Chapter 6
Temporal Data Warehousing

A *temporal data warehousing* is a warehousing technique where the temporal (or historical) aspect of records is incorporated into the warehouse. For example, if a book is a dimension (in a bookshop setting), the book price changes over time. If we would like to keep track of the book prices in the data warehouse, then a temporal data warehouse technique must be used. Without this temporal or historical aspect of the book price, the analysis made from the data warehouse is only based on the current book price.

This chapter focuses on temporal data warehousing covering star schemas, which maintain a temporal aspect of the data. Temporal data warehousing is also known as *slowly changing dimensions* (*SCD*). This chapter will also describe various techniques that can be used to deal with SCD.

6.1 A Bookshop Case Study

To learn the concept of temporal data warehousing, let's use the Bookshop example as a case study. The management of a bookshop with a number of branches would like to build a data warehouse to analyse their book sales. They have already stored all book sales transactions in an operational database. The management would particularly like to analyse their book sales performance from various perspectives, such as from a monthly basis, from a book basis and from a branch basis. The first task to do this is to define a star schema.

A simple star schema for this Bookshop case study is shown in Fig. 6.1. It has three dimensions: Time, Branch and Book Dimensions, and the fact has one fact measure, called Number of Books Sold. With this star schema, the management is able to retrieve the number of books sold based on the month, based on the branch and based on the book.

D. Taniar, W. Rahayu, *Data Warehousing and Analytics*, Data-Centric Systems and Applications, https://doi.org/10.1007/978-3-030-81979-8_6

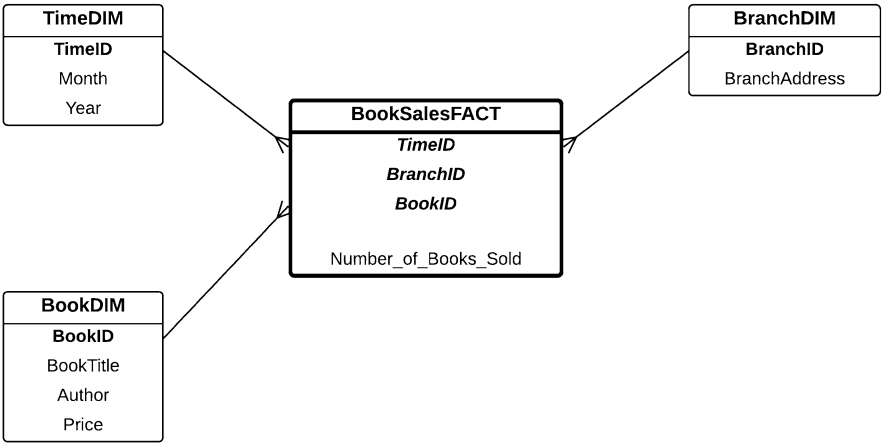

Fig. 6.1 The Bookshop star schema

The Fact Table would probably look like the following (see Table 6.1). Note that the sales figures are for illustrative purposes only, and the following entries in the tables are not necessarily comprehensive.

The Fact Table can be joined with the dimensions to produce a more comprehensive report. In the example below, the Fact Table is joined with the Book Dimension table, and consequently the attributes in the Book Dimension table will appear in the report. Assume the Book Dimension table has the following data (see Table 6.2).

The new report is shown in Table 6.3. Note the three additional attributes: Book Title, Author and Price from the Book Dimension table. Pay attention to the Price column in Report 1. In this case, the Price column contains the *Current Price* of each book, and it is likely to be the *Original Price*. However, from time to time, a book is sold at a discounted price. The above sales report does not reflect this; therefore, the analysis from this report can be misleading.

To incorporate the temporal values of the Price, we need to add a *Bridge Table* to the Book Dimension to store the history of book prices (which is the Book Price Dimension). Note that the Book Price Dimension (the Bridge Table) is implemented as a *Weak Entity*, with a composite primary key of BookID, Start Date and End Date (refer to Fig. 6.2).

The Book Dimension and Book Price Dimension tables are as follows (refer to Tables 6.4 and 6.5). In the Book Price Dimension table, a *temporal* attribute Price is managed, where for each price, the period (Start and End Dates) and the remarks are recorded. Note that we use Dec9999 to indicate that the end date is still open until now.

Table 6.1 BookSalesFact Table

TimeID	BranchID	BookID	Number of books sold
Mar2008	City	C1	5
Mar2008	City	H6	15
Mar2008	City	DV	23
Mar2008	City	...	...
Mar2008	Chadstone	C1	15
Mar2008	Chadstone	H6	3
Mar2008	Chadstone	DV	2
Mar2008	Chadstone	...	...
Mar2008	Camberwell	C1	1
Mar2008	Camberwell	H6	1
Mar2008	Camberwell	DV	2
Mar2008	Camberwell	...	...
Mar2008	...	...	...
...	...	...	...
...	...	...	...
Dec2007	City	C1	15
Dec2007	City	H6	6
Dec2007	City	DV	6
Dec2007	City	...	...
Dec2007	Chadstone	C1	10
Dec2007	Chadstone	H6	8
Dec2007	Chadstone	DV	1
Dec2007	Chadstone	...	...
Dec2007	Camberwell	C1	18
Dec2007	Camberwell	H6	3
Dec2007	Camberwell	DV	2
Dec2007	Camberwell	...	...
Dec2007	...	...	...
...	...	...	...

Table 6.2 Book Dimension table

BookID	Book title	Author	Price
C1	CSIRO Diet	CSIRO Team	$45.95
H6	Harry Potter 6	Rowling	$30.95
DV	Da Vinci Code	Dan Brown	$27.95
...	...	...	...

Table 6.3 Report 1 (Book Sales Fact with Book Dimension)

TimeID	BranchID	BookID	Book title	Author	Price	Number of books sold
Mar2008	City	C1	CSIRO Diet	CSIRO Team	$45.95	5
Mar2008	City	H6	Harry Potter 6	Rowling	$30.95	15
Mar2008	City	DV	Da Vinci Code	Dan Brown	$27.95	23
Mar2008	City	...	...	...	...	...
Mar2008	Chadstone	C1	CSIRO Diet	CSIRO Team	$45.95	15
Mar2008	Chadstone	H6	Harry Potter 6	Rowling	$30.95	3
Mar2008	Chadstone	DV	Da Vinci Code	Dan Brown	$27.95	2
Mar2008	Chadstone	...	...	...	...	...
Mar2008	Camberwell	C1	CSIRO Diet	CSIRO Team	$45.95	1
Mar2008	Camberwell	H6	Harry Potter 6	Rowling	$30.95	1
Mar2008	Camberwell	DV	Da Vinci Code	Dan Brown	$27.95	2
Mar2008	Camberwell	...	...	...	...	...
Mar2008	...	...	...	...	...	...
...	...	...	...	...	...	...
...	..	...	...	...	...	...
Dec2007	City	C1	CSIRO Diet	CSIRO Team	$45.95	15
Dec2007	City	H6	Harry Potter 6	Rowling	$30.95	6
Dec2007	City	DV	Da Vinci Code	Dan Brown	$27.95	6
Dec2007	City	..	...	...	...	...
Dec2007	Chadstone	C1	CSIRO Diet	CSIRO Team	$45.95	10
Dec2007	Chadstone	H6	Harry Potter 6	Rowling	$30.95	8
Dec2007	Chadstone	DV	Da Vinci Code	Dan Brown	$27.95	1
Dec2007	Chadstone	..	...	...	...	...
Dec2007	Camberwell	C1	CSIRO Diet	CSIRO Team	$45.95	18
Dec2007	Camberwell	H6	Harry Potter 6	Rowling	$30.95	3
Dec2007	Camberwell	DV	Da Vinci Code	Dan Brown	$27.95	2
Dec2007	Camberwell	..	...	...	...	...
Dec2007	...	...	...	...	...	...
...	..	...	...	...	...	...

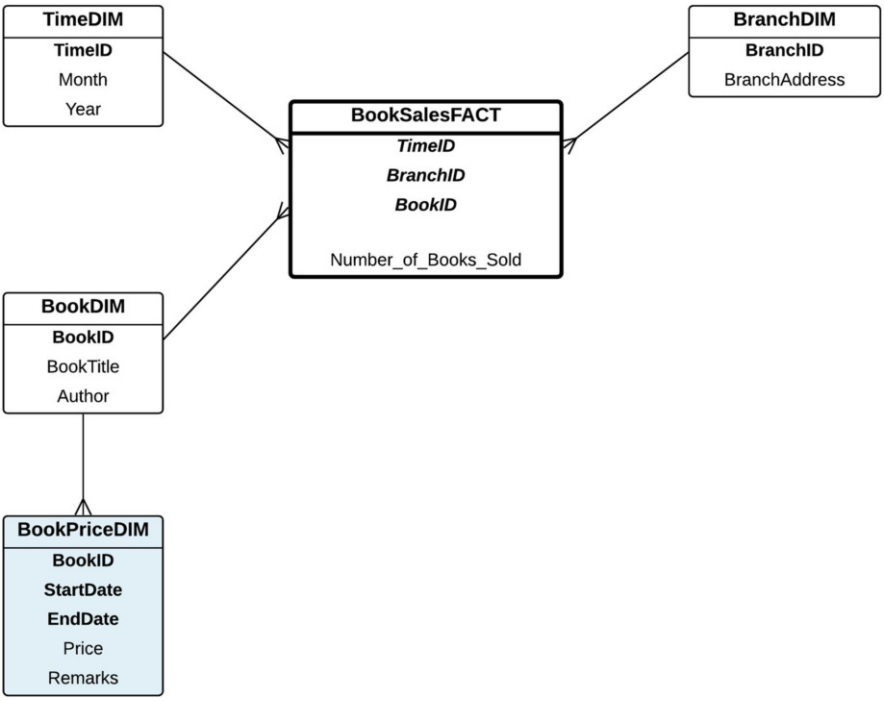

Fig. 6.2 A temporal data warehousing star schema

Table 6.4 Book Dimension table

BookID	Book title	Author
C1	CSIRO Diet	CSIRO Team
H6	Harry Potter 6	Rowling
DV	Da Vinci Code	Dan Brown
...	...	...

Table 6.5 Book Price Dimension table

BookID	Start date	End date	Price	Remarks
C1	Jan2007	July2007	$45.95	Full Price
C1	Aug2007	Oct2007	$36.75	20% Discount
C1	Nov2007	Jan2008	$23.00	Half Price
C1	Feb2008	**Dec9999**	$45.95	Full Price
H6	Jan2007	Mar2007	$21.95	Launching
H6	Apr2007	Jan2008	$30.95	Full Price
H6	Feb2008	**Dec9999**	$10.00	End of Product Sale
DV	Jan2007	**Dec9999**	$27.95	Full Price
...	...	...	...	...

