


Database Design with MySQL Workbench

A Round Trip Software Engineering Case

Jerzy Letkowski

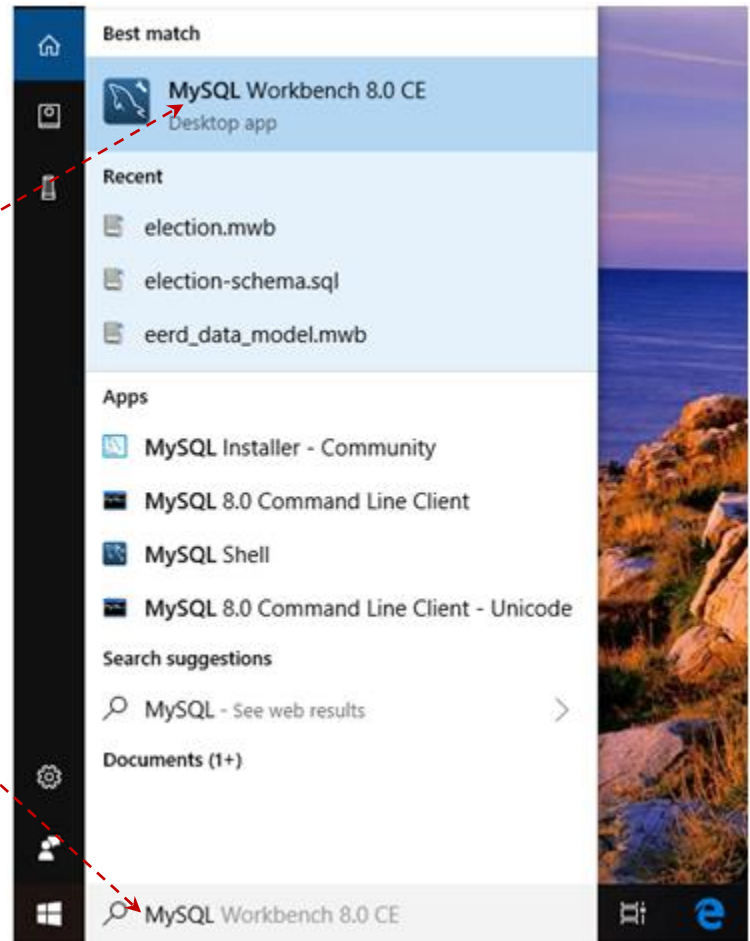
Prerequisites - Downloads

- MySQL Server and MySQL Workbench are expected to be installed. The setup program(s) can be downloaded from the MySQL Website.
- This instruction uses the server and workbench installed using the July 5, 2018 installer, downloaded from <http://www.mysql.com/>:
 - Tab: Downloads (GA)
 - [Scroll Down] MySQL Community Edition (GPL):
Community (GPL) Downloads »
 - MySQL Community Server (GPL)
 - Windows (x86, 32, 64-bit), MySQL Installer MSI : Go to Download Page
 - [Scroll Down] Windows (x86, 32-bit), MSI Installer 8.0.11 230.0M 
 - No thanks, just start my download.
- The **mysql-installer-community-8.0.11.0.msi** version was available, at the time of writing this instruction.
- Other major operating systems are also supported, including Mac OS X and Linux.

Prerequisites - Installation

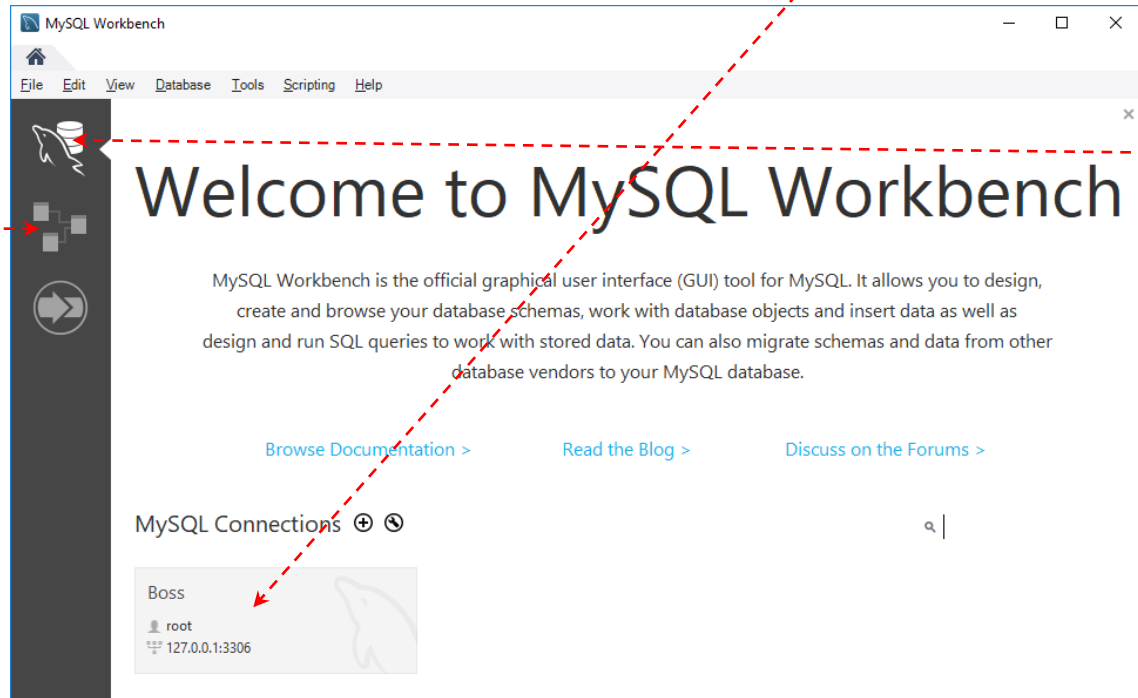
- Using the standard Window installer is pretty straight forward. Just follow the default developer's options.
- It is very important that you do not forget the **root** password.
- When done, follow the remaining slides to learn more about database design and implementation, using MySQL Workbench and Community Server.


You can **start MySQL Workbench** by using Windows search for **MySQL** and picking it from the list.

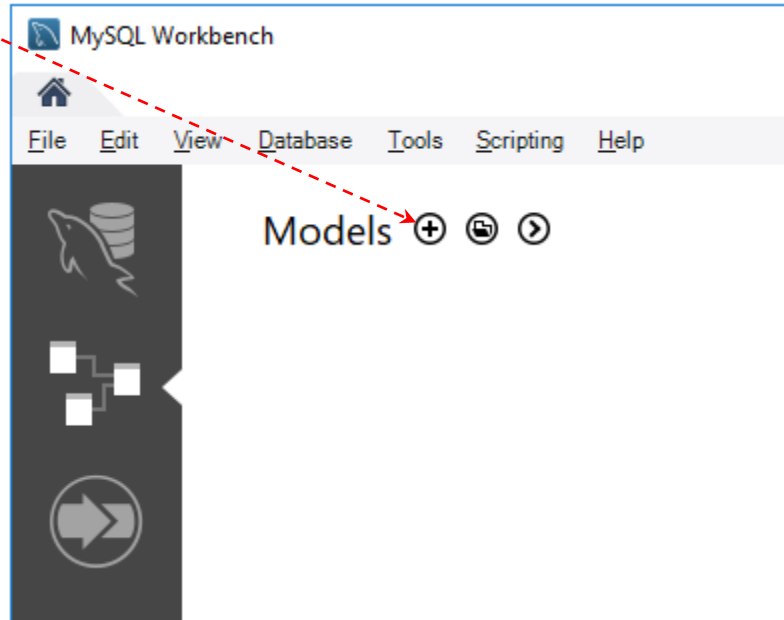


The front page of the **MySQL Workbench** program **shows** the initial (root) **connection** , the **Models**' interface.

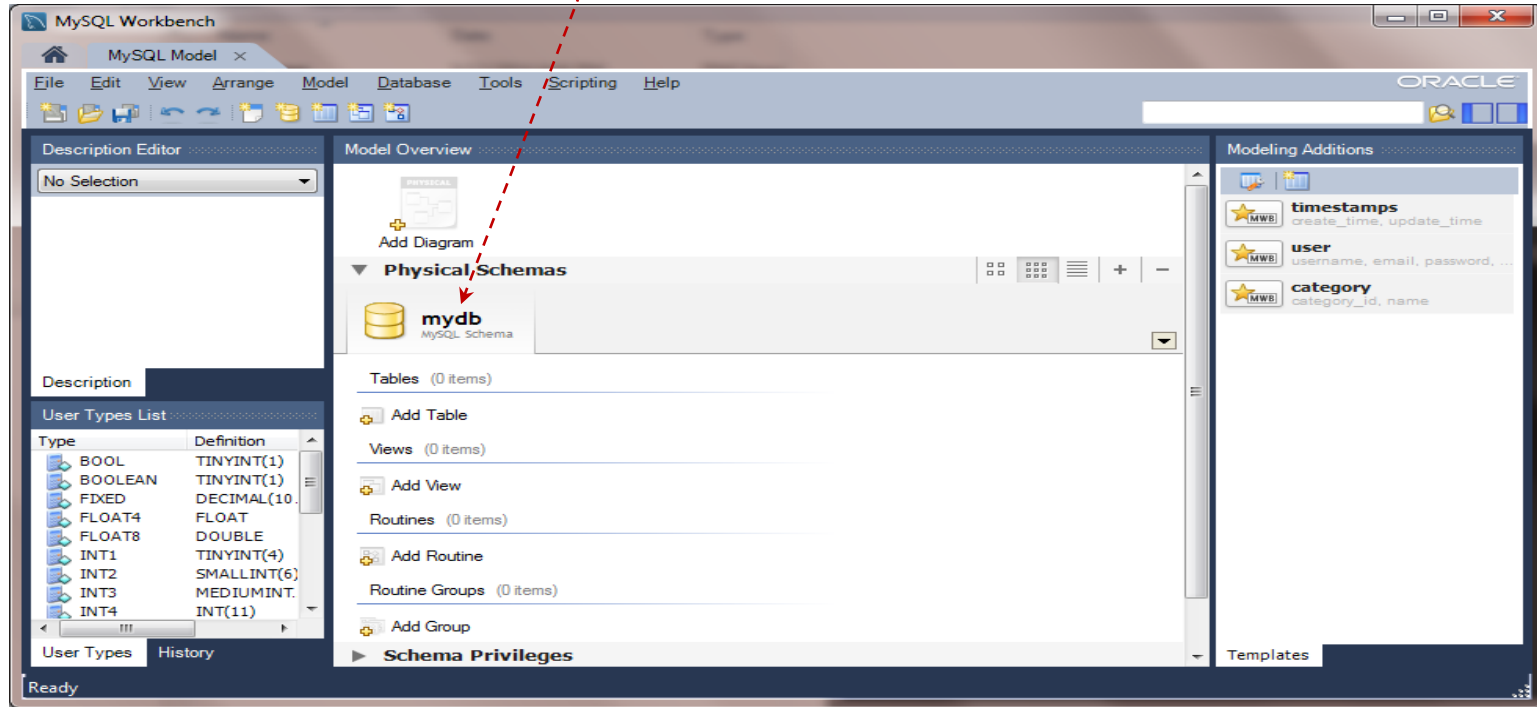
Click the **Models** icon to open the data modeling facility.



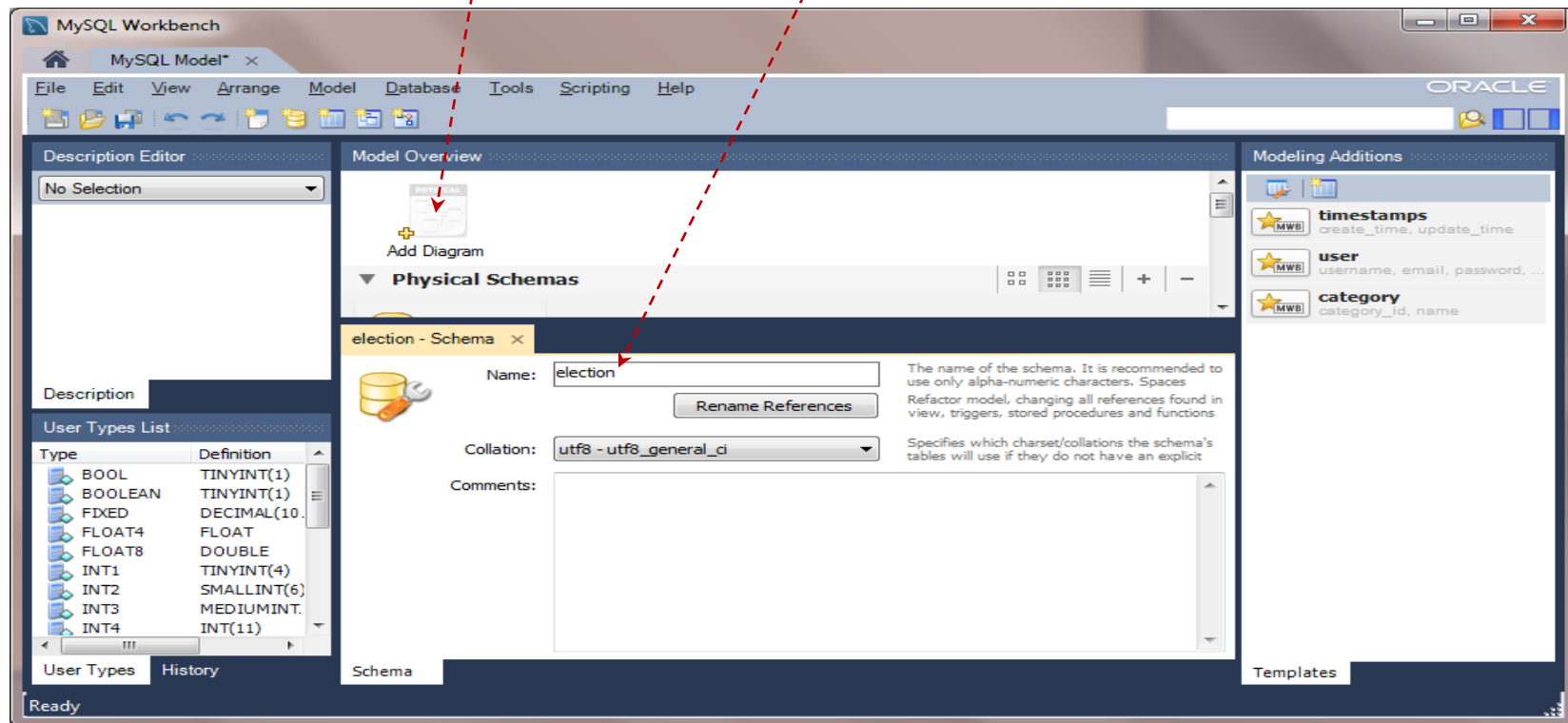
Click the **Models**  icon in order to create a new relational database model.



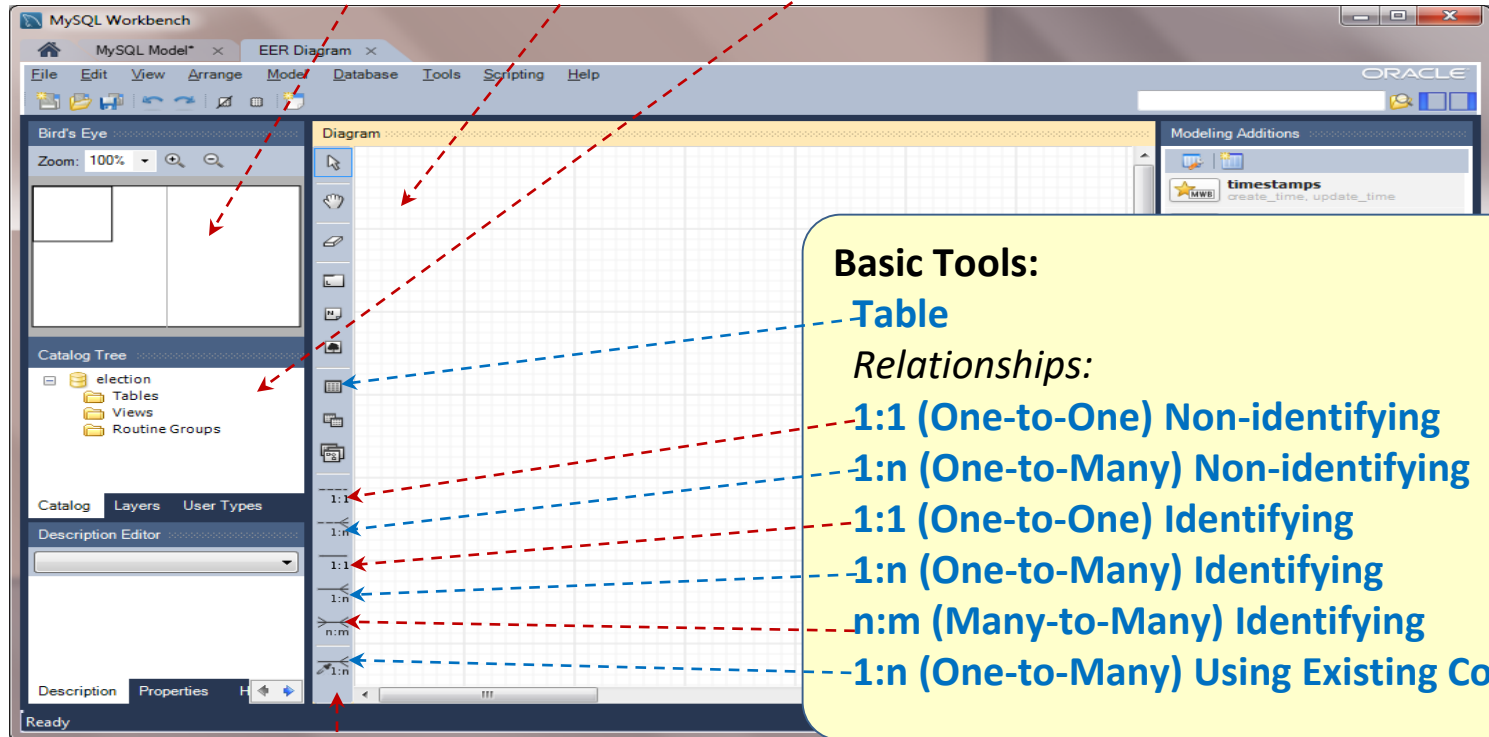
Double-click the **default schema (database) name** tab (**mydb**) in order to create a new database name.



Type name **election**,
and **double-click** the **Add Diagram** icon.



MySQL Workbench **opens** the **Diagram canvas** along with other useful panels, including the **Birds Eye view panel**, and **Catalog Tree panel**.



Basic Tools:

Table

Relationships:

1:1 (One-to-One) Non-identifying

1:n (One-to-Many) Non-identifying

1:1 (One-to-One) Identifying

1:n (One-to-Many) Identifying

n:m (Many-to-Many) Identifying

1:n (One-to-Many) Using Existing Columns

The **Toolbar** contains graphical tools that are used to paint **Enhanced Entity Relationship Diagrams (EERDs)**.

Problem Statement

A Database for an Online Voting System

Our mission is to design a database for an on-line election system that will be utilized to conduct election of new leaders of a non-profit organization. The organization has members some of which hold leadership positions (president, vice president, treasurer, newsletter editor, annual meeting coordinator, secretary, etc.). Some of the members have been nominated to run for the positions. Assume that they have already accepted their nominations, thus becoming official candidates for the available positions. The organization's statute states that each candidate may only run for one position and each member may cast no more than one vote for each of the positions. The database should facilitate the voting process and record all votes assigned to the candidates but it should not tell which member has voted for which candidate.

Entities

From the problem definition one could identify the following base entities:

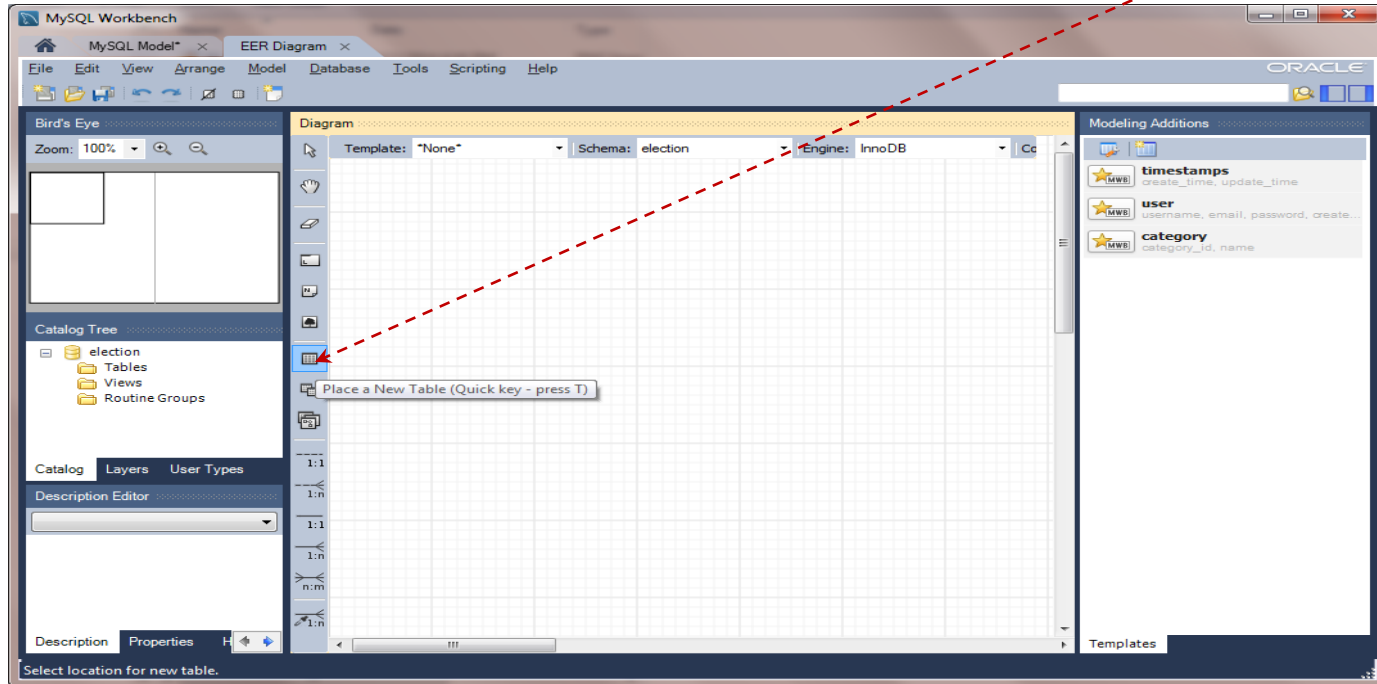
- **Member**: a list (set) of members of the organization, some of whom are candidates;
- **Office**: a list of positions ('president', 'treasurer', 'editor', etc.);

Other entities result from associations between the base entities.

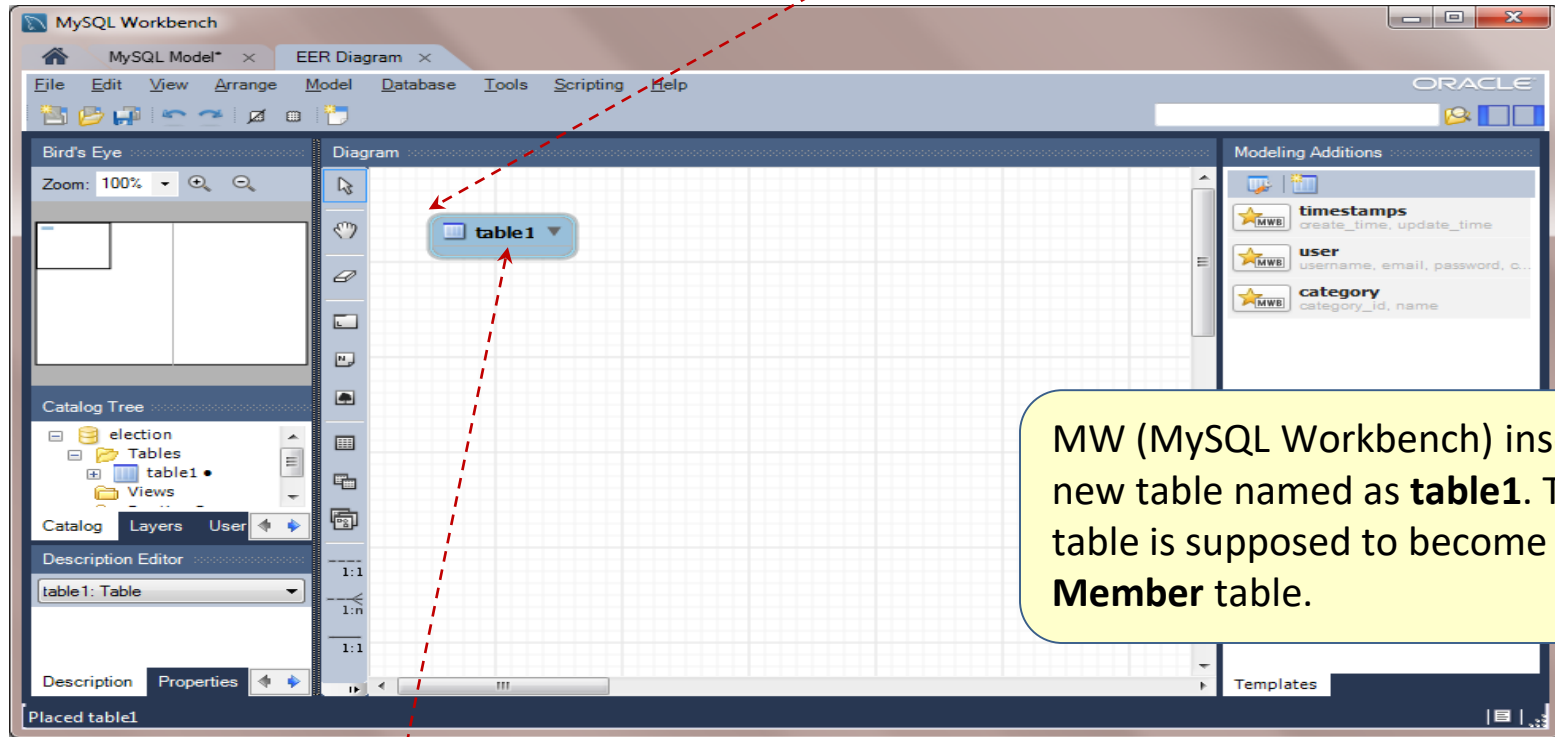
Our mission is to design a database for an on-line election system that will be utilized to conduct election of new leaders of a non-profit organization. The organization has **members** some of which hold **leadership positions** (president, vice president, treasurer, newsletter editor, annual meeting coordinator, secretary, etc.). Some of the **members have been nominated** to run for the positions. Assume that they have already accepted their nominations, thus becoming official **candidates** for the available **positions**. The organization's statute states that each **candidate** may only run for one **position (office)** and each **member** may cast no more than one **vote** for each of the **positions**. The database should facilitate the voting process and record all **votes** assigned to the candidates but it should not tell which member has voted for which candidate.

Entity Member

To add entity (table) **Member** to the data model (EER diagram) *click* the **New Table** icon.



Next **click** anywhere on the **Diagram canvas**.



MW (MySQL Workbench) inserts the new table named as **table1**. This table is supposed to become the **Member** table.

In order to change the name of the table and add its attributes, **double-click** table **table1**.

Type name **Member**.

Add column name **mid**
as (Datatype) **INT**.

Set column **mid** as the
Primary Key (PK).

Add other columns:
firstName as **VARCHAR(45)**,
lastName as **VARCHAR(90)**,
pass as **CHAR(12)**,
email as **VARCHAR(90)**.

When done, **close** the **Member - Table** definition panel.

