiche

Come si può immaginare dal nome, le *funzioni matematiche* restituiscono il risultato di un'operazione matematica e solitamente richiedono come parametro di input un'espressione numerica, che può essere un valore letterale, un valore numerico di colonna di tabella o qualsiasi espressione (compreso l'output di un'altra funzione) che generi un valore numerico.

SIGN

La funzione SIGN prende un'espressione numerica e restituisce uno dei seguenti valori, in base al segno del numero in input:

VALORE RESTITUITO	SIGNIFICATO
-1	Il numero in input è negativo
0	Il numero in input è zero
1	Il numero in input è positivo
null	Il numero in input è null

Di seguito un esempio:

```
SELECT LATE_OR_LOSS_FEE,
SIGN(LATE_OR_LOSS_FEE) AS FEE_SIGN
FROM MOVIE_RENTAL
WHERE LATE_OR_LOSS_FEE IS NOT NULL;
```

```
LATE_OR_LOSS_FEE FEE_SIGN
-----
29.99           1
4              1
4              1
29.98           1
```

SQRT

La funzione SQRT prende una singola espressione numerica e restituisce la sua radice quadrata. La sintassi generale è la seguente:

SQRT (*espressione_numerica*)

Il risultato non ha alcun significato, ma a scopo illustrativo si prende la radice quadrata dei prezzi non null per restituzione in ritardo o perdita esaminati nell'esempio precedente:

```
SELECT LATE_OR_LOSS_FEE,
       SQRT(LATE_OR_LOSS_FEE) AS FEE_SQRT
FROM MOVIE_RENTAL
WHERE LATE_OR_LOSS_FEE IS NOT NULL;
```

```
LATE_OR_LOSS_FEE FEE_SQRT
-----
29.99 5.47631263
      4          2
      4          2
29.98 5.47539953
```

CEILING (CEIL)

La funzione CEILING restituisce l'intero minimo, maggiore o uguale al valore dell'espressione numerica fornita come parametro di input. In altre parole, arrotonda al numero intero successivo. Esistono alcune interessanti questioni di compatibilità di nomi tra le diverse implementazioni SQL: Microsoft SQL Server utilizza il nome CEILING, Oracle utilizza il nome CEIL, DB2 e MySQL ammettono l'utilizzo di entrambi i nomi (CEIL o CEILING).

Come esempio, si applica CEILING ai prezzi di consegna in ritardo o perdita (se si utilizza Oracle, modificare CEILING in CEIL):

```
SELECT LATE_OR_LOSS_FEE,
       CEILING(LATE_OR_LOSS_FEE) AS FEE_CEILING
FROM MOVIE_RENTAL
WHERE LATE_OR_LOSS_FEE IS NOT NULL;
```

```
LATE_OR_LOSS_FEE FEE_CEILING
-----
4.00             4
4.00             4
29.99           30
29.98           30
```

FLOOR

La funzione FLOOR è l'opposto logico della funzione CEILING: restituisce l'intero minore o uguale al valore dell'espressione numerica fornita come parametro di input. In altre parole, arrotonda al numero intero *precedente*. Di seguito un esempio che mostra FLOOR applicata ai costi di consegna in ritardo o perdita:

```
SELECT LATE_OR_LOSS_FEE,
       FLOOR(LATE_OR_LOSS_FEE) AS FEE_FLOOR
FROM MOVIE_RENTAL
WHERE LATE_OR_LOSS_FEE IS NOT NULL;
```

LATE_OR_LOSS_FEE	FEE_FLOOR
4.00	4
4.00	4
29.99	29
29.98	29

Funzioni di data e ora

Esiste pochissima coerenza nelle *funzioni di data e ora* tra i diversi produttori di DBMS. In gran parte questo dipende dal fatto che molti produttori hanno sviluppato i tipi di dati per data e ora prima dello sviluppo degli standard. A causa di tale diversità, le funzioni di data e ora sono presentate in forma sommaria per Microsoft SQL Server, Oracle, DB2 e MySQL. Come sempre, consultare la documentazione del produttore per avere spiegazioni dettagliate sull'utilizzo di tali funzioni. I termini indicati in corsivo sono definiti nelle note, dopo ciascuna tabella. Nel paragrafo viene utilizzato il termine "data/ora" per indicare una stringa di caratteri contenente una data e un'ora in un formato accettabile per il particolare DBMS.

