

# Reading from SQL databases

# SQL (Structured Query Language)

- ▶ Pronounced *Ess Queue Ell* or *Sequel*
- ▶ The package **RODBC** is used to read SQL databases (and other database formats).
- ▶ Load required package  
> `library(RODBC)`
- ▶ Get an overview of the package: **library(help=RODBC)**

## Common functions used

Function	Description
odbcDriverConnect()	Open a connection to an ODBC database
sqlQuery()	Submit a query to an ODBC database and return the results
sqlTables()	List Tables on an ODBC Connection
sqlFetch()	Read a table from an ODBC database into a data frame
sqlColumns()	Query Column Structure in ODBC Tables
close(connection)	Close the connection

## Connecting to SQL server databases

- ▶ A remote database to play with:

Server Name: msedxeus.database.windows.net

Database: DAT209x01

Login: RLogin

Password: P@ssw0rd

- ▶ Connect to database

- ▶ Create connection string

```
> connStr <- paste(  
+   "Server=msedxeus.database.windows.net",  
+   "Database=DAT209x01",  
+   "uid=Rlogin",  
+   "pwd=P@ssw0rd",  
+   "Driver={SQL Server}",  
+   sep=";"  
+ )
```

- ▶ Connect

```
> conn <- odbcDriverConnect(connStr)
```

## Connecting to a local SQL Database on your harddisk:

- ▶ Replace server name with the SQL server name on the local machine;
- ▶ With the default SQL installation, this is equal to the **name of the local machine**:

```
>connStr <- paste(  
+   "Server=My_Machine",  
+   "Database=DAT209x01",  
+   "uid=Rlogin",  
+   "pwd=P@ssw0rd",  
+   "Driver={SQL Server}",  
+   sep=";"  
+ )
```

```
>conn <- odbcDriverConnect(connStr)
```

## Other operating systems

### Instructions for Ubuntu Linux 14.04:

- ▶ Install the required drivers and ODBC package via the *commandline*:

```
sudo apt-get install r-cran-rodbc unixodbc-bin unixodbc odbcinst freetds-bin tdsodbc  
sudo odbcinst -i -d -f /usr/share/tdsodbc/odbcinst.ini
```

- ▶ From **R** use a modified connection string:

```
> connStr <- paste(  
+   "Server=msedxeus.database.windows.net",  
+   "Database=DAT209x01",  
+   "uid=Rlogin",  
+   "pwd=P@ssw0rd",  
+   "Driver=FreeTDS",  
+   "TDS_Version=8.0",  
+   "Port=1433",  
+   sep=";"  
+ )
```

- ▶ Connect

```
> conn <- odbcDriverConnect(connStr)
```

## A first query

- ▶ Use **sqlTables** to list all tables in the database.
- ▶ Submit query from **R** :

```
> tab <- sqlTables(conn)
> head(tab)
```

	TABLE_CAT	TABLE_SCHEM	TABLE_NAME	TABLE_TYPE	REMARKS
1	DAT209x01	bi	date	TABLE	<NA>
2	DAT209x01	bi	geo	TABLE	<NA>
3	DAT209x01	bi	manufacturer	TABLE	<NA>
4	DAT209x01	bi	product	TABLE	<NA>
5	DAT209x01	bi	salesFact	TABLE	<NA>
6	DAT209x01	bi	sentiment	TABLE	<NA>

## Getting a table

- ▶ Use **sqlFetch** to get a table from the database.
- ▶ Get the table 'manufacturer' from SCHEMA 'bi':

```
> mf <- sqlFetch(conn, "bi.manufacturer")  
> mf
```

	ManufacturerID	Manufacturer
1	1	Abbas
2	2	Aliqui
3	3	Barba
4	4	Currus
5	5	Fama
6	6	Leo
7	7	VanArsdel
8	8	Natura
9	9	Palma
10	10	Pirum
11	11	Pomum
12	12	Quibus
13	13	Salvus
14	14	Victoria



## Submit real SQL

- ▶ Use **sqlQuery** for more advanced queries.

- ▶ SQL syntax example:

```
SELECT Manufacturer FROM bi.manufacturer WHERE ManufacturerID < 10
```

- ▶ Submit query to **R** :

```
> query <- "  
+   SELECT Manufacturer  
+   FROM   bi.manufacturer  
+   WHERE  ManufacturerID < 10  
+   "  
> sqlQuery(conn, query)
```

```
Manufacturer  
1      Abbas  
2     Aliqui  
3     Barba  
4     Currus  
5      Fama  
6      Leo  
7 VanArsdel  
8     Natura  
9      Palma
```

## Large tables

- ▶ A common use case: Fetching entire table is infeasible
- ▶ Get some info without complete fetch:

- ▶ Count number of rows in table 'salesFact':

```
> sqlQuery(conn, "SELECT COUNT(*) FROM bi.salesFact")
```

```
1 10439386
```

- ▶ Show some column info:

```
> sqlColumns(conn, "bi.salesFact")[c("COLUMN_NAME", "TYPE_NAME")]
```

	COLUMN_NAME	TYPE_NAME
1	ProductID	bigint
2	Date	date
3	Zip	varchar
4	Units	int
5	Revenue	numeric

- ▶ Show first two rows:

```
> sqlQuery(conn, "SELECT TOP 2 * FROM bi.salesFact")
```

	ProductID	Date	Zip	Units	Revenue
1	1	2012-10-20	30116	1	412.125
2	1	2012-10-10	90630	1	412.125

- ▶ Fetch a subset

```
> df <- sqlQuery(conn, "SELECT * FROM bi.salesFact WHERE Zip='30116'")  
> dim(df)
```

```
[1] 1000    5
```

## Data types

- ▶ Classes of variables on the **R** side:

```
> sapply(df, class)
```

```
ProductID      Date      Zip      Units      Revenue  
"integer"  "factor" "integer" "integer" "numeric"
```

- ▶ Recall that the variable 'Zip' was stored as the SQL specific type 'varchar'. When read into **R** it became an integer.
- ▶ This type conversion is similar to the behaviour of **read.table()**. To avoid conversion you may want to pass **as.is=TRUE** to your SQL query.
- ▶ For the present database the following conversion rules apply:

SQL type	R type	R type (as.is=TRUE)
smallint	integer	integer
int	integer	integer
bigint	integer	character
numeric	numeric	character
date	factor	character
varchar	integer or factor	character
datetime	POSIXct	character

## SQL summary statistics

- Some useful SQL summary statistics:

SQL command	R equivalent
SUM(x)	sum(x)
AVG(x)	mean(x)
STDEV(x)	sd(x)
COUNT(x)	length(x)

- Example

```
> df <- sqlQuery(conn,  
+   "SELECT      AVG(Revenue), STDEV(Revenue), Zip  
+   FROM        bi.salesFact  
+   GROUP BY    Zip"  
+ )  
> colnames(df) <- c("AVG(Revenue)", "STDEV(Revenue)", "Zip")
```

## End of session

- ▶ Close the connection  
    > `close(conn)`