Notice that the primary key is automatically set as **Not NULL (NN)**. Data types **INT**, **VARCHAR** and **CHAR** stand for an integer, a variable-length string (text) and a fixed-length string, respectively.

Some individual have hard time of visualizing abstract entities. The attached Excel macro-workbook, [**election.xlsm**](#), contains different depictions of the entities.

Each worksheet shows three representations for each of the election-database entities: **ERD (UML)**, **SQL** and **Excel Table**.

The **election.xlsm** workbook also contains a macro-function,

```
sqlInsert(table_name_abs_ref,data_type_abs_ref,record_ref),
```

that transforms rows of the tables into their **SQL-Insert** statements. In order to facilitate this function, a row with "high-level" data types is added (right above the column names), where **n** stands for numeric and **c**—nonnumeric.

Note that the last part of this instruction shows how to transform the **ERD** view into the **SQL** statements. Typically, when designing a new database, the **SQL** representations are not available until the **ERD** model is completed and transformed to it. At this point, **SQL (Structured Query Language)** may be somewhat mysterious (it will be explored in detail later). Suffice to say, it is the official language for managing relational database systems. It is shown here for completeness.

The **Member** worksheet of the **election.xlsm** workbook shows three views of the Member entity: **UML**, **SQL Schema**, **SQL Data**, and **Excel Table**.

The **sqlInsert()** function transforms the rows of the **Excel Table** into **SQL-Insert** statements.

Member	
mid	INT
firstName	VARCHAR(45)
lastName	VARCHAR(90)
pass	CHAR(12)
email	VARCHAR(90)
Indexes	

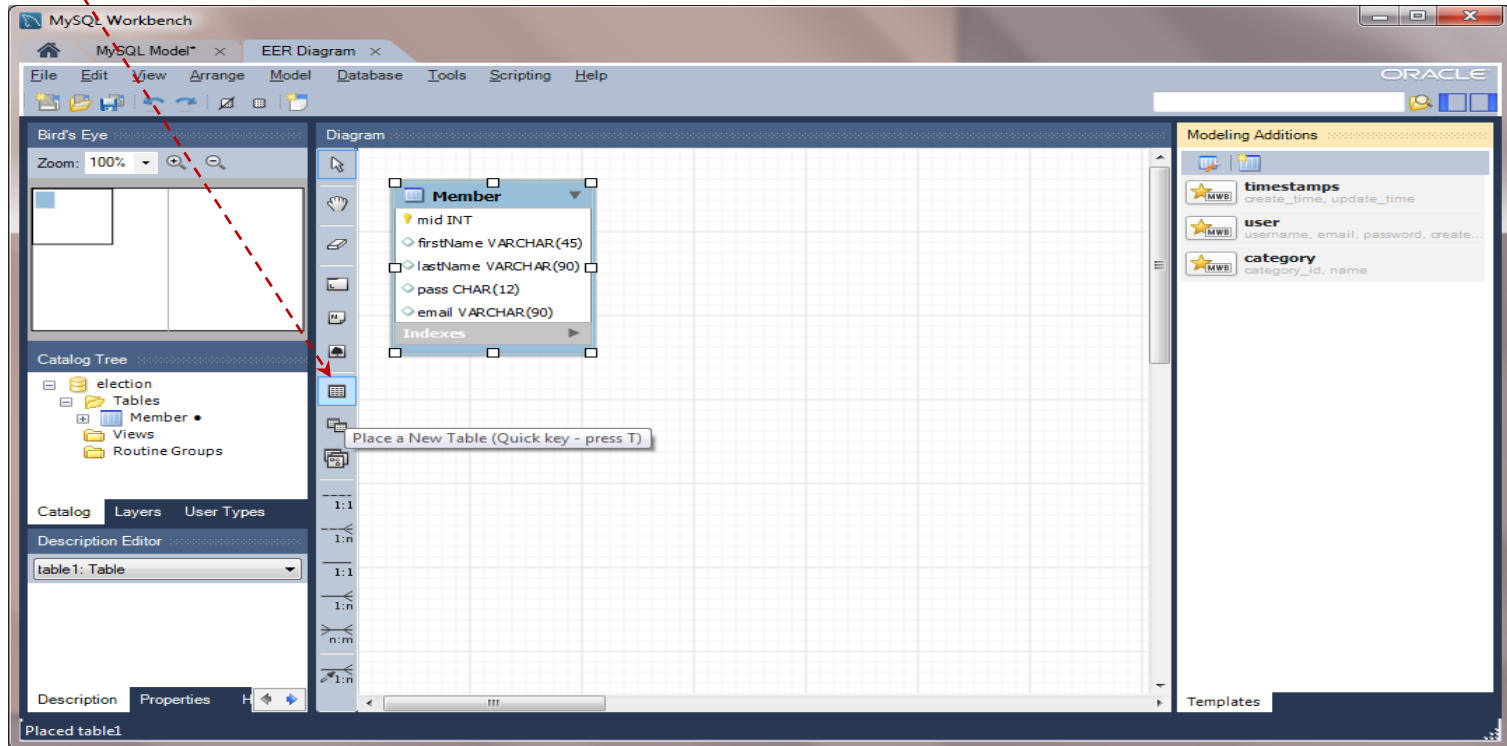
```
CREATE TABLE Member
(
    mid int primary key,
    firstName varchar(45) NOT NULL,
    lastName varchar(90) NOT NULL,
    pass char(12) NOT NULL,
    email varchar(90)
);
```

n	c	c	c	c
mid	firstName	lastName	pass	email
1	Ann	Anson	aa01119	aanson@misor.org
2	Ben	Jamin	bj11122	bjamin@misor.org
3	Cir	Cus	cc22211	ccus@misor.org
4	Don	Donski	dd00413	ddonski@misor.org
5	Eve	Lady	el98765	elady@misor.org
6	Fin	End	fe00011	fend@misor.org
7	Gin	Rum	gr12345	grum@misor.org

```
Insert into Member(mid,firstName,lastName,pass,email) Values(1,'Ann','Anson','aa01119','aanson@misor.org');
Insert into Member(mid,firstName,lastName,pass,email) Values(2,'Ben','Jamin','bj11122','bjamin@misor.org');
Insert into Member(mid,firstName,lastName,pass,email) Values(3,'Cir','Cus','cc22211','ccus@misor.org');
Insert into Member(mid,firstName,lastName,pass,email) Values(4,'Don','Donski','dd00413','ddonski@misor.org');
Insert into Member(mid,firstName,lastName,pass,email) Values(5,'Eve','Lady','el98765','elady@misor.org');
```


Entity Office

To add entity (table) Office to the data model (EER diagram) **click** the **New Table** icon.



Change the name of this table
to **Office**.

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Arrange, Model, Database, Tools, Scripting, and Help. The main workspace is divided into three panes: Bird's Eye (left), Diagram (center), and Modeling Additions (right). The Diagram pane shows an EER diagram with two tables: 'Member' and 'Office'. The 'Member' table has columns: mid INT, firstName VARCHAR(45), lastName VARCHAR(90), pass CHAR(12), and email VARCHAR(90). The 'Office' table has columns: old INT and title VARCHAR(45). The 'Office - Table' panel is open at the bottom, showing the table definition for 'Office' in the 'election' schema. The table has two columns: 'oid' (INT, PK, NN, UI) and 'title' (VARCHAR(45), NN, UI). The 'title' column is highlighted in blue. The 'Columns' tab is selected in the bottom pane.

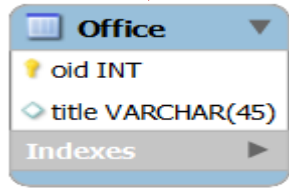
Add column `oid` as INT and PK.

Add column `title` as VARCHAR(45).

When done, **close**
the **Office – Table**
definition panel.

The **Office** worksheet of the [election.xlsm](#) workbook shows three views of each of the **Office** entity: **UML**, **SQL Schema**, **SQL Data**, and **Excel Table**.

The `sqlInsert()` function transforms the rows of the **Excel Table** into **SQL-Insert** statements.



```
CREATE TABLE Office
(
    oid int primary key,
    title varchar(50) NOT NULL
);
```

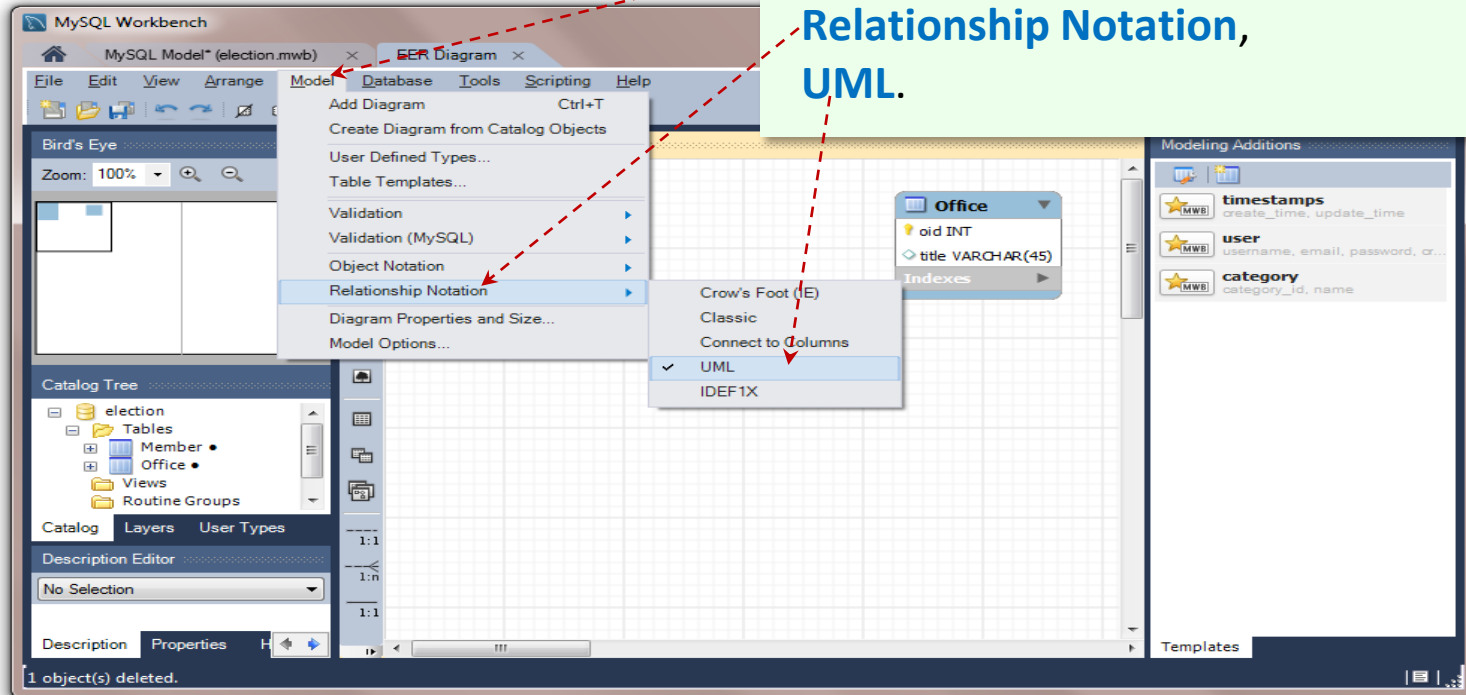
```
Insert into Office(oid,title) Values(1,'President');
Insert into Office(oid,title) Values(2,'Vice-President');
Insert into Office(oid,title) Values(3,'Treasurer');
Insert into Office(oid,title) Values(4,'Editor');
Insert into Office(oid,title) Values(5,'Web Manager');
```

Table: Office	
n	c
oid	title
1	President
2	Vice-President
3	Treasurer
4	Editor
5	Web Manager

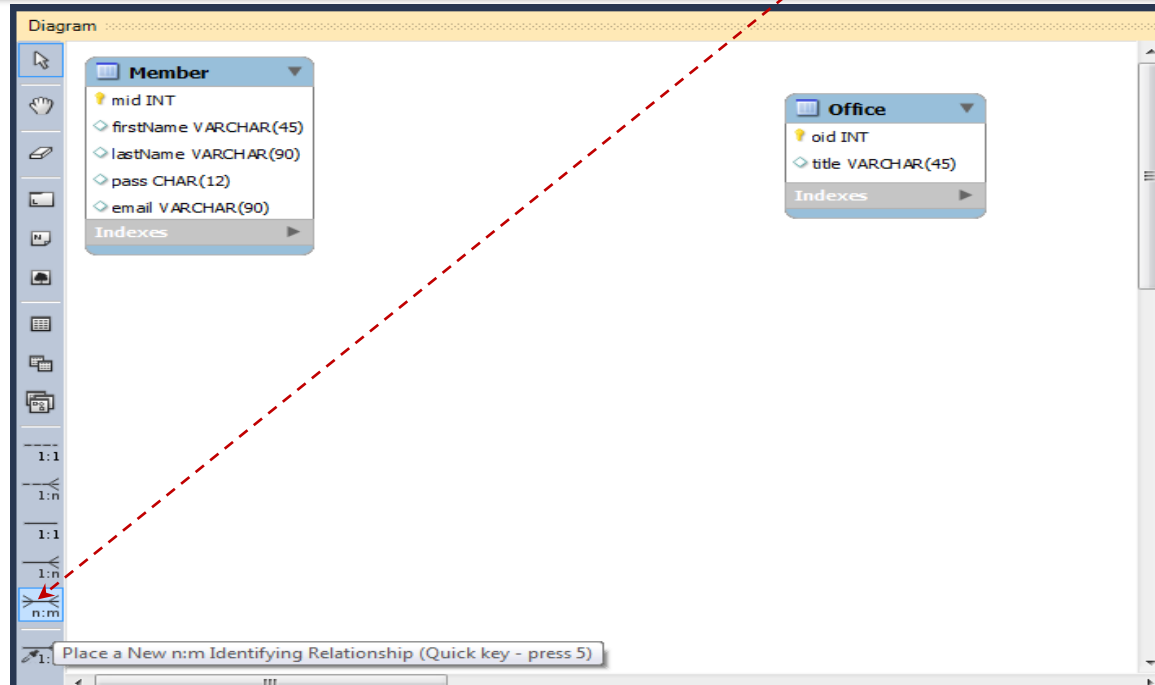
This instruction shows how to develop an **EER** diagram (**ERD**) in which relationships are modeled using the **UML** notation. Make sure that your notation is the same.

Select the following menu options:

Model
Relationship Notation,
UML.



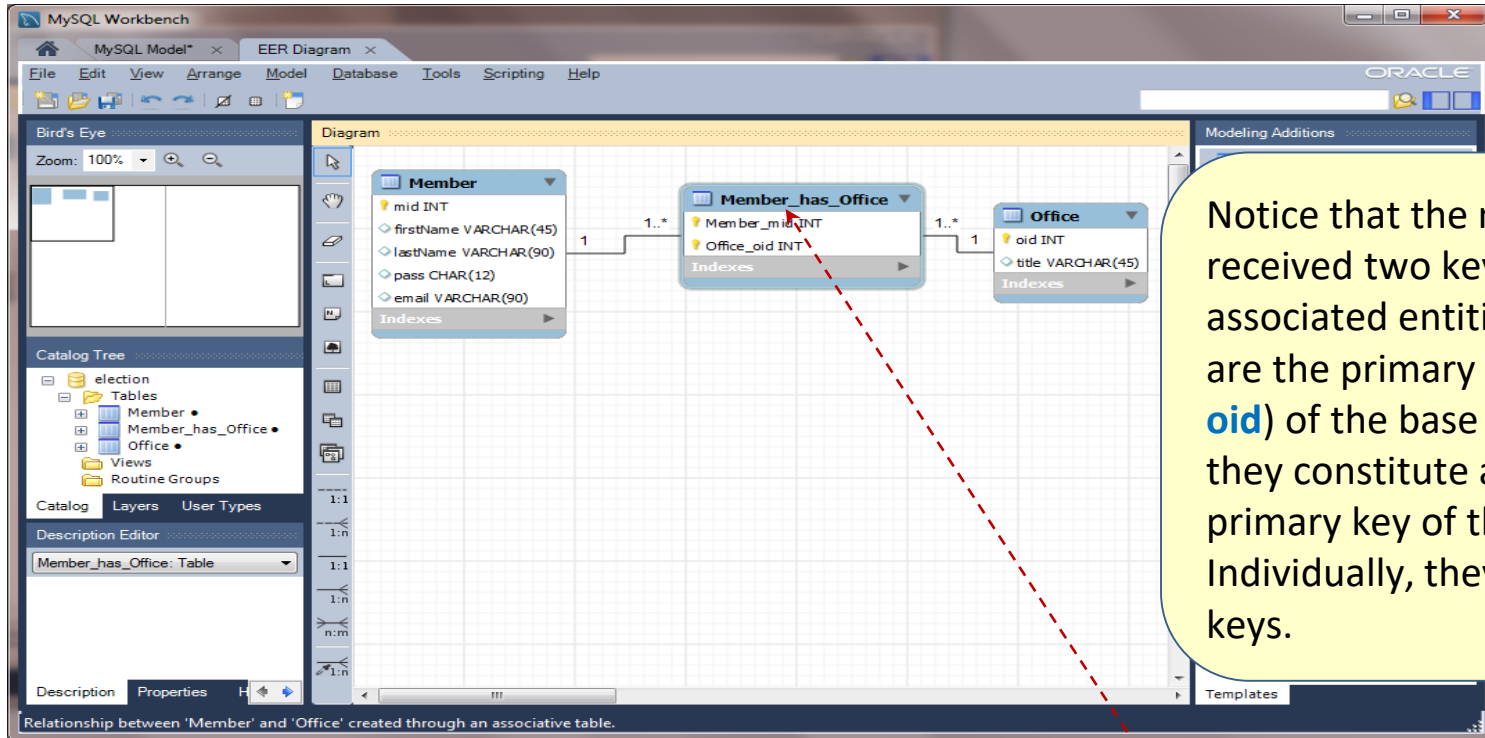
To add a relationship between entities **Member** and **Office**, *click* the **m:n Identifying Relationship** button.



Notice that **m:n** stands for **Many-to-Many**. Since a **member** (instance of entity **Member**) participates in election of many **offices** (positions) and a given **office** (instance of entity **Office**) is being elected by many members, this is a **Many-to-Many** relationship.

Note: Despite selecting the **UML** notation, the relationship tools are shown, using the **Crow-Foot** notation.

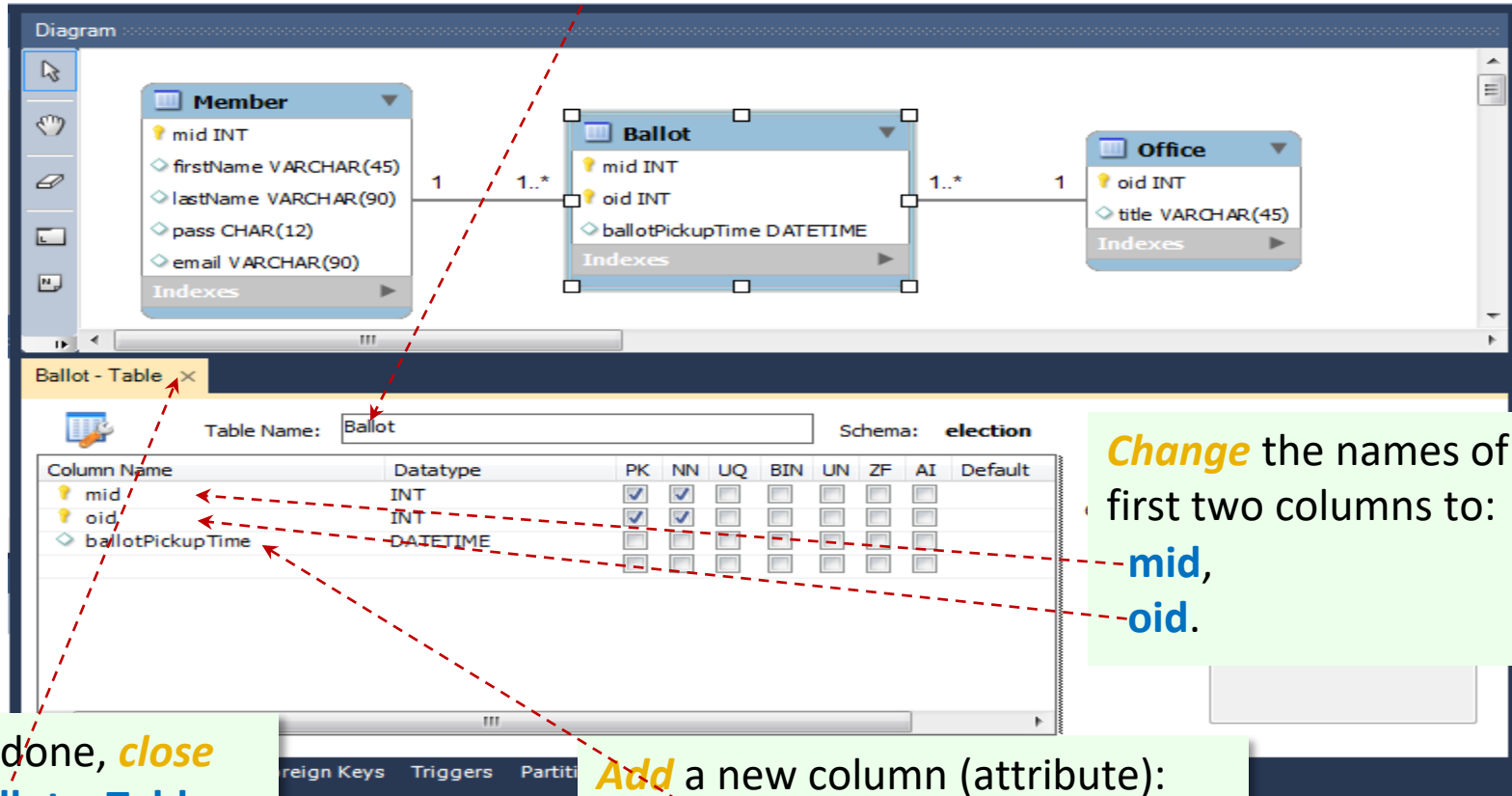
MW inserts a new [default] entity, **Member-has-Office** and sets the relationships of this entity with **Member** and **Office** to **Many-to-One**. This is a typical resolution of the **Many-to-Many** relationships as required by the relational database model.



Notice that the new entity has received two keys from the associated entities. Their sources are the primary keys (**mid** and **oid**) of the base entities. Jointly, they constitute a [composite] primary key of the new entity's. Individually, they are foreign keys.

In order to customize the **Member-has-Office** entity, *double-click* this entity.

Change the name of this entity to **Ballot**.



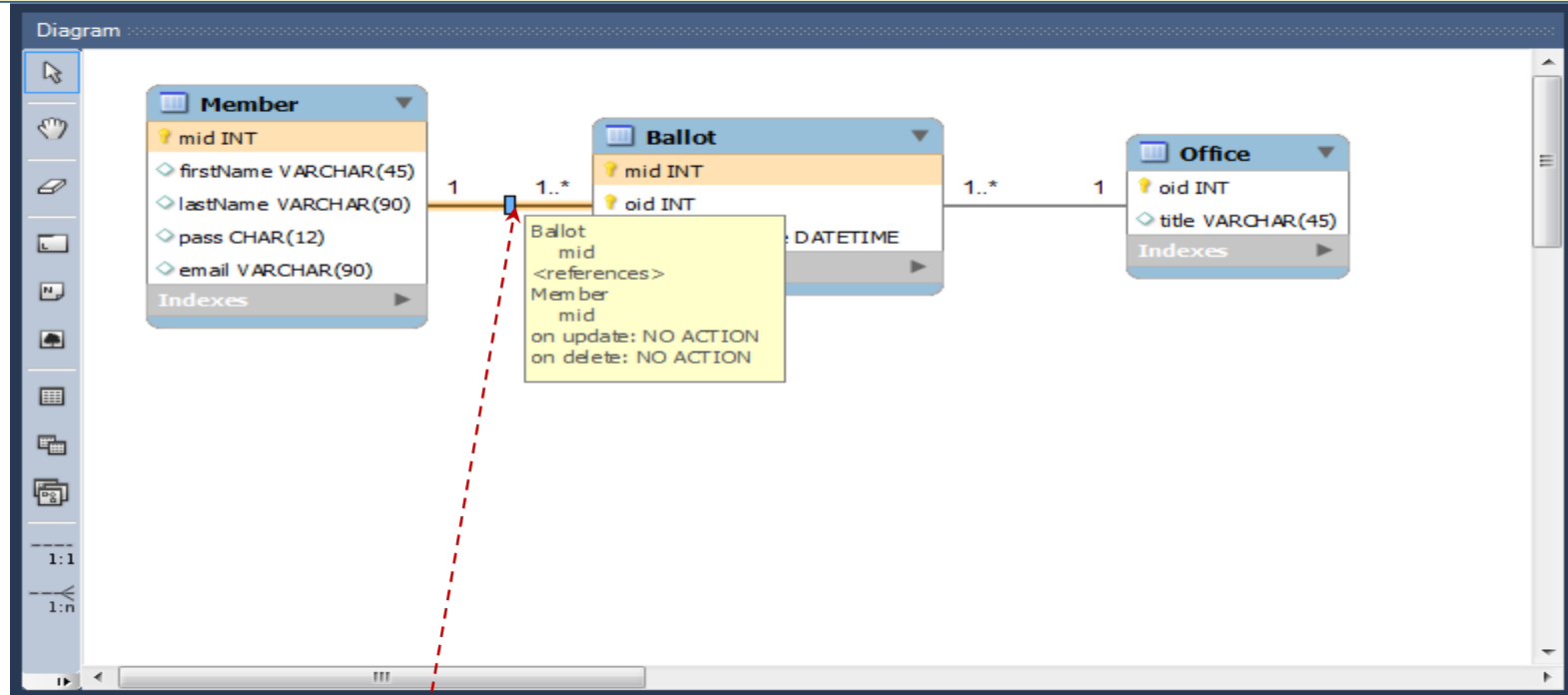
Change the names of the first two columns to:

mid,
oid.

When done, **close** the **Ballot – Table** definition panel.

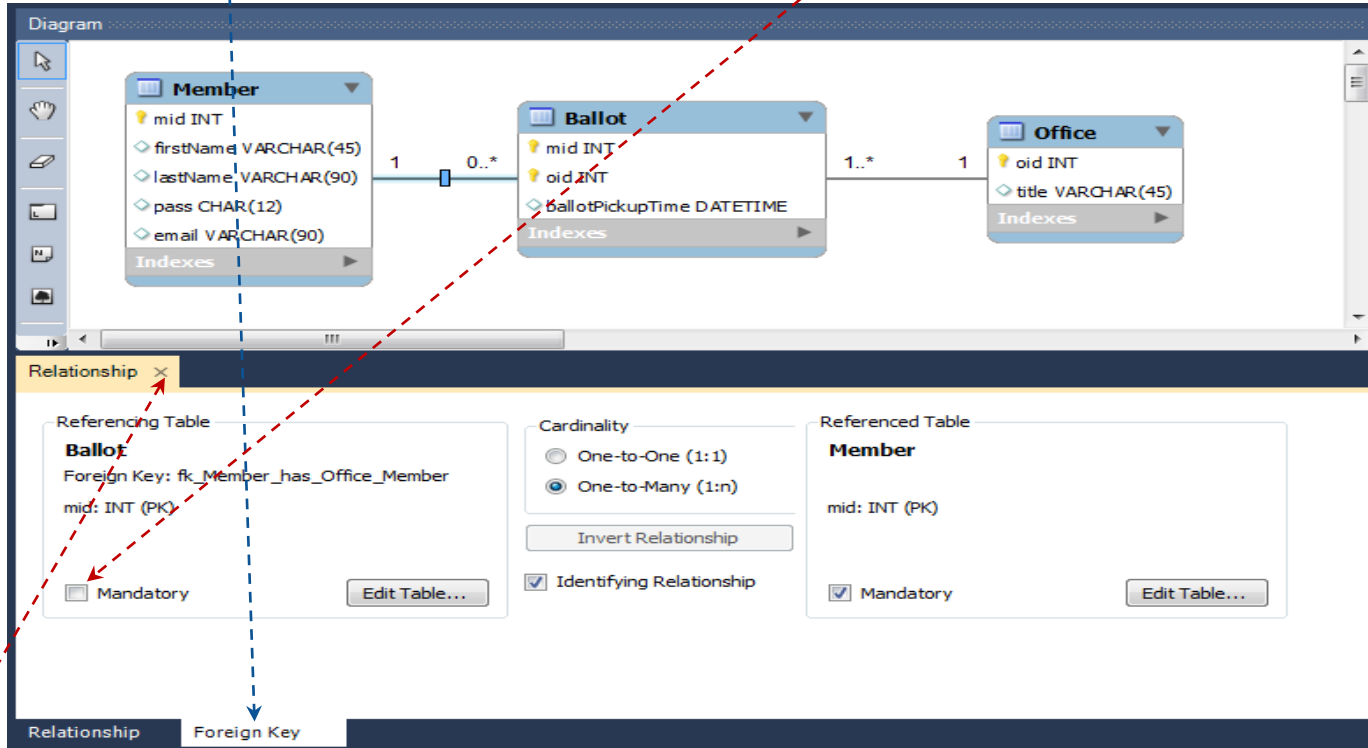
Add a new column (attribute):
ballotPickupTime as **DATETIME**.