Funzioni di data e ora per Microsoft SQL Server

Microsoft SQL Server offre le funzioni di data e ora indicate nella tabella di seguito:

FUNZIONE	SCOPO	PARAMETRI DI INPUT
DATEADD	Restituisce una nuova data/ora calcolata sommando un intervallo alla parte_data della data fornita	parte_data, quantità intervallo, data/ora
DATEDIFF	Restituisce il numero di delimitazioni di data/ora attraversate da due date	parte_data, data/ora iniziale, data/ora finale
DATENAME	Restituisce un nome di testo che rappresenta la parte_data selezionata della data/ora di input	parte_data, data/ora
DATEPART	Restituisce un intero che rappresenta la parte_data selezionata della data/ora di input	parte_data, data/ora
DAY	Restituisce un intero che rappresenta il giorno contenuto nella data/ora fornita	data/ora
GETDATE	Restituisce la data/ora di sistema corrente	Nessuno
GETUTCDATE	Restituisce la data/ora UTC (Universal Coordinated Time, ora coordinata universale) corrente	Nessuno
MONTH	Restituisce un intero che rappresenta il mese contenuto nella data/ora fornita	data/ora
YEAR	Restituisce un intero (quattro cifre) che rappresenta l'anno contenuto nella data/ora fornita	data/ora

NOTA *parte_data* è un parametro che specifica una parte di una data, quale anno, mese, giorno, ora, minuto, secondo e millisecondo. Per valori e opzioni, consultare la documentazione Microsoft SQL Server.

Funzioni di data e ora per Oracle

Oracle dispone di oltre 24 funzioni di data e ora. Si ricordi che sebbene Oracle chiami il tipo di dati DATE, tutte le date contengono un componente orario; quando non viene utilizzato è impostato su zero (a rappresentare la mezzanotte). Le funzioni più utilizzate sono elencate nella tabella di seguito:

FUNZIONE	SCOPO	PARAMETRI DI INPUT
ADD_MONTHS	Aggiunge alla data fornita il numero di mesi fornito	data, numero di mesi (valore positivo o negativo)
CURRENT_DATE	Restituisce la data corrente nel fuso orario impostato per la sessione di database	Nessuno
EXTRACT	Estrae dalla data fornita il campo di data/ora specificato	parola chiave campo data/ora, data
LAST_DAY	Restituisce la data fornita con il giorno spostato sull'ultimo giorno del mese	data
MONTHS_BETWEEN	Restituisce il numero di mesi (comprese le parti frazionarie) tra due date fornite; il risultato è negativo se la seconda data è antecedente alla prima	prima data, seconda data
SYSDATE	Restituisce la data e l'ora di sistema corrente	Nessuno
TO_CHAR	Quando utilizzata con una data, converte la data in una stringa di caratteri specificata da <code>stringa_formato</code>	data, <code>stringa_formato</code>
TO_DATE	Converte la stringa di caratteri fornita in una data con formattazione interna Oracle, utilizzando <code>stringa_formato</code> come modello per l'interpretazione dei contenuti della stringa di caratteri	data, <code>stringa_formato</code>
TRUNC	Tronca una data all'unità di tempo specificata nella parola chiave di campo per data/ora. Se la parola chiave viene omessa, la data viene troncata al giorno corrente	data, parola chiave campo data/ora

NOTA

- *parola chiave campo data/ora* è una parola chiave che specifica uno dei campi contenuti in una data Oracle, quale YEAR, MONTH, DAY, HOUR, MINUTE e SECOND;
- *stringa_formato* è una stringa di caratteri costituita da simboli, che specificano il formato da utilizzare per la data quando viene convertita da o in una stringa di caratteri. Esistono oltre 40 simboli diversi che possono essere utilizzati in *stringa_formato* (per un elenco completo consultare la documentazione Oracle). Per esempio, *stringa_formato* "MM/DD/YYYY HH:MI" si riferisce a una stringa di caratteri data del tipo "12/01/2004 11:58", mentre *stringa_formato* "DD-MON-RR" (il formato Oracle predefinito) si riferisce a una stringa del tipo "01-Dec-04";
- *TO_CHAR* può anche essere utilizzata per convertire valori numerici in stringhe di caratteri;
- *TRUNC* può anche essere utilizzata per troncatura valori numerici, per eliminare le cifre a destra del punto decimale.

Funzioni di data e ora per MySQL

MySQL dispone di oltre 30 funzioni di data e ora. Le funzioni più utilizzate sono elencate nella tabella di seguito:

