While being extremely useful for every application that does paging, or just to limit result sets to reasonable sizes, this clause is not yet part of any SQL standard (up until SQL:2008). Hence, there exist a variety of possible implementations in various SQL dialects, concerning this limit clause. jOOQ chose to implement the LIMIT .. OFFSET clause as understood and supported by MySQL, H2, HSQLDB, Postgres, and SQLite. Here is an example of how to apply limits with jOOQ:

```
create.select().from(BOOK).limit(1).offset(2);
```

This will limit the result to 1 books starting with the 2nd book (starting at offset 0!). limit() is supported in all dialects, offset() in all but Sybase ASE, which has no reasonable means to emulate it. This is how jOOQ trivially emulates the above query in various SQL dialects with native OFFSET pagination support:

```
-- MySQL, H2, HSQLDB, Postgres, and SQLite
SELECT * FROM BOOK LIMIT 1 OFFSET 2

-- CUBRID supports a MySQL variant of the LIMIT .. OFFSET clause
SELECT * FROM BOOK LIMIT 2, 1

-- Derby, SQL Server 2012, Oracle 12c, the SQL:2008 standard
SELECT * FROM BOOK OFFSET 2 ROWS FETCH NEXT 1 ROWS ONLY

-- Ingres (almost the SQL:2008 standard)
SELECT * FROM BOOK OFFSET 2 FETCH FIRST 1 ROWS ONLY

-- Firebird
SELECT * FROM BOOK ROWS 2 TO 3

-- Sybase SQL Anywhere
SELECT TOP 1 ROWS START AT 3 * FROM BOOK

-- DB2 (almost the SQL:2008 standard, without OFFSET)
SELECT * FROM BOOK FETCH FIRST 1 ROWS ONLY

-- Sybase ASE, SQL Server 2008 (without OFFSET)
SELECT TOP 1 * FROM BOOK
```

Things get a little more tricky in those databases that have no native idiom for OFFSET pagination (actual queries may vary):

```
-- DB2 (with OFFSET), SQL Server 2008 (with OFFSET)
SELECT * FROM (
  SELECT BOOK.*,
    ROW_NUMBER() OVER (ORDER BY ID ASC) AS RN
  FROM BOOK
) AS X
WHERE RN > 1
AND RN <= 3

-- DB2 (with OFFSET), SQL Server 2008 (with OFFSET)
SELECT * FROM (
  SELECT DISTINCT BOOK.ID, BOOK.TITLE
    DENSE_RANK() OVER (ORDER BY ID ASC, TITLE ASC) AS RN
  FROM BOOK
) AS X
WHERE RN > 1
AND RN <= 3

-- Oracle 11g and less
SELECT *
FROM (
  SELECT b.*, ROWNUM RN
  FROM (
    SELECT *
```

```
      FROM BOOK
      ORDER BY ID ASC
    ) b
    WHERE ROWNUM <= 3
)
WHERE RN > 1
```

As you can see, jOOQ will take care of the incredibly painful ROW_NUMBER() OVER() (or ROWNUM for Oracle) filtering in subselects for you, you'll just have to write limit(1).offset(2) in any dialect.

# SQL Server's ORDER BY, TOP and subqueries

As can be seen in the above example, writing correct SQL can be quite tricky, depending on the SQL dialect. For instance, with SQL Server, you cannot have an ORDER BY clause in a subquery, unless you also have a TOP clause. This is illustrated by the fact that jOOQ renders a TOP 100 PERCENT clause for you. The same applies to the fact that ROW_NUMBER() OVER() needs an ORDER BY windowing clause, even if you don't provide one to the jOOQ query. By default, jOOQ adds ordering by the first column of your projection.