Since theoretically there may be members who will not participate in the election of their representatives (officers), the participation of the **Ballot** entity in the relationship with entity **Member** is optional. The cardinality constraint at the **Ballot** side should be changed from many-mandatory (**1..***) to many-optional (**0..***).



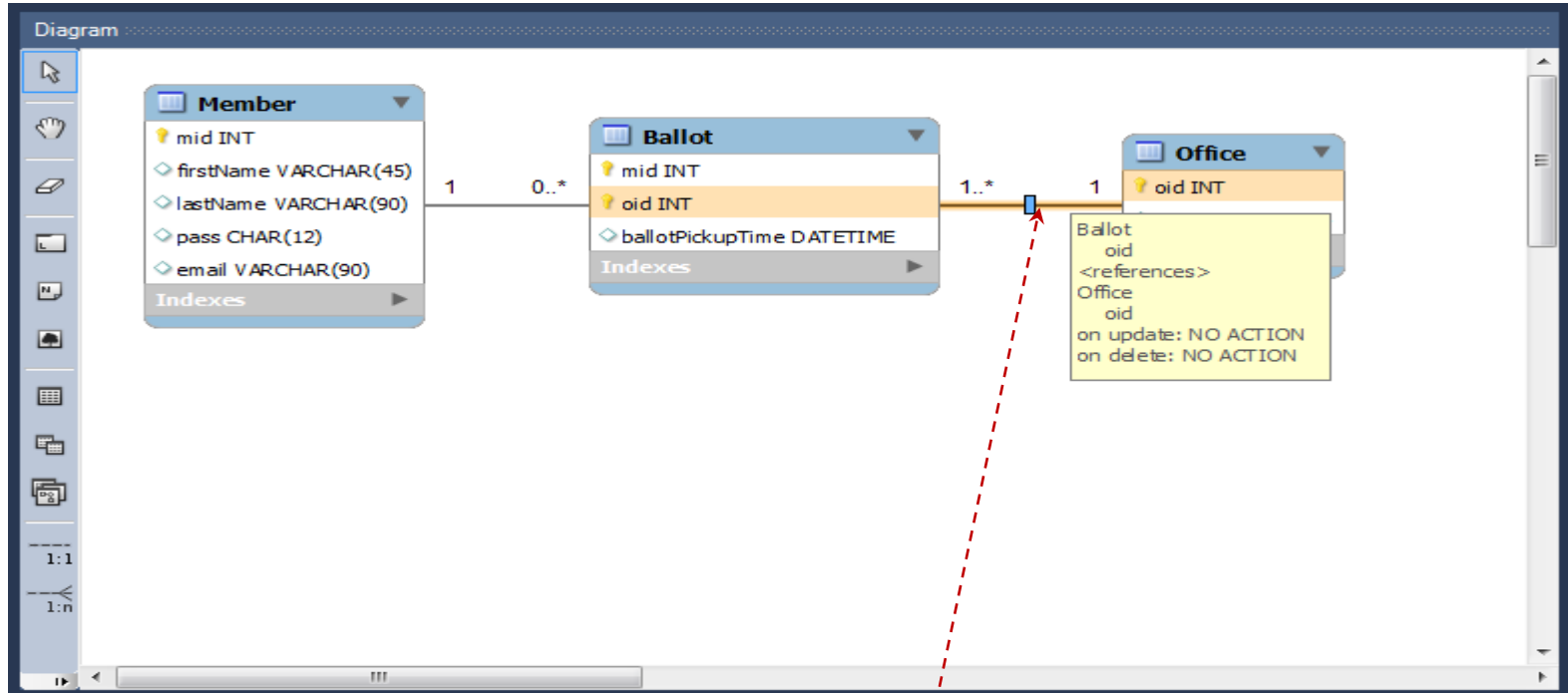
Double-click the **Member - Ballot** link.

Select the **Foreign Key** tab and **unchecked** option **Mandatory** in the **Ballot** panel.



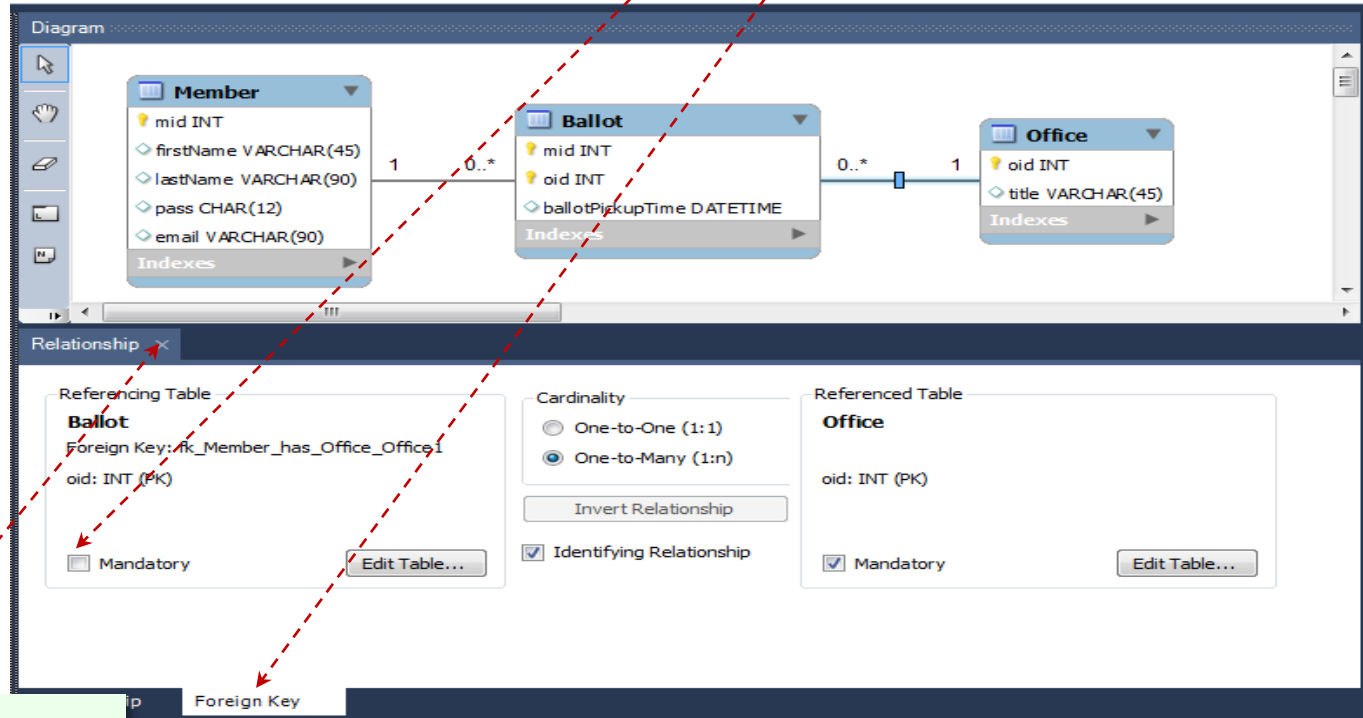
When done, **close** the **Relationship** panel.

Since theoretically there may be Office instances (positions) that will not receive any votes (at any given time), the participation of the **Ballot** entity in the relationship with entity **Office** should be optional. The cardinality constraint at the **Ballot** side should be changed from many-mandatory (**1..***) to many-optional (**0..***).



Double-click the **Ballot – Office** link.

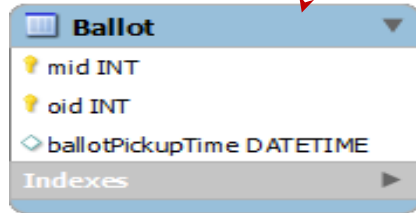
Select the **Foreign Key** tab and **uncheck** option **Mandatory** in the **Ballot** panel.



When done, **close** the **Relationship** panel.

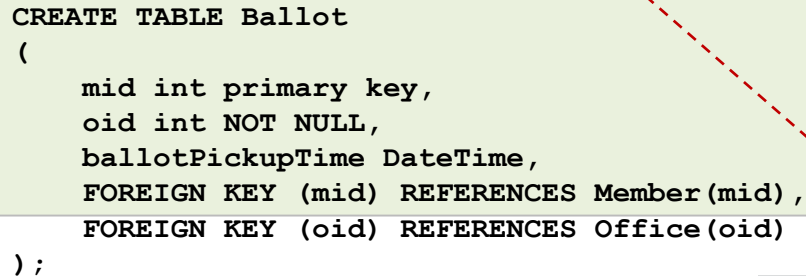
The **Ballot** worksheet of the **election.xlsm** workbook shows three views of each of the **Ballot** entity: **UML**, **SQL** and **Excel Table**.

The `sqlInsert()` function transforms the rows of the table into **SQL-Insert** statements.



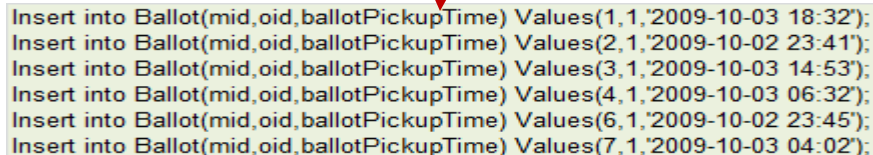
A screenshot of a software interface showing the 'Ballot' entity. It lists three fields: 'mid' of type 'INT', 'oid' of type 'INT', and 'ballotPickupTime' of type 'DATETIME'. Below the fields is a tab labeled 'Indexes' with a right-pointing arrow.

Ballot	
mid	INT
oid	INT
ballotPickupTime	DATETIME
Indexes	



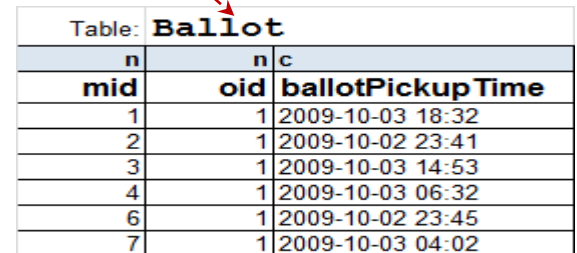
A screenshot of a code editor showing the SQL statement to create the 'Ballot' table. The statement defines 'mid' as the primary key, 'oid' as a non-null integer, and 'ballotPickupTime' as a datetime. It also includes foreign key constraints for 'mid' and 'oid'.

```
CREATE TABLE Ballot
(
    mid int primary key,
    oid int NOT NULL,
    ballotPickupTime DateTime,
    FOREIGN KEY (mid) REFERENCES Member(mid),
    FOREIGN KEY (oid) REFERENCES Office(oid)
);
```



A screenshot of a code editor showing six SQL INSERT statements for the 'Ballot' table. Each statement provides values for 'mid', 'oid', and 'ballotPickupTime'.

```
Insert into Ballot(mid,oid,ballotPickupTime) Values(1,1,'2009-10-03 18:32');
Insert into Ballot(mid,oid,ballotPickupTime) Values(2,1,'2009-10-02 23:41');
Insert into Ballot(mid,oid,ballotPickupTime) Values(3,1,'2009-10-03 14:53');
Insert into Ballot(mid,oid,ballotPickupTime) Values(4,1,'2009-10-03 06:32');
Insert into Ballot(mid,oid,ballotPickupTime) Values(6,1,'2009-10-02 23:45');
Insert into Ballot(mid,oid,ballotPickupTime) Values(7,1,'2009-10-03 04:02');
```



A screenshot of an Excel table titled 'Table: Ballot'. It has three columns: 'mid', 'oid', and 'ballotPickupTime'. The table contains seven rows of data.

Table: Ballot		
n	n	c
mid	oid	ballotPickupTime
1	1	2009-10-03 18:32
2	1	2009-10-02 23:41
3	1	2009-10-03 14:53
4	1	2009-10-03 06:32
6	1	2009-10-02 23:45
7	1	2009-10-03 04:02

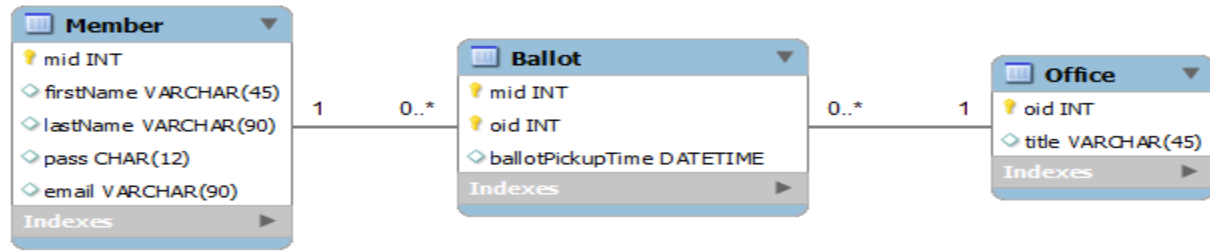


Table: Member				
n	c	c	c	c
mid	firstName	lastName	pass	email
1	Ann	Anson	aa01119	aanson@misor.org
2	Ben	Jamin	bj11122	bjamin@misor.org
3	Cir	Cus	cc22211	ccus@misor.org
4	Don	Donski	dd00413	ddonski@misor.org
5	Eve	Lady	el98765	elady@misor.org
6	Fin	End	fe00011	fend@misor.org
7	Gin	Rum	gr12345	grum@misor.org

Table: Ballot		
n	n	c
mid	oid	ballotPickup Time
1	1	2009-10-03 18:32
2	1	2009-10-02 23:41
3	1	2009-10-03 14:53
4	1	2009-10-03 06:32
6	1	2009-10-02 23:45
7	1	2009-10-03 04:02

Table: Office	
n	c
oid	title
1	President
2	Vice-President
3	Treasurer
4	Editor
5	Web Manager

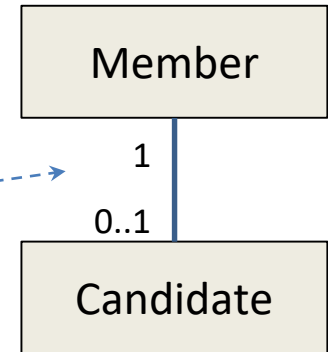
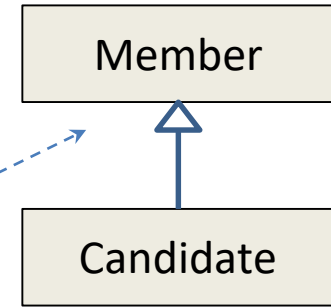
Looking at the relationships between the entities defined, using the primary and foreign keys, one can see that, for example, 'Ann' (pk: **mid**=1 in **Member**) picked her ballot for the office of 'President' (pk: **oid**=1 in **Office**) at **ballotPickupTime**='2009-10-03 18:32'. In table **Ballot**, the foreign keys, **mid** and **oid**, are set for this relationship to the values of the primary keys of 'Ann' and 'President', respectively. Jointly, in table **Ballot**, the keys **mid** and **oid** constitute the primary key.

There is no requirement for a foreign key to have the same name as its related primary key. Nonetheless, having the same names may simplify documentation and query development as long as it is clear which are the primary keys and which are the foreign keys. Some of the **SQL** queries, involving more than one related tables can be simplified when the keys have the same names.

Notice that **MW** has generated the foreign key with names made of the entity names and their primary-key names (**Member_mid** and **Office_oid**) and we changed them to **mid** and **oid**, respectively.

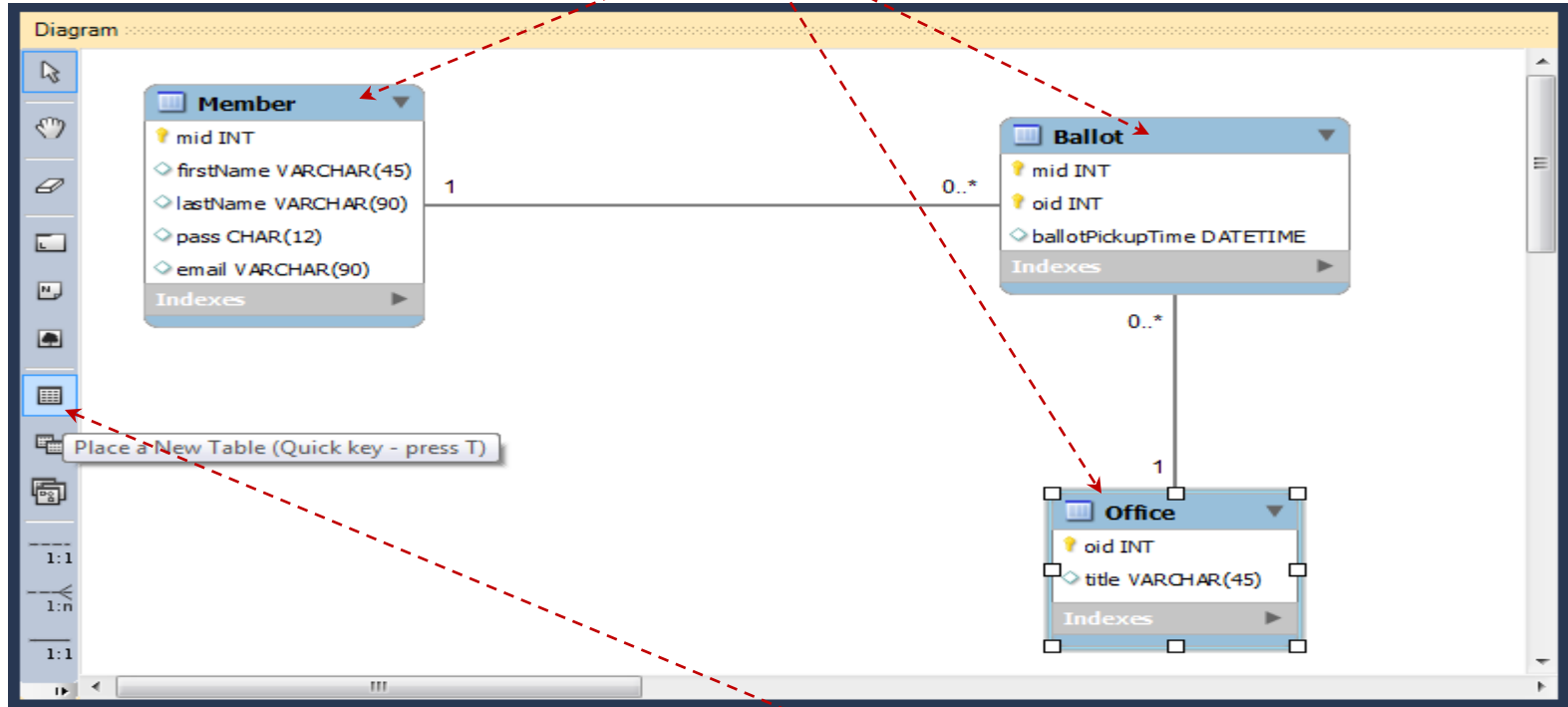
In this instruction, all primary keys and their corresponding foreign keys will have the same name. Thus, the primary key, **mid**, of the **Member** table will propagate to other entities as a foreign key with the same name (**mid**). The primary key, **oid**, of the **Office** table, will also have the same name for all related foreign keys. It will make **SQL** queries look more slick and compact.

The model developed so far does not include important individuals: candidates who are running for offices. Each candidate must be a member (an instance of entity **Member**) but only some of the members are candidates (instances of entity **Candidate**). The relationship between **Candidate** and **Member** is hierarchical. **Candidates** inherit all the attributes from related members. Such a relationship is modeled in **UML** as **specialization** (a candidate is a specialization of a member) also known as a **Super-type – Sub-type** association*. **MySQL Workbench** does not support such a notation. However, it supports this type of the relationship as an **One-to-One** relationship with optional participation of the **subtype** entity (here **Candidate**).



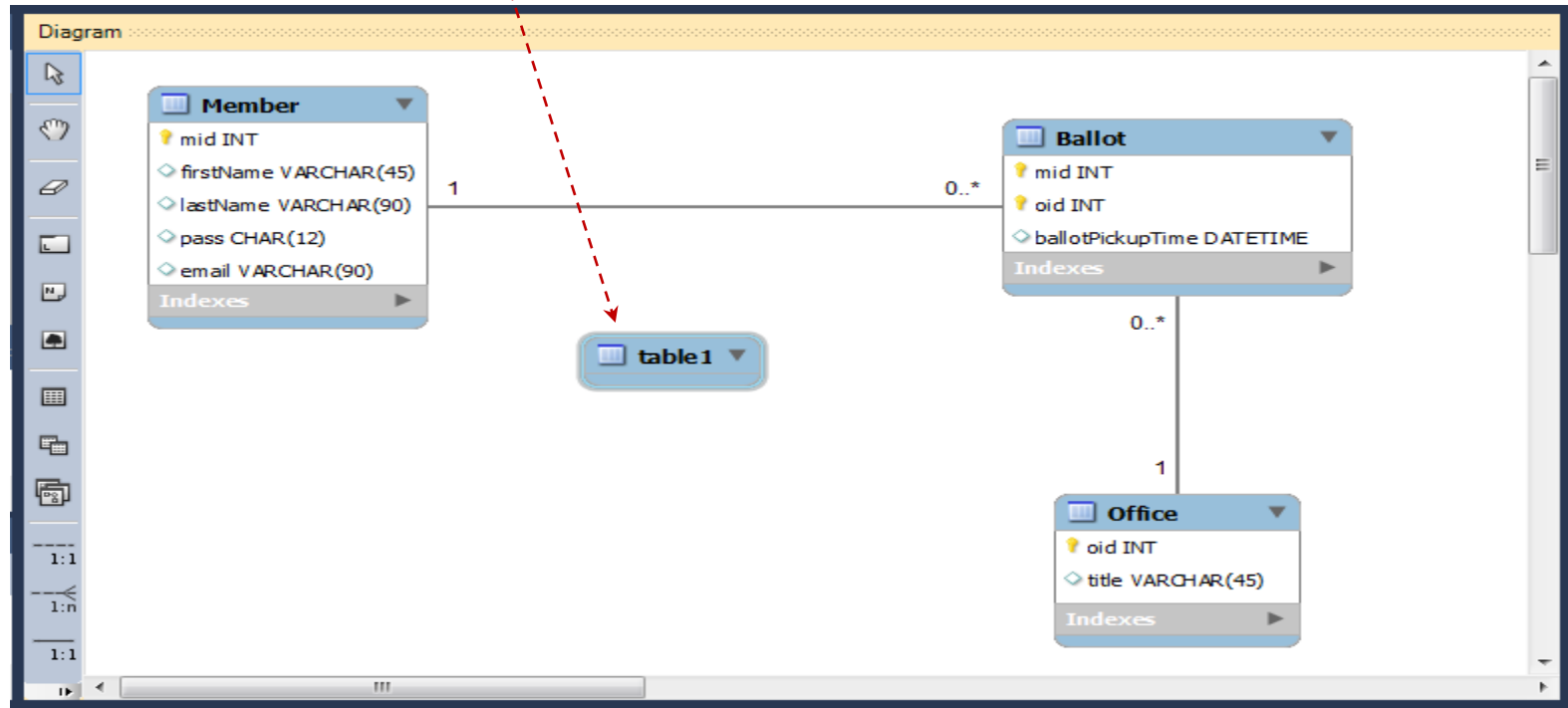
* In an object-oriented language, **Member** would be a super-class and **Candidate**—a sub-class. **Java** would say that class **Candidate** extends class **Member**.

Since there will be more entities and relationships in the model, why don't we **rearrange** the entities approximately as shown below.
It can be done by simply **dragging** the **entities** around.



Now, **click** the **New Table** icon.

Click the **diagram canvas** to add a table.



Double-click the **new table** and *change* its name to **Candidate**.

The screenshot shows a database design tool interface. At the top, a diagram displays three tables: **Member**, **Ballot**, and **Candidate**. The **Member** table has columns: mid INT, firstName VARCHAR(45), lastName VARCHAR(90), pass CHAR(12), and email VARCHAR(90). The **Ballot** table has columns: mid INT, oid INT, and ballotPickupTime DATETIME. The **Candidate** table is newly created and currently has no columns. A red dashed arrow points from the text 'Double-click the new table' to the **Candidate** table in the diagram. Another red dashed arrow points from the text 'change its name to Candidate' to the 'Table Name' field in the 'Candidate - Table' panel, which already contains the text 'Candidate'. The 'Candidate - Table' panel also shows a 'Schema' field set to 'election'. Below the panel, there are tabs for 'Columns', 'Indexes', 'Foreign Keys', 'Triggers', 'Partitioning', 'Options', 'Inserts', and 'Privileges'. The 'Columns' tab is currently selected.