FUNZIONE	SCOPO	PARAMETRI DI INPUT
ADDDATE	Somma due espressioni di data, intervallo o data/ora, generando una nuova data	espressione 1, espressione 2
ADDTIME	Somma due espressioni di ora, generando una nuova ora	espressione 1, espressione 2
CURDATE	Restituisce la data corrente nel formato AAAA-MM-GG	Nessuno
DATE	Restituisce la parte di data di un'espressione di data/ora	espressione di data/ora
DATEDIFF	Restituisce il numero di giorni tra due date	data iniziale, data finale
DATE_FORMAT	Formatta una data secondo una stringa di formato	data, stringa di formato
DAYNAME	Restituisce il nome per il giorno della settimana contenuto in una data	data
DAYOFMONTH	Restituisce il giorno del mese, nell'intervallo tra 1 e 31	data
DAYOFWEEK	Restituisce un indice numerico per il giorno della settimana contenuto in una data (1 per domenica, 2 per lunedì e così via)	data
DAYOFYEAR	Restituisce il giorno dell'anno per il giorno contenuto in una data con un intervallo valido tra 1 e 366	data
LAST_DAY	Modifica il giorno in una data nell'ultimo giorno del mese	data
MONTH	Restituisce il mese contenuto in una data con un intervallo valido tra 1 e 12	data
MONTHNAME	Restituisce il nome del mese contenuto in una data	data
NOW	Restituisce la data e l'ora corrente	Nessuno
STR_TO_DATE	Converte una stringa di caratteri in un elemento di data in formato data/ora; la stringa di formato indica il formato delle informazioni di data nella stringa di caratteri di input	stringa di caratteri, stringa di formato
TIME	Restituisce la parte oraria di un'espressione di data/ora	data/ora
TIMEDIFF	Restituisce la differenza oraria tra due parametri data/ora o espressioni di ora	espressione 1, espressione 2
TIME_FORMAT	Formatta una data secondo la stringa di formato	ora, stringa di formato
UTC_DATE	Restituisce la data UTC (Universal Coordinated Time, ora coordinata universale) corrente	Nessuno
UTC_TIME	Restituisce l'ora UTC (Universal Coordinated Time, ora coordinata universale) corrente	Nessuno
WEEKOFYEAR	Restituisce le settimana dall'anno, nell'intervallo tra 1 e 54	data

NOTA *stringa di formato è una stringa di caratteri che indica le opzioni di formattazione per parti della data. Per ulteriori informazioni consultare la documentazione MySQL.*

Funzioni di data e ora per DB2

DB2 UDB dispone di oltre 20 funzioni di data e ora. Le funzioni più utilizzate sono elencate nella tabella di seguito:

FUNZIONE	SCOPO	PARAMETRI DI INPUT
DATE	Converte un'espressione in una data	espressione
DAY	Restituisce la parte di giorno di un'espressione di data/ora	espressione di data/ora
DAYNAME	Restituisce il nome del giorno della settimana per una data o un'espressione di data/ora	espressione di data/ora
DAYOFWEEK	Restituisce il giorno della settimana (1 per domenica, 2 per lunedì e così via) per un'espressione di data/ora	espressione di data/ora
DAYS	Restituisce una rappresentazione intera di una data	espressione di data/ora
MINUTE	Restituisce la parte in minuti di un'espressione di data/ora	espressione di data/ora
MONTH	Restituisce la parte di mese di un'espressione di data/ora	espressione di data/ora
MONTHNAME	Restituisce il nome del mese per una data o un'espressione di data/ora	espressione di data/ora
QUARTER	Restituisce un intero nell'intervallo tra 1 e 4, che rappresenta il trimestre di calendario in cui rientra la data	espressione di data/ora
SECOND	Restituisce la parte in secondi di un'espressione di data/ora	espressione di data/ora
TIME	Restituisce la parte in ore di un'espressione di data/ora	espressione di data/ora
WEEK	Restituisce la settimana dall'anno come intero, nell'intervallo tra 1 e 54	espressione di data/ora
YEAR	Restituisce la parte di anno di un'espressione di data/ora	espressione di data/ora

6.2 Sfruttamento delle viste

Nel Capitolo 1 si è detto che una vista è una query di database memorizzata, che offre a un utente di database un sottoinsieme personalizzato dei dati di una o più tabelle nel database. Si noti che Microsoft Access utilizza il termine *query* in luogo di vista. Il vantaggio delle viste è rappresentato dal fatto che, una volta create, possono essere interrogate come le tabelle. Di fatto, l'utente non deve sapere di utilizzare una vista in luogo di una tabella reale. Utilizzando le viste si possono presentare alcune limitazioni quando si cerca di effettuare alcune manipolazioni (inserimenti, aggiornamenti e cancellazioni) dei dati (consultare la documentazione del proprio DBMS), ma le query funzionano allo stesso modo sulle viste e sulle tabelle. Inoltre, le viste offrono numerosi vantaggi:

- nascondere colonne a un utente che non ha necessità di vederle o non deve vederle;
- nascondere righe a un utente che non ha necessità di vederle o non deve vederle;
- nascondere operazioni complesse, quali le unioni;
- migliorare le prestazioni della query (in alcuni RDBMS, per esempio Microsoft SQL Server).

