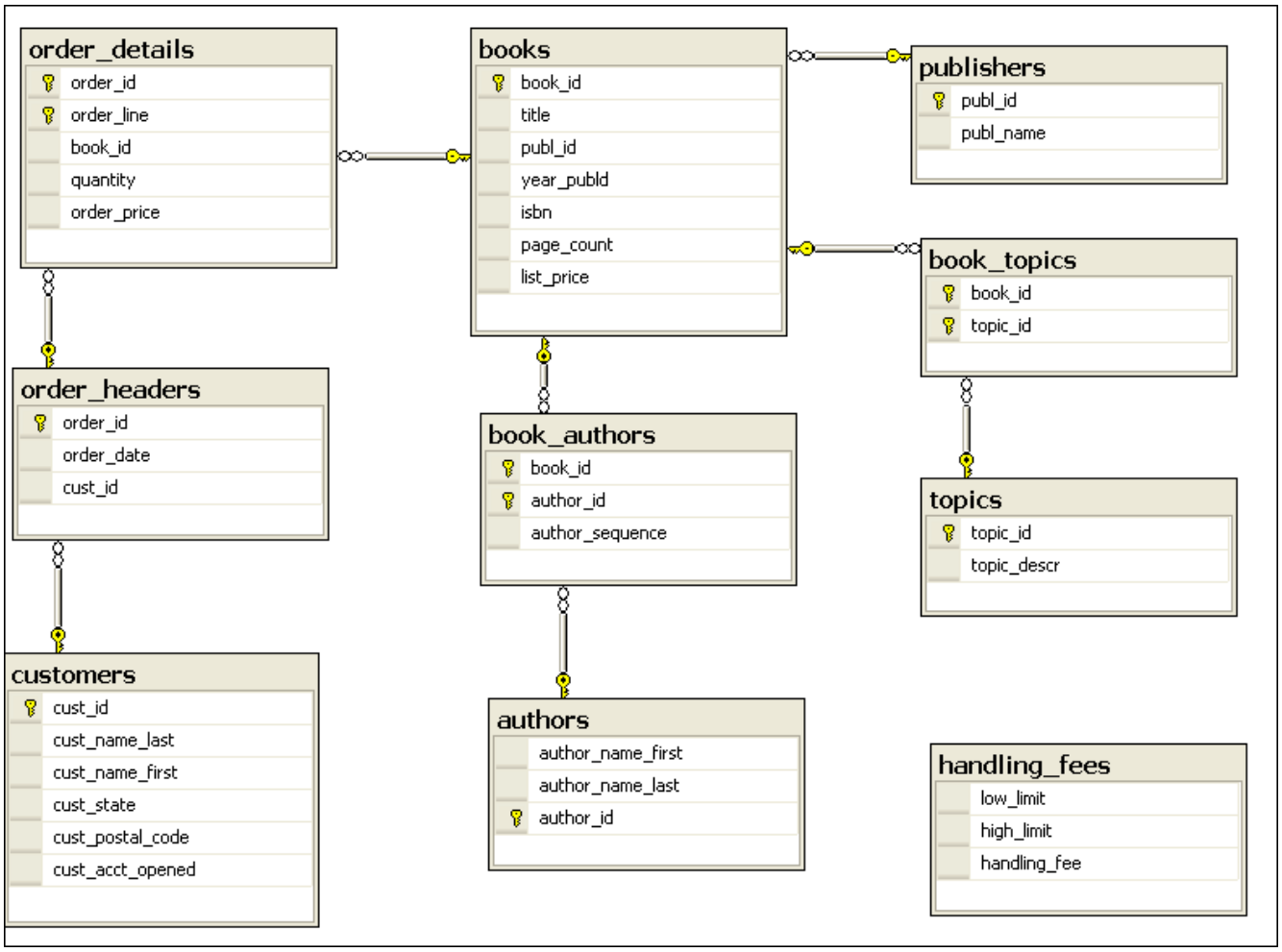
The books database is used to store data for a small bookstore.

These are the tables for the books database. Try reading through the create table statements to see how the tables are related to each other. You could experiment with some select statements and try a few joins. Experimenting is a good way to get a feel for these tables.



The create script also contains the definition of a view we will use in some assignments.

Some general rules regarding this database and the assignments.

- When an assignment asks to find a book on a certain topic such as a database book, use the topic id attribute. This attribute is found in both the topics table and the book_topics table. If the tasks asks you to find Database books, you use the book_topics table and filter for the topic_id DB. You do *not* use the topics table and filter on the topic_descr attribute and you do *not* filter the book title. The topic_id values are defined values and a list of the allowed topic_id values and the topic descriptions is in the topics table.
- You use the topics table only if the tasks asks you to display the topic description or specifically says to filter on that attribute.
- Some assignments may ask you to use **book categories**. A book category is a defined sets of topic id values.; use these lists for those tasks. Some topics are not included in a category; some topics are in more than one category. Note that you are *not* allowed to make up your own lists for these categories. These category terms are not stored in the tables.