Table 6.6 Report 2 with the correct Book Price

TimeID	BranchID	BookID	Book title	Author	Price	Number of books sold
Mar2008	City	C1	CSIRO Diet	CSIRO Team	$45.95	5
Mar2008	City	H6	Harry Potter 6	Rowling	**$10.00**	15
Mar2008	City	DV	Da Vinci Code	Dan Brown	$27.95	23
Mar2008	City	...	...	...	...	...
Mar2008	Chadstone	C1	CSIRO Diet	CSIRO Team	$45.95	15
Mar2008	Chadstone	H6	Harry Potter 6	Rowling	**$10.00**	3
Mar2008	Chadstone	DV	Da Vinci Code	Dan Brown	$27.95	2
Mar2008	Chadstone	...	...	...	...	...
Mar2008	Camberwell	C1	CSIRO Diet	CSIRO Team	$45.95	1
Mar2008	Camberwell	H6	Harry Potter 6	Rowling	**$10.00**	1
Mar2008	Camberwell	DV	Da Vinci Code	Dan Brown	$27.95	2
Mar2008	Camberwell	...	...	...	...	...
Mar2008	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...
Dec2007	City	C1	CSIRO Diet	CSIRO Team	**$23.00**	15
Dec2007	City	H6	Harry Potter 6	Rowling	$30.95	6
Dec2007	City	DV	Da Vinci Code	Dan Brown	$27.95	6
Dec2007	City	..	...	...	...	...
Dec2007	Chadstone	C1	CSIRO Diet	CSIRO Team	**$23.00**	10
Dec2007	Chadstone	H6	Harry Potter 6	Rowling	$30.95	8
Dec2007	Chadstone	DV	Da Vinci Code	Dan Brown	$27.95	1
Dec2007	Chadstone	..	...	...	...	...
Dec2007	Camberwell	C1	CSIRO Diet	CSIRO Team	**$23.00**	18
Dec2007	Camberwell	H6	Harry Potter 6	Rowling	$30.95	3
Dec2007	Camberwell	DV	Da Vinci Code	Dan Brown	$27.95	2
Dec2007	Camberwell	..	...	...	...	...
Dec2007	...	...	...	...	...	...
...	...	...	...	...	...	...

Since each book has a temporal feature (i.e. temporal attribute), the previous Report 1 can be revised to incorporate the correct sale price of the book. Hence, the new report (Report 2) is shown in Table 6.6. Note that now we can understand why more copies of CSIRO Diet were sold in December 2007 as it was half price. Also note that the sale of Harry Potter is coming to an end (e.g. not much demand). The Da Vinci Code has always been sold for the full price.

6.2 Implementation of Temporal Data Warehousing

This section will discuss the SQL implementation details of the temporal data warehousing. Consider an E/R diagram of the operational database as shown in Fig. 6.3.

The star schema was previously shown in Fig. 6.2. The SQL commands to create the dimension tables are as follows:

```
create table BranchDim as
select * from Branch;

create table BookDim as
select * from Book;

create table TimeDim as
select distinct
  to_char(TransactionDate, 'MonYYYY') as TimeID,
  to_char(TransactionDate, 'Mon') as Month,
  to_char(TransactionDate, 'YYYY') as Year
from BookTransaction;

create table BookPriceDim as
select  * from BookPriceHistory;
```

The SQL to create the Fact Table is as follows:

```
create table BookSalesFact1 as
select
  to_char(T.TransactionDate, 'MonYYYY') as TimeID,
  BK.BookID,
  BR.BranchID,
  sum(T.Quantity) as Number_of_Books_Sold
from BookTransaction T, Book BK, Branch BR
where T.BranchID = BR.BranchID
and T.BookID = BK.BookID
group by
  to_char(T.TransactionDate, 'MonYYYY'),
  BK.BookID,
  BR.BranchID;
```

Calculating Number of Books Sold is quite straightforward, that is, the sum of the attribute quantity from the Book Transaction table in the operational database.

Suppose we would like to have one additional fact measure, called *Total Sales*. Therefore, the new star schema is shown in Fig. 6.4.

The SQL command to create the second Fact Table (we call it *BookSalesFact2*) is as follows:

```
create table BookSalesFact2 as
select * from BookSalesFact1;

alter table BookSalesFact2
add (Total_Sales number);
```

```
declare
  cursor PriceCursor is
    select *
    from BookPriceDim;
begin
  for Item in PriceCursor loop
    -- update value for Total_Sales in BookSalesFact2
    update BookSalesFact2
    set Total_Sales = Number_Of_Books_Sold * Item.Price
    where BookID = Item.BookID
    and to_date(TimeID, 'MonYYYY') >=
      to_date(Item.StartDate, 'MonYYYY')
    and to_date(TimeID, 'MonYYYY') <=
      to_date(Item.EndDate, 'MonYYYY');
  end loop;
end;
/
```

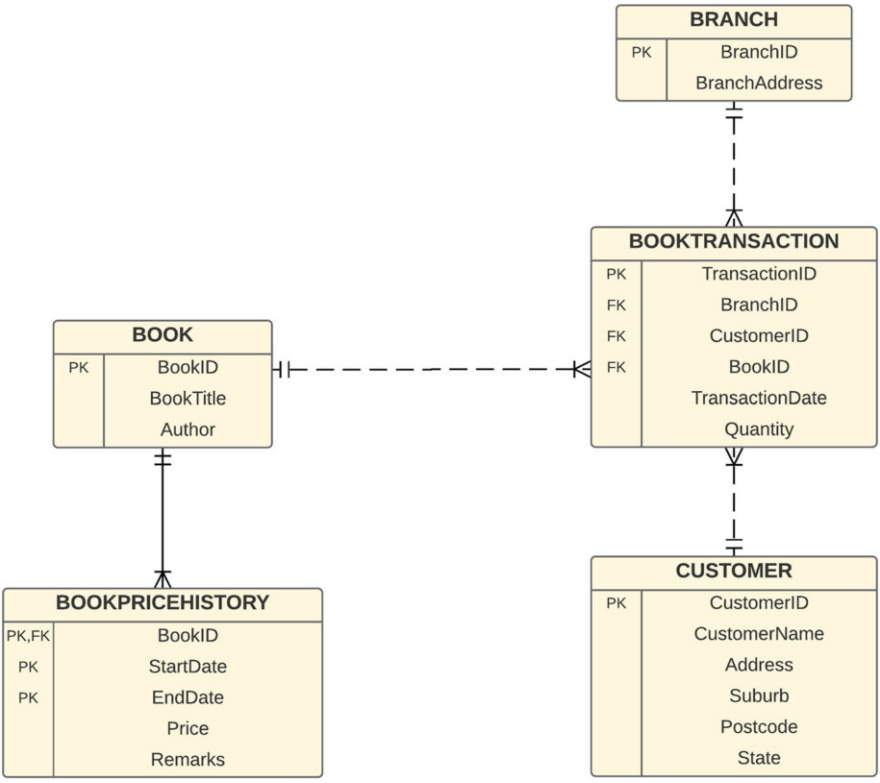

Fig. 6.3 An E/R diagram of the operational database

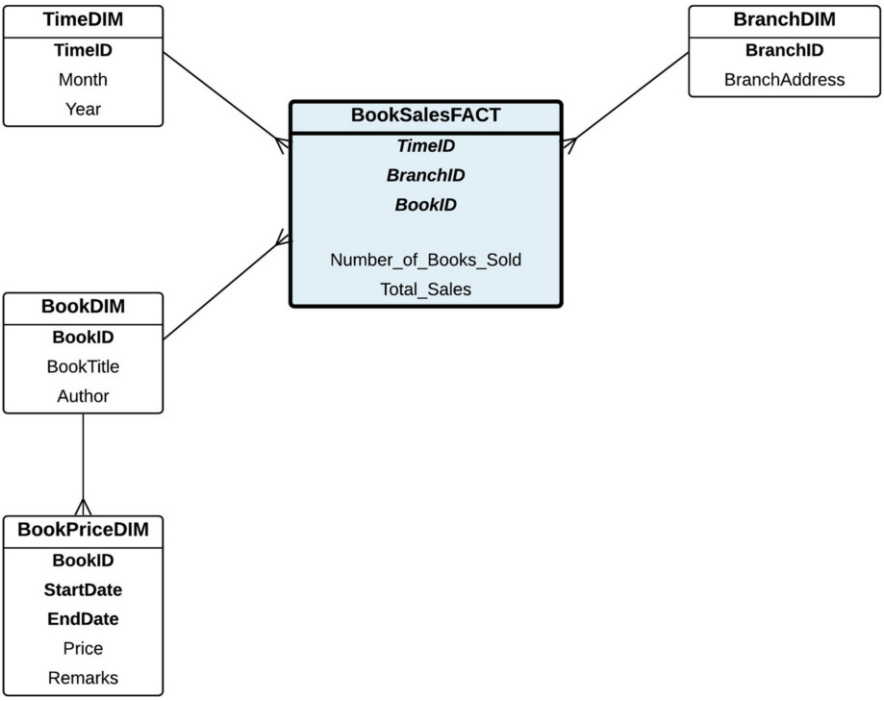

Fig. 6.4 Star Schema with a new fact attribute: Total Sales

For each record in the BookPriceDim table, update the BookSalesFact2 table and calculate the total sales by multiplying Number of Books Sold in the BookSalesFact2 and the "correct" price of the BookPriceDim (where the TimeID in the fact should be between the start date and end date of the book price).

The aforementioned method assumes that BookSalesFact1 has been created. BookSalesFact2 is created by copying from BookSalesFact1, and then update it by adding new column called Total Sales. However, if we do not assume that BookSalesFact1 has already been created, we can create BookSalesFact2 from scratch, using the following create table statement:

```
create table BookSalesFact2 as
select
   to_char(T.TransactionDate, 'MonYYYY') as TimeID,
   BK.BookID,
   BR.BranchID,
   sum(T.Quantity) as Number_Of_Books_Sold,
   sum(T.Quantity * BP.Price) as Total_Sales
from
  BookTransaction T,
  Book BK,
  Branch BR,
  BookPriceHistory BP
```

```
where T.BranchID = BR.BranchID
and T.BookID = BK.BookID
and BK.BookID = BP.BookID
and T.TransactionDate >= to_date(BP.StartDate, 'MonYYYY')
and T.TransactionDate <= to_date(BP.EndDate, 'MonYYYY')
group by
   to_char(T.TransactionDate, 'MonYYYY'),
   BK.BookID,
   BR.BranchID;
```

6.3 Temporal Attributes and Temporal Dimensions

6.3.1 Temporal Attributes

In the above case study, Book Price in the Book Price Dimension table is called a **temporal attribute**. A temporal attribute is an attribute in which the value of that attribute has a lifetime. In this example, each book price has a lifetime, and the lifetime is determined by the Start Date and End Date attributes in the Book Price Dimension table. For example, the book price of $45.95 for Book C1 is only valid between January 2007 and July 2007.

The Book Price Dimension table is a Relational DBMS (RDBMS) implementation of temporal data warehousing, which is implemented as a Bridge Table. The reason why the Book Price Dimension is separated from the original Book Dimension is because one book may have many prices in different time periods. Because the relational model does not permit a nested table, the information about book prices has to be separated into another table. The Book Price Dimension table was shown in Table 6.5.

As stated in the previous section, when we produce a report that joins the Fact Table with the dimension tables (e.g. joining the BookSalesFact Table with the Book Dimension and Book Price Dimension tables), we can produce Report 2 (refer to Table 6.6) which contains all of the attributes from the BookSalesFact Table with the addition of Book Title and Author attributes from the Book Dimension table and the Price attribute from the Book Price Dimension table. This report is produced by the following SQL statement:

```
select
  F.TimeID,
  F.BranchID,
  F.BookID,
  B.BookTitle,
  B.Author,
  P.Price,
  F.Number_of_Books_Sold
from BookSalesFact F, BookDim B, BookPriceDim P
where F.BookID = B.BookID
and B.BookID = P.BookID
```

```
and to_date(F.TimeID, 'MonYYYY') >=
  to_date(P.StartDate, 'MonYYYY')
and to_date(F.TimeID, 'MonYYYY') <=
  to_date(P.EndDate, 'MonYYYY');
```

This SQL command joins the three mentioned tables (i.e. Book Sales Fact, Book Dimension and Book Price Dimension) and checks if TimeID (which is Month and Year) in the BookSalesFact Table falls in the period of the book price as stated by the Start Date and End Date in the Book Price Dimension table. As a result, the correct price will be displayed in the report which matches with the month (i.e. TimeID) of the fact record.