Gli esempi del presente paragrafo mostrano viste che offrono i vantaggi citati in precedenza. Come si è visto nel Capitolo 4, la sintassi generale per la creazione di una vista è la seguente:

```
CREATE [OR REPLACE] VIEW nome_vista AS query_sql;
```

Di seguito alcuni esempi pratici che utilizzano il database di esempio per la videoteca:

- Evidenti regole sulla privacy impongono al responsabile del negozio di mantenere strettamente confidenziali gli ID per le imposte (numeri di sicurezza sociale) e le paghe dei dipendenti. Tuttavia esistono numerose applicazioni basate sul Web che richiedono alcune informazioni sui dipendenti, quali il nome del dipendente e il nome del responsabile corrispondente (se esiste). In tali casi è necessaria una vista che possa essere utilizzata da tali applicazioni Web, senza che uno sviluppatore che utilizza i dati possa accidentalmente rivelare informazioni confidenziali. La query nella vista deve unirsi alla tabella PERSON per trovare il nome del dipendente ed effettuare una nuova unione (utilizzando un'unione esterna) per ricavare il nome del responsabile del dipendente. Inoltre i nomi di colonna devono essere il più possibile leggibili per un team di sviluppo Web. Di seguito la dichiarazione SQL per la creazione della vista:

```
CREATE VIEW EMPLOYEE_LIST AS
SELECT A.PERSON_ID AS ID,
       B.PERSON_GIVEN_NAME AS FIRST_NAME,
       B.PERSON_MIDDLE_NAME AS MIDDLE_NAME,
       B.PERSON_FAMILY_NAME AS LAST_NAME,
       C.PERSON_GIVEN_NAME AS MANAGER_FIRST_NAME,
       C.PERSON_FAMILY_NAME AS MANAGER_LAST_NAME
FROM EMPLOYEE A JOIN PERSON B
     ON A.PERSON_ID = B.PERSON_ID
LEFT OUTER JOIN PERSON C
     ON A.SUPERVISOR_PERSON_ID = C.PERSON_ID
```

View created.

Quando la vista è stata creata, è possibile scrivere query su di essa come si fa per le tabelle. Di seguito una semplice query sulla vista creata in precedenza:

```
SELECT LAST_NAME, MANAGER_LAST_NAME
FROM EMPLOYEE_LIST
ORDER BY LAST_NAME;
```

LAST_NAME	MANAGER_LAST_NAME
Alexander	
Bernstein	Alexander
Chung	Alexander

- Per un elenco di film per un catalogo, si desiderano elencare MPAA Rating Description e Movie Genre Description. È evidente che tale unione di

tre tabelle è necessaria molto spesso, quindi si può aiutare lo staff della videoteca creando una vista con le unioni già pronte. Di seguito la dichiarazione SQL per la creazione della vista:

```
CREATE VIEW MOVIE_LISTING AS
SELECT A.MOVIE_ID, B.MOVIE_GENRE_DESCRIPTION AS GENRE,
       C.MPAA_RATING_CODE AS RATING,
       C.MPAA_RATING_DESCRIPTION AS RATING_DESC,
       A.MOVIE_TITLE, A.RETAIL_PRICE_VHS,
       A.RETAIL_PRICE_DVD, A.YEAR_PRODUCED
FROM MOVIE A JOIN MOVIE_GENRE B ON
       A.MOVIE_GENRE_CODE = B.MOVIE_GENRE_CODE
       JOIN MPAA_RATING C ON
       A.MPAA_RATING_CODE = C.MPAA_RATING_CODE
```

Di seguito una semplice query che utilizza la vista invece delle tabelle:

```
SELECT GENRE, RATING, MOVIE_TITLE
FROM MOVIE_LISTING
ORDER BY GENRE, RATING, MOVIE_TITLE;
```

GENRE	RATING	MOVIE_TITLE
Action and Adventure	PG-13	Master and Commander: The Far
Action and Adventure	PG-13	Pirates of the Caribbean: The
Action and Adventure	PG-13	The Day After Tomorrow
Action and Adventure	PG-13	The Italian Job
Action and Adventure	R	Kill Bill: Vol. 1
Action and Adventure	R	Man on Fire
Action and Adventure	R	The Last Samurai
Comedy	PG-13	50 First Dates
Comedy	PG-13	Matchstick Men
Comedy	PG-13	Something's Gotta Give
Comedy	PG-13	The School of Rock
Drama	PG-13	Big Fish
Drama	R	Cold Mountain
Drama	R	Lost in Translation
Drama	R	Monster
Drama	R	Mystic River
Drama	R	Road to Perdition
Foreign	R	Das Boot
Romance	PG-13	13 Going on 30
Romance	PG-13	Two Weeks Notice

20 rows selected.