Diagram

Member

- mid INT
- firstName VARCHAR(45)
- lastName VARCHAR(90)
- pass CHAR(12)
- email VARCHAR(90)

Ballot

- mid INT
- oid INT
- ballotPickupTime DATETIME

Candidate

Candidate - Table

Table Name:

Schema: **election**

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Collation:

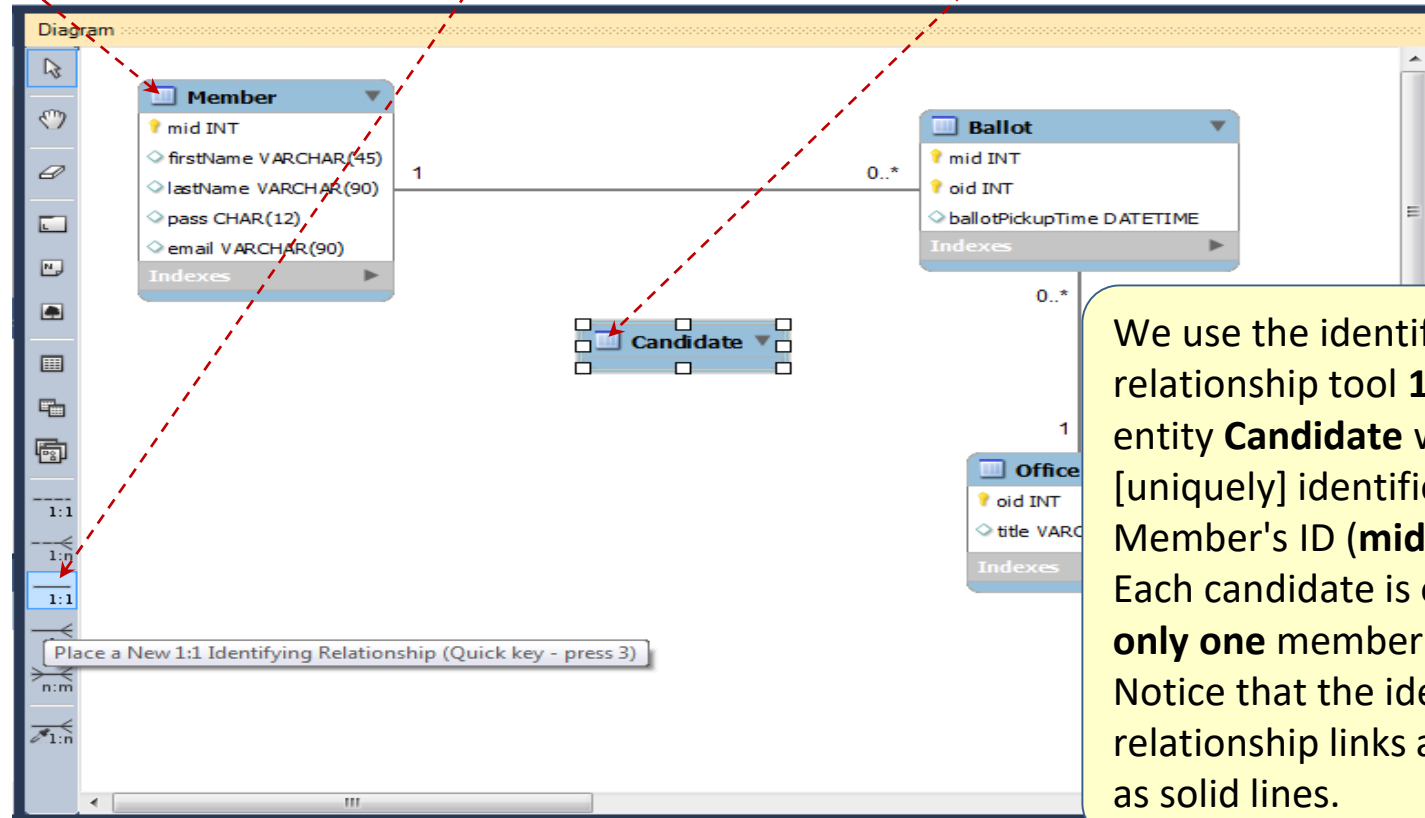
Comments:

Columns | Indexes | Foreign Keys | Triggers | Partitioning | Options | Inserts | Privileges

When done, *close* the **Candidate – Table** panel.

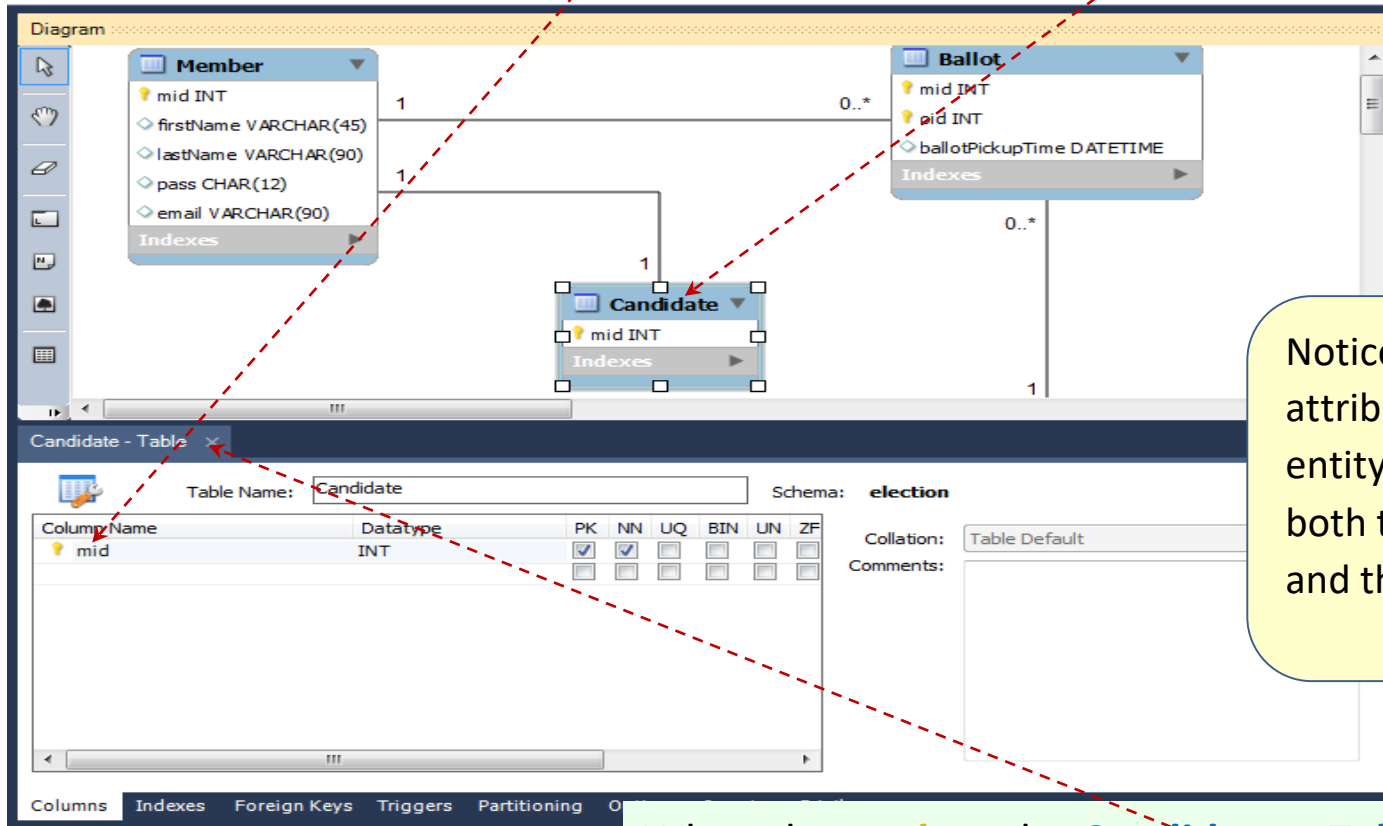
Connect entity **Candidate** with entity **Member** using the **1:1 Identifying Relationship** tool.

More specifically, first **click** the **1:1 tool**, next **click** table **Candidate** and finally **click** table **Member**.



We use the identifying relationship tool **1:1** since entity **Candidate** will be [uniquely] identified by the Member's ID (**mid**). Each candidate is **one and only one** member. Notice that the identifying relationship links are shown as solid lines.

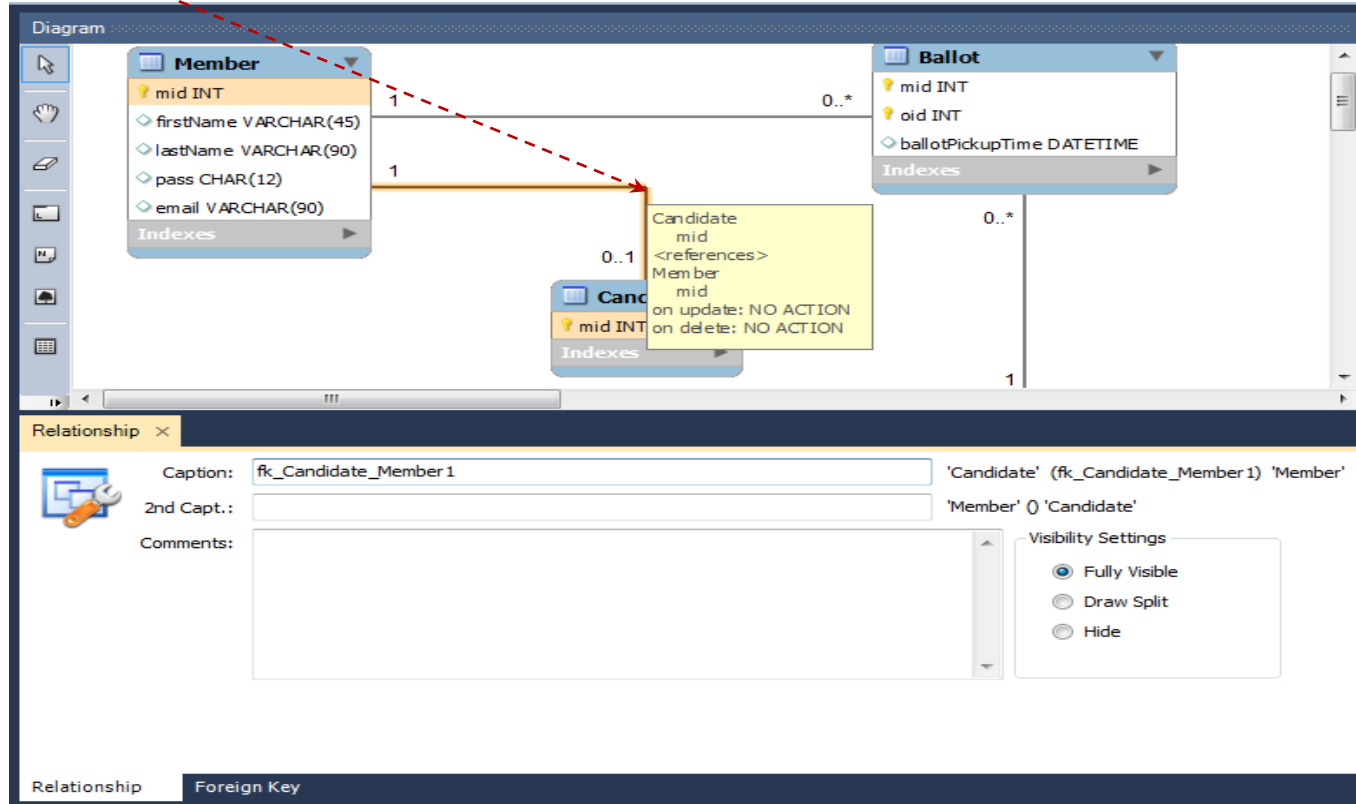
To **change** the name of the inherited table, **double-click** entity **Candidate** and change column-name **Member_mid** to **mid**.



Notice that the inherited attribute, **mid**, serves in entity **Candidate** as both the **primary key** and the **foreign key**.

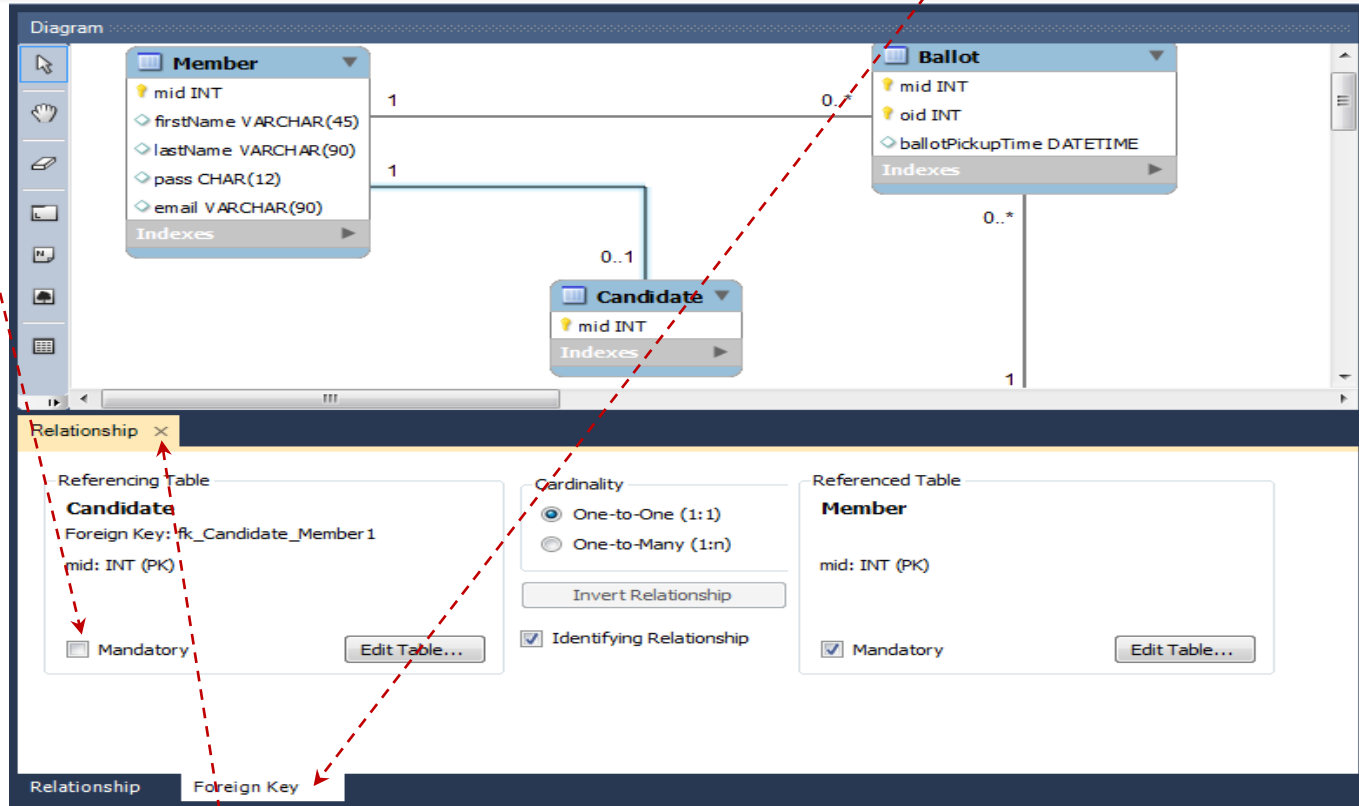
When done, **close** the **Candidate – Table** panel.

To **change** the cardinality constraint at the **Candidate** entity, first **double-click** the **Member - Candidate** link.



option **Mandatory** in the **Candidate** panel.

Select the **Foreign Key** tab and **uncheck**



When done, **close** the **Relationship** panel.

A **One-to-Many** relationship is directional. When connecting the related entities, **MW** requires that the entity on the **Many** side be selected (clicked on) prior to selection of the entity at the **One** side. The latter maps its primary key to the foreign key of the former.

The relationship we are about to create is:

Office (1) — *is-being-run-for-by* — (1..*) **Candidate**

(In order for an instance of office, e.g. '**President**', to be part of the election, there must be at least one candidate running for it.)

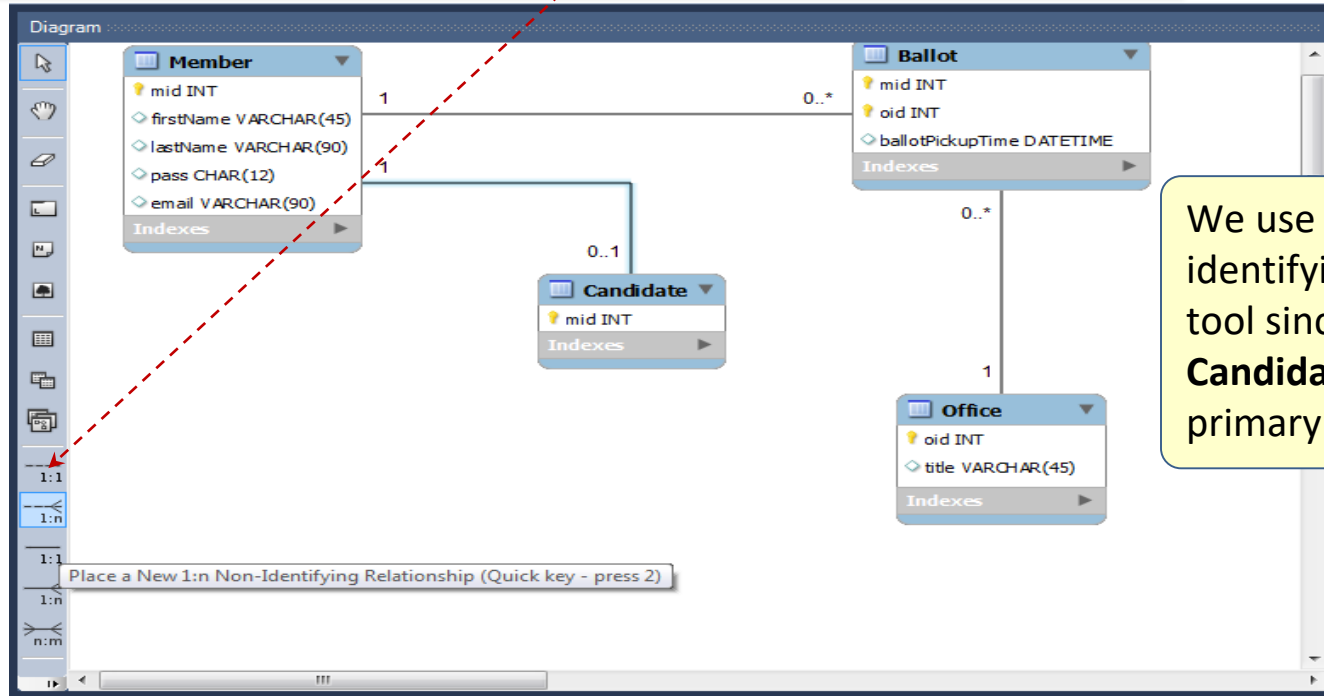
This relationship can also be defined as:

Candidate (1..*) — *runs-for* — (1) **Office**

Since an office may have many candidates, it is responsibility of the candidates to "know" which offices they are running for. Thus each instance of entity **Candidate** must include "information" about the office s/he is running for. This information is nothing else but the foreign key in the **Candidate** table pointing to the primary key in the **Office** table.

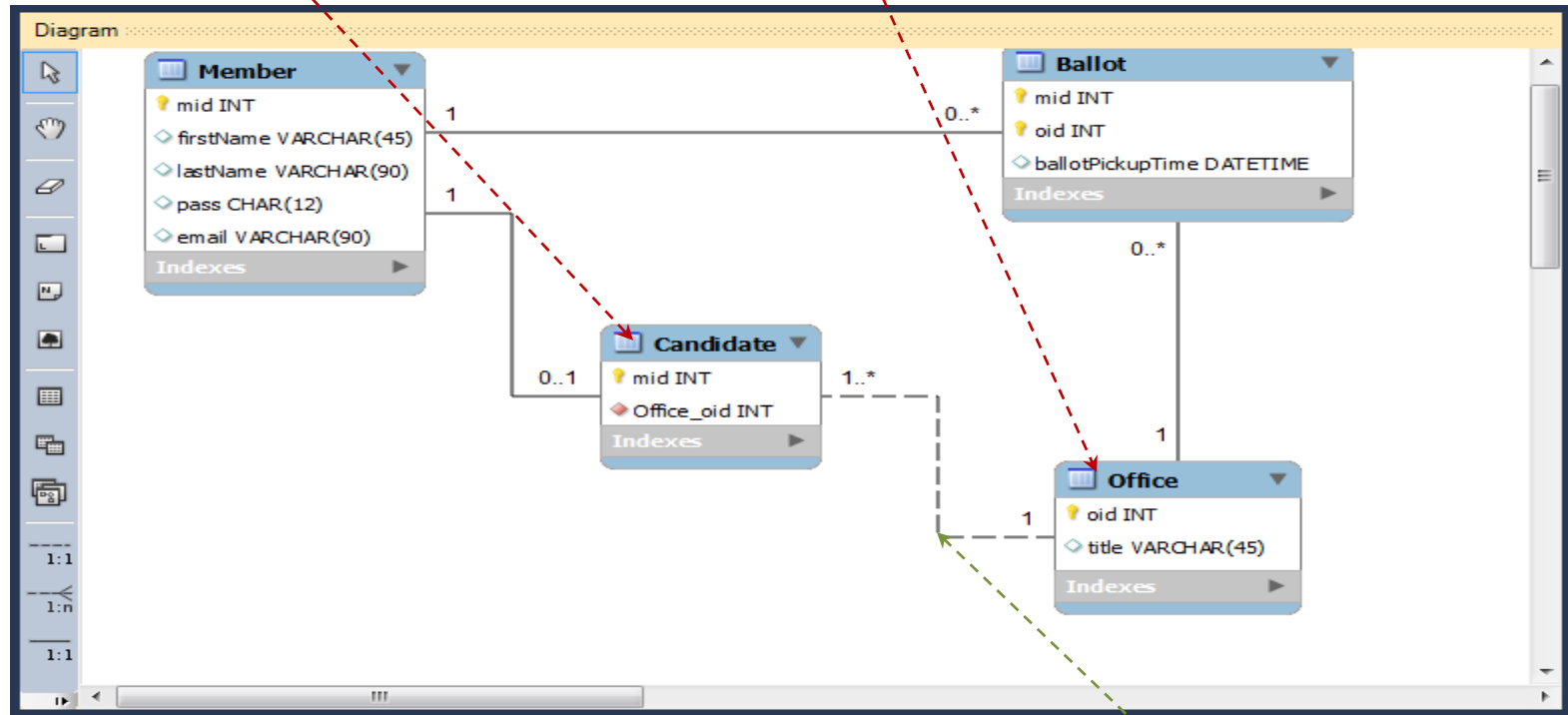
Now it is a good time to further specialize entity **Candidate**. What distinguishes a candidate from a member is that the former is [deterministically] running for an office. Since a candidate may run for only one office and an office may have many candidates, the appropriate relationship between **Office** and **Candidate** is a **One-To-Many (1:n) Non-identifying Relationship**.

Select (click) the **Non-identifying One-To-Many Relationship** tool.



We use the non-identifying relationship tool since entity **Candidate** already has a primary key.

Click the **Candidate** table and then *click* the **Office** table.



MW connects entity **Office** with entity **Candidate** with a One-To-Many (1 : 1..*) Non-identifying Relationship. Notice that this non-identifying relationship link is shown using a dashed line.

Double-click the **Candidate** table and then *change* column name **Office_oid** to **oid**.

The screenshot displays a database management interface. The top section, titled "Diagram", shows a relationship diagram with three tables: "pass", "Candidate", and "Office". The "pass" table has columns "pass CHAR(12)" and "email VARCHAR(90)". The "Candidate" table has columns "mid INT" and "oid INT". The "Office" table has a column "oid INT". Relationships are shown with lines and cardinalities: "pass" (1) to "Candidate" (0..1), "Candidate" (1..*) to "Office" (1), and "Candidate" (1) to "Office" (1). A red dashed arrow points from the text "Double-click the Candidate table" to the "Candidate" table in the diagram. Another red dashed arrow points from the text "change column name Office_oid to oid" to the "oid" column in the "Candidate" table's table editor.

The bottom section, titled "Candidate - Table", shows the table editor for the "Candidate" table. The table name is "Candidate" and the schema is "election". The table has two columns: "mid" (INT) and "oid" (INT). The "oid" column is highlighted. The table editor includes a table with columns: Column Name, Datatype, PK, NN, UQ, BIN, UN, and ZF. The "oid" column is highlighted in the table.

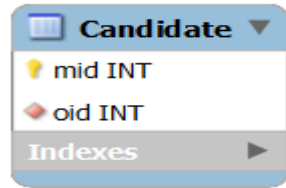
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF
mid	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
oid	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

The table editor also includes a "Collation" dropdown set to "Table Default" and a "Comments" text area. The bottom of the interface shows tabs for "Columns", "Indexes", "Foreign Keys", "Triggers", "Partitioning", "Options", "Inserts", and "Privileges".

When done, *close* the **Candidate – Table** panel.

The **Candidate** worksheet of the **election.xlsm** workbook shows three views of each of the **Candidate** entity: **UML**, **SQL** and **Excel Table**.

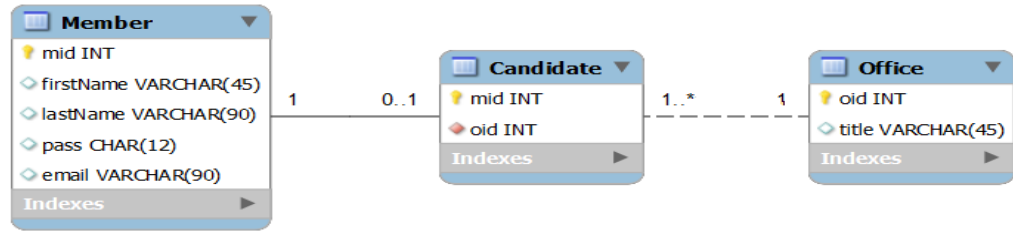
The **sqlInsert()** function transforms the rows of the table into **SQL-Insert** statements.



```
CREATE TABLE Candidate
(
    mid int primary key,
    oid int NOT NULL,
    FOREIGN KEY (mid) REFERENCES Member(mid),
    FOREIGN KEY (oid) REFERENCES Office(oid)
);
```

```
Insert into Candidate(mid,oid) Values(1,4);
Insert into Candidate(mid,oid) Values(3,1);
Insert into Candidate(mid,oid) Values(4,2);
Insert into Candidate(mid,oid) Values(7,4);
Insert into Candidate(mid,oid) Values(8,3);
Insert into Candidate(mid,oid) Values(9,3);
Insert into Candidate(mid,oid) Values(10,2);
```

