* NVARCHAR(100) -> CHARACTER VARYING(100)
* NVARCHAR(MAX) -> TEXT
* DATETIME -> TIMESTAMP WITHOUT TIME ZONE
* BIT -> BOOLEAN
* DECIMAL(18, 7) -> NUMERIC(18, 7)
* FLOAT -> DOUBLE PRECISION
* IDENTITY -> SERIAL

<https://www.devbridge.com/articles/migrating-from-mssql-to-postgresql/>

There are **no clustered indexes** in PostgreSQL. All indexes are non-clustered by default.

* Multiple SQL statements in one query must be separated with semi-colons. In Microsoft SQL Server 2014 T-SQL statement terminator is still optional, even if Microsoft highly recommends the use of a semi-colon because in future versions it will become required.
* Boolean (Bit) values in MSSQL are 0 and 1. In PostgreSQL they have to be replaced with true / false or the equivalent of “t” and “f”.
* String concatenation in MSSQL is “SELECT FirstName + LastName FROM...”. In PostgreSQL equivalent is “SELECT FirstName || LastName FROM...”.
* There is no ISNULL function in PostgreSQL. Use COALESCE instead.

**Case sensitivity.**In MSSQL case sensitivity is configurable via the database or table column collation. In PostgreSQL, however, it’s not so straight-forward and there is no way how to set database case sensitivity. There are several techniques how to make constraints case insensitive:

* Instead of LIKE constraint use ILIKE (Insensitive like)
* Use LOWER/UPPER functions for equality constraints, e.g. “… WHERE LOWER(Name) = LOWER(‘John’)”
* Use CITEXT extension. The CITEXT extension allows you to create case insensitive columns, so you can avoid things like the lower function. In fact, this extension calls the lower function automatically so that you don’t have to do it manually.

**Order By with NULLs.**  Nulls are ordered differently on MSSQL and PostgreSQL. In MSSQL nulls are ordered first and in PostgreSQL nulls are last by default. PostgreSQL does have the flexibility to choose how nulls should be ordered. The NULLS FIRST and NULLS LAST options can be used to determine whether nulls appear before or after non-null values in the sort ordering, e.g.:

[?](https://www.devbridge.com/articles/migrating-from-mssql-to-postgresql/)

|  |  |
| --- | --- |
| 1 | SELECT Id, Name FROM Foo ORDER BY Name NULLS FIRST; |

**STEP BY STEP**

* Created RDS instances with initial database schema (tables, views and functions).
* Deployed all websites with PostgreSQL changes to production. App\_Offline.htm file was included to notify users of a migration taking place and the fact that no data could be read or written at the time.
* Exported data from MSSQL to CSV files table by table using the [BCP](https://msdn.microsoft.com/en-us/library/ms162802.aspx) tool. Database sizes were relatively small – 1st shard was about 35GB in total, so export to CSV files worked well for us. BCP does have one pitfall – it doesn’t support export to CSV with UTF-8 encoding. So we had to export to CSV using UTF-16 encoding and later convert it to UTF-8.
* Changed CSV file encoding from UTF-16 to UTF-8 using [iconv](https://www.gnu.org/software/libiconv/) tool.
* Imported CSV files to corresponding tables using PSQL (PostgreSQL interactive terminal) COPY command.
* Created foreign keys and indexes. These objects were created after data migration due to better performance.
* Removed App\_Offline.htm from all websites. The next request into the application caused IIS to load the application and app-domain again, and operations continued along as normal.

<https://github.com/dalibo/sqlserver2pgsql>