- La videoteca è interessata all'installazione di un computer nella sezione bambini per consentire ai più giovani di cercare i film online. Tuttavia il responsabile del negozio desidera essere certo del fatto che su tale computer siano visualizzati solo i film con classificazione G, PG e PG-13.

È semplice rispettare la regola nel programma applicativo che controlla il computer, tuttavia se viene utilizzata una vista di database e l'applicazione per il computer è costretta a utilizzare solo la vista, non possono verificarsi errori nella logica di programmazione dell'applicazione ed è semplice modificare la vista in un secondo momento, se l'MPAA cambia nuovamente il sistema di classificazione (come ha già fatto molte volte in passato). Invece di ripetere in questa vista la logica di unione della vista MOVIE_LISTING, per creare la nuova vista è possibile utilizzare semplicemente la vista MOVIE_LISTING, poiché è possibile creare una vista basata su una vista esistente, disponendo in tal modo di una flessibilità infinita. Di seguito l'esempio:

```
CREATE VIEW CHILDRENS_MOVIE_LISTING AS
SELECT * FROM MOVIE_LISTING
WHERE RATING IN ('G','PG','PG-13')
```

Utilizzando la stessa selezione utilizzata per MOVIE_LISTING, di seguito il nuovo risultato:

```
SELECT GENRE, RATING, MOVIE_TITLE
FROM CHILDRENS_MOVIE_LISTING
ORDER BY GENRE, RATING, MOVIE_TITLE;
```

GENRE	RATING	MOVIE_TITLE
Action and Adventure	PG-13	Master and Commander: The Far
Action and Adventure	PG-13	Pirates of the Caribbean: The
Action and Adventure	PG-13	The Day After Tomorrow
Action and Adventure	PG-13	The Italian Job
Comedy	PG-13	50 First Dates
Comedy	PG-13	Matchstick Men
Comedy	PG-13	Something's Gotta Give
Comedy	PG-13	The School of Rock
Drama	PG-13	Big Fish
Romance	PG-13	13 Going on 30
Romance	PG-13	Two Weeks Notice

11 rows selected.

Si noti come la clausola WHERE nella vista CHILDRENS_MOVIE_LISTING blocca le righe non desiderate dai risultati di query. Tuttavia è necessario prestare attenzione alla creazione di viste basate su altre viste: la vista CHILDRENS_MOVIE_LISTING dipende dalla vista MOVIE_LISTING e può divenire invalida o funzionare in modo errato, se viene modificata la vista MOVIE_LISTING. Senza pianificazione e controllo, si può ottenere come risultato un castello di carte, che crolla ogni qualvolta viene effettuata una modifica. In ogni caso le viste sono uno strumento potente che non può essere ignorato.

6.3 Utilizzo di SQL per la generazione di SQL

La maggior parte dei prodotti RDBMS viene fornita con un insieme di viste di catalogo, che consentono alle query di accedere ai metadati che descrivono gli oggetti di database contenuti nel database. Gli amministratori di database esperti sanno come sfruttare i dati nel catalogo e utilizzano dichiarazioni SQL per generare altre dichiarazioni SQL.

Generazione di SQL in Oracle

In Oracle, la vista USER_TABLES contiene informazioni su tutte le tabelle appartenenti all'utente di database corrente. È possibile utilizzare il comando "DESCRIBE USER_TABLES" per vedere la definizione della vista, oppure consultare il manuale Oracle Server Reference per descrizioni di questa e altre viste di catalogo.

Di seguito una dichiarazione SQL che crea un comando DROP TABLE per ogni tabella trovata in USER_TABLES. La clausola WHERE è stata aggiunta per eliminare alcune tabelle interne di Oracle, che possono altrimenti comparire nel set di risultati. Esistono anche tecniche disponibili per inviare i risultati di query a un file, che può quindi essere utilizzato come script, ma vanno oltre gli scopi del presente volume (per ulteriori informazioni consultare il comando Oracle SQL*Plus SP00L).

```
SELECT 'DROP TABLE ' || TABLE_NAME ||
       ' CASCADE CONSTRAINTS;' AS SQL
FROM USER_TABLES
WHERE TABLE_NAME NOT LIKE 'BIN$%';
```