Table: Candidate	
n	n
mid	oid
1	4
3	1
4	2
7	4
8	3
9	3
10	2



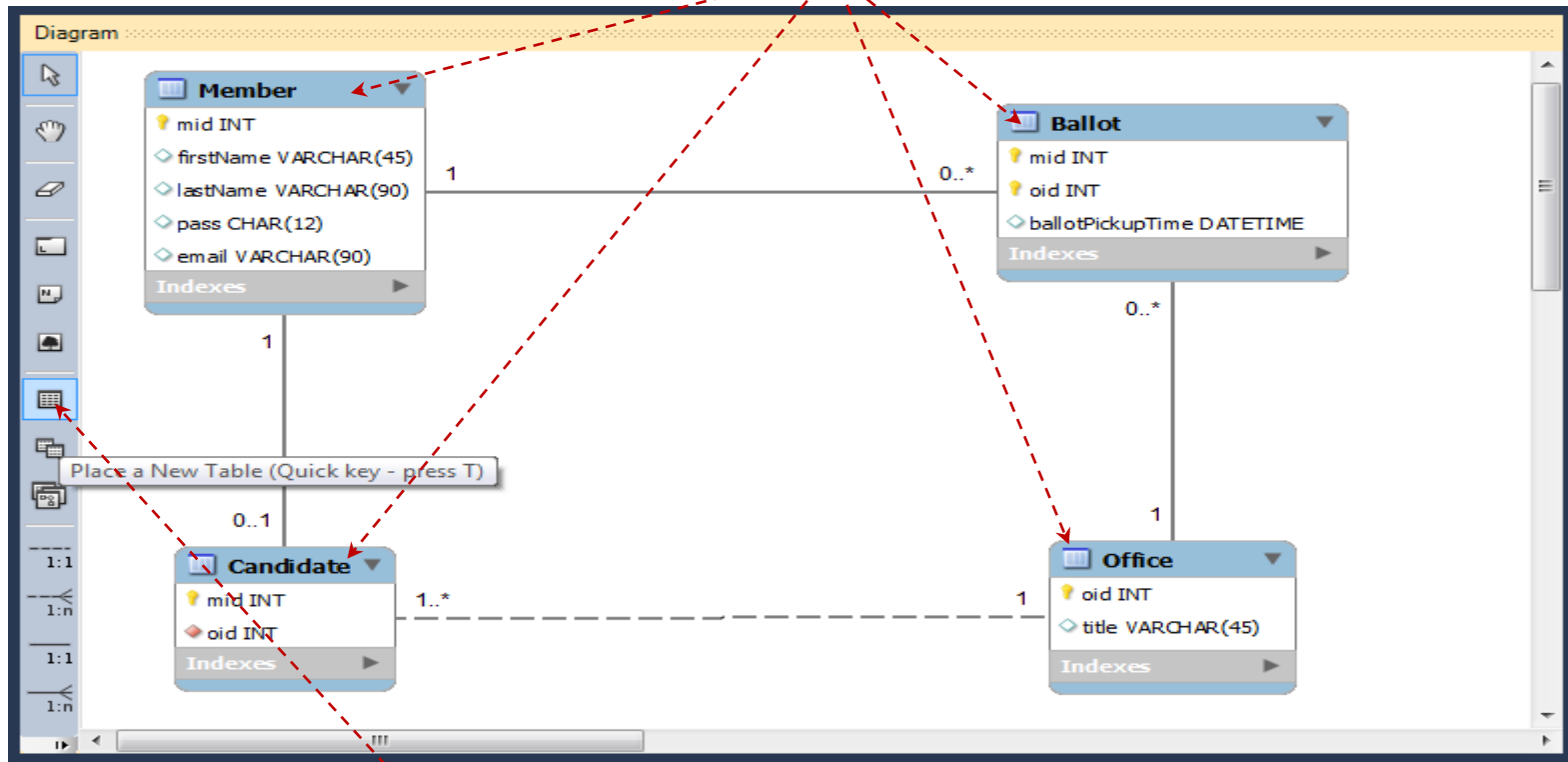
mid	firstName	lastName	pass	email
1	Ann	Anson	aa01119	aanson@misor.org
2	Ben	Jamin	bj11122	bjamin@misor.org
3	Cir	Cus	cc22211	ccus@misor.org
4	Don	Donski	dd00413	ddonski@misor.org
5	Eve	Lady	el98765	elady@misor.org
6	Fin	End	fe00011	fend@misor.org
7	Gin	Rum	gr12345	grum@misor.org
8	Hal	Bar	hb01011	hbar@misor.org
9	Ian	Yan	iy99900	iyan@misor.org
10	Jan	Osik	jo33322	josik@misor.org

mid	oid
1	4
3	1
4	2
7	4
8	3
9	3
10	2

oid	title
1	President
2	Vice-President
3	Treasurer
4	Editor
5	Web Manager

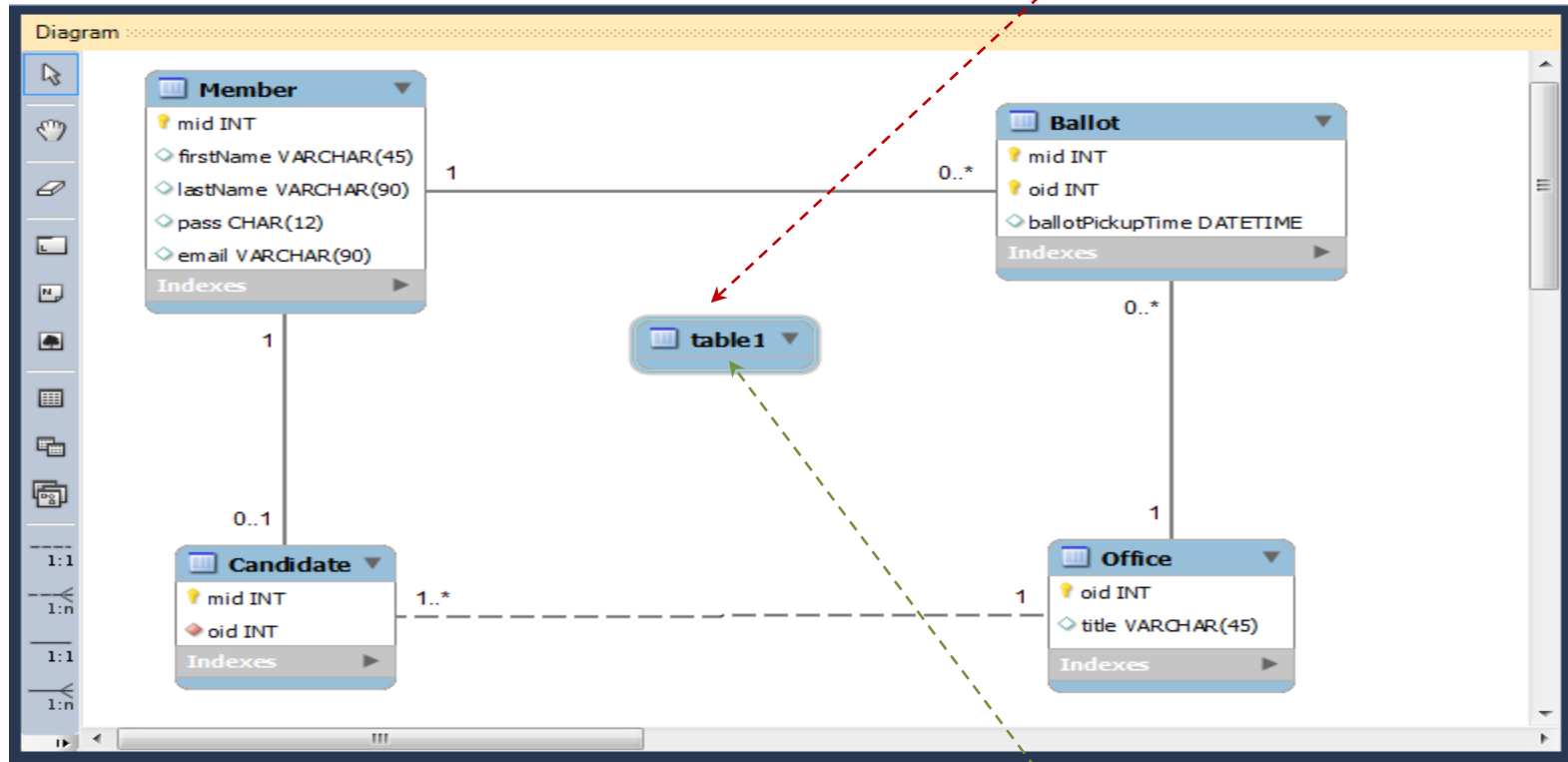
Looking at the relationships between the entities, defined by means of primary and foreign keys, one can see that, for example, 'Ann' (pk: **mid**=1 in **Member**) is a candidate (pk: **mid**=1, having fk: **mid**=1 and **oid**=4 in **Candidate**) who is running for office of 'Editor' (pk: **oid**=4) in **Office**. In table **Candidate**, the foreign keys, **mid** and **oid**, are set for this relationship to the values of the primary keys of 'Ann' and 'Editor', respectively. In table **Candidate**, the key **mid** plays a dual role. It is both the primary and foreign key (resulting from an identifying relationship).

Since there will be one more entity and relationship in the model, why don't we **rearrange** the entities approximately as shown below. It can be done by simply **dragging** the **entities** around.



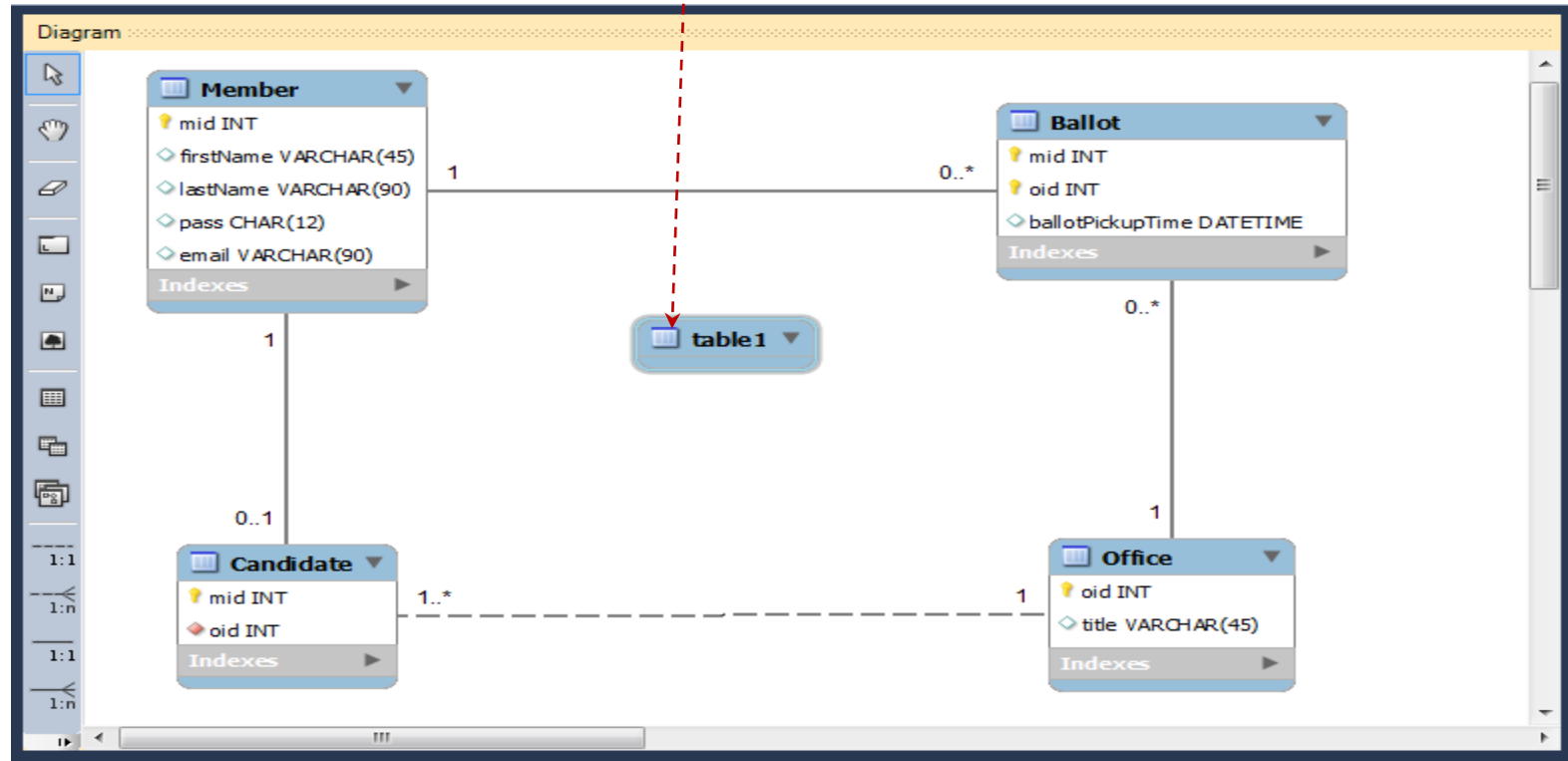
When done, **click** the **New Table** icon.

Click in the middle of the diagram to place a **new table** there.

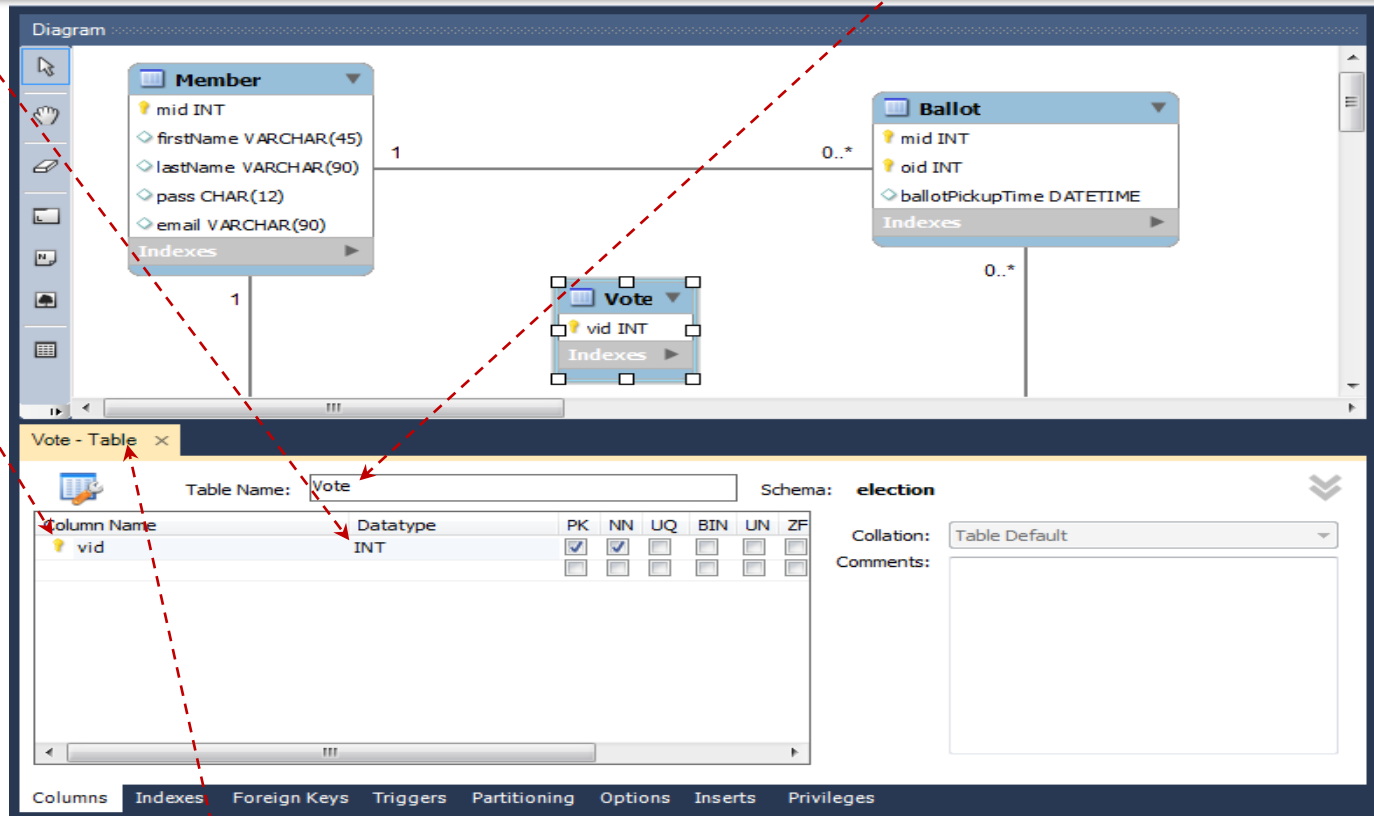


MW adds a new **entity** (**table1**) to the diagram.

Double-click the new table.

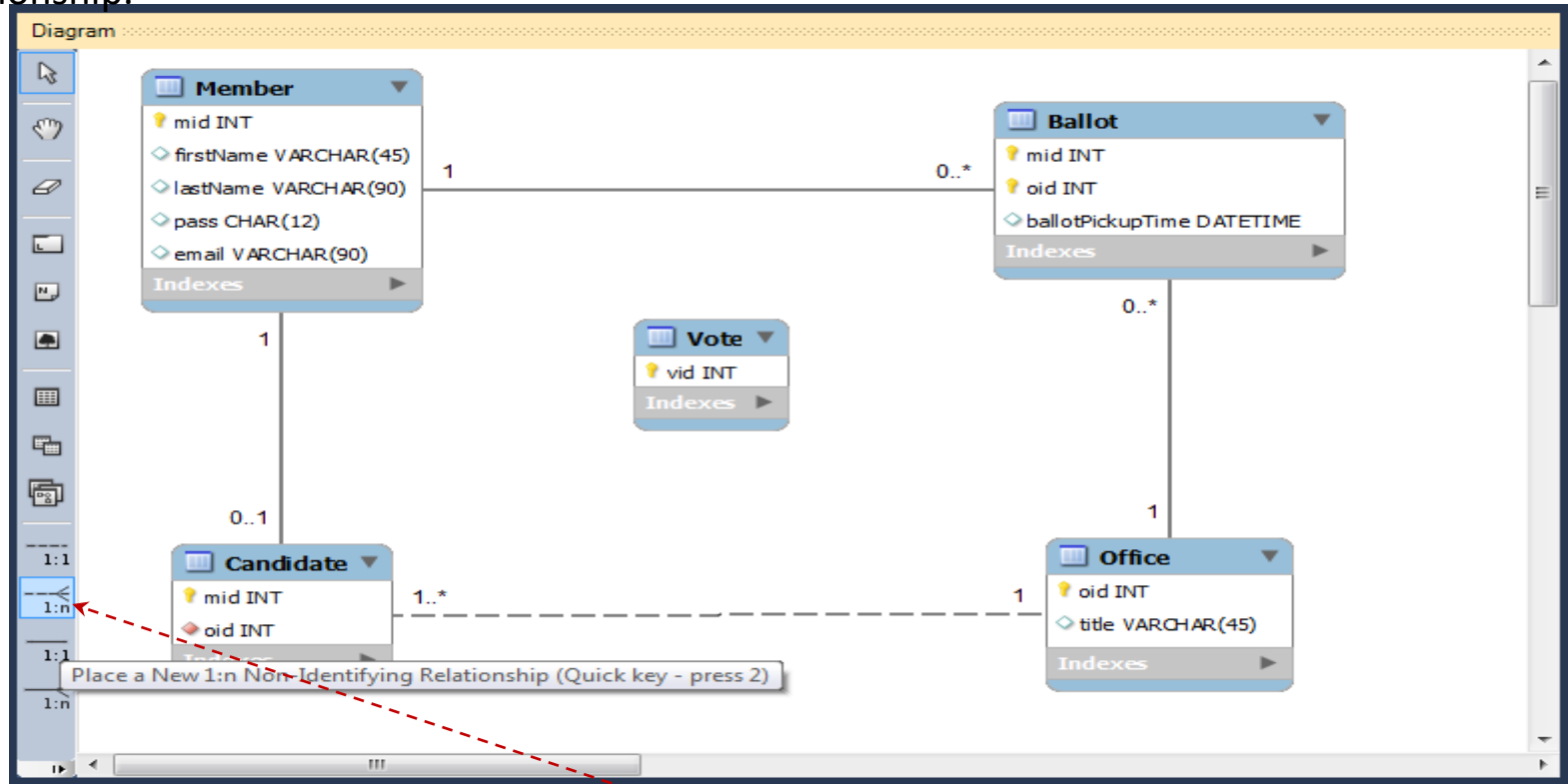


Change the name of this table to **Vote** and *add* column **vid** of type **INT** as a **primary key**.



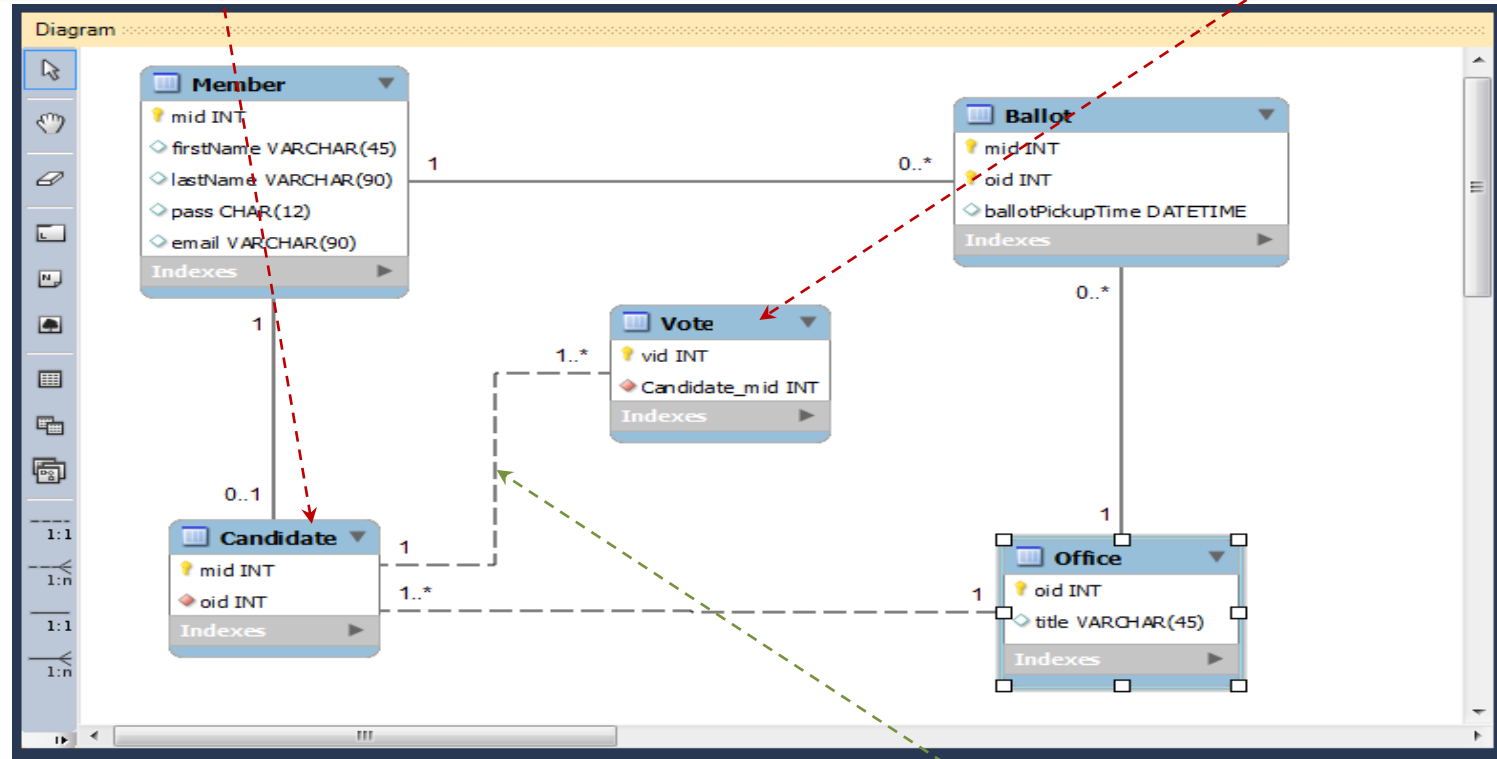
When done, *close* the **Vote - Table** panel.

Entity **Vote** serves here as a set of votes cast for the candidates. Since a candidate may get many votes and each vote is cast exactly for one candidate, **Candidate – Vote** is a **One-to-Many** relationship.



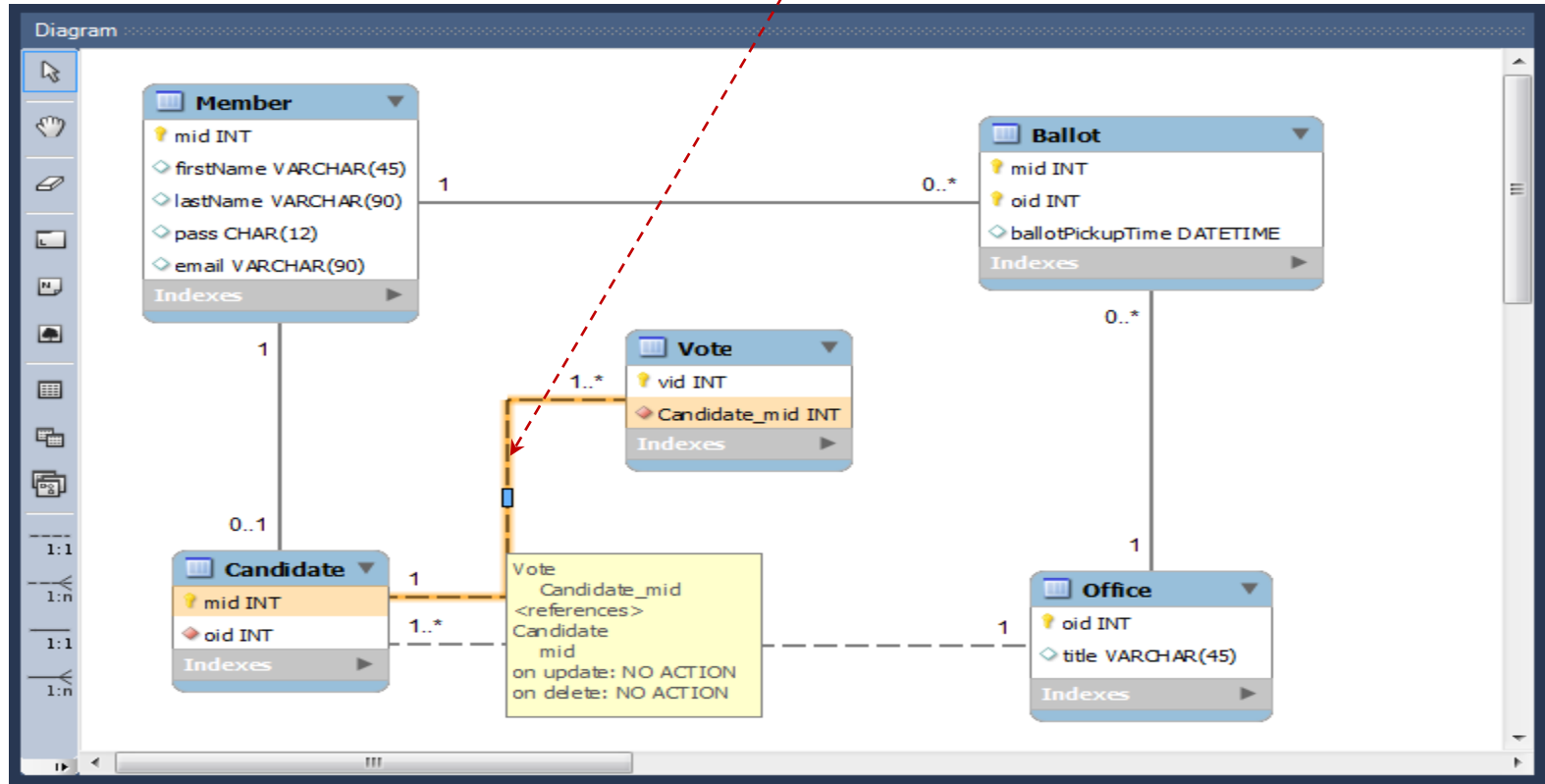
Select (click) the 1:n Non-Identifying Relationship tool.

First **click** the **Vote** table and
then **click** the **Candidate** table.

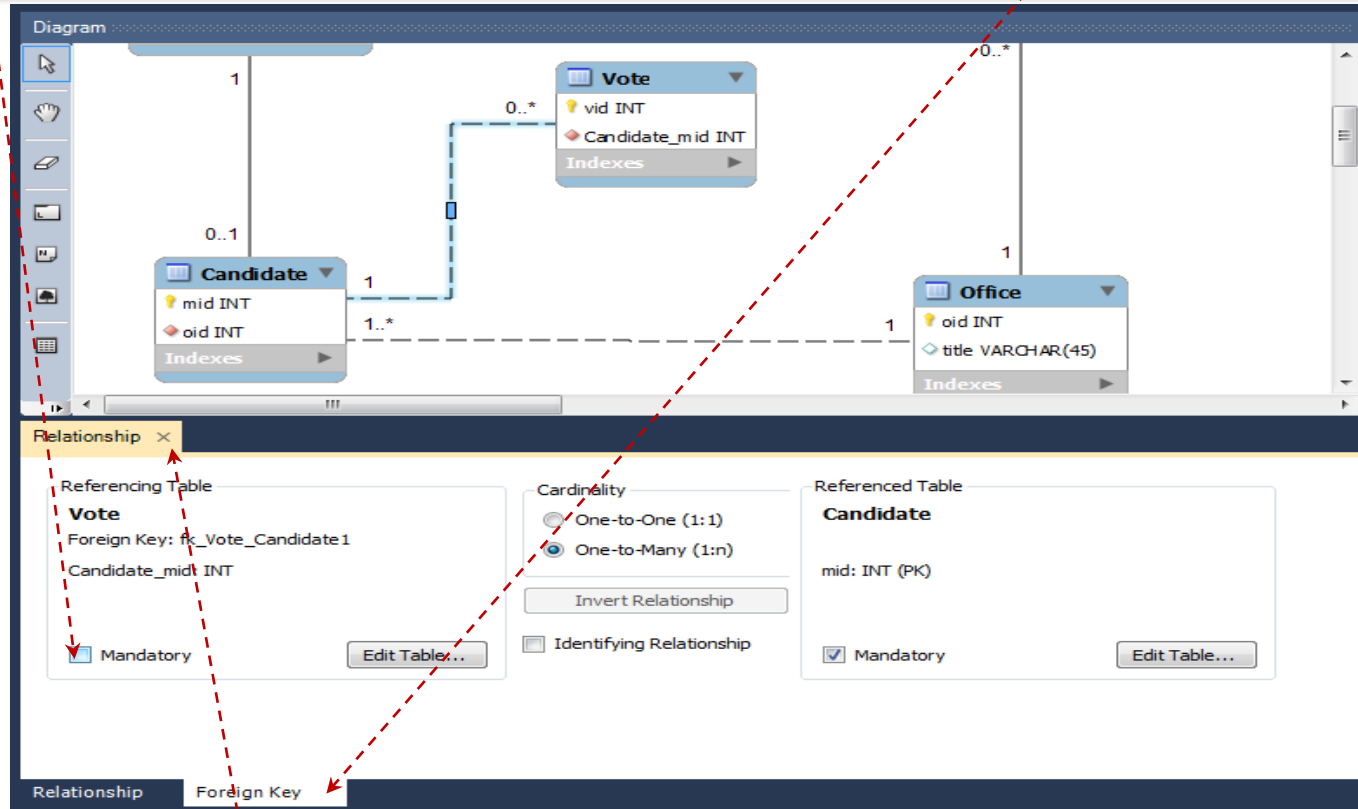


MW connects entity **Candidate** with entity **Vote** with a One-To-Many (1 : 1..*) Non-identifying Relationship. This relationship needs to be “relaxed” as there may be candidates who will not receive any votes.