However, a problem will arise if the granularity of time that is used by the book price and the fact is different. For example, if the price of BookID C1 is $23.00 but not from November 2007 to January 2008, but from November 2007 to 15 January 2008 (consequently the price increased to $45.95 as stated from 16 January 2008 instead of February 2008; refer to Table 6.7), then the `where` clause condition in the above SQL becomes:

```
and to_date(F.TimeID, 'MonYYYY') >=
  to_date(P.StartDate, 'MonYYYY')
and to_date(F.TimeID, 'MonYYYY') <=
  to_date(P.EndDate, 'MonYYYY');
```

This `where` clause condition will however produce an incorrect report because TimeID 200701 for January 2007 will match with two records in the Book Price Dimension table, one with a price of $23.00 and the other with a price of $45.95. Consequently, in the report, there will be two records for book C1 in January 2008, in each branch, one with the price of $23.00 and the other with the price of $45.95, simply because there are two book prices for the month of January 2008 (see Report 3 in Table 6.8).

This problem can only be solved if the granularity of TimeID in the fact is the same as the Start Date and End Date in the Book Price Dimension table. If TimeID is on the month level, then the period of the book price must be exclusively based on month. On the other hand, if the period of the book price is at a date granularity

Table 6.7 BookPriceDim table

BookID	Start date	End date	Price	Remarks
C1	Jan2007	Jul2007	$45.95	Full Price
C1	Aug2007	Oct2007	$36.75	20% Discount
C1	Nov2007	**15Jan2008**	$23.00	Half Price
C1	**16Jan2008**	Dec9999	$45.95	Full Price
H6	Jan2007	Mar2007	$21.95	Launching
H6	Apr2007	Jan2008	$30.95	Full Price
H6	Feb2008	Dec9999	$10.00	End of Product Sale
DV	Jan2007	Dec9999	$27.95	Full Price
...	...	...	...	...

Table 6.8 Report 3: an incorrect report

TimeID	BranchID	BookID	Book title	Author	Price	Number of books sold
Jan2008	**City**	**C1**	**CSIRO Diet**	**CSIRO Team**	**$23.00**	**25**
Jan2008	**City**	**C1**	**CSIRO Diet**	**CSIRO Team**	**$45.95**	**25**
Jan2008	City	H6	Harry Potter 6	Rowling	$30.95	10
Jan2008	City	DV	Da Vinci Code	Dan Brown	$27.95	7
Jan2008	City	...	...	...	...	...
Jan2008	**Chadstone**	**C1**	**CSIRO Diet**	**CSIRO Team**	**$23.00**	**30**
Jan2008	**Chadstone**	**C1**	**CSIRO Diet**	**CSIRO Team**	**$45.05**	**30**
Jan2008	Chadstone	H6	Harry Potter 6	Rowling	$30.95	15
Jan2008	Chadstone	DV	Da Vinci Code	Dan Brown	$27.95	5
Jan2008	Chadstone	...	...	...	...	...
Jan2008	**Camberwell**	**C1**	**CSIRO Diet**	**CSIRO Team**	**$23.00**	**20**
Jan2008	**Camberwell**	**C1**	**CSIRO Diet**	**CSIRO Team**	**$45.05**	**20**
Jan2008	Camberwell	H6	Harry Potter 6	Rowling	$30.95	5
Jan2008	Camberwell	DV	Da Vinci Code	Dan Brown	$27.95	5
Jan2008	Camberwell	...	...	...	...	...
Jan2008	...	...	...	...	...	...
...	...	...	...	...	...	...

level, then the TimeID in the fact must also be at a date granularity level. If the level of granularity is different, the aforementioned problem will occur. Another way to "solve" this problem is by not allowing the report to join with the Book Price Dimension table. In other words, the Book Price Dimension table is purely treated as additional information in the data warehouse in case the management wants to drill down for information on certain books.

The above problem of mismatched TimeID granularity between the Book Price Dimension table and the Fact can be solved if we display "two" prices on the same record in the report (see Table 6.9). For example, in the first record in Table 6.9, it has two book prices because for that month (e.g. January 2008), there was a change in the book price from $23.00 to $45.95.

Report 4 can be created by the following SQL command:

```
select
   F.TimeID,
   F.BranchID,
   F.BookID,
   B.BookTitle,
   B.Author,
   listagg(P.Price, ';') within group (order by P.Price)
     as Price,
   F.Number_of_Books_Sold
from BookSalesFact F, BookDim B, BookPriceDim P
where F.BookID = B.BookID
and B.BookID = P.BookID
and to_date(F.TimeID, 'MonYYYY') >=
  to_date(P.StartDate, 'MonYYYY')
```

```
and to_date(F.TimeID, 'MonYYYY') <=
  to_date(P.EndDate, 'MonYYYY')
group by
   F.TimeID,
   F.BranchID,
   F.BookID,
   B.BookTitle,
   B.Author,
   F.Number_of_Books_Sold;
```

The above problem in Report 3 occurs because we joined the Fact Table with the Book Price Dimension table; there is no problem with the records in the Fact Table itself. One might ask what happened with the Total Sales attribute in the Fact Table. The calculation for the Total Sales is still correct, as shown in the following SQL statement:

```
create table BookSalesFact2 as
select
  to_char(T.TransactionDate, 'MonYYYY') as TimeID,
  BK.BookID,
  BR.BranchID,
  sum(T.Quantity) as Number_of_Books_Sold,
  sum(T.Quantity * BP.Price) as Total_Sales
from
  BookTransaction T,
  Book BK,
  Branch BR,
  BookPriceHistory BP
where T.BranchID = BR.BranchID
and T.BookID = BK.BookID
and BK.BookID = BP.BookID
and T.TransactionDate >= BP.StartDate
and T.TransactionDate <= BP.EndDate
group by
  to_char(T.TransactionDate, 'MonYYYY'),
  BK.BookID,
  BR.BranchID;
```

This SQL joins the four tables in the operational database, namely, Book Transaction, Book, Branch and Book Price History. While joining these tables, it obtains the correct book price by comparing Transaction Date with Start Date and End Date of the book price. In this case, we assume that Start Date and End Date, as well as Transaction Date, have a date granularity. Once these tables are joined, the grouping is then based on month. The individual book total sales are correct, and the grouping is then based on month. Hence, the problem of an incorrect report is not due to the incorrect fact but because of the join between the fact and the dimension that stores the temporal attribute.

Table 6.9 Report 4: multiple book prices on one month

TimeID	BranchID	BookID	Book title	Author	Price	Number of books Sold
Jan2008	City	C1	CSIRO Diet	CSIRO Team	**$23.00;$45.95**	25
Jan2008	City	H6	Harry Potter 6	Rowling	$30.95	10
Jan2008	City	DV	Da Vinci Code	Dan Brown	$27.95	7
Jan2008	City	...	...	...	...	...
Jan2008	Chadstone	C1	CSIRO Diet	CSIRO Team	**$23.00;$45.95**	30
Jan2008	Chadstone	H6	Harry Potter 6	Rowling	$30.95	15
Jan2008	Chadstone	DV	Da Vinci Code	Dan Brown	$27.95	5
Jan2008	Chadstone	...	...	...	...	...
Jan2008	Camberwell	C1	CSIRO Diet	CSIRO Team	**$23.00;$45.95**	20
Jan2008	Camberwell	H6	Harry Potter 6	Rowling	$30.95	5
Jan2008	Camberwell	DV	Da Vinci Code	Dan Brown	$27.95	5
Jan2008	Camberwell	...	...	...	...	...
Jan2008	...	...	...	...	...	...
...	...	...	...	...	...	...

6.3.2 Temporal Dimensions

If a temporal attribute is an attribute where the value has a lifetime, a *temporal dimension* is a dimension where the record of the dimension has a lifetime. For example, if BookID C1 appeared in 2007, disappeared in 2008 and then reappeared again in 2009, then the book dimension needs a temporal dimension. In another situation, if a branch opens and closes several times, then the branch dimension needs a temporal dimension. However, these two examples are not realistic because it is very rare that books appear and disappear and branches open and close. So, in order to highlight the temporal dimensions, we are going to use a more realistic case study of a Calendar shop.

A *Calendar shop* has a number of branches. It sells different types of calendars and diaries. Some branches are seasonal; they open a stall in major shopping malls, and these stalls are only open for certain months (for instance, some stalls open from October to February, while others may open from November to January). This company also has a small number of permanent shops which are open all year round.

This Calendar shop company wants to analyse their sales, similar to the Bookshop case study. The star schema for the Calendar shop has three dimensions: Time, Branch and Merchandise Dimensions (instead of Book Dimension in the Bookshop case study). For the Merchandise Dimension, if we want to keep track of the changes in prices, like we did with the Book Price Dimension, we could have the merchandise price as a temporal attribute in the Merchandise Price Dimension.

For the Branch Dimension, because some branches have a certain lifetime, Branch Dimension needs a temporal dimension called the Branch History Dimension. In the Branch History Dimension, the entire branch record has a temporal

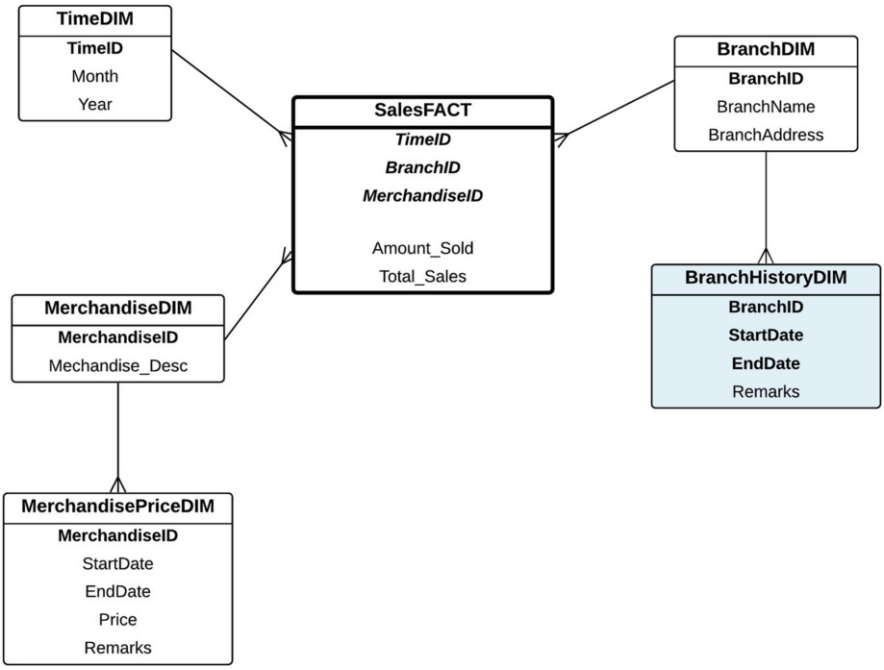

Fig. 6.5 Star schema for the Calendar shop case study

Table 6.10 Branch Dimension table

BranchID	Branch name	Branch address
MEL	Melbourne Central	88 Lonsdale St
CH	Chadstone Mall	109 Dandenong Rd
DOW	Doncaster Westfield	75 Doncaster Rd
...	...	...

element, indicated by the Start and End Dates. The star schema for the Calendar shop is shown in Fig. 6.5.