SQL

```
-----
DROP TABLE MOVIE_RENTAL CASCADE CONSTRAINTS;
DROP TABLE CUSTOMER_TRANSACTION CASCADE CONSTRAINTS;
DROP TABLE CUSTOMER_ACCOUNT_PERSON CASCADE CONSTRAINTS;
DROP TABLE EMPLOYEE CASCADE CONSTRAINTS;
DROP TABLE MOVIE_LANGUAGE CASCADE CONSTRAINTS;
DROP TABLE MOVIE_COPY CASCADE CONSTRAINTS;
DROP TABLE MOVIE CASCADE CONSTRAINTS;
DROP TABLE PERSON CASCADE CONSTRAINTS;
DROP TABLE MPAA_RATING CASCADE CONSTRAINTS;
DROP TABLE MOVIE_GENRE CASCADE CONSTRAINTS;
DROP TABLE LANGUAGE CASCADE CONSTRAINTS;
DROP TABLE CUSTOMER_ACCOUNT CASCADE CONSTRAINTS;
```

Generazione di SQL in Microsoft SQL Server

Microsoft utilizza il termine "tabelle di sistema" per le tabelle SQL Server contenenti metadati. È possibile trovarne le descrizioni in Books Online

alla voce “system tables.” Di seguito l’equivalente SQL Server dell’esempio precedente (eliminazione di tutte le tabelle) con l’utilizzo della tabella SQL Server SYSOBJECTS. La clausola WHERE filtra tutti gli oggetti, ad eccezione di quelli con tipo di oggetto “user table”.

```
SELECT 'DROP TABLE ' + NAME + ';'
FROM SYSOBJECTS
WHERE XTYPE='U'
```

```
DROP TABLE PERSON;
DROP TABLE MOVIE;
DROP TABLE MOVIE_COPY;
DROP TABLE MOVIE_LANGUAGE;
DROP TABLE EMPLOYEE;
DROP TABLE CUSTOMER_ACCOUNT_PERSON;
DROP TABLE CUSTOMER_TRANSACTION;
DROP TABLE MOVIE_RENTAL;
DROP TABLE CUSTOMER_ACCOUNT;
DROP TABLE LANGUAGE;
DROP TABLE MOVIE_GENRE;
DROP TABLE MPAA_RATING;
```

NOTE *Queste dichiarazioni DROP generate non vengono eseguite senza ulteriore lavoro, a causa dei vincoli referenziali. SQL Server non supporta la clausola “CASCADE CONSTRAINTS” con il comando DROP, quindi i vincoli referenziali devono essere eliminati prima che possano essere eliminate le tabelle altrimenti, per essere eseguite, le dichiarazioni DROP devono essere indicate con l’ordine esatto.*

6.4 L’espressione CASE

L’espressione CASE è un’aggiunta recente ma importante allo standard SQL: per la prima volta, parti di dichiarazioni SQL possono essere eseguite in modo condizionale. Per esempio, una colonna nei risultati di query può essere formattata in base ai valori contenuti in un’altra colonna. Tuttavia la propria implementazione SQL potrebbe non supportare ancora tale funzione, perché è molto recente.

L’espressione CASE ammette due forme generali.

Espressione CASE semplice

Di seguito la sintassi generale della forma semplice dell’espressione CASE:

```
CASE espressione_input
  WHEN espressione_confronto THEN espressione_risultato
  [WHEN espressione_confronto THEN espressione_risultato ...]
```

```
[ELSE espressione_risultato]
END
```

NOTA

- ogni condizione *WHEN* viene valutata come *espressione_input* = *espressione_confronto*, se il risultato è un *TRUE* (vero) logico, viene restituita *espressione_risultato* e non viene valutata alcuna altra condizione *WHEN*;
- se nessuna delle condizioni *WHEN* viene valutata come *TRUE* (vero), ed esiste una condizione *ELSE*, viene restituita *espressione_risultato* associata alla condizione *ELSE*;
- se nessuna delle condizioni *WHEN* viene valutata come *TRUE* (vero), e non esiste una condizione *ELSE*, viene restituito un valore null.

Come esempio, è possibile utilizzare l'espressione *CASE* per tradurre *MPAA Rating Code* in un semplice messaggio che può essere visualizzato alla cassa della videoteca, per ricordare ai cassieri di controllare l'età dei clienti per i film con classificazione superiore a *PG-13*. Si noti l'inserimento della parola chiave *AS* dopo la parola chiave *END* per assegnare un nome di colonna alla colonna generata nel set di risultati. Di seguito l'esempio:

```
SELECT MOVIE_ID, MPAA_RATING_CODE AS RATING,
       CASE MPAA_RATING_CODE
         WHEN 'G' THEN 'All ages'
         WHEN 'PG' THEN 'Parental guidance'
         WHEN 'PG-13' THEN 'Ages 13 and up'
         ELSE 'MUST be at least 17'
       END AS RATING_DESC
FROM MOVIE
ORDER BY MOVIE_ID;
```