Double-click the **Candidate - Vote** relationship link.

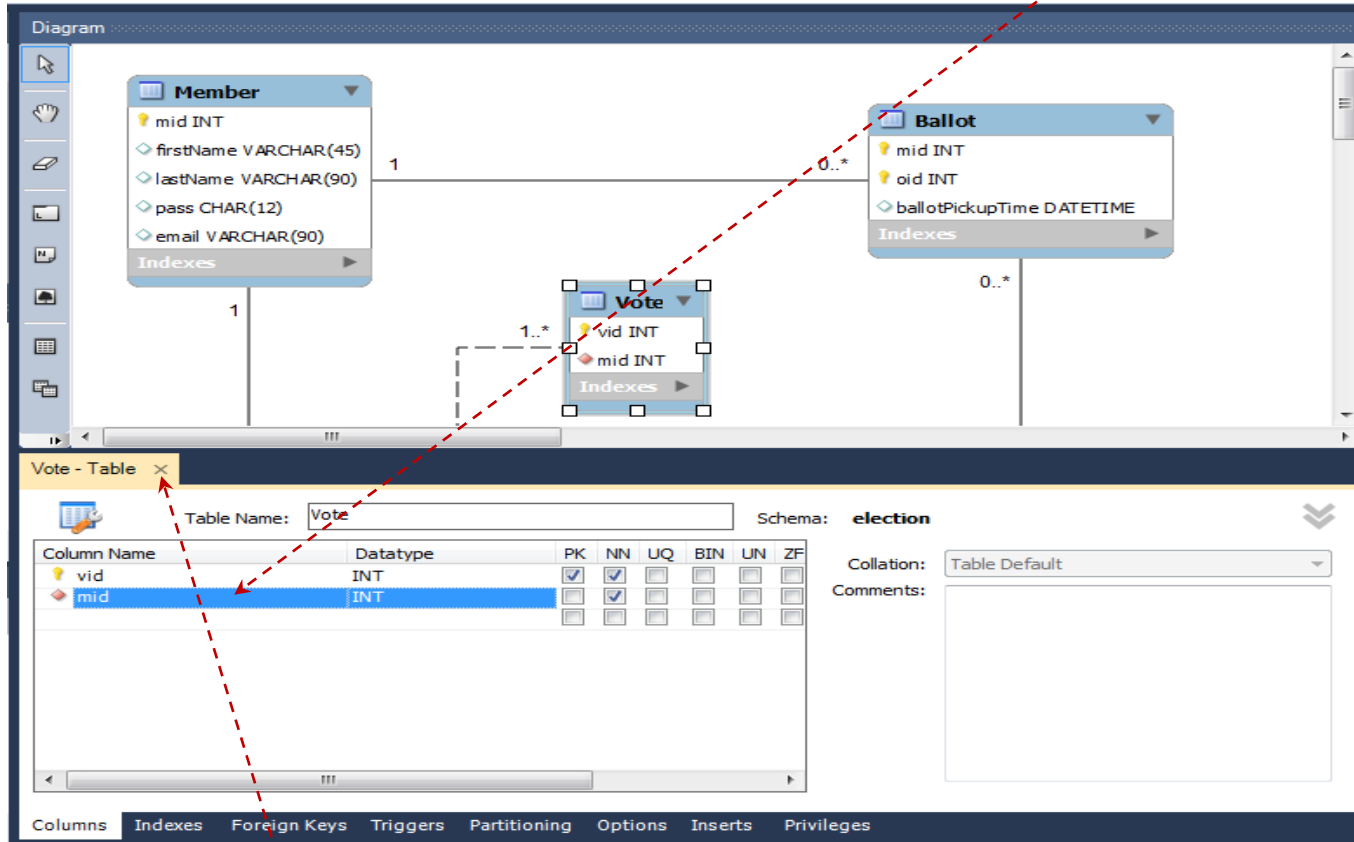


option **Mandatory** in the **Candidate** panel. *Select* the **Foreign Key** tab and *uncheck*



When done, *close* the **Relationship** panel.

Change the name of column **Candidate_mid** to **mid**.



When done, close the **Vote – Table** panel.

The **Vote** worksheet of the **election.xlsm** workbook shows three views of each of the **Vote** entity: **UML**, **SQL** and **Excel Table**.

The `sqlInsert()` function transforms the rows of the table into **SQL-Insert** statements.



```
CREATE TABLE Vote
(
    id int primary key auto_increment,
    mid int NOT NULL,
    FOREIGN KEY (mid) REFERENCES
Candidate (mid)
);
```

```
Insert into Vote(vid,mid) Values(1,13);
Insert into Vote(vid,mid) Values(2,22);
Insert into Vote(vid,mid) Values(3,22);
Insert into Vote(vid,mid) Values(4,22);
Insert into Vote(vid,mid) Values(5,13);
Insert into Vote(vid,mid) Values(6,3);
Insert into Vote(vid,mid) Values(7,22);
Insert into Vote(vid,mid) Values(8,13);
Insert into Vote(vid,mid) Values(9,13);
Insert into Vote(vid,mid) Values(10,13);
```

Table: **Vote**

n	n
vid	mid
1	13
2	22
3	22
4	22
5	13
6	3
7	22
8	13
9	13
10	13

Examining the relationships between the entities, defined by means of primary and foreign keys, one can see that, for example, candidate **13** (fk: **mid**=13 in **Candidate**) has received 5 votes (pk: **vid**=2, 3, 7, 11, 13, with fk: **mid**=13 in **Vote**).

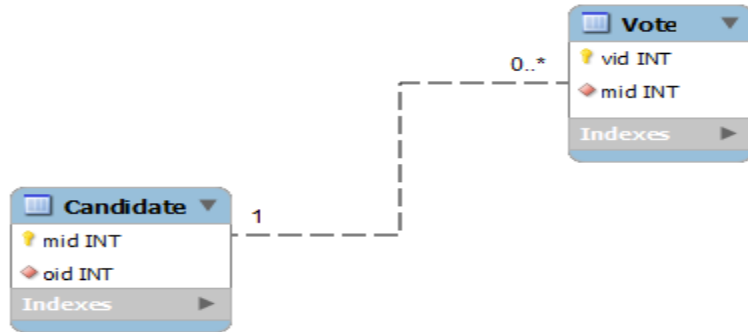


Table: Candidate	
n	n
mid	oid
1	4
3	1
4	2
7	4
8	3
9	3
10	2
12	3
13	1
15	2
18	5
20	4
21	4
22	1
23	4
24	2
25	4
33	5
37	5
40	5

Table: Vote	
n	n
vid	cid
1	22
2	13
3	13
4	22
5	22
6	3
7	13
8	22
9	3
10	22
11	13
12	22
13	13
14	3
15	3
16	3
17	22

The logical phase of the database design is done! It has enough details (tables, attributes, primary keys, foreign keys and cardinality constraints) so that it can be transformed into an executable **SQL** model. **MW** can do this transformation via its **Forward Engineer SQL** export facility (menu options):

File > Export > Forward Engineer SQL Create Script ...

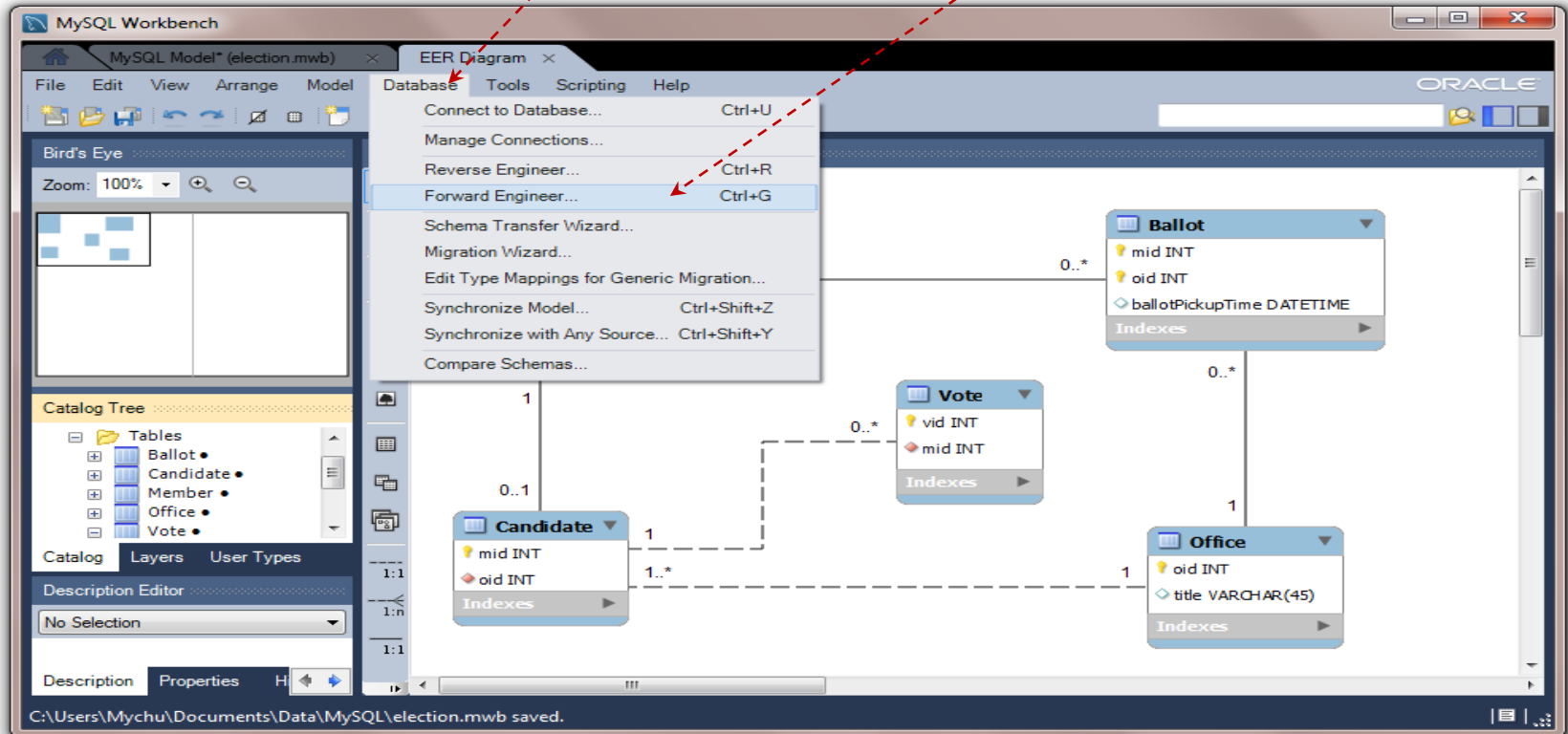
or

Database > Forward Engineer ...

The following instruction shows how to do it using the latter command.

Make sure that your **EER** diagram is saved!

Select menu options **Database** followed by **Forward Engineer**.



Use your **root** connection and **click** button **Next**.

Forward Engineer to Database

Connection Options

- Options
- Select Objects
- Review SQL Script
- Commit Progress

Set Parameters for Connecting to a DBMS

Stored Connection: Select from saved connection settings

Connection Method: Method to use to connect to the RDBMS

Parameters ☒ SSL ☐ Advanced

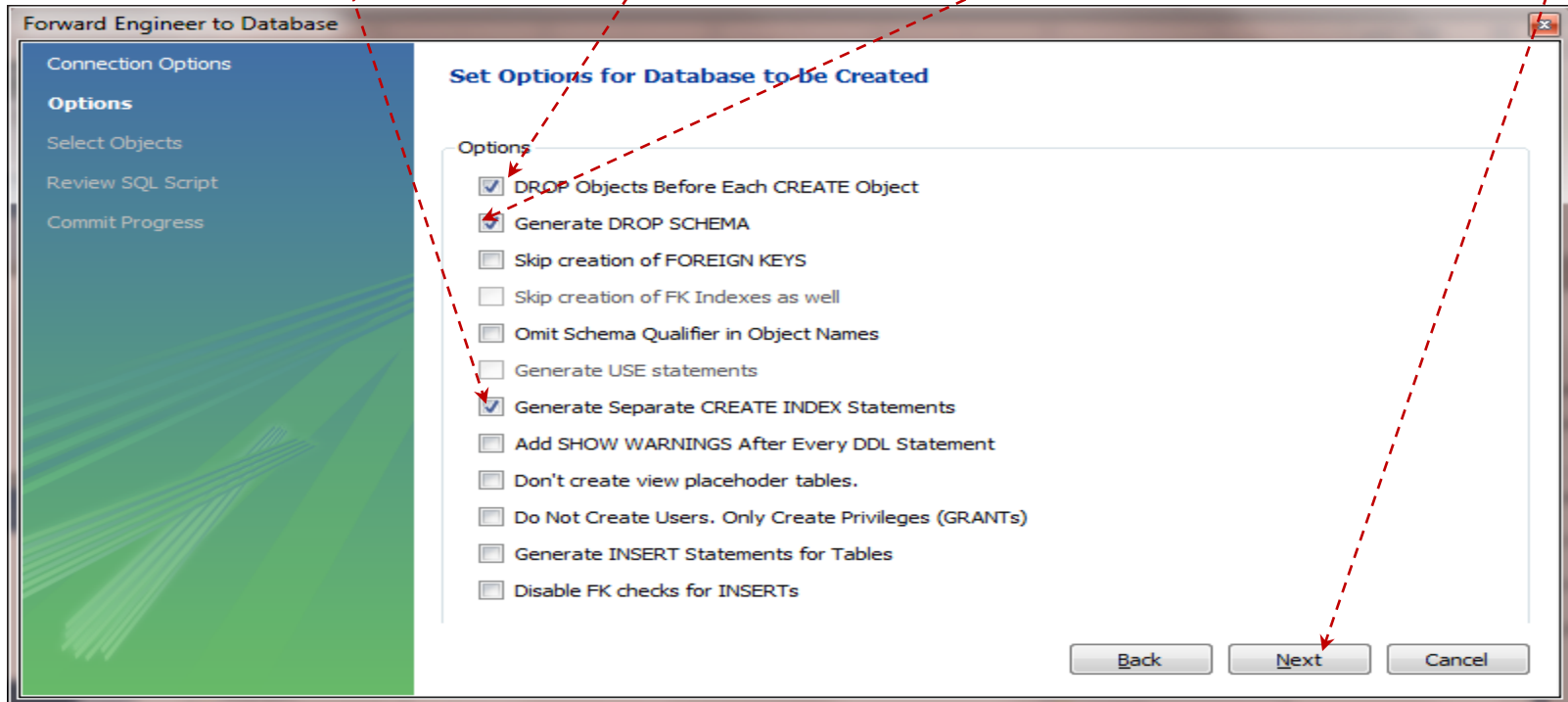
Hostname: Port: Name or IP address of the server host. - TCP

Username: Name of the user to connect with.

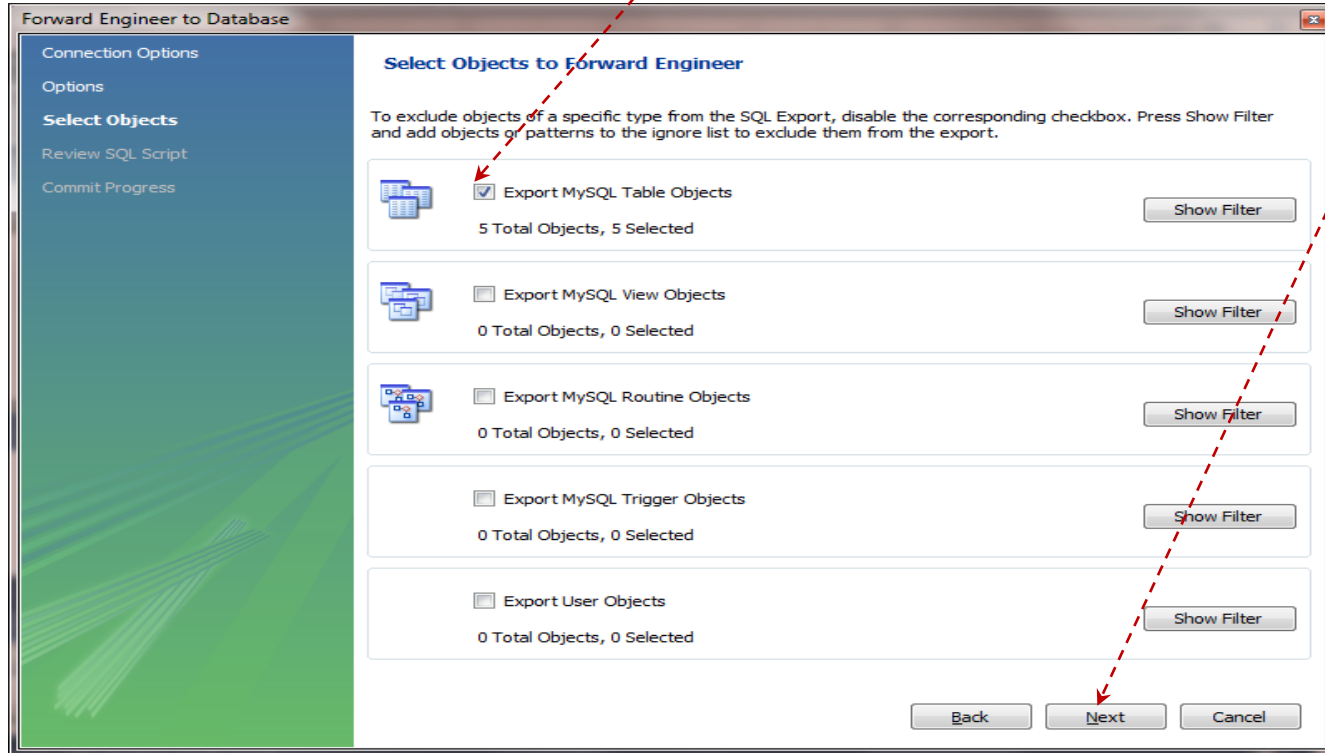
Password: The user's password. Will be requested later

Default Schema: The schema to use as default schema. Leave

Select (check) options Drop Objects Before Each CREATE Object, Generate DROP SCHEMA, and Generate Separate CREATE INDEX Statements and then click button Next.

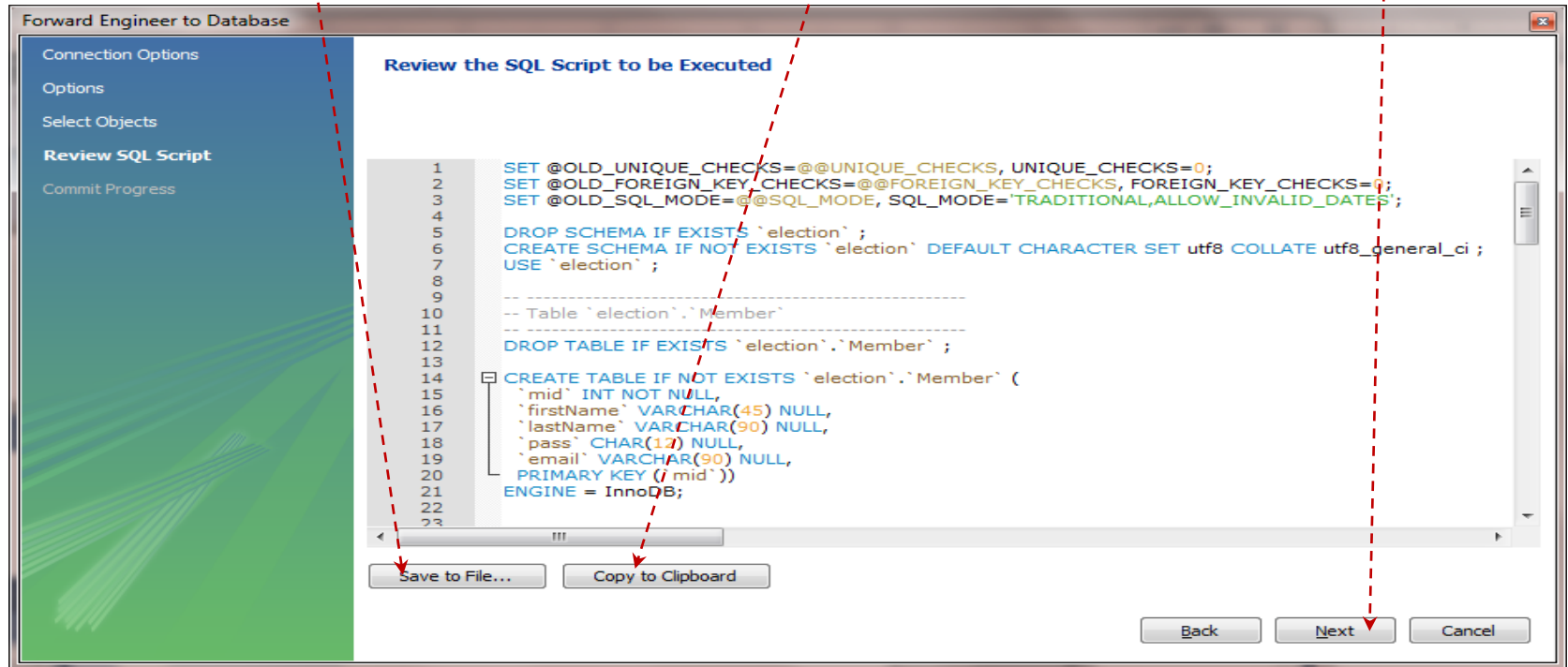


Select (check) just one option, **Export MySQL Table Objects**, and then click button **Next**.



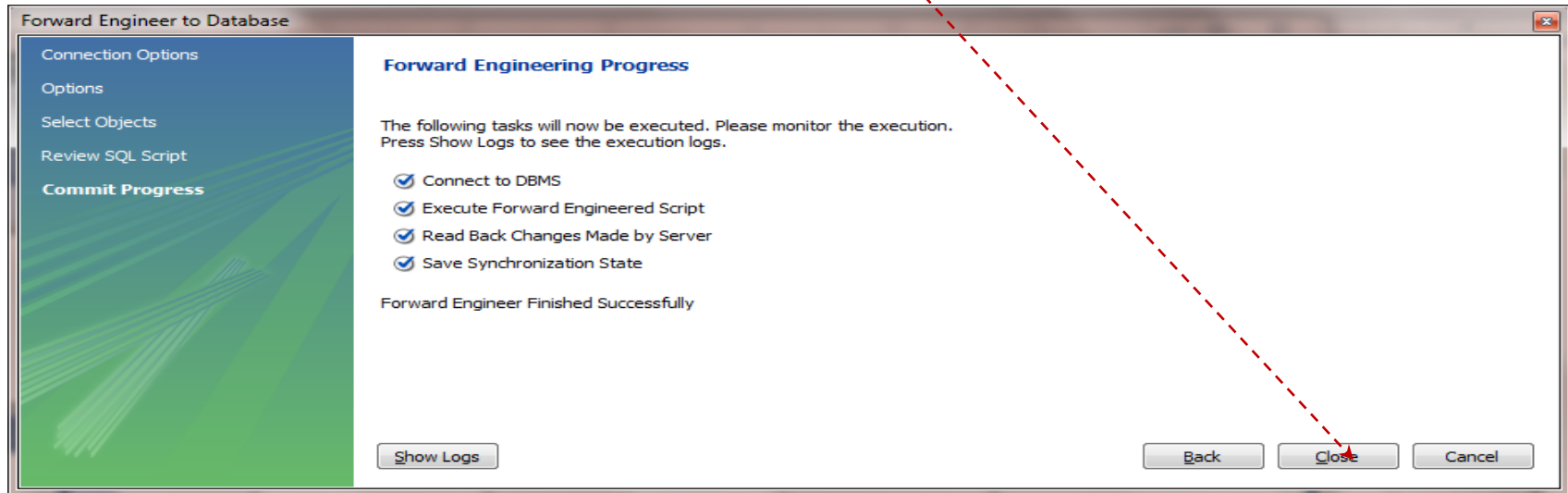
MW generates an **SQL** script, including all statements necessary to create the database, **election**, tables (**Member**, **Office**, **Ballot**, **Candidate** and **Vote**) as well as indexes of the foreign key for tables (**Ballot**, **Candidate** and **Vote**).

Save the script to a **file**, **click** button **Copy to Clipboard**, and then **click** button **Next**.



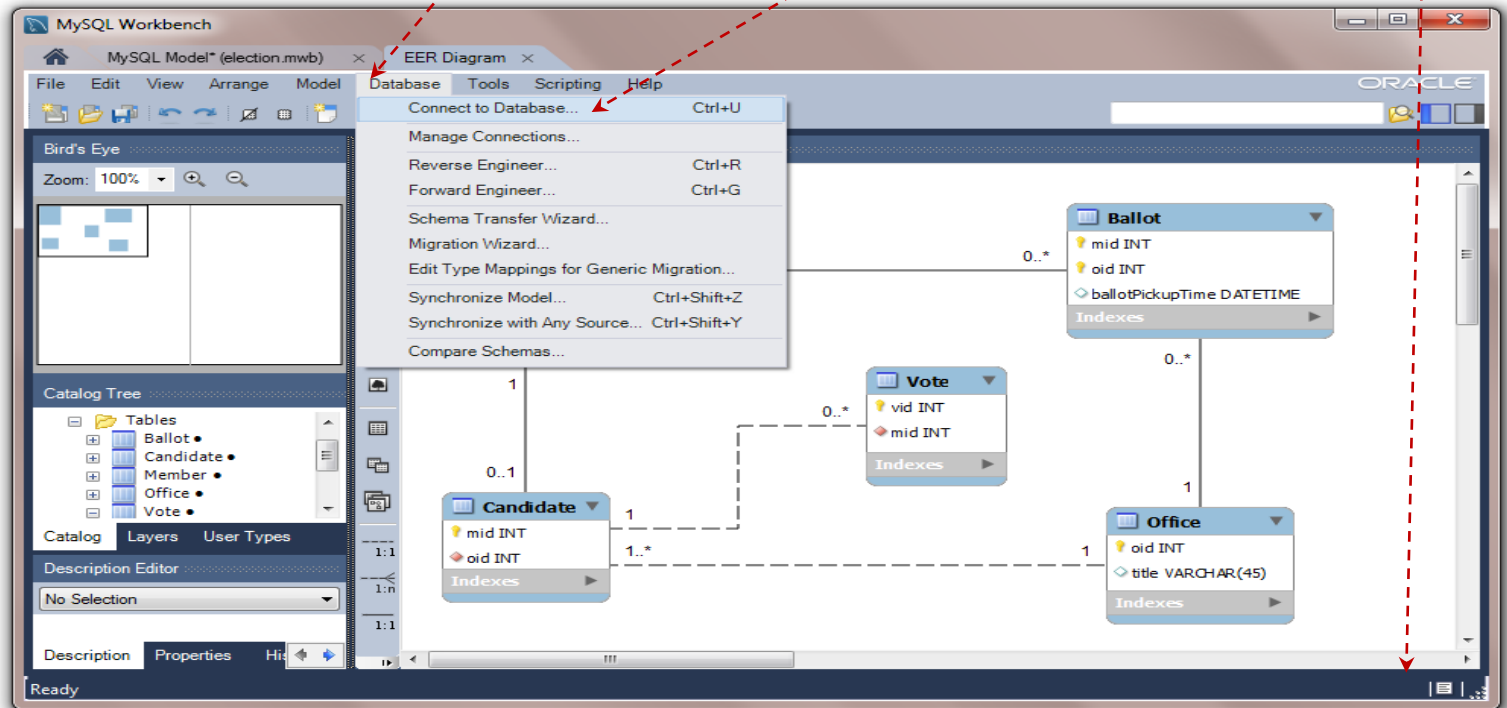
MW connects to the server (**DBMS**) and executes the **SQL** script. The logical design has just been transformed [automatically] into a physical database.