The sample records of Branch Dimension and Branch History Dimension are shown in Tables 6.10 and 6.11.

In the Branch History Dimension table, Start Date and End Date refer to the BranchID or the entire record, rather than to a specific attribute. This means, for example, the Chadstone Branch (BranchID CH) opened several times in the last few years, as indicated by the Start Date and End Date, whereas the Melbourne Central shop (BranchID MEL) is open all year round. The Remarks attribute is only used for remarks for each occurrence of the branch entry in the Branch History Dimension table. The Contact Number attribute shows the contact number of a particular branch at a particular lifetime. In this example, it shows that the Chadstone branch (CH) had a different contact phone number (a mobile number) every year, whereas the

Table 6.11 BranchHistoryDim table

BookID	Start date	End date	Remarks	Contact number
MEL	Jan0000	Dec9999	Main shop	(03) 9859 8070
CH	Oct2007	Mar2008		0411 848 821
CH	Oct2008	Feb2009	Under re-construction	0413 356 665
CH	Oct2009	Feb2010		0412 313 313
DOW	Nov2007	Feb2008		0427 123 456
DOW	Nov2008	Feb2009		0427 123 456
DOW	Oct2009	Feb2010		0427 123 456
...	...	...	...	...

Doncaster branch (DOW) has had the same contact mobile number every year since it opened.

When Sales fact is joined with Branch Dimension and Branch History Dimension, the report may look like that shown in Table 6.12. For simplicity, MerchandiseID is left empty. The report starts in October 2007 with only two shops open (Melbourne Central and Chadstone Mall), then in November 2007, an additional shop opened (Doncaster Westfield). Then in April 2008, only Melbourne Central was opened, as Chadstone Mall re-opened in October 2008.

If we compare temporal attribute and temporal dimension in this Calendar shop case study, both look very similar. The only difference is that in the case of the temporal attribute, there is a temporal attribute in the history dimension table (e.g. book price in the Book Price Dimension table), whereas in the temporal dimension case, there is no such attribute. The attributes in the temporal dimension are only the key identifier of the parent dimension and possibly any remarks attributes. Or in other words, we can say that a temporal dimension is where all attributes in the dimension are all, collectively, temporal attributes.

6.3.3 Another Temporal Dimension

Temporal dimensions or dimensions with temporal attributes are normally implemented as a Bridge Table, as discussed in the previous sections. In this section, we are going to see an example where a temporal dimension is not implemented as a Bridge Table, rather as a normal dimension.

Suppose there is a simple star schema that maintains a list of courses/degrees (e.g. Bachelor of Computer Science, Master of Information Technology, etc.). A course or a degree structure changes from time to time, simply due to the nature of the discipline, especially those which are moving at a reasonably high pace, like information technology. Each course has a course code. When the course changes significantly, the university or the faculty usually introduces a new course (or degree) and phases out the old course (or degree) rather than "updating" the

Table 6.12 Report: SalesFact joined with Branch Dimension and Branch History Dimension

Time ID	Branch ID	Branch name	Branch address	Start date	End date	Remarks	Contact number	Merchandise	Amount sold	Total sales
Oct2007	MEL	Melbourne Central	88 Lonsdale Street	Jan0000	Dec9999	Main shop	...	...	...	...
Oct2007	CH	Chadstone Mall	109 Dandenong Road	Oct2007	Mar2008		...	...	...	...
Nov2007	MEL	Melbourne Central	88 Lonsdale Street	Jan0000	Dec9999	Main shop	...	...	...	...
Nov2007	CH	Chadstone Mall	109 Dandenong Road	Oct2007	Mar2008		... ...	....	... ...	... ...
Nov2007	DOW	Doncaster Westfield	75 Doncaster Road	Nov2007	Feb2008		... ...	. ...	... ...	... ...
Dec2007	MEL	Melbourne Central	88 Lonsdale Street	Jan0000	Dec9999	Main shop	... ...	... ...	... ...	... ...
Dec2007	CH	Chadstone Mall	109 Dandenong Road	Oct2007	Mar2008		... ...	....	... ...	... ...
Dec2007	DOW	Doncaster Westfield	75 Doncaster Road	Nov2007	Feb2008		... ...	....	... ...	... ...
Jan2008	MEL	Melbourne Central	88 Lonsdale Street	Jan0000	Dec9999	Main shop	... ...	... ...	... ...	... ...
Jan2008	CH	Chadstone Mall	109 Dandenong Road	Oct2007	Mar2008		... ...	....	... ...	... ...
Jan2008	DOW	Doncaster Westfield	75 Doncaster Road	Nov2007	Feb2008		... ...	....	... ...	... ...

(continued)

Table 6.12 (continued)

Time ID	Branch ID	Branch name	Branch address	Start date	End date	Remarks	Contact number	Merchandise	Amount sold	Total sales
Feb2008	MEL	Melbourne Central	88 Lonsdale Street	Jan0000	Dec9999	Main shop	... ...	... ...	... ...	... ...
Feb2008	CH	Chadstone Mall	109 Dandenong Road	Oct2007	Mar2008		... ...	....	... ...	... ...
Feb2008	DOW	Doncaster Westfield	75 Doncaster Road	Nov2007	Feb2008		... ...	....	... ...	... ...
Mar2008	MEL	Melbourne Central	88 Lonsdale Street	Jan0000	Dec9999	Main shop	... ...	... ...	... ...	... ...
Mar2008	CH	Chadstone Mall	109 Dandenong Road	Oct2007	Mar2008		... ...	....	... ...	... ...
Apr2008	MEL	Melbourne Central	88 Lonsdale Street	Jan0000	Dec9999	Main shop	... ...	... ...	... ...	... ...
May2008	MEL	Melbourne Central	88 Lonsdale Street	Jan0000	Dec9999	Main shop	... ...	... ...	... ...	... ...
...	...	...	...	...	...	...	...	...	...	...
Oct2008	MEL	Melbourne Central	88 Lonsdale Street	Jan0000	Dec9999	Main shop	... ...	... ...	... ...	... ...
Oct2008	CH	Chadstone Mall	109 Dandenong Road	Oct2008	Feb2009	Under re-construction	....	... ...	... ...	... ...
...	...	...	...	...	...	...	...	...	...	...

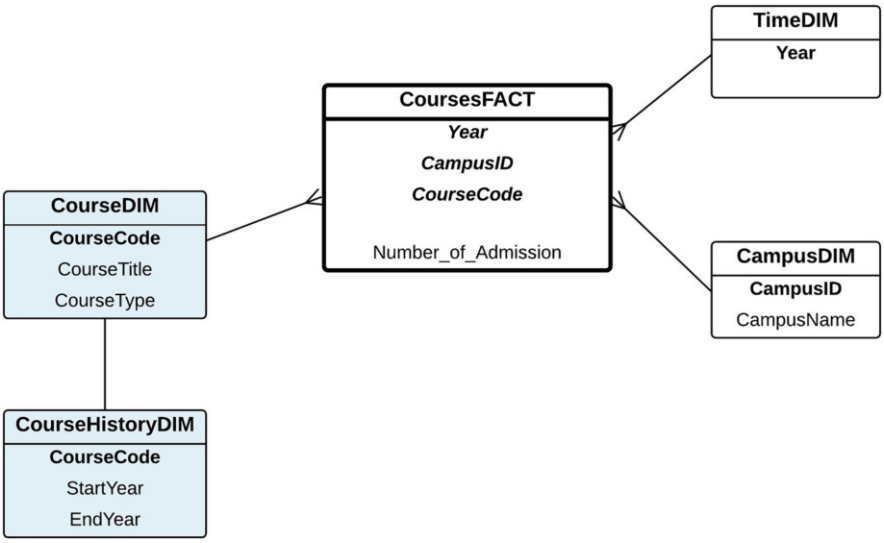

Fig. 6.6 Course History as a temporal dimension: using a Bridge Table

existing degree. When this happens, students entering this course/degree will be admitted to the new course (rather than the old course), while existing students will still be taught using the old degree structure until there are no more students in the old course. The new course will have a new course code, even though the course name might remain the same. Hence, it is common that a course or a degree has a number of course codes, but only one of them can admit new students.

Figure 6.6 shows a simple star schema with a temporal dimension, called Course History Dimension, with a Course Code attribute as the identifier. The year when the course is introduced is stored in the Start Year attribute. When the course no longer admits new students (in other words, the course is being phased out or closed), the End Year is recorded. Course without End Year means that the course is an active and current course. The star schema in Fig. 6.6 is able to analyse the number of "new" students (admission) to enrol in courses, for each year and each campus. Because a course has only "one" history, the relationship between Course and Course History Dimension is a 1-1 relationship.

This case study is rather different from the Calendar Shop case study in the previous section where a branch (or a shop) is open for several months a year and is closed for the rest of the year. However, the shop opens again next year. This means that the shop opens and closes, hence, a shop has many instances in the history of the shop.

On the other hand, when a course is closed, it never re-opens. Hence, a course has only one instance in the history. If the End Year is not specified, the course is still active. If there is an End Year (which is normally a past year), the course is already closed and does not admit any new students. Because the relationship cardinality

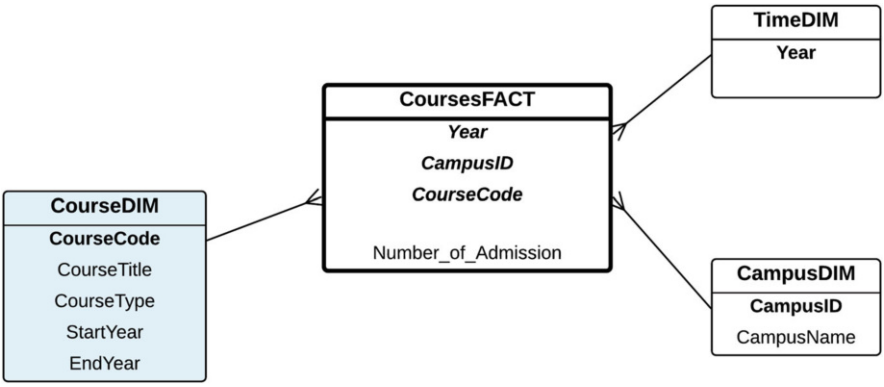

Fig. 6.7 Course History as a temporal dimension: not as a Bridge Table

between Course and Course History is a 1-1 relationship, both dimensions can be combined into one dimension called the Course Dimension (see Fig. 6.7). Hence, the temporal dimension, which is in this case the Course Dimension, is directly linked to the Fact, so this temporal dimension is actually a "Normal" dimension, even though there is a lifetime for each record in this Course Dimension table.

6.4 Slowly Changing Dimensions

Temporal data warehousing is often known as *slowly changing dimensions* or *SCD*. Slowly changing dimensions are dimensions where the records of these dimensions change slowly over a period of time. Using the Bookshop case study earlier in this chapter, the Book Dimension has price information and it is common that the price of a book changes "slowly" over time, which is why the Book Dimension is an example of a slowly changing dimension. It is called "slowly" because changes to price do not occur, for example, every hour or even every week; rather, book prices change over a longer period of time, such as months or even years.