MOVIE_ID	RATING	RATING_DESC
1	R	MUST be at least 17
2	R	MUST be at least 17
3	PG-13	Ages 13 and up
4	PG-13	Ages 13 and up
5	R	MUST be at least 17
6	PG-13	Ages 13 and up
7	PG-13	Ages 13 and up
8	R	MUST be at least 17
9	PG-13	Ages 13 and up
10	R	MUST be at least 17
11	PG-13	Ages 13 and up
12	PG-13	Ages 13 and up
13	PG-13	Ages 13 and up
14	R	MUST be at least 17
15	R	MUST be at least 17
16	PG-13	Ages 13 and up

```

17 PG-13 Ages 13 and up
18 R      MUST be at least 17
19 PG-13 Ages 13 and up
20 R      MUST be at least 17

```

Espressione CASE cercata

La cosiddetta espressione CASE cercata consente di avere condizioni di confronto più flessibili, perché ciascuna di esse è scritta come condizione completa, compreso l'operatore di confronto. Di seguito la sintassi generale:

```

CASE
  WHEN condizione THEN espressione_risultato
  [WHEN condizione THEN espressione_risultato ...]
  [ELSE espressione_risultato]
END

```

NOTA

- *ciascuna condizione può essere qualsiasi espressione SQL che dia come risultato TRUE o FALSE;*
- *ogni condizione WHEN viene valutata in sequenza, se una di esse viene calcolata come TRUE (vero), viene restituita espressione _risultato associata e non viene valutata alcuna altra condizione WHEN;*
- *se nessuna delle condizioni WHEN viene valutata come TRUE (vero), ed esiste una condizione ELSE, viene restituita espressione _risultato associata alla condizione ELSE;*
- *se nessuna delle condizioni WHEN viene valutata come TRUE (vero), e non esiste una condizione ELSE, viene restituito un valore null.*

Come esempio, di seguito si trova una query che classifica i film in VHS per intervallo di prezzo:

```

SELECT MOVIE_ID, RETAIL_PRICE_VHS,
       CASE
         WHEN RETAIL_PRICE_VHS IS NULL THEN 'Not Available'
         WHEN RETAIL_PRICE_VHS < 10 THEN 'Bargain'
         WHEN RETAIL_PRICE_VHS < 20 THEN 'Budget'
         WHEN RETAIL_PRICE_VHS < 40 THEN 'Average'
         ELSE 'Premium'
       END AS PRICE_CATEGORY
FROM MOVIE
ORDER BY MOVIE_ID;

```

MOVIE_ID	RETAIL_PRICE_VHS	PRICE_CATEGORY
1	58.97	Premium
2	15.95	Budget
3	14.95	Budget

4	11.95 Budget
5	24.99 Average
6	24.99 Average
7	14.95 Budget
8	50.99 Premium
9	12.98 Budget
10	49.99 Premium
11	6.93 Bargain
12	9.95 Bargain
13	6.93 Bargain
14	24.99 Average
15	9.99 Bargain
16	11.69 Budget
17	14.94 Budget
18	24.99 Average
19	12.98 Budget
20	17.99 Budget

6.5 Quiz

Scegliere le risposte corrette per ciascuna domanda a risposta multipla. Si noti che per ogni domanda è possibile avere più risposte esatte.

1. Le funzioni SQL:

- Restituiscono un set di valori.
- Restituiscono un singolo valore.
- Possono essere utilizzate nella clausola WHERE di una dichiarazione SQL.
- Possono essere utilizzate come alias di nome di tabella in una dichiarazione SQL.
- Possono essere utilizzate nell'elenco di colonne di una dichiarazione SQL.

2. La funzione REPLACE:

- Sostituisce un nome di tabella con un nome di vista.
- Sostituisce un nome di colonna con un alias di colonna.
- Sostituisce una stringa di caratteri in una colonna con un'altra stringa di caratteri.
- Sostituisce tutti i valori in una colonna con un nuovo set di valori.
- Sostituisce tutte le righe in una vista con righe contenenti dati da un'altra tabella.

3. La funzione di valore null:

- Viene chiamata NVL nei database Oracle.
- Viene chiamata ISNULL nei database IBM DB2.
- Viene chiamata ISNULL nei database Microsoft SQL Server.

- d. Viene chiamata ISNULL nei database MySQL.
- e. Viene chiamata IFNULL nello standard SQL.

4. La funzione LTRIM:

- a. Elimina gli spazi finali dalle stringhe di caratteri.
- b. Elimina gli spazi iniziali dalle stringhe di caratteri.
- c. Può essere annidata con altre funzioni.
- d. Sostituisce valori null con altri valori nelle stringhe di caratteri.
- e. Elimina gli spazi iniziali e finali dalle stringhe di caratteri.