Click button **Close**.

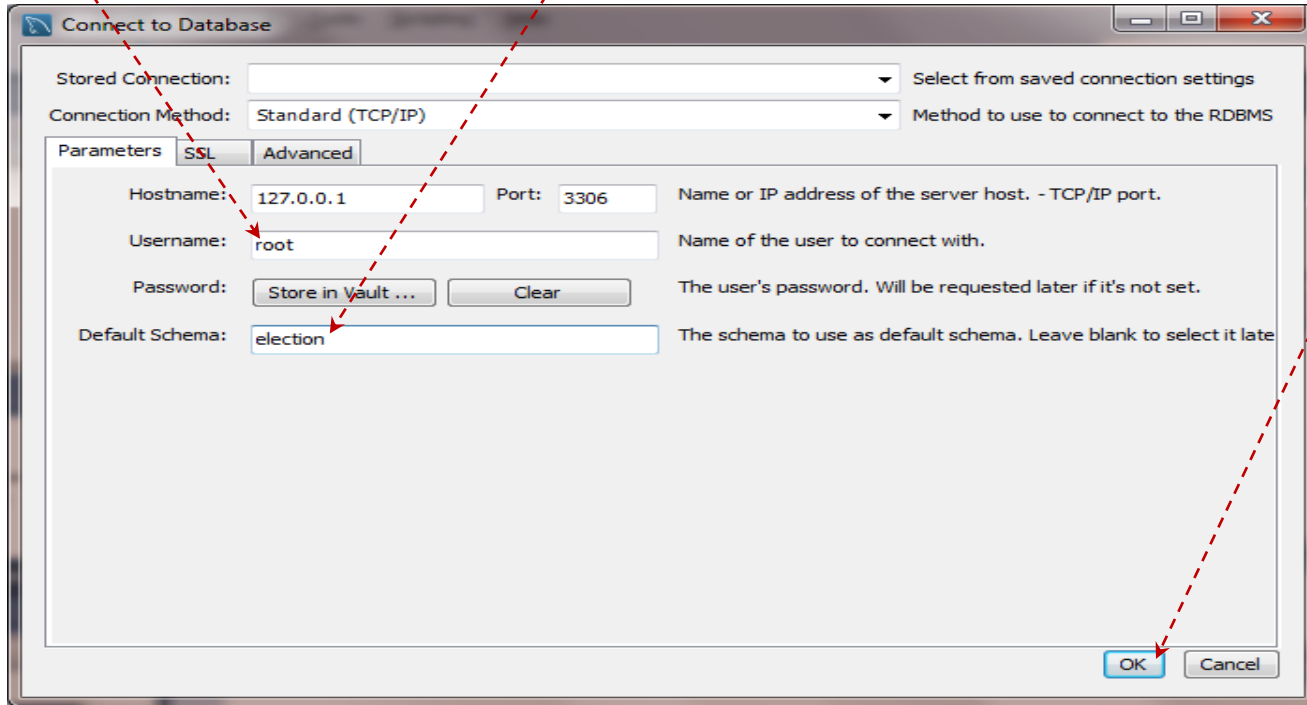


Why don't we connect to the **election** database and add a few records to the tables. The **election.xlsm** workbook contains sample **Insert** statements.

Select (click) menu options **Database** and **Connect to Database ...** .



Use the **root** connection. **Type** name **election** for Default Schema and then **click** button **OK**.



The screenshot shows a 'Connect to Database' dialog box with the following fields and annotations:

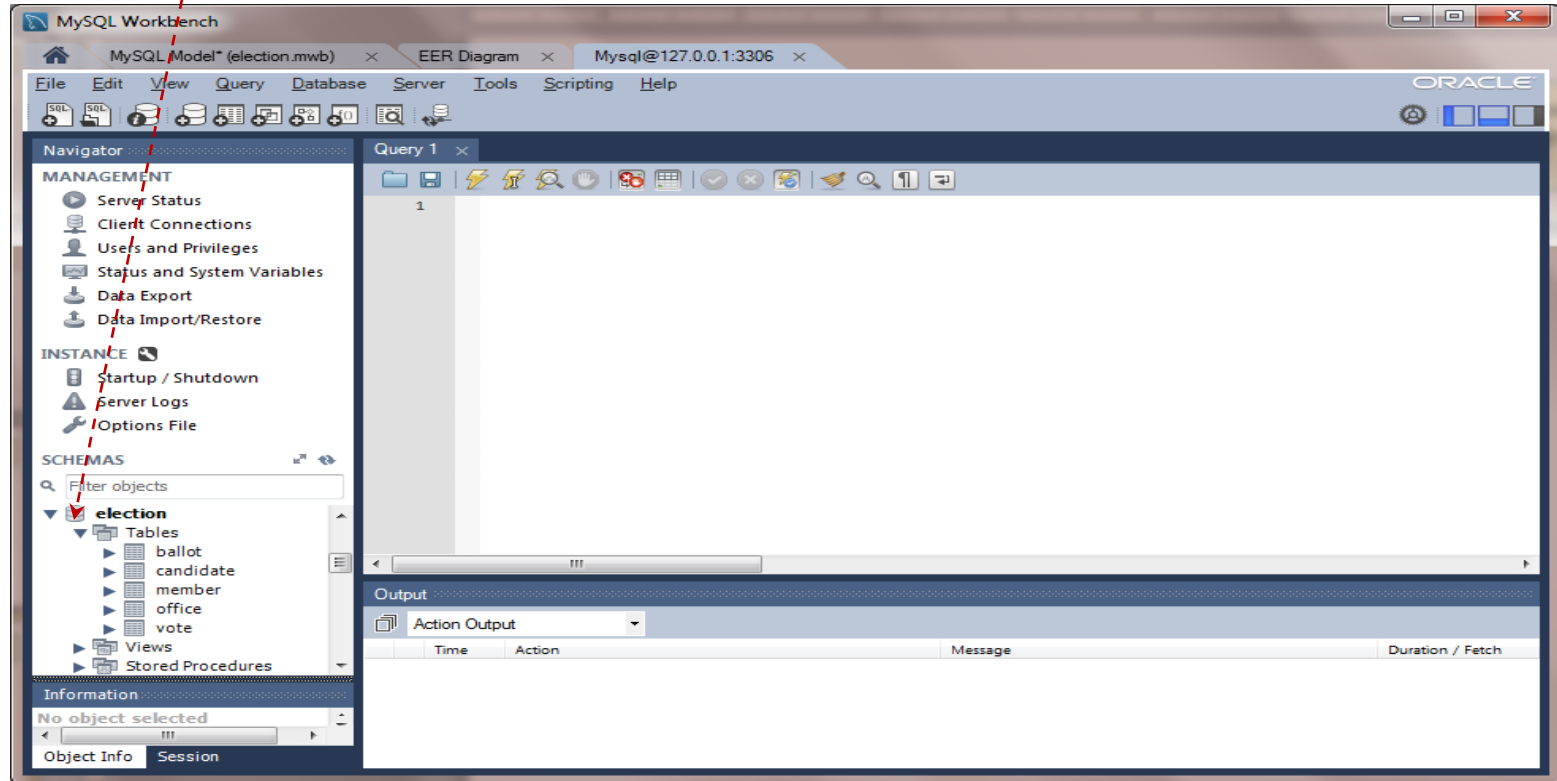
- Stored Connection:** A dropdown menu with a downward arrow. A red dashed arrow points from the word 'root' in the instruction above to this field.
- Connection Method:** A dropdown menu showing 'Standard (TCP/IP)' with a downward arrow. A red dashed arrow points from the word 'Type' in the instruction above to this field.
- Parameters tab:** The 'Parameters' tab is selected, with 'SSL' and 'Advanced' tabs also visible.
- Hostname:** A text field containing '127.0.0.1'. A red dashed arrow points from the word 'election' in the instruction above to this field.
- Port:** A text field containing '3306'.
- Username:** A text field containing 'root'. A red dashed arrow points from the word 'root' in the instruction above to this field.
- Password:** A section with a 'Store in Vault ...' button and a 'Clear' button. A red dashed arrow points from the word 'click' in the instruction above to the 'OK' button.
- Default Schema:** A text field containing 'election'. A red dashed arrow points from the word 'election' in the instruction above to this field.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right. A red dashed arrow points from the word 'OK' in the instruction above to the 'OK' button.

Help text for fields:

- Stored Connection: Select from saved connection settings
- Connection Method: Method to use to connect to the RDBMS
- Hostname: Name or IP address of the server host. - TCP/IP port.
- Username: Name of the user to connect with.
- Password: The user's password. Will be requested later if it's not set.
- Default Schema: The schema to use as default schema. Leave blank to select it later.

MW opens a Query panel and reveals all existing databases in the **SCHEMAS** panel. A full expansion of the **election** tree will show more details, including column names.

Expand the **election** tree in order to reveal the tables.



Switch to the **election.xlsm** workbook and select the **Member** worksheet.

Select and copy all the **Insert Statements**.

election.xlsm [Compatibility Mode] - Microsoft Excel

File Home Insert Page Layout Formulas Data Review View Developer PDF Archi Team

Clipboard Font Alignment Number Cells Editing

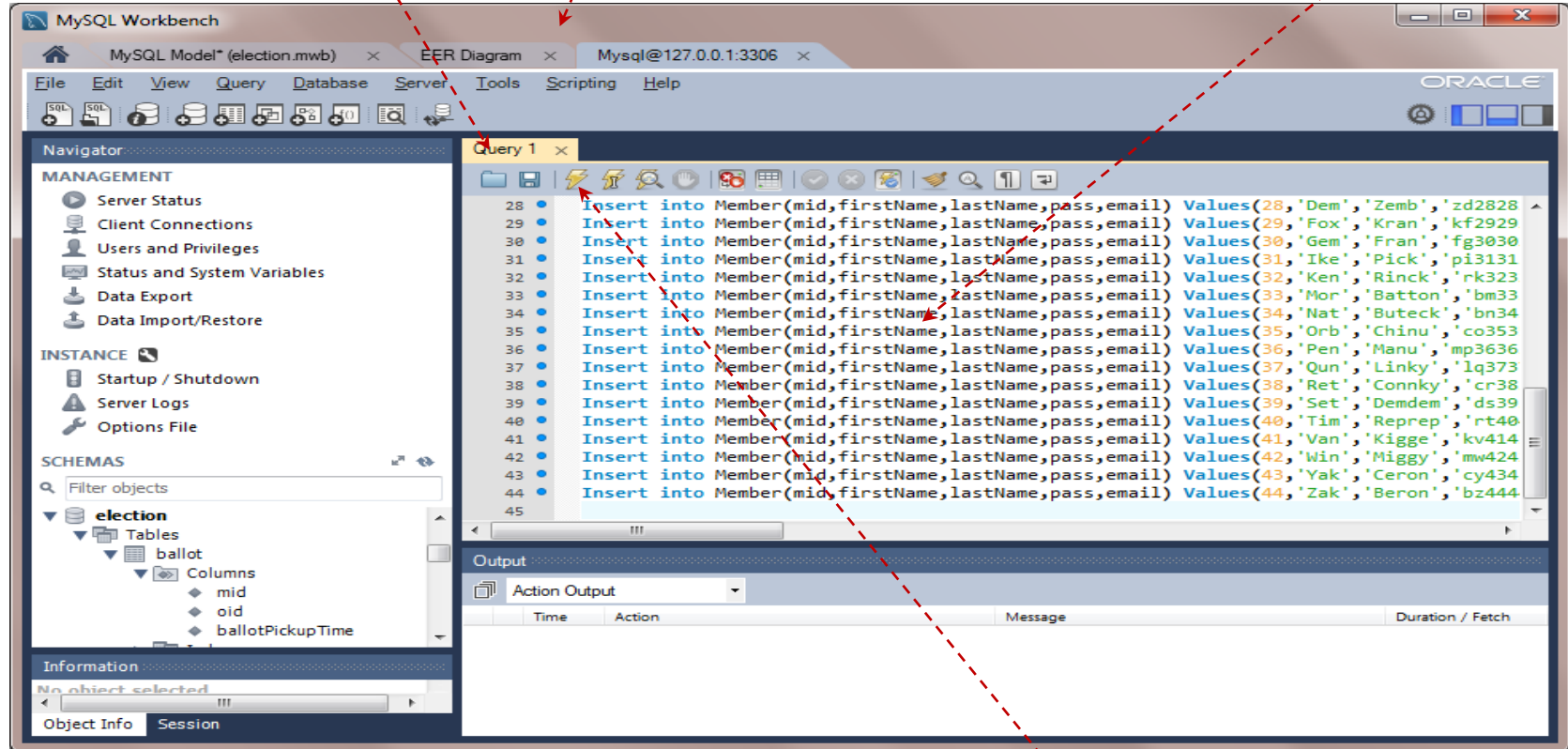
General

qlInsert(\$B\$13,\$A\$14:\$E\$15,A16:E16)

Member

Name	Email	Phone
John Doe	john.doe@member.org	1234567890
Jane Smith	jane.smith@member.org	9876543210
Bob Johnson	bob.johnson@member.org	5555555555
Alice Brown	alice.brown@member.org	1111111111
Charlie White	charlie.white@member.org	2222222222
Diana Prince	diana.prince@member.org	3333333333
Frank Miller	frank.miller@member.org	4444444444
Grace Lee	grace.lee@member.org	6666666666
Henry King	henry.king@member.org	7777777777
Ivy Green	ivy.green@member.org	8888888888
Jack Black	jack.black@member.org	9999999999
Karen Blue	karen.blue@member.org	1010101010
Liam Grey	liam.grey@member.org	2020202020
Mia Gold	mia.gold@member.org	3030303030
Noah Silver	noah.silver@member.org	4040404040
Olivia Bronze	olivia.bronze@member.org	5050505050
Peter Copper	peter.copper@member.org	6060606060
Quinn Iron	quinn.iron@member.org	7070707070
Rachel Steel	rachel.steel@member.org	8080808080
Sam Tin	sam.tin@member.org	9090909090
Tina Lead	tina.lead@member.org	0101010101
Uma Alloy	uma.alloy@member.org	1212121212
Victor Metal	victor.metal@member.org	2323232323
Wendy Nickel	wendy.nickel@member.org	3434343434
Xavier Zinc	xavier.zinc@member.org	4545454545
Yara Cadmium	yara.cadmium@member.org	5656565656
Zoe Silver	zoe.silver@member.org	6767676767
Adam Gold	adam.gold@member.org	7878787878
Eve Bronze	eve.bronze@member.org	8989898989
Frank Copper	frank.copper@member.org	9090909090
Grace Iron	grace.iron@member.org	0101010101
Henry Steel	henry.steel@member.org	1212121212
Ivy Tin	ivy.tin@member.org	2323232323
Jack Lead	jack.lead@member.org	3434343434
Karen Alloy	karen.alloy@member.org	4545454545
Liam Metal	liam.metal@member.org	5656565656
Mia Nickel	mia.nickel@member.org	6767676767
Noah Zinc	noah.zinc@member.org	7878787878
Olivia Cadmium	olivia.cadmium@member.org	8989898989
Peter Silver	peter.silver@member.org	9090909090
Quinn Gold	quinn.gold@member.org	0101010101
Rachel Bronze	rachel.bronze@member.org	1212121212
Sam Copper	sam.copper@member.org	2323232323
Tina Iron	tina.iron@member.org	3434343434
Uma Steel	uma.steel@member.org	4545454545
Victor Tin	victor.tin@member.org	5656565656
Wendy Lead	wendy.lead@member.org	6767676767
Xavier Alloy	xavier.alloy@member.org	7878787878
Yara Metal	yara.metal@member.org	8989898989
Zoe Nickel	zoe.nickel@member.org	9090909090
Adam Zinc	adam.zinc@member.org	0101010101
Eve Cadmium	eve.cadmium@member.org	1212121212
Frank Silver	frank.silver@member.org	2323232323
Grace Gold	grace.gold@member.org	3434343434
Henry Bronze	henry.bronze@member.org	4545454545
Ivy Copper	ivy.copper@member.org	5656565656
Jack Iron	jack.iron@member.org	6767676767
Karen Steel	karen.steel@member.org	7878787878
Liam Tin	liam.tin@member.org	8989898989
Mia Lead	mia.lead@member.org	9090909090
Noah Alloy	noah.alloy@member.org	0101010101
Olivia Metal	olivia.metal@member.org	1212121212
Peter Nickel	peter.nickel@member.org	2323232323
Quinn Zinc	quinn.zinc@member.org	3434343434
Rachel Cadmium	rachel.cadmium@member.org	4545454545
Sam Silver	sam.silver@member.org	5656565656
Tina Gold	tina.gold@member.org	6767676767
Uma Bronze	uma.bronze@member.org	7878787878
Victor Copper	victor.copper@member.org	8989898989
Wendy Iron	wendy.iron@member.org	9090909090
Xavier Steel	xavier.steel@member.org	0101010101
Yara Tin	yara.tin@member.org	1212121212
Zoe Lead	zoe.lead@member.org	2323232323
Adam Alloy	adam.alloy@member.org	3434343434
Eve Metal	eve.metal@member.org	4545454545
Frank Nickel	frank.nickel@member.org	5656565656
Grace Zinc	grace.zinc@member.org	6767676767
Henry Cadmium	henry.cadmium@member.org	7878787878
Ivy Silver	ivy.silver@member.org	8989898989
Jack Gold	jack.gold@member.org	9090909090
Karen Bronze	karen.bronze@member.org	0101010101
Liam Copper	liam.copper@member.org	1212121212
Mia Iron	mia.iron@member.org	2323232323
Noah Steel	noah.steel@member.org	3434343434
Olivia Tin	olivia.tin@member.org	4545454545
Peter Lead	peter.lead@member.org	5656565656
Quinn Alloy	quinn.alloy@member.org	6767676767
Rachel Metal	rachel.metal@member.org	7878787878
Sam Nickel	sam.nickel@member.org	8989898989
Tina Zinc	tina.zinc@member.org	9090909090
Uma Cadmium	uma.cadmium@member.org	0101010101
Victor Silver	victor.silver@member.org	1212121212
Wendy Gold	wendy.gold@member.org	2323232323
Xavier Bronze	xavier.bronze@member.org	3434343434
Yara Copper	yara.copper@member.org	4545454545
Zoe Iron	zoe.iron@member.org	5656565656
Adam Steel	adam.steel@member.org	6767676767
Eve Tin	eve.tin@member.org	7878787878
Frank Lead	frank.lead@member.org	8989898989
Grace Alloy	grace.alloy@member.org	9090909090
Henry Metal	henry.metal@member.org	0101010101
Ivy Nickel	ivy.nickel@member.org	1212121212
Jack Zinc	jack.zinc@member.org	2323232323
Karen Cadmium	karen.cadmium@member.org	3434343434
Liam Silver	liam.silver@member.org	4545454545
Mia Gold	mia.gold@member.org	5656565656
Noah Bronze	noah.bronze@member.org	6767676767
Olivia Copper	olivia.copper@member.org	7878787878
Peter Iron	peter.iron@member.org	8989898989
Quinn Steel	quinn.steel@member.org	9090909090
Rachel Tin	rachel.tin@member.org	0101010101
Sam Lead	sam.lead@member.org	1212121212
Tina Alloy	tina.alloy@member.org	2323232323
Uma Metal	uma.metal@member.org	3434343434
Victor Nickel	victor.nickel@member.org	4545454545
Wendy Zinc	wendy.zinc@member.org	5656565656
Xavier Cadmium	xavier.cadmium@member.org	6767676767
Yara Silver	yara.silver@member.org	7878787878
Zoe Gold	zoe.gold@member.org	8989898989
Adam Bronze	adam.bronze@member.org	9090909090
Eve Copper	eve.copper@member.org	0101010101
Frank Iron	frank.iron@member.org	1212121212
Grace Steel	grace.steel@member.org	2323232323
Henry Tin	henry.tin@member.org	3434343434
Ivy Lead	ivy.lead@member.org	4545454545
Jack Alloy	jack.alloy@member.org	5656565656
Karen Metal	karen.metal@member.org	6767676767
Liam Nickel	liam.nickel@member.org	7878787878
Mia Zinc	mia.zinc@member.org	8989898989
Noah Cadmium	noah.cadmium@member.org	9090909090
Olivia Silver	olivia.silver@member.org	0101010101
Peter Gold	peter.gold@member.org	1212121212
Quinn Bronze	quinn.bronze@member.org	2323232323
Rachel Copper	rachel.copper@member.org	3434343434
Sam Iron	sam.iron@member.org	4545454545
Tina Steel	tina.steel@member.org	5656565656
Uma Tin	uma.tin@member.org	6767676767
Victor Lead	victor.lead@member.org	7878787878
Wendy Alloy	wendy.alloy@member.org	8989898989
Xavier Metal	xavier.metal@member.org	9090909090
Yara Nickel	yara.nickel@member.org	0101010101
Zoe Zinc	zoe.zinc@member.org	1212121212
Adam Cadmium	adam.cadmium@member.org	2323232323
Eve Silver	eve.silver@member.org	3434343434
Frank Gold	frank.gold@member.org	4545454545
Grace Bronze	grace.bronze@member.org	5656565656
Henry Copper	henry.copper@member.org	6767676767
Ivy Iron	ivy.iron@member.org	7878787878
Jack Steel	jack.steel@member.org	8989898989
Karen Tin	karen.tin@member.org	9090909090
Liam Lead	liam.lead@member.org	0101010101
Mia Alloy	mia.alloy@member.org	1212121212
Noah Metal	noah.metal@member.org	2323232323
Olivia Nickel	olivia.nickel@member.org	3434343434
Peter Zinc	peter.zinc@member.org	4545454545
Quinn Cadmium	quinn.cadmium@member.org	5656565656
Rachel Silver	rachel.silver@member.org	6767676767
Sam Gold	sam.gold@member.org	7878787878
Tina Bronze	tina.bronze@member.org	8989898989
Uma Copper	uma.copper@member.org	9090909090
Victor Iron	victor.iron@member.org	0101010101
Wendy Steel	wendy.steel@member.org	1212121212
Xavier Tin	xavier.tin@member.org	2323232323
Yara Lead	yara.lead@member.org	3434343434
Zoe Alloy	zoe.alloy@member.org	4545454545
Adam Metal	adam.metal@member.org	5656565656
Eve Nickel	eve.nickel@member.org	6767676767
Frank Zinc	frank.zinc@member.org	7878787878
Grace Cadmium	grace.cadmium@member.org	8989898989
Henry Silver	henry.silver@member.org	9090909090
Ivy Gold	ivy.gold@member.org	0101010101
Jack Bronze	jack.bronze@member.org	1212121212
Karen Copper	karen.copper@member.org	2323232323
Liam Iron	liam.iron@member.org	3434343434
Mia Steel	mia.steel@member.org	4545454545
Noah Tin	noah.tin@member.org	5656565656
Olivia Lead	olivia.lead@member.org	6767676767
Peter Alloy	peter.alloy@member.org	7878787878
Quinn Metal	quinn.metal@member.org	8989898989
Rachel Nickel	rachel.nickel@member.org	9090909090
Sam Zinc	sam.zinc@member.org	0101010101
Tina Cadmium	tina.cadmium@member.org	1212121212
Uma Silver	uma.silver@member.org	2323232323
Victor Gold	victor.gold@member.org	3434343434
Wendy Bronze	wendy.bronze@member.org	4545454545
Xavier Copper	xavier.copper@member.org	5656565656
Yara Iron	yara.iron@member.org	6767676767
Zoe Steel	zoe.steel@member.org	7878787878
Adam Tin	adam.tin@member.org	8989898989
Eve Lead	eve.lead@member.org	9090909090
Frank Alloy	frank.alloy@member.org	0101010101
Grace Metal	grace.metal@member.org	1212121212
Henry Nickel	henry.nickel@member.org	2323232323
Ivy Zinc	ivy.zinc@member.org	3434343434
Jack Cadmium	jack.cadmium@member.org	4545454545
Karen Silver	karen.silver@member.org	5656565656
Liam Gold	liam.gold@member.org	6767676767
Mia Bronze	mia.bronze@member.org	7878787878
Noah Copper	noah.copper@member.org	8989898989
Olivia Iron	olivia.iron@member.org	9090909090
Peter Steel	peter.steel@member.org	0101010101
Quinn Tin	quinn.tin@member.org	1212121212
Rachel Lead	rachel.lead@member.org	2323232323
Sam Alloy	sam.alloy@member.org	3434343434
Tina Metal	tina.metal@member.org	4545454545
Uma Nickel	uma.nickel@member.org	5656565656
Victor Zinc	victor.zinc@member.org	6767676767
Wendy Cadmium	wendy.cadmium@member.org	7878787878
Xavier Silver	xavier.silver@member.org	8989898989
Yara Gold	yara.gold@member.org	9090909090
Zoe Bronze	zoe.bronze@member.org	0101010101
Adam Copper	adam.copper@member.org	1212121212
Eve Iron	eve.iron@member.org	2323232323
Frank Steel	frank.steel@member.org	3434343434
Grace Tin	grace.tin@member.org	4545454545
Henry Lead	henry.lead@member.org	5656565656
Ivy Alloy	ivy.alloy@member.org	6767676767
Jack Metal	jack.metal@member.org	7878787878
Karen Nickel	karen.nickel@member.org	8989898989
Liam Zinc	liam.zinc@member.org	9090909090
Mia Cadmium	mia.cadmium@member.org	0101010101
Noah Silver	noah.silver@member.org	1212121212
Olivia Gold	olivia.gold@member.org	2323232323
Peter Bronze	peter.bronze@member.org	3434343434
Quinn Copper	quinn.copper@member.org	4545454545
Rachel Iron	rachel.iron@member.org	5656565656
Sam Steel	sam.steel@member.org	6767676767
Tina Tin	tina.tin@member.org	7878787878
Uma Lead	uma.lead@member.org	8989898989
Victor Alloy	victor.alloy@member.org	9090909090
Wendy Metal	wendy.metal@member.org	0101010101
Xavier Nickel	xavier.nickel@member.org	1212121212
Yara Zinc	yara.zinc@member.org	2323232323
Zoe Cadmium	zoe.cadmium@member.org	3434343434
Adam Silver	adam.silver@member.org	4545454545
Eve Gold	eve.gold@member.org	5656565656
Frank Bronze	frank.bronze@member.org	6767676767
Grace Copper	grace.copper@member.org	7878787878
Henry Iron	henry.iron@member.org	8989898989
Ivy Steel	ivy.steel@member.org	9090909090
Jack Tin	jack.tin@member.org	0101010101
Karen Lead	karen.lead@member.org	1212121212
Liam Alloy	liam.alloy@member.org	2323232323
Mia Metal	mia.metal@member.org	3434343434
Noah Nickel	noah.nickel@member.org	4545454545
Olivia Zinc	olivia.zinc@member.org	5656565656
Peter Cadmium	peter.cadmium@member.org	6767676767
Quinn Silver	quinn.silver@member.org	7878787878
Rachel Gold	rachel.gold@member.org	8989898989
Sam Bronze	sam.bronze@member.org	9090909090
Tina Copper	tina.copper@member.org	0101010101
Uma Iron	uma.iron@member.org	1212121212
Victor Steel	victor.steel@member.org	2323232323
Wendy Tin	wendy.tin@member.org	3434343434
Xavier Lead	xavier.lead@member.org	4545454545
Yara Alloy	yara.alloy@member.org	5656565656
Zoe Metal	zoe.metal@member.org	6767676767
Adam Nickel	adam.nickel@member.org	7878787878
Eve Zinc	eve.zinc@member.org	8989898989
Frank Cadmium	frank.cadmium@member.org	9090909090
Grace Silver	grace.silver@member.org	0101010101
Henry Gold	henry.gold@member.org	1212121212
Ivy Bronze	ivy.bronze@member.org	2323232323
Jack Copper	jack.copper@member.org	3434343434
Karen Iron	karen.iron@member.org	4545454545
Liam Steel	liam.steel@member.org	5656565656
Mia Tin	mia.tin@member.org	6767676767
Noah Lead	noah.lead@member.org	7878787878
Olivia Alloy	olivia.alloy@member.org	8989898989
Peter Metal	peter.metal@member.org	9090909090
Quinn Nickel	quinn.nickel@member.org	0101010101
Rachel Zinc	rachel.zinc@member.org	1212121212
Sam Cadmium	sam.cadmium@member.org	2323232323
Tina Silver	tina.silver@member.org	3434343434
Uma Gold	uma.gold@member.org	4545454545
Victor Bronze	victor.bronze@member.org	5656565656
Wendy Copper	wendy.copper@member.org	6767676767
Xavier Iron	xavier.iron@member.org	7878787878
Yara Steel	yara.steel@member.org	8989898989
Zoe Tin	zoe.tin@member.org	9090909090
Adam Lead	adam.lead@member.org	0101010101
Eve Alloy	eve.alloy@member.org	1212121212
Frank Metal	frank.metal@member.org	2323232323
Grace Nickel	grace.nickel@member.org	3434343434
Henry Zinc	henry.zinc@member.org	4545454545
Ivy Cadmium	ivy.cadmium@member.org	5656565656
Jack Silver	jack.silver@member.org	6767676767
Karen Gold	karen.gold@member.org	7878787878
Liam Bronze	liam.bronze@member.org	8989898989
Mia Copper	mia.copper@member.org	9090909090
Noah Iron	noah.iron@member.org	0101010101
Olivia Steel	olivia.steel@member.org	1212121212
Peter Tin	peter.tin@member.org	2323232323
Quinn Lead	quinn.lead@member.org	3434343434
Rachel Alloy	rachel.alloy@member.org	4545454545
Sam Metal	sam.metal@member.org	5656565656
Tina Nickel	tina.nickel@member.org	6767676767
Uma Zinc	uma.zinc@member.org	7878787878
Victor Cadmium	victor.cadmium@member.org	8989898989
Wendy Silver	wendy.silver@member.org	9090909090
Xavier Gold	xavier.gold@member.org	0101010101
Yara Bronze	yara.bronze@member.org	1212121212
Zoe Copper	zoe.copper@member.org	2323232323
Adam Iron	adam.iron@member.org	3434343434
Eve Steel	eve.steel@member.org	4545454545
Frank Tin	frank.tin@member.org	5656565656
Grace Lead	grace.lead@member.org	6767676767
Henry Alloy	henry.alloy@member.org	7878787878
Ivy Metal	ivy.metal@member.org	8989898989
Jack Nickel	jack.nickel@member.org	9090909090
Karen Zinc	karen.zinc@member.org	0101010101
Liam Cadmium	liam.cadmium@member.org	1212121212
Mia Silver	mia.silver@member.org	2323232323
Noah Gold	noah.gold@member.org	3434343434
Olivia Bronze	olivia.bronze@member.org	4545454545
Peter Copper	peter.copper@member.org	5656565656
Quinn Iron		