The topic of slowly changing dimensions is different from attributes or records that are "rapidly" changing, such as the location attribute of a moving taxi, which changes very frequently (e.g. changes to the location coordinate are recorded every minute in the operational database), or the price of shares in the stock market, which may change every few seconds, etc. The topic of these rapid changes of records or values is often studied in the area of Real-Time Data Warehousing or Stream Data Warehousing, which is not the scope of this chapter.

There are three basic types of SCD called Type 1, Type 2 and Type 3. Each of these types treats an implementation of SCD differently. However, recently, data warehousing practitioners have added new types, called Type 0, Type 4 and Type 6, which enrich the implementation options for SCD.

6.4.1 SCD Type 0 and Type 1

SCD Type 0 and Type 1 are quite similar; they do not actually record the history of changes in the dimension.

In **Type 0**, the dimension stores the "Original or Initial" value of the records when the data warehouse is built. If the value of the dimension attributes changes, the changes are not recorded. The records remain the same as when the records were first inserted into the data warehouse.

Using the Bookshop case study, these values are the "Full Price". For many books, the original or initial price was the full price, but for some books, the price listed initially may not be the full price. So, the Book Dimension table using SCD Type 0 which lists the full price is shown in Table 6.13. If the book prices change after this, the new price will not be recorded in the data warehouse.

For other systems, it is more desirable to store the original or initial value rather than the so-called full price as in this example.

Because SCD Type 0 does not record the history of the book price, the star schema does not have a temporal dimension. The star schema for the Bookshop case study is shown in Fig. 6.1.

Type 1 also does not record the history of changes; rather it only records the latest value of the record. Using the Book Dimension example, the book price in the Book Dimension will be the latest price. This means that when there is a change of price, the old price will be overwritten by the new price in the Book Dimension table. Table 6.14 shows the contents of the Book Dimension table using SCD Type 1. In this case, the $10.00 price of BookID H6 is the latest price. Note that the other two books in this example have the same price as the full price in SCD Type 0, but this does not mean that the prices were never changed. This Book Dimension table only tells us that these are the current price of the books.

Similar to SCD Type 0, the SCD Type 1 star schema does not maintain a temporal dimension. The star schema is the one shown in Fig. 6.1.

Because SCD Type 0 and Type 1 do not maintain the history of book prices, if we need to produce a report that joins the Book Dimension table with the Book Sales

Table 6.13 Book Dimension table (SCD Type 0)

BookID	Book title	Author	Price
C1	CSIRO Diet	CSIRO Team	$45.95
H6	Harry Potter 6	Rowling	$30.95
DV	Da Vinci Code	Dan Brown	$27.95
...	...	...	...

Table 6.14 Book Dimension Table (SCD Type 1)

BookID	Book title	Author	Price
C1	CSIRO Diet	CSIRO Team	$45.95
H6	Harry Potter 6	Rowling	**$10.00**
DV	Da Vinci Code	Dan Brown	$27.95
...	...	...	...

Fact Table, we need to be careful not to draw an association between the book price from Book Dimension and the TimeID from the Book Sales Fact, because the book price in the report may not necessarily be the book price at that particular TimeID. In such a report, the column title for the book price can be changed to "original book price" (for SCD Type 0) or "latest book price" (for SCD Type 1) to avoid any misunderstanding in interpreting the report.

6.4.2 SCD Type 2

SCD Type 2 keeps track of the history but does not separate the history from the main dimension; instead, the new records are continually added to the dimension. Using the Book Dimension as an example, when the price of a book is changed, it creates "another book" with the same details but with the new BookID and of course the new price. In addition to the Start Date and End Date, it also has a Current Flag to indicate whether a record is the current or a past record. Any additional information, such as remarks, may also be included. The Book Dimension table for SCD Type 2 is shown in Table 6.15.

Note that the same book has a different BookID for a different book price and period. Usually, the BookID attribute is implemented as a Surrogate Key, but in this example, we just added a sequence number to the original BookID to differentiate the same book in a different time period. Because of these multiple BookIDs for the same book, the report that joins the Book Sales Fact and the Book Dimension tables will appear as follows (see report 3 SCD Type 2 in Table 6.16).

Note that the first record in Report 3, the BookID, is C1_4, because this was the BookID of the CSIRO Diet book in March 2008. Because SCD Type 2 only changes the original Book Dimension table, the star schema looks similar to that of Fig. 6.1, but the Book Dimension table has additional attributes (see Fig. 6.8).

Table 6.15 Book Dimension table (SCD Type 2)

BookID	Book title	Author	Start Date	End Date	Price	Remarks	Current Flag
C1_1	CSIRO Diet	CSIRO Team	Jan2007	Jul2007	$45.95	Full Price	N
C1_2	CSIRO Diet	CSIRO Team	Aug2007	Oct2007	$36.75	20% Discount	N
C1_3	CSIRO Diet	CSIRO Team	Nov2007	Jan2008	$23.00	Half Price	N
C1_4	CSIRO Diet	CSIRO Team	Feb2008	Dec9999	$45.95	Full Price	Y
H6_1	Harry Potter 6	Rowling	Jan2007	Mar2007	$21.95	Launching	N
H6_2	Harry Potter 6	Rowling	Apr2007	Jan2008	$30.95	Full Price	N
H6_3	Harry Potter 6	Rowling	Feb2008	Dec9999	$10.00	End of Product Sale	Y
DV_1	Da Vinci Code	Dan Brown	Jan2007	Dec9999	$27.95	Full Price	Y
...	...	...	...	...	...	...	...

Table 6.16 Report 3 (SCD Type 2)

TimeID	BranchID	BookID	Book title	Author	Price	Number of books sold
Mar2008	City	**C1_4**	CSIRO Diet	CSIRO Team	$45.95	5
Mar2008	City	**H6_3**	Harry Potter 6	Rowling	$10.00	15
Mar2008	City	**DV_1**	Da Vinci Code	Dan Brown	$27.95	23
Mar2008	City	...	...	...	...	...
Mar2008	Chadstone	**C1_4**	CSIRO Diet	CSIRO Team	$45.95	15
Mar2008	Chadstone	**H6_3**	Harry Potter 6	Rowling	$10.00	3
Mar2008	Chadstone	**DV_1**	Da Vinci Code	Dan Brown	$27.95	2
Mar2008	Chadstone	...	...	...	...	...
Mar2008	Camberwell	**C1_4**	CSIRO Diet	CSIRO Team	$45.95	1
Mar2008	Camberwell	**H6_3**	Harry Potter 6	Rowling	$10.00	1
Mar2008	Camberwell	**DV_1**	Da Vinci Code	Dan Brown	$27.95	2
Mar2008	Camberwell	...	...	...	...	...
Mar2008	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...
Dec2007	City	**C1_3**	CSIRO Diet	CSIRO Team	$23.00	15
Dec2007	City	**H6_2**	Harry Potter 6	Rowling	$30.95	6
Dec2007	City	**DV_1**	Da Vinci Code	Dan Brown	$27.95	6
Dec2007	City	..	...	...	...	...
Dec2007	Chadstone	**C1_3**	CSIRO Diet	CSIRO Team	$23.00	10
Dec2007	Chadstone	**H6_2**	Harry Potter 6	Rowling	$30.95	8
Dec2007	Chadstone	**DV_1**	Da Vinci Code	Dan Brown	$27.95	1
Dec2007	Chadstone	..	...	...	...	...
Dec2007	Camberwell	**C1_3**	CSIRO Diet	CSIRO Team	$23.00	18
Dec2007	Camberwell	**H6_2**	Harry Potter 6	Rowling	$30.95	3
Dec2007	Camberwell	**DV_1**	Da Vinci Code	Dan Brown	$27.95	2
Dec2007	Camberwell	..	...	...	...	...
Dec2007	...	...	...	...	...	...
...	...	...	...	...	...	...

6.4.3 SCD Type 3

SCD Type 3 is a simplification of Type 2. Unlike Type 2 which maintains multiple records of the same book, Type 3 does not have multiple records for the same book. One book has one entry in the Book Dimension table. For the price, it only records the current price and the previous price. In other words, it does not maintain the entire history of price changes; rather, it only keeps the last two prices. The Book Dimension table for SCD Type 3 is shown in Table 6.17. Note that for a book which does not have any previous price, a Null value will be recorded in the Previous Price column.

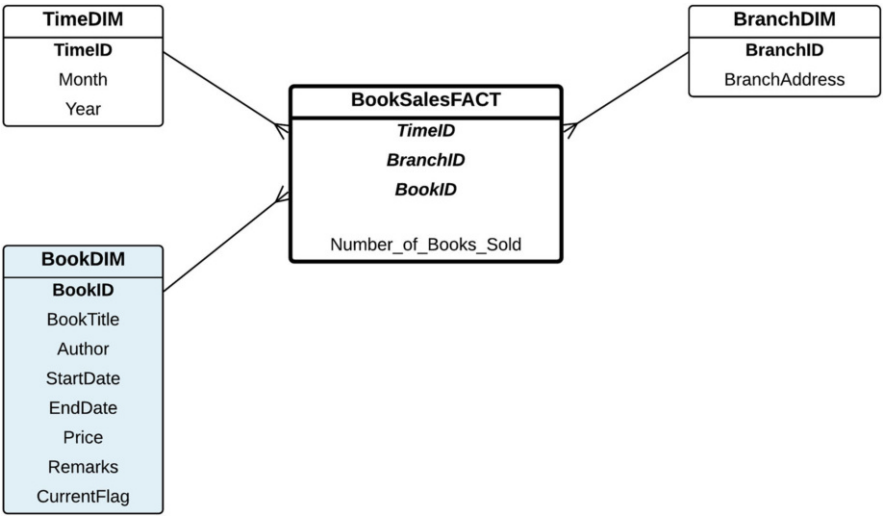

Fig. 6.8 The Bookshop star schema with SCD Type 2 for Book Dimension

Table 6.17 Book Dimension table (SCD Type 3)

BookID	Book title	Author	Current price	Previous price
C1	CSIRO Diet	CSIRO Team	**$45.95**	**$23.00**
H6	Harry Potter 6	Rowling	**$10.00**	**$30.95**
DV	Da Vinci Code	Dan Brown	**$27.95**	**Null**
...	...	...	...	...

The main rationale for adopting SCD Type 3 is that most of the analysis will be based on the current price and, at most, one past price, this being the previous price. Perhaps, this is so it can be compared with the pricing trend. It is assumed that analysing the complete history is not necessary.

Additionally, although we can store information on when the price was changed, the original SCD Type 3 does not record this. It only records the current and the previous prices. Without keeping the date when the price was changed, it is not possible to correlate the book price with the TimeID information in the Book Sales Fact Table. Consequently, the report that can be produced will need to pay particular attention to the potential mismatch between the information in the book price column (from the Book Dimension table) and the TimeID column (from the Book Sales Fact Table).

The star schema for SCD Type 3 is shown in Fig. 6.9.