5. La funzione CHAR:

- a. In alcune implementazioni SQL viene chiamata CHR.
- b. In alcune implementazioni SQL è identica alla funzione ASCII.
- c. Restituisce il valore del set di caratteri ASCII per un carattere.
- d. Restituisce il carattere per un valore di set di caratteri ASCII.
- e. Converte un valore numerico in una stringa di caratteri.

6. La funzione SIGN:

- a. Restituisce -1 se il parametro fornito ha un valore negativo.
- b. Restituisce 0 se il parametro fornito ha un valore pari a zero.
- c. Restituisce $+1$ se il parametro fornito ha un valore maggiore o uguale a zero.
- d. Restituisce 0 se il parametro fornito è null.
- e. Restituisce un valore null se il parametro fornito non è un numero.

7. La funzione CEILING:

- a. Arrotonda un numero per difetto al numero intero successivo.
- b. Arrotonda un numero per eccesso al numero intero successivo.
- c. Restituisce sempre un intero.
- d. Restituisce un intero o un valore null.
- e. In alcune implementazioni SQL viene chiamata CEIL.

8. La funzione FLOOR:

- a. Arrotonda un numero per difetto al numero intero successivo.
- b. Arrotonda un numero per eccesso al numero intero successivo.
- c. Restituisce sempre un intero.
- d. Restituisce un intero o un valore null.
- e. In alcune implementazioni SQL viene chiamata FLR.

9. Le funzioni di data e ora:

- a. Sono molto simili tra le implementazioni di produttori diversi.
- b. Variano considerevolmente tra le implementazioni di produttori diversi.
- c. Contengono funzioni che formattano gli elementi di data e ora per la visualizzazione.
- d. Contengono funzioni che convertono stringhe di caratteri in date e ore.

- e. Contengono funzioni che convertono elementi di data e ora in stringhe di caratteri.

10. Le espressioni CASE:

- a. Consentono l'esecuzione condizionale di clausole in una dichiarazione SQL.
- b. Hanno due forme, chiamate statica e dinamica.
- c. Hanno due forme, chiamate cercata e non cercata.
- d. Hanno due forme, chiamate semplice e cercata.
- e. Hanno due forme, chiamate standard e cercata.

Scrivere dichiarazioni SQL per risolvere ciascuno dei problemi di seguito.

- 11. Elencare i valori MPAA _ RATING _ CODE delle tabella MPAA _ RATING, con i trattini tradotti in spazi.
- 12. Utilizzando la funzione CHAR (chiamata CHR in Oracle), elencare MOVIE _ ID e MOVIE _ TITLE per tutti i film contenenti una virgoletta semplice (carattere ASCII 39) nel nome.
- 13. Trovare il valore del set di caratteri ASCII per un punto esclamativo (!)
- 14. Trovare il prezzo medio di un film in DVD (colonna DVD _ RETAIL _ PRICE nella tabella MOVIE), con la media calcolata arrotondata per eccesso al dollaro.
- 15. Trovare il prezzo medio di un film in VHS (colonna VHS _ RETAIL _ PRICE nella tabella MOVIE), con la media calcolata arrotondata per difetto al dollaro.
- 16. Per un database Oracle, generare i comandi SQL per eliminare tutti i vincoli referenziali posseduti dall'utente corrente. La vista di catalogo di Oracle si chiama USER _ CONSTRAINTS, i vincoli referenziali hanno un CONSTRAINT _ TYPE pari a "R". Il nome di tabella su cui è basato il vincolo si trova nella colonna TABLE _ NAME della vista di catalogo. Ricordare che è necessario utilizzare il comando ALTER TABLE per eliminare i vincoli di tabella.
- 17. Per un database Microsoft SQL Server, elencare tutti i vincoli di chiave esterna. Utilizzare la tabella di sistema SYSOBJECTS, in cui NAME è il nome del vincolo e XTYPE ha valore "F" per vincoli di chiave esterna.
- 18. Scrivere una dichiarazione SQL che elenchi tutti i Customer Account (CUSTOMER _ ACCOUNT _ ID) con una stringa di caratteri basata sul valore di CHILD _ RENTAL _ ALLOWED _ INDIC dove la stringa indica "Child Rental OK"

se il valore è “Y” e “NO CHILD RENTAL” se il valore è “N”. Suggerimento: una semplice espressione CASE funziona molto bene.

19. Scrivere una dichiarazione SQL che elenchi tutti i film (MOVIE_ID) insieme al decennio in base al valore di YEAR_PRODUCED (80s, 90s, 00s, Unknown). Risulta utile un'espressione CASE cercata.
20. Scrivere una dichiarazione SQL che elenchi tutti i noleggi di film (tabella MOVIE_RENTAL) con LATE_OR_LOSS_FEE catalogato come di seguito: Nessuno (valore di dati null o 0), Minore (valore di dati < 10), Maggiore (valore di dati >=10).