Category	Topic id
Science	SCI
Programming	PGM, VB, NET, ADO
Database Systems	DB, SQL, SSRV, MySQL, ORA, ADO
Literature	FCT, POE
Data Storage Techniques	NOSQL, XML, DB

- Assignments will ask you to display various aspects of books and book orders. These are some general definitions that will not be repeated in each assignment.
  - Books are identified by the book_id attribute; orders are identified by the order_id attribute; customers are identified by the cust_id attribute- etc.
  - The price at which the book is sold or ordered is the order_price; it is not the list_price
  - The order_price in the books table is the price for 1 copy of the book. If the customer orders more than one copy; they have to pay for each copy. If the order_ price is 12.00 and the customer orders 4 copies, the extended cost (extended price) is 48.00.
  - If the query asks for the total number of copies of a book ordered, that is the sum of the quantity attributes.
  - If the query asks for the total sales amount of a book ordered that is the sum of the extended cost values.
  - If the query asks for the total sales amount for an order that is the sum of the extended cost values for all of the books for that order.
  - If a query asks how many orders we have, that refers to the numbers of orders, not the quantity of books. An order with one detail row with a quantity of 500 books is 1 order.
  -
  - The term 'current month' refers to the month and year in which the query is run. it is derived from the system date and is not hard coded.
  - The term 'previous month' refers to the entire month before the current month. The term 'next month' refers to the entire month after the current month.
  - For example, if you run your query in December 2013, the current month is the entire month of December 2013 (2013-12-01 through 2013-12-31). The previous month is the entire month of November 2013 (2013-11-01 through 2013-11-30). The following month is the entire month of January 2014 (2014-01-01 through 2014-01-31).These values are all derived for the system date and are not hard coded. However if the tasks asks for the first month in the current year, the first month is always January; you derive the current year.

We have a books table containing basic data about the books we carry. The tables each have a primary key, except for the handling_fees table. You should read the create table statements to find how these rules are described in SQL.

Books table

We have several attributes for books. The primary key is the book_id. The title is required. The publisher id is nullable. This means we could carry books that are not published by an established publisher. The year published is a required field and we do not carry any book published before 1850. The ISBN is nullable- some books might not have an ISBN:

We have a list_price; a book can be free; this is a nullable field which means we could enter a book for which we do not know the list price and we could enter a book with a list_price of 0. The page count has to be greater than or equal to0 and is also nullable.

A book is associated with only one publisher.

Publishers table:

We are storing only the publisher name. This is used as a lookup table so that we can store the shorter publ_id values in the books table. The publisher name is required. We could also store more information about publishers in the table- such as mailing address, phone numbers for our contact person at the publisher.

Categories of Books:

Books can be classified under multiple categories - so a book that compared Oracle SQL to T-SQL could be classified under the categories SQL, ORA and SSRV.

Topics table:

In this table we have the short topic_id field and the longer descriptive field. This is used as a lookup table so that we can store the shorter topic_id in the books table. The descriptive field is unique and not null; we do not want to have two different topics with the same descriptive name

Book Topics table:

Since a book can have multiple topics and a topic can be used for multiple books, we use a junction table bookTopics to handle this. This table has only two columns- one to identify the book and the other to indentify the topic. These columns are fk to the books and topics table. Together they make up the pk. This means that a book which has one topic would have one row in the bookTopics table while a book which has three topics would have three rows in the bookTopics table. It is possible that a book has no rows in the bookTopics table if it does not fit in one of our predefined topic area. And we can have set up topic areas and not have any books classified as that topic.

This table is said to be "all-keys" since all of its columns are FK to other tables and make up the PK.

A book could have multiple authors and an author could have written multiple books.

Authors table:

We are storing the authors first and last name.

Book authors table

This is similar in nature to the bookTopics table since it joins the books and authors tables which have a M:N relationship. It also includes a column, author sequence, which indicates the order of the listing of the authors of the book. Authors care about the order in which their name is listed.

These are some simplifying assumptions we are making about books. We are not keeping information about the various editions of a book or that the same book might have been published in different formats over the years

(hardcopy , ebook, books on tape) or have been taken over by different publishers. We assume each of these variations would get its own book id and we are not tracking that these rows refer to essentially the same book. And these are not rare books where each individual copy of a book would have its own row.

We are storing only a few fields for customers (mostly to keep the data table smaller). We have the customer's name, state, and postal code and the date they opened an account with us.

Customers can place orders for books and each order belongs to only one customer. An order can be for one or more books and a customer can order multiple copies of a book. This database is not concerned with the cash type sales that might occur at a bricks and mortar store.

customers, order headers, order details tables:

These are standard table structures. Read the sql for these. The attributes in the order tables are not null. If we delete a row in the order table that also deletes the related rows in the order details table. The OrderDetails table has a compound primary key.

handling_fees

This is the one table that does not have a primary key. It is a range lookup table. Suppose we are shipping an order with 10 books. We can find which range in the handling fees lookup table is matched and can determine the fee. One issue with this type of table is the need for a value for the highest volume. We have an assumption here that we will never ship more than 999 books on an order. That assumption could cause us problems.

We could set the low_limit column as a pk but that is less meaningful than the pk columns for the other tables.