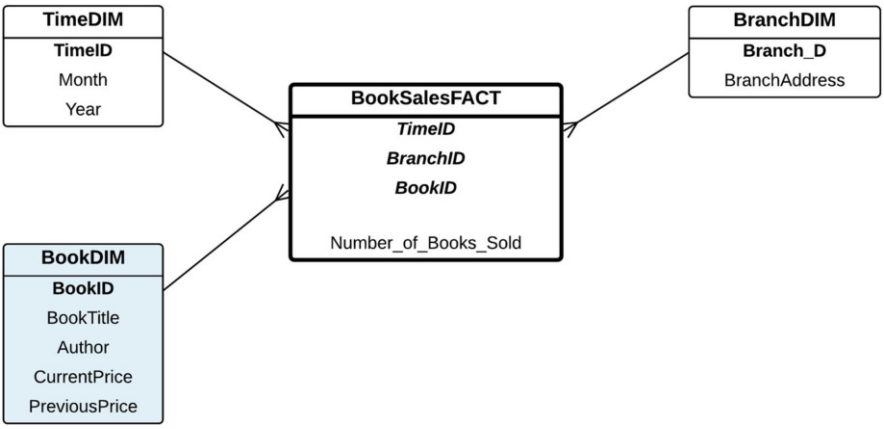

Fig. 6.9 The Bookshop star schema with SCD Type 3 for Book Dimension

Table 6.18 Book Dimension table

BookID	Book title	Author
C1	CSIRO Diet	CSIRO Team
H6	Harry Potter 6	Rowling
DV	Da Vinci Code	Dan Brown
...	...	...

Table 6.19 Book Price Dimension table

BookID	Start date	End date	Price	Remarks
C1	Jan2007	Jul2007	$45.95	Full Price
C1	Aug2007	Oct2007	$36.75	20% Discount
C1	Nov2007	Jan2008	$23.00	Half Price
C1	Feb2008	Dec9999	$45.95	Full Price
H6	Jan2007	Mar2007	$21.95	Launching
H6	Apr2007	Jan2008	$30.95	Full Price
H6	Feb2008	Dec9999	$10.00	End of Product Sale
DV	Jan2007	Dec9999	$27.95	Full Price
...	...	...	...	...

6.4.4 SCD Type 4

SCD Type 4 is a new way to treat SCD that was not in the original SCD theory. In SCD Type 4, we create a new dimension to maintain the history of attribute value changes. Basically, the temporal dimension that is described in the beginning of this chapter is the SCD Type 4. In Type 4, the original Book Dimension table is kept without the price attribute. The price attribute, Start Date and End Date are separated into another table, which is the Book Price Dimension table. The Book Dimension and the Book Price Dimension tables are shown in Tables 6.18 and 6.19.

The main advantage of Type 4 is that we do not need to have a different BookID for the same book. Additionally, the entire history of changes is kept. As shown earlier, this method will guarantee that the report that joins the information from the Book Sales Fact Table and the dimension tables will be accurate and will reflect the correct book price at a certain TimeID. The star schema for SCD Type 4 was shown previously in Fig. 6.2.

6.4.5 SCD Type 6

SCD Type 6 is a combination of Type 2 and Type 3. In Type 3, only the current price and the previous price are recorded, but not the entire history. In Type 2, the entire history of changes is maintained, but a separate identifier (e.g. the Surrogate Key) is needed. In Type 6, a separate identifier for the same book is not needed (as in Type 3), but the entire history is kept (similar to Type 2). SCD Type 6 for the Book Dimension table is shown in Table 6.20.

In Type 6, there is no need to maintain a separate history table. The history itself is kept in the original dimension table. The star schema for SCD Type 6 is shown in Fig. 6.10. Note that the Book Dimension table has a composite key comprising BookID, Start Date and End Date.

Table 6.20 Book Dimension (SCD Type 6)

Book ID	Book Title	Author	Start Date	End Date	Current Price	Previous Price	Remarks	Current Flag
C1	CSIRO Diet	CSIRO Team	Jan2007	Jul2007	$45.95	Null	Full Price	N
C1	CSIRO Diet	CSIRO Team	Aug2007	Oct2007	$36.75	$45.95	20% Discount	N
C1	CSIRO Diet	CSIRO Team	Nov2007	Jan2008	$23.00	$36.75	Half Price	N
C1	CSIRO Diet	CSIRO Team	Feb2008	Dec9999	$45.95	$23.00	Full Price	Y
H6	Harry Potter 6	Rowling	Jan2007	Mar2007	$21.95	Null	Launching	N
H6	Harry Potter 6	Rowling	Apr2007	Jan2008	$30.95	$21.95	Full Price	N
H6	Harry Potter 6	Rowling	Feb2008	Dec9999	$10.00	$30.95	End of Product	Y
DV	Da Vinci Code	Dan Brown	Jan2007	Dec9999	$27.95	Null	Full Price	Y
...	...	...	...	...	...	...	...	...

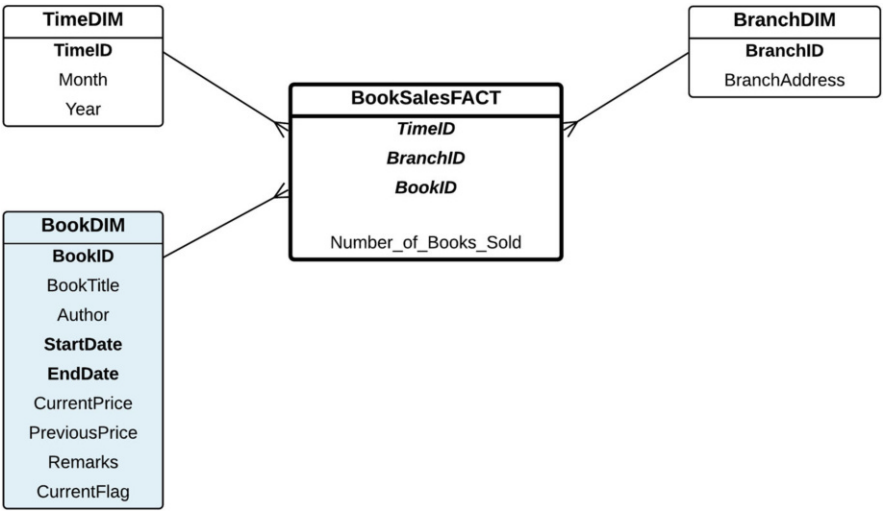

Fig. 6.10 The Bookshop star schema with SCD Type 6 for Book Dimension

If we examine this star schema (and the Book Dimension table), the Book Dimension table has a composite key, which comprises BookID, Start Date and End Date, whereas the Fact Table uses only BookID as the reference to the Book Dimension table. Taking BookID C1 as an example, there are four records of C1 in the Book Dimension table, and there are many records of C1 in the Fact Table. The cardinality relationship between the Book Dimension table and Fact Table is no longer 1-*m*, but *m*-*m*. Normally, there are three possible options to address this issue:

1. Add a new surrogate key to the Book Dimension table, and consequently the Fact Table will have this surrogate key from the Book Dimension as a reference. This surrogate key can also simply be a concatenation between BookID, Start Date and End Date. Or, the surrogate key is BookID with a sequence number, like in SCD Type 2.
2. If we do not want a surrogate key in the Book Dimension table, then the Fact Table must also include the Start Date and End Date, in addition to the BookID. But this will be messy because the Fact already had TimeID, referencing the Time Dimension table.
3. The last option is to have an associative table (or a bridge table) between Book Dimension and Fact, because the cardinality between Book Dimension and Fact is *m*-*m*. This middle table that links Book Dimension and Fact will have a composite key comprising the identifier from Book Dimension and Fact. This is also a messy option, because we are trying to use a relational theory to solve this *m*-*m* issue.

The easiest way to address this issue is by adopting the first option, making SCD Type 6 almost identical to SCD Type 2. In other words, SCD Type 6 is unnecessary or useless. But, if we insist on implementing SCD Type 6 as shown in the star schema in Fig. 6.10, since no PK-FK is explicitly implemented in the tables, it is still possible for the two tables, Book Dimension and Fact, to exist independently. However, when we want to produce a report that joins the Fact and Book Dimension tables, the join must include these conditions: (*i*) the BookID of both tables must be the same, and (*ii*) the TimeID of the Fact must be in between the Start Date and End Date of the Book Dimension. These join conditions are rather similar to the join conditions used in SCD Type 4, or the temporal data warehousing discussed earlier in this chapter, where the date is part of the join condition between the Fact, Book Dimension and Book Price Dimension. On this note, if we examine SCD Type 6 even more deeply, there is a strong similarity between SCD Type 6 and SCD Type 4. SCD Type 4 is actually a normalised version of SCD Type 6, where the Book Dimension table acts like an intermediate table between Book Price Dimension and Fact. SCD Type 6, in contrast, is a non-normalised version of SCD Type 4, where everything is lumped into one table, namely, the Book Dimension table. The join conditions to join the Book Dimension and the Fact are identical.

6.4.6 Implementation of SCD in SQL

The concepts of SCD are very easy to digest. However, implementing SCD in SQL has several challenging technical aspects. In this section, we are going to examine the SQL to create the six SCD types.

The source tables in the operational database are the Book table and the Book Price History table, as shown in the E/R diagram (refer to Fig. 6.1). The structures of these two tables are:

- **Book** (BookID, BookTitle, Author)
- **BookPriceHistory** (BookID, StartDate, EndDate, Price, Remarks)

SCD Type 0 is a non-temporal dimension. The Book Dimension table using SCD Type 0 is shown in Table 6.13. Note that the "original price" may not be the "initial price" because some books have a launching price which can be heavily discounted. Therefore, the Book Dimension table using SCD Type 0 checks for the Remarks "Full Price".

```
create table SCD0 as
select distinct
  B.BookID, B.BookTitle,
  B.Author, H.Price as OriginalPrice
from Book B, BookPriceHistory H
where B.BookID = H.BookID
and H.Remarks = 'Full Price';
```

SCD Type 1 uses the latest price (refer to Table 6.14). We could choose a book where the End Date is "Dec9999". Alternatively, we can sort by the Start Date and choose the "latest" Start Date. In the following SQL, the inner query ranks the book records based on the Start Date in descending order, and the outer query chooses those books which have a rank of 1.

```
create table SCD1 as
select
  T.BookID, T.BookTitle,
  T.Author, T.Price as CurrentPrice
from (
  select
    B.BookID, B.BookTitle, B.Author,
    to_date(H.StartDate, 'MonYYYY'), H.Price,
    rank() over( partition by B.BookID
       order by to_date(H.StartDate, 'MonYYYY') desc)
       as Rank
  from Book B, BookPriceHistory H
  where B.BookID = H.BookID) T
where T.Rank = 1;
```

SCD Type 2 is a join between the Book table and Book Price History table, so that we can get the book details as well as the Start Date and End Date. However, a book with the same BookID should be added with a sequence number, as previously shown in Table 6.15.

The SQL for SCD Type 2 uses the rank function as well as the partition clause, so the same book will be ranked according to their Start Date. The Current Flag column will identify whether the book record has the current price or not.

```
create table SCD2 as
select  B.BookID  || '_' ||
   rank() over(partition by B.BookID
     order by to_date(H.StartDate, 'MonYYYY') asc)
     as BookID,
  B.BookTitle, B.Author, H.StartDate,
  H.EndDate, H.Price, H.Remarks,
  case H.EndDate when 'Dec9999' then 'Y' else 'N'
  end as CurrentFlag
from Book B, BookPriceHistory H
where B.BookID = H.BookID;
```

SCD Type 3 is quite complex as it not only has the Current Price (similar to SCD Type 2) but also the Previous Price. If SCD Type 2 uses rank 1, SCD Type 3 needs rank 2 (for the Previous Price) as well. Hence, SCD Type 3 joins the table with rank 1 and the table with rank 2. Since some books do not have the Previous Price, we need to use an outer join instead of an inner join.

```
create table SCD3 as
select
  T1.BookID, T1.BookTitle, T1.Author,
  T1.CurrentPrice, T2.CurrentPrice as PreviousPrice
from (
```

```
select
  T.BookID, T.BookTitle,
  T.Author, T.Price as CurrentPrice
from (
   select
     B.BookID, B.BookTitle,
     B.Author, to_date(H.StartDate, 'MonYYYY'),
     H.Price,
     rank() over( partition by B.BookID
       order by to_date(H.StartDate, 'MonYYYY') desc)
       as Rank
   from Book B, BookPriceHistory H
   where B.BookID = H.BookID) T
where T.Rank = 1) T1,
(select
  T.BookID, T.BookTitle,
  T.Author, T.Price as CurrentPrice
from (
  select
    B.BookID, B.BookTitle, B.Author,
    to_date(H.StartDate, 'MonYYYY'), H.Price,
    rank() over( partition by B.BookID
      order by to_date(H.StartDate, 'MonYYYY') desc)
      as Rank
  from Book B, BookPriceHistory H
  where B.BookID = H.BookID) T
where T.Rank = 2) T2
where T1.BookID = T2.BookID(+);
```

SCD Type 4 is actually the temporal data warehousing discussed in the first part of this chapter. The SQL to create the Book Dimension is an entire extraction from the Book Price History table from the operational database.

```
create table SCD4 as
select * from BookPriceHistory;
```

SCD Type 6 is a combination of SCD Type 2 and SCD Type 3. However, the implementation of SCD Type 6 in SQL can be very challenging. We cannot simply join SCD Type 2 and SCD Type 3 because SCD Type 2 has the complete history of prices, whereas SCD Type 3 has only the current and previous prices. Additionally, joining between SCD Type 2 and SCD Type 3 cannot use an equi-join. Instead, we need to check if the BookID of SCD Type 3 is part of the BookID of SCD Type 2, using the `like` operator.

Creating SCD Type 6 from SCD Type 2 and SCD Type 3 needs a couple of steps. The first step is to join SCD Type 2 and SCD Type 3 using the `like` operator. The join results use the BookID attribute from SCD Type 3, and the rest of the attributes are from SCD Type 2. Additionally, an Order Number attribute will have a sequence number for each group of BookID. The results of this join query are shown in Table 6.21.

Table 6.21 Join between SCD Type 2 and SCD Type 3

Book ID	Book Title	Author	Start Date	End Date	Price	Remarks	Current Flag	Order Number
C1	CSIRO Diet	CSIRO Team	Jan2007	Jul2007	$45.95	Full Price	N	**1**
C1	CSIRO Diet	CSIRO Team	Aug2007	Oct2007	$36.75	20% Discount	N	**2**
C1	CSIRO Diet	CSIRO Team	Nov2007	Jan2008	$23.00	Half Price	N	**3**
C1	CSIRO Diet	CSIRO Team	Feb2008	Dec9999	$45.95	Full Price	Y	**4**
DV	Da Vinci Code	Dan Brown	Jan2007	Dec9999	$27.95	Full Price	Y	**1**
H6	Harry Potter 6	Rowling	Jan2007	Mar2007	$21.95	Launching	N	**1**
H6	Harry Potter 6	Rowling	Apr2007	Jan2008	$30.95	Full Price	N	**2**
H6	Harry Potter 6	Rowling	Feb2008	Dec9999	$10.00	End of Product	Y	**3**
...	...	...	...	...	...	...	...	...

```
select
  SCD3.BookID,
  SCD2.BookTitle, SCD2.Author,
  SCD2.StartDate, SCD2.EndDate,
  SCD2.Price, SCD2.Remarks, SCD2.CurrentFlag,
  row_number() over
    (partition by SCD3.BookID
     order by SCD3.BookID, SCD2.StartDate) as
     OrderNumber
  from SCD2, SCD3
where SCD2.BookID like SCD3.BookID||'_%'
order by SCD3.BookID, SCD2.StartDate;
```

The next step is to do a self-join of Table 6.21. The Price from the first table will become the Current Price, and the Price from the second table will become the Previous Price. There are two join conditions for this self-join query. The first condition is that the BookID must be the same. In order for the Price from the second table to become the Previous Price, we need to add the Order Number in the second table by one, and then the second join condition is to match the Order Number from these two tables. The SQL for the self-join is as follows, whereas the result is shown in Table 6.22.

```
select
  T1.BookID, T1.BookTitle, T1.Author,
  T1.StartDate, T1.EndDate,
  T1.Price as CurrentPrice,
  T2.Price as PreviousPrice,
  T1.Remarks, T1.CurrentFlag
from (
  select SCD3.BookID,
    SCD2.BookTitle, SCD2.Author, SCD2.StartDate,
```

```
    SCD2.EndDate, SCD2.Price, SCD2.Remarks,
    SCD2.CurrentFlag,
    row_number() over
      (partition by SCD3.BookID
       order by SCD3.BookID, SCD2.StartDate) as
       OrderNumber
    from SCD2, SCD3
  where SCD2.BookID like SCD3.BookID||'_%'
  order by SCD3.BookID, SCD2.StartDate) T1,
  (
  select SCD3.BookID,
    SCD2.BookTitle, SCD2.Author, SCD2.StartDate,
    SCD2.EndDate, SCD2.Price, SCD2.Remarks,
    SCD2.CurrentFlag,
    row_number() over
      (partition by SCD3.BookID
       order by SCD3.BookID, SCD2.StartDate) + 1 as
       OrderNumber
    from SCD2, SCD3
  where SCD2.BookID like SCD3.BookID||'_%'
  order by SCD3.BookID, SCD2.StartDate) T2
where T1.BookID = T2.BookID
and T1.OrderNumber = T2.OrderNumber
order by T1.BookID, T1.StartDate;
```

However, the self-join results excluded the first record of each book, because there is no Previous Price for the first record of each book. In order to fix this mistake, the self-join query needs to use an outer join, instead of an inner join. The correct SQL query for SCD Type 6 that uses an outer join is very similar to the above SQL query that uses an inner join, with only one difference, that is, the join

Table 6.22 Self-join results

Book ID	Book Title	Author	Start Date	End Date	Current Price	Previous Price	Remarks	Current Flag
C1	CSIRO Diet	CSIRO Team	Aug2007	Oct2007	$36.75	$45.95	20% Discount	N
C1	CSIRO Diet	CSIRO Team	Nov2007	Jan2008	$23.00	$36.75	Half Price	N
C1	CSIRO Diet	CSIRO Team	Feb2008	Dec9999	$45.95	$23.00	Full Price	Y
H6	Harry Potter 6	Rowling	Apr2007	Jan2008	$30.95	$21.95	Full Price	N
H6	Harry Potter 6	Rowling	Feb2008	Dec9999	$10.00	$30.95	End of Product	Y
...	...	...	...	...	...	...	...	...

Table 6.23 The correct self-join results (SCD Type 6)

Book ID	Book Title	Author	Start Date	End Date	Current Price	Previous Price	Remarks	Current Flag
C1	CSIRO Diet	CSIRO Team	Jan2007	Jul2007	$45.95	Null	Full Price	N
C1	CSIRO Diet	CSIRO Team	Aug2007	Oct2007	$36.75	$45.95	20% Discount	N
C1	CSIRO Diet	CSIRO Team	Nov2007	Jan2008	$23.00	$36.75	Half Price	N
C1	CSIRO Diet	CSIRO Team	Feb2008	Dec9999	$45.95	$23.00	Full Price	Y
DV	Da Vinci Code	Dan Brown	Jan2007	Dec9999	$27.95	Null	Full Price	Y
H6	Harry Potter 6	Rowling	Jan2007	Mar2007	$21.95	Null	Launching	N
H6	Harry Potter 6	Rowling	Apr2007	Jan2008	$30.95	$21.95	Full Price	N
H6	Harry Potter 6	Rowling	Feb2008	Dec9999	$10.00	$30.95	End of Product	Y
...	...	...	...	...	...	...	...	...

condition is now an outer join (with a (+) sign at the end of each join condition), which is as follows:

```
where T1.BookID = T2.BookID (+)
and T1.OrderNumber = T2.OrderNumber (+)
```

The final results are shown in Table 6.23.

6.4.7 Creating the Fact Tables

After tackling the implementation challenges of SCD, the next step is to examine the SQL commands to create the Fact Tables. For SCD Types 0, 1, 3 and 4 (see the sample records in Tables 6.13, 6.14, 6.17, 6.18 and 6.19), the BookID attributes are exactly the same as the BookID in the "non-temporal" version of data warehousing (refer to Table 6.2 at the beginning of this chapter). Consequently, the Fact Table will not be affected when we use SCD Types 0, 1, 3 or 4. The original Book Dimension, as shown in the non-temporal version, has three books with BookIDs: C1 (CSIRO Diet), H6 (Harry Potter) and DV (Da Vinci Code). The Book Dimensions in SCD Types 0, 1, 3 and 4 also have the same BookIDs as the original Book Dimension, and the number of records in these tables is the same.

However, when producing a report, by joining the Fact Table and the Book Dimension table, when all the attributes from the Book Dimension are present (including the Price), we cannot draw a correlation between the TimeID in the Fact and the Book Price in the Book Dimension table, because the Book Price displayed on the report might not be the price on the TimeID. This problem was illustrated in Table 6.3. This is why instead of using SCD Types 0, 1 or 3, we use SCD Type 4, which was discussed earlier in this chapter. With SCD Type 4, we not only maintain the complete history of Book Price; we are also able to produce correct reports when joining with the Fact Table.

For SCD Type 2 (see the sample records in Table 6.15), the number of records is not the same as the original Book Dimension table. This is because every time the Book Price is changed, a "new" book is inserted into the dimension table. This has an impact on the contents of the Fact Table. For example, in the Fact Table, it should not be BookID C1 for CSIRO Diet Book, but C1_1 or C1_2, etc., to refer to different Book instance in the history. Therefore, when SCD Type 2 is used, the contents of the Fact Table must contain the correct BookID. The SQL command to create the Fact Table for SCD Type 2 is as follows:

```
create table BookSalesFactWithSCD2 as
select
  to_char(T.TransactionDate, 'MonYYYY') as TimeID,
  BK.BookID,
  BR.BranchID,
  sum(T.Quantity) as Number_of_Books_Sold
from BookTransaction T, SCD2 BK, Branch BR
where T.BranchID = BR.BranchID
and BK.BookID like T.BookID||'_%'
and to_date(BK.StartDate, 'MonYYYY') <= T.TransactionDate
and T.TransactionDate <= to_date(BK.EndDate, 'MonYYYY')
group by
  to_char(T.TransactionDate, 'MonYYYY'),
  BK.BookID,
  BR.BranchID;
```

There are three things to note: (*i*) It joins with the Book Dimension SCD Type 2. The reason for this is that the Book Table in the operational database does not have the new BookIDs. The new BookIDs only exist in SCD Type 2, because the new BookIDs are created when the Book Price is changed. (*ii*) The join condition uses a LIKE and a wildcard to match the new BookID in SCD Type 2 and the BookID in the Boo Transaction table from the operational database. (*iii*) The join condition must include checking the dates so that the correct record in the Book Dimension SCD Type 2 is used when joining with the records in the Book Transaction table from the operational database. The Fact Table is shown in Table 6.24.

SCD Type 6 is similar to SCD Type 2, that is, it contains the same number of records. The only difference is that in SCD Type 6, BookID does not change, and the original BookID is used. Therefore, the SQL to create the Fact Table is slightly simpler. In the join condition, it simply compares the BookID from SCD Type 6 and the Book Transaction table (note that in SCD Type 2 Fact Table, it uses the LIKE and wildcard). The SQL command to create the Fact Table for SCD Type 6 is as follows:

```
create table BookSalesFactWithSCD6 as
select
  to_char(T.TransactionDate, 'MonYYYY') as TimeID,
  BK.BookID,
  BR.BranchID,
  sum(T.Quantity) as Number_of_Books_Sold
from BookTransaction T, SCD6 BK, Branch BR
where T.BranchID = BR.BranchID
```

Table 6.24 Fact Table (SCD Type 2)

TimeID	BranchID	BookID	Number of books sold
Mar2008	City	**C1_4**	5
Mar2008	City	**H6_3**	15
Mar2008	City	**DV_1**	23
Mar2008	City	...	...
Mar2008	Chadstone	**C1_4**	15
Mar2008	Chadstone	**H6_3**	3
Mar2008	Chadstone	**DV_1**	2
Mar2008	Chadstone	...	...
Mar2008	Camberwell	**C1_4**	1
Mar2008	Camberwell	**H6_3**	1
Mar2008	Camberwell	**DV_1**	2
Mar2008	Camberwell	...	...
Mar2008	...	...	...
...	...	...	...
...	...	...	...
Dec2007	City	**C1_3**	15
Dec2007	City	**H6_2**	6
Dec2007	City	**DV_1**	6
Dec2007	City	..	...
Dec2007	Chadstone	**C1_3**	10
Dec2007	Chadstone	**H6_2**	8
Dec2007	Chadstone	**DV_1**	1
Dec2007	Chadstone	..	...
Dec2007	Camberwell	**C1_3**	18
Dec2007	Camberwell	**H6_2**	3
Dec2007	Camberwell	**DV_1**	2
Dec2007	Camberwell	..	...
Dec2007	...	...	...
...	...	...	...

```
and BK.BookID = T.BookID
and to_date(BK.StartDate, 'MonYYYY') <= T.TransactionDate
and T.TransactionDate <= to_date(BK.EndDate, 'MonYYYY')
group by
  to_char(T.TransactionDate, 'MonYYYY'),
  BK.BookID,
  BR.BranchID;
```

The Fact Table is shown in Table 6.25. It has almost the same records as Table 6.24, except that here the original BookID is kept.

It is worthwhile to note that when we join SCD Type 2 and the Fact Table, or when we join SCD Type 6 and the Fact Table, the correct Book Price will be shown (as Book Price matches with the TimeID), and therefore, there will not be any incorrect correlation between these two attributes in the report. The reason for

Table 6.25 Fact Table (SCD Type 6)

TimeID	BranchID	BookID	Number of books sold
Mar2008	City	**C1**	5
Mar2008	City	**H6**	15
Mar2008	City	**DV**	23
Mar2008	City	...	...
Mar2008	Chadstone	**C1**	15
Mar2008	Chadstone	**H6**	3
Mar2008	Chadstone	**DV**	2
Mar2008	Chadstone	...	...
Mar2008	Camberwell	**C1**	1
Mar2008	Camberwell	**H6**	1
Mar2008	Camberwell	**DV**	2
Mar2008	Camberwell	...	...
Mar2008	...	...	...
...	...	...	...
...	...	...	...
Dec2007	City	**C1**	15
Dec2007	City	**H6**	6
Dec2007	City	**DV**	6
Dec2007	City	..	...
Dec2007	Chadstone	**C1**	10
Dec2007	Chadstone	**H6**	8
Dec2007	Chadstone	**DV**	1
Dec2007	Chadstone	..	...
Dec2007	Camberwell	**C1**	18
Dec2007	Camberwell	**H6**	3
Dec2007	Camberwell	**DV**	2
Dec2007	Camberwell	..	...
Dec2007	...	...	...
...	...	...	...

this is that both SCD Type 2 and Type 6 maintain the complete history of Book Prices; they keep the StartDate and EndDate of each Book Price.

SCD Type 4 (our temporal data warehousing version) also keeps the complete history of Book Prices, and therefore, the report when joining the Fact Table and the Book Dimension Table is also correct, as shown previously in Table 6.6. SCD Type 2 and SCD Type 6 will also produce the same report. The only difference is the way the Book Price history is maintained. SCD Type 4 uses two tables (i.e. Book Dimension and Book Price Dimension), whereas SCD Type 2 and SCD Type 6 use one table only, where the Book Price history is maintained in the Book Dimension.

6.5 Summary

In this chapter, we focus on incorporating historical data in the data warehouse. This is called *temporal data warehousing*. A temporal data warehousing uses the concept of the Bridge Table (or a Weak Entity), where the history is maintained in a bridge table. Maintaining the history of certain attributes is important in order to make associative analysis more accurate when analysing the reports produced by the fact and dimensions. However, a certain degree of caution is needed when joining the Fact Table and the temporal dimension, especially when the level of granularity of time between the fact and the temporal dimension is not the same.

Temporal data warehousing is also known as *slowly changing dimensions* (*SCD*). In this chapter, various treatments and types for SCD are presented. Different types will serve different purposes of data warehousing.

6.6 Exercises

6.1 It is very common for professors in a university to embark on a consulting project with a company. This consulting job is often regarded as a research activity, as the professor's research expertise is being utilised by a company in a particular project. The research office in the university keeps track of all consulting jobs that professors undertake. A simple E/R diagram to do this is shown in Fig. 6.11. Each consulting project has Project Number and Title, as well as the Company who provided the money, the amount and the Year when the consulting was conducted. For research reporting purposes, the research office at the university assigned a research field for each consulting job. Examples of some research fields are AI, Big Data, Visualisation, Web Development, Mobile Apps, etc.

Also note that the professors' position history is also maintained. This is particularly useful to see the trend of the amount of money that the professor secures from the company as he/she moves up the ladder of the professor ranks (e.g. Assistant Professor, Associate Professor and Professor).

The task is to design a star schema to keep track of the Total Consulting Amount for each year, each research field as well as each professor. Write the SQL commands to create the dimensions and Fact Tables.

6.2 Employee Dimension is a *slowly changing dimension* (*SCD*). For simplicity, the dimension contains only these attributes: EmployeeID, Name and Salary. SCD has the six types: Types 0, 1, 2, 3, 4 and 6. Draw sample tables for Employee Dimension using each of the SCD types. Assume that the Salary attribute is the *Temporal attribute*, which may change from time to time, due to promotion, increase of salary, etc. Add more attributes to the table, whenever required by the SCD. For each type, you need to have at least two employee records, and these are the two employees:

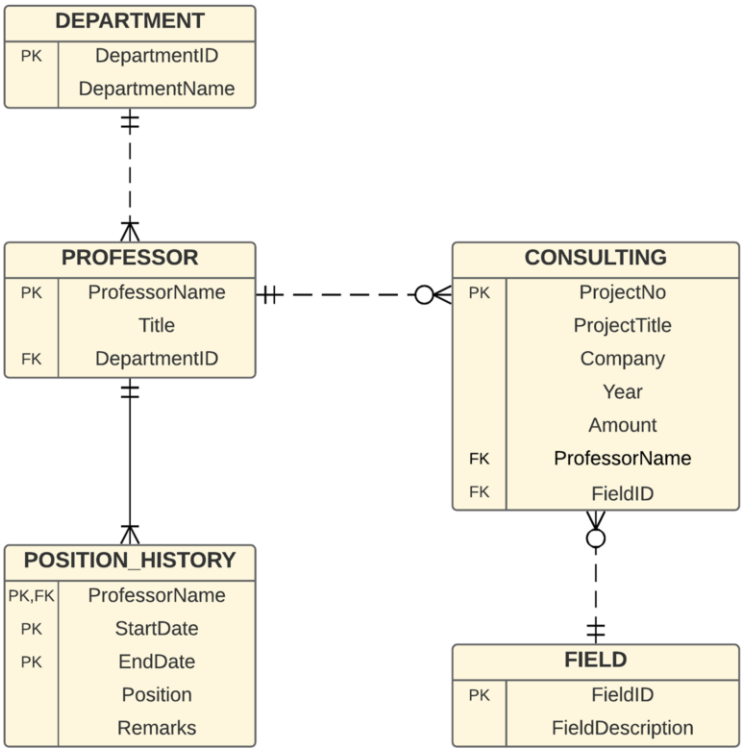

Fig. 6.11 Consulting E/R diagram

- The first employee is *Adam* with an EmployeeID: A1. He started his job in 1 March 2016 with a salary of $3900. After his probation ended in 31 May 2016, his salary was increased to $4300. At the beginning of 2017, he received a pay increase, and his salary became $5000. From 1 July 2017 (until now), his salary has been $5500.
- The second employee is *Ben* with an EmployeeID: B2. He started his job in 1 February 2017 with an initial salary of $4000. After his probation ended in 31 May 2017, his salary became $4750 which is his current salary.

6.7 Further Readings

Slowly changing dimension (SCD) has been discussed and explained in various data warehousing books, such as [1–7] and [8]. There have been discussions to extend the SCD from Type 6 to more complex SCDs, by combining several basic SCDs.

[9] discusses an object-relational implementation of temporal data warehouses, using multi-valued attributes, objects and classes. Further details about object-

relational implementation of an RDBMS, such as Oracle, can be found in the following book: [10].

Very early work on temporal databases can be found in [11–14] and [15].

References

1. C. Adamson, *Star Schema The Complete Reference* (McGraw-Hill Osborne Media, 2010)
2. R. Laberge, *The Data Warehouse Mentor: Practical Data Warehouse and Business Intelligence Insights* (McGraw-Hill, New York, 2011)
3. M. Golfarelli, S. Rizzi, *Data Warehouse Design: Modern Principles and Methodologies* (McGraw-Hill, New York, 2009)
4. R. Kimball, M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (Wiley, London, 2013)
5. R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, B. Becker, *The Data Warehouse Lifecycle Toolkit* (Wiley, London, 2011)
6. W.H. Inmon, *Building the Data Warehouse*. ITPro Collection (Wiley, London, 2005)
7. M. Jarke, *Fundamentals of Data Warehouses*, 2nd edn. (Springer, Berlin, 2003)
8. A. Vaisman, E. Zimányi, *Data Warehouse Systems: Design and Implementation*. Data-Centric Systems and Applications (Springer, Berlin, 2014)
9. E. Malinowski, E. Zimányi, *Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications*. Data-Centric Systems and Applications (Springer, Berlin, 2008)
10. J.W. Rahayu, D. Taniar, E. Pardede, *Object-oriented Oracle*. Solutions For IT Professionals (IRM Press, 2006)
11. R.T. Snodgrass, I. Ahn, Temporal databases. Computer **19**(9), 35–42 (1986)
12. R.T. Snodgrass, Temporal databases, in *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, International Conference GIS—From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning, Pisa, Italy, September 21–23, 1992, Proceedings*, ed. by A.U. Frank, I. Campari, U. Formentini. Lecture Notes in Computer Science, vol. 639 (Springer, Berlin, 1992), pp. 22–64
13. C.S. Jensen, R.T. Snodgrass, Temporal data management. IEEE Trans. Knowl. Data Eng. **11**(1), 36–44 (1999)
14. R.T. Snodgrass, I. Ahn, A taxonomy of time in databases, in *Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data, Austin, Texas, USA, May 28–31, 1985*, ed. by S.B. Navathe (ACM Press, New York, 1985), pp. 236–246
15. R.T. Snodgrass, *Developing Time-Oriented Database Applications in SQL* (Morgan Kaufmann, Los Altos, 1999)