Switch back to the Query panel in MW and press Ctrl+V, in order to paste the Insert statements.



With all the Insert statements in the Query panel, click the Execute button.

The **Output** panel shows feedback for each of the executes statements.

The screenshot displays the MySQL Workbench interface. The 'Query 1' panel contains a series of 17 'Insert into Member' statements, each with unique values for mid, firstName, lastName, pass, and email. The 'Output' panel at the bottom shows the results of the last four statements (lines 41-44), each indicating '1 row(s) affected' with a duration of 0.000 sec. A red dashed arrow points from the text 'The Output panel shows feedback for each of the executes statements.' to the 'Output' panel. A blue dashed arrow points from the text 'In order to clear the Query panel, select all the Insert statements (press Ctrl+A) and press the Delete key.' to the 'Query 1' panel.

Query 1

```
28 Insert into Member(mid,firstName,lastName,pass,email) Values(28,'Dem','Zemb','zd2828
29 Insert into Member(mid,firstName,lastName,pass,email) Values(29,'Fox','Kran','kf2929
30 Insert into Member(mid,firstName,lastName,pass,email) Values(30,'Gem','Fran','fg3030
31 Insert into Member(mid,firstName,lastName,pass,email) Values(31,'Ike','Pick','pi3131
32 Insert into Member(mid,firstName,lastName,pass,email) Values(32,'Ken','Rinck','rk3223
33 Insert into Member(mid,firstName,lastName,pass,email) Values(33,'Mor','Botton','bm33
34 Insert into Member(mid,firstName,lastName,pass,email) Values(34,'Nat','Buteck','bn34
35 Insert into Member(mid,firstName,lastName,pass,email) Values(35,'Orb','Chinu','co353
36 Insert into Member(mid,firstName,lastName,pass,email) Values(36,'Pen','Manu','mp3636
37 Insert into Member(mid,firstName,lastName,pass,email) Values(37,'Qun','Linky','lq373
38 Insert into Member(mid,firstName,lastName,pass,email) Values(38,'Ret','Connky','cr38
39 Insert into Member(mid,firstName,lastName,pass,email) Values(39,'Set','Demdem','ds39
40 Insert into Member(mid,firstName,lastName,pass,email) Values(40,'Tim','Reprep','rt40
41 Insert into Member(mid,firstName,lastName,pass,email) Values(41,'Van','Kigge','kv414
42 Insert into Member(mid,firstName,lastName,pass,email) Values(42,'Win','Miggy','mw424
43 Insert into Member(mid,firstName,lastName,pass,email) Values(43,'Yak','Ceron','cy434
44 Insert into Member(mid,firstName,lastName,pass,email) Values(44,'Zak','Beron','bz444
45
```

Output

	Time	Action	Message	Duration / Fetch
✓ 41	11:50:03	Insert into Member(mid,firstName,lastName,pa...	1 row(s) affected	0.000 sec
✓ 42	11:50:03	Insert into Member(mid,firstName,lastName,pa...	1 row(s) affected	0.000 sec
✓ 43	11:50:03	Insert into Member(mid,firstName,lastName,pa...	1 row(s) affected	0.015 sec
✓ 44	11:50:03	Insert into Member(mid,firstName,lastName,pa...	1 row(s) affected	0.000 sec

In order to clear the **Query** panel, **select** all the **Insert** statements (**press Ctrl+A**) and **press** the **Delete** key.

In a similar way, bring the other **Insert** statements (for tables **Office**, **Ballot**, **Candidate** and **Vote**) from the Excel workbook, **election.xlsm** , to the **MW's Query** panel. Execute the statements and explore the database by running a few queries.

After all the sample records have been added, run a few queries. The first example shows all positions stored in the **Office** table.

Type statement **SELECT * FROM Office;** into the **Query** panel and *click* the **Execute** button.

The screenshot shows a database query tool interface. At the top, a toolbar contains various icons, including a lightning bolt icon for executing queries. Below the toolbar, the query editor displays the SQL statement `SELECT * FROM Office;`. The results pane shows a table with two columns, `oid` and `title`, containing five rows of data. Below the results pane, the `Office 4` tab is visible. At the bottom, the `Output` pane shows the query execution status, including the time, action, message, and duration.

oid	title
1	President
2	Vice-President
3	Treasurer
4	Editor
5	Web Manager
NULL	NULL

Time	Action	Message	Duration / Fetch
12:23:05	SELECT * FROM Office LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Query result!

Query status!

The second example shows all the candidates.

Type statement

`SELECT * FROM Member WHERE mid IN (SELECT mid FROM Candidate);`
into the **Query** panel
and **click** the **Execute** button.

The screenshot shows a database query tool interface. At the top, a tab labeled 'Query 1' is active. Below it is a toolbar with various icons, including a magnifying glass and a play button. The query editor displays the SQL statement: `SELECT * FROM Member WHERE mid IN (SELECT mid FROM Candidate);`. Below the query editor is a 'Result Set Filter' section. The main area displays a table with the following data:

	mid	firstName	lastName	pass	email
▶	3	Cir	Cus	cc22211	ccus@misor.org
	13	Mac	Intosh	mi77788	mintosh@misor.org
	22	Vin	Cent	cv00001	vcent@misor.org
	4	Don	Donski	dd00413	ddonski@misor.org
	10	Jan	Osik	jo33322	josik@misor.org
	15	Ore	Gon	og33344	ogon@misor.org
	24	Xer	Xes	xx12345	xxes@misor.org
	8	Hal	Bar	hb01011	hbar@misor.org

Below the table is a 'Member 5' tab. At the bottom, there is an 'Output' section with a dropdown menu set to 'Action Output'. The output table shows the following data:

	Time	Action	Message	Duration / Fetch
✓	1 12:23:05	SELECT * FROM Office LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
✓	2 12:30:00	SELECT * FROM Member WHERE mid IN (SE...	20 row(s) returned	0.000 sec / 0.000 sec

The same result can be done by an explicit JOIN statement:

`SELECT * FROM Member JOIN
Candidate USING(mid);`

The third example shows all the candidates and offices they are running for.

Type statement

`SELECT * FROM Member JOIN Candidate USING(mid) JOIN Office USING(oid);`
into the **Query** panel
and **click** the **Execute** button.

The screenshot shows a database query tool interface. At the top, a toolbar contains various icons, including a lightning bolt icon for executing a query. Below the toolbar, the 'Query 1' panel displays the SQL statement: `SELECT * FROM Member JOIN Candidate USING(mid) JOIN Office USING(oid);`. A red dashed arrow points from the text 'click the Execute button' in the preceding block to the lightning bolt icon in the toolbar. Below the query panel, the 'Result Set Filter' section is empty. The main area displays a table of results with the following columns: oid, mid, firstName, lastName, pass, email, and title. The table contains 10 rows of data. Below the table, the 'Result 7' panel is visible, showing a 'Read Only' status. The 'Output' panel at the bottom shows a log of actions, including the execution of the query and the number of rows returned.

oid	mid	firstName	lastName	pass	email	title
1	3	Cir	Cus	cc22211	ccus@misor.org	President
1	13	Mac	Intosh	mi77788	mintosh@misor.org	President
1	22	Vin	Cent	cv00001	vcent@misor.org	President
2	4	Don	Donski	dd00413	ddonski@misor.org	Vice-President
2	10	Jan	Osik	jo33322	josik@misor.org	Vice-President
2	15	Ore	Gon	og33344	ogon@misor.org	Vice-President
2	24	Xer	Xes	xx12345	xxes@misor.org	Vice-President
3	8	Hal	Bar	hb01011	hbar@misor.org	Treasurer

	Time	Action	Message	Duration / Fetch
✓	2 12:30:00	SELECT * FROM Member WHERE mid IN (...)	20 row(s) returned	0.000 sec / 0.000 sec
✓	3 12:34:39	SELECT * FROM Member JOIN Candidate U...	20 row(s) returned	0.000 sec / 0.000 sec
✓	4 12:35:54	SELECT * FROM Member JOIN Candidate U...	20 row(s) returned	0.000 sec / 0.000 sec

The fourth example shows all the members who have not yet picked any of their ballots.

Type statement

```
SELECT mid, lastName FROM Member WHERE mid NOT IN (SELECT mid FROM Ballot);
```

into the **Query** panel

and **click** the **Execute** button.

The screenshot shows a database query tool interface. At the top, a 'Query 1' panel contains the SQL statement: `SELECT mid, lastName FROM Member WHERE mid NOT IN (SELECT mid FROM Ballot);`. Below this, the 'Result Set Filter' section displays a table with the following data:

	mid	lastName
▶	5	Lady
	19	Bee
	29	Kran
	42	Miggy
*	NULL	NULL

At the bottom, the 'Member 9' panel shows the 'Output' section with a table of execution logs:

	Time	Action	Message	Duration / Fetch
✓	4 12:35:54	SELECT * FROM Member JOIN Candidate U...	20 row(s) returned	0.000 sec / 0.000 sec
✓	5 13:25:40	SELECT mid, lastName FROM Member WH...	5 row(s) returned	0.000 sec / 0.000 sec
✓	6 13:38:36	SELECT mid, lastName FROM Member WH...	4 row(s) returned	0.000 sec / 0.000 sec

Using an existential query:

```
SELECT mid, lastName
```

```
FROM Member
```

```
WHERE NOT EXISTS
```

```
(SELECT * FROM Ballot WHERE Ballot.mid=Member.mid);
```

This final example shows the number of positions the members have already voted for.

Type statement

`SELECT mid, Count(Ballot.mid) FROM Ballot GROUP BY mid;`
into the **Query** panel
and **click** the **Execute** button.

The screenshot shows a database query interface. At the top, a toolbar contains various icons, including a lightning bolt icon for executing queries. Below the toolbar, the query editor displays the SQL statement: `SELECT mid, Count(Ballot.mid) FROM Ballot GROUP BY mid;`. The results are shown in a table with two columns: `mid` and `Count(Ballot.mid)`. The table contains 10 rows of data. A green dashed arrow points from the text 'Notice that one of the members have picked up only one ballot.' to the row where `mid` is 45 and `Count(Ballot.mid)` is 1. Below the results table, the 'Output' panel shows a log of database actions, including the execution of the query and the number of rows returned.

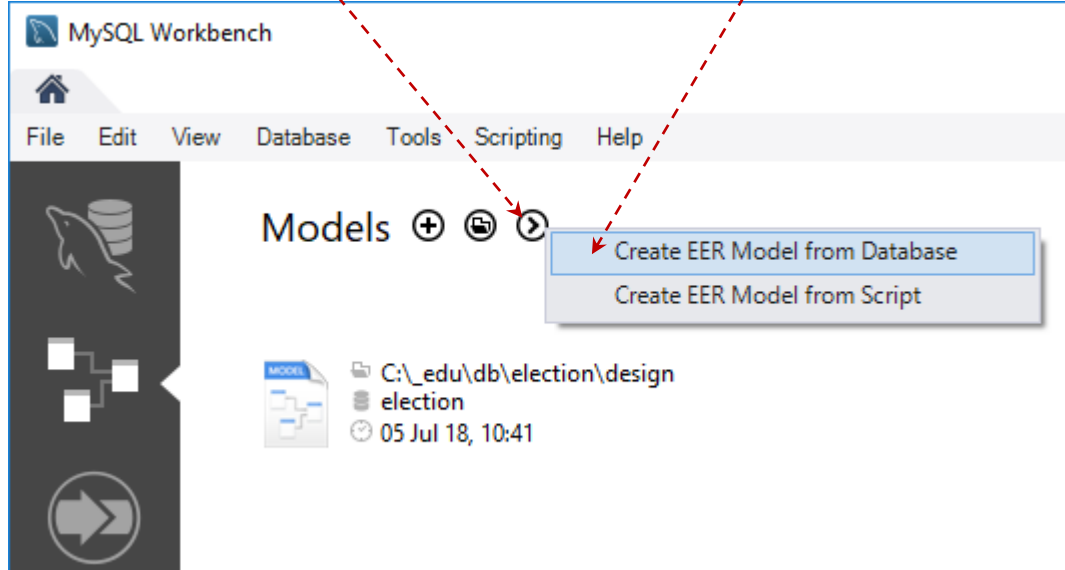
mid	Count(Ballot.mid)
38	5
39	5
40	5
41	5
43	5
44	5
45	1

Time	Action	Message	Duration / Fetch
5 13:25:40	SELECT mid, lastName FROM Member WH...	5 row(s) returned	0.000 sec / 0.000 sec
6 13:38:36	SELECT mid, lastName FROM Member WH...	4 row(s) returned	0.000 sec / 0.000 sec
7 13:43:33	SELECT mid, Count(Ballot.mid) FROM Ballot ...	41 row(s) returned	0.000 sec / 0.000 sec

This concludes the **Forward Engineering** case, including the logical design and schema generation, extended by populating the databases with sample records and running queries. The remaining slides show how to **Reverse Engineer** an existing [physical] database. It is a good time to save and close **MW (MySQL Workbench)**. Then start a new instance of **MW** (a fresh start).

Start MySQL Workbench and select the **Data Model** option (see slides [4](#) and [5](#) for details).

Click Button  and select the **Create EER Model from Database**.



Use your **root** connection and **click** button **Next**.

The screenshot shows the 'Reverse Engineer Database' dialog box. A green callout box at the top contains the text 'Use your **root** connection and **click** button **Next**.' Two red dashed arrows originate from this box: one points to the 'root' text in the 'Username' field, and the other points to the 'Next' button at the bottom right of the dialog.

Reverse Engineer Database

Connection Options

- Connect to DBMS
- Select Schemas
- Retrieve Objects
- Select Objects
- Reverse Engineer
- Results

Set Parameters for Connecting to a DBMS

Stored Connection: Select from saved connection settings

Connection Method: Method to use to connect to the RDBMS

Parameters ☒ SSL ☐ Advanced

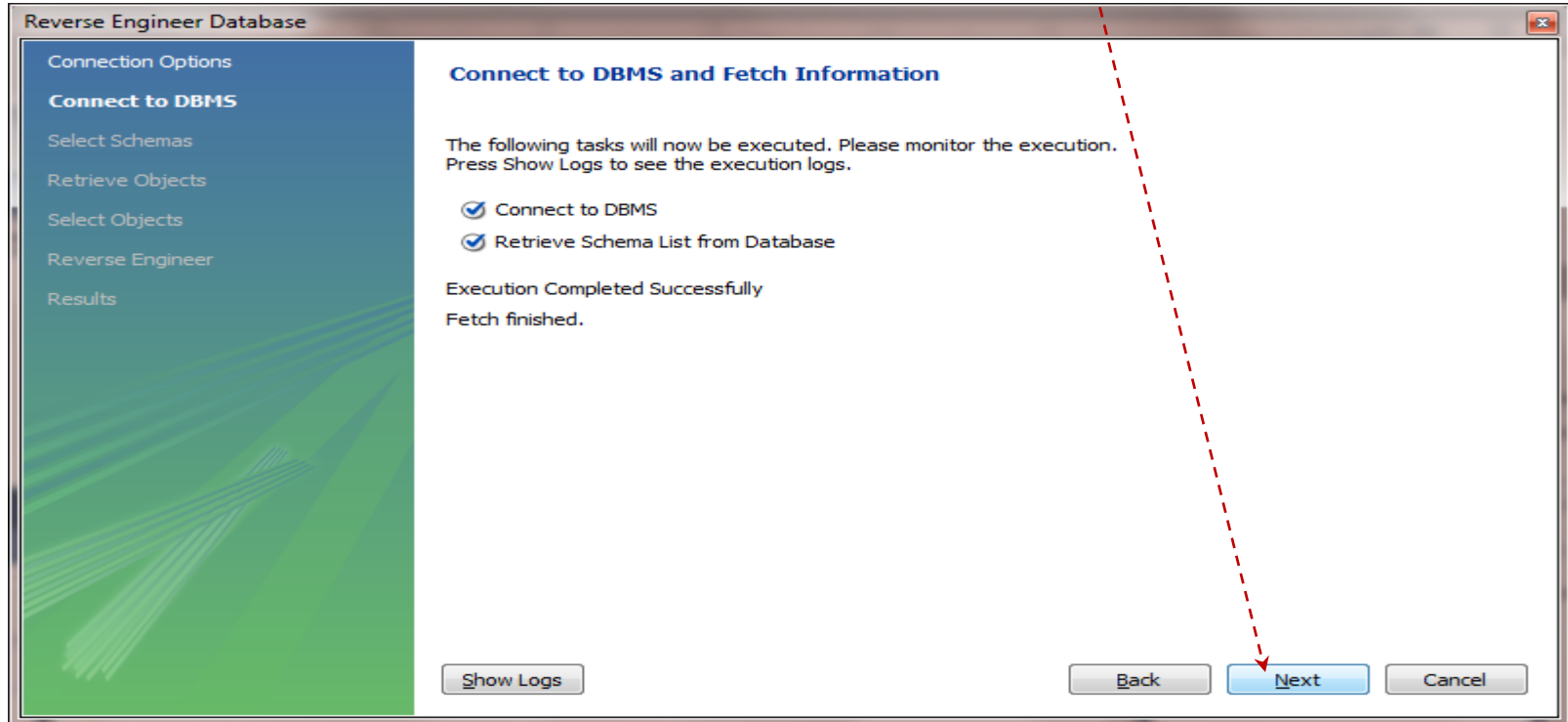
Hostname: Port: Name or IP address of the server host. - TCP/IP p

Username: Name of the user to connect with.

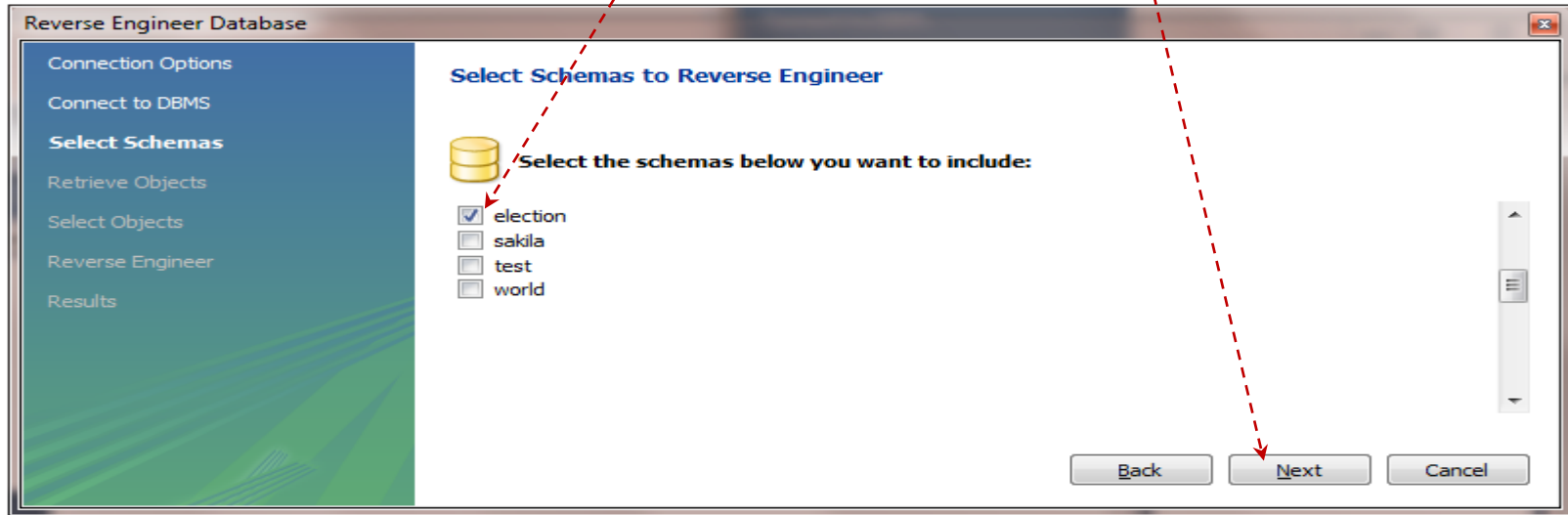
Password: The user's password. Will be requested later if it's

MW is connecting to the server and retrieving the existing schemas from the **DBMS**.

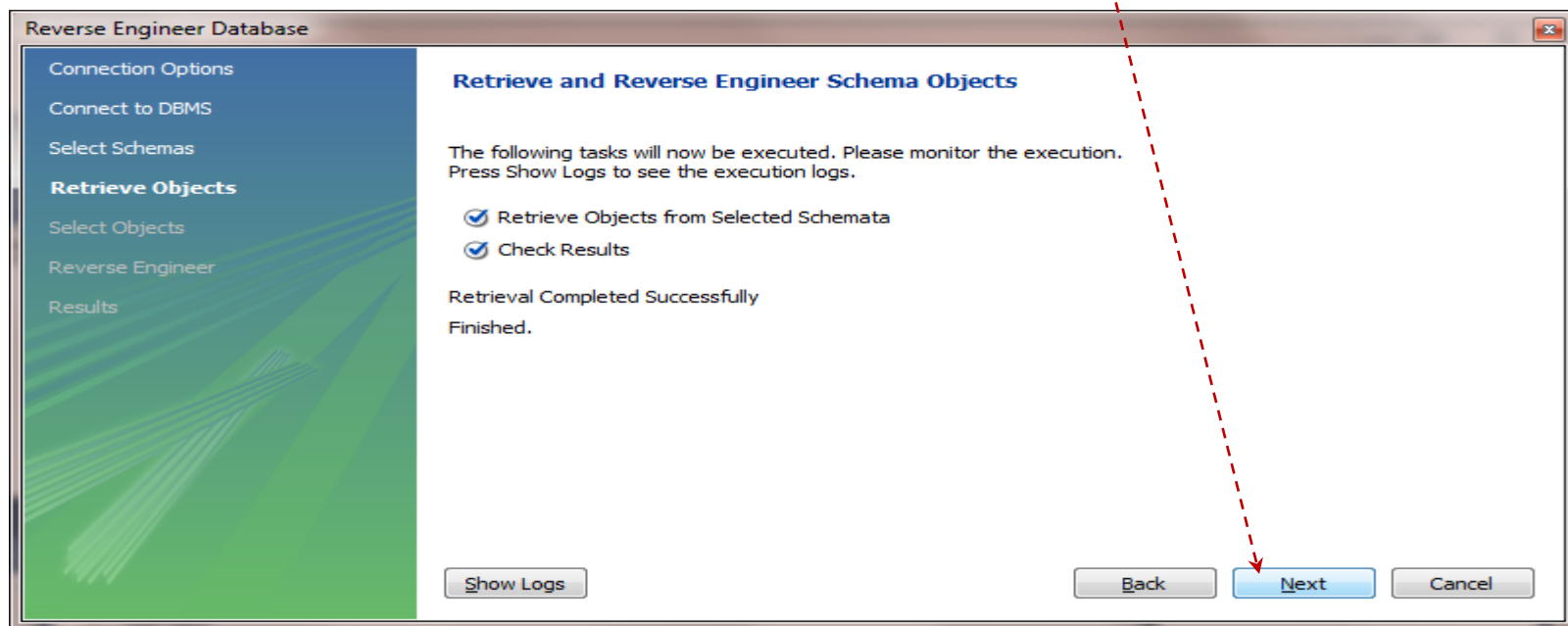
Click button **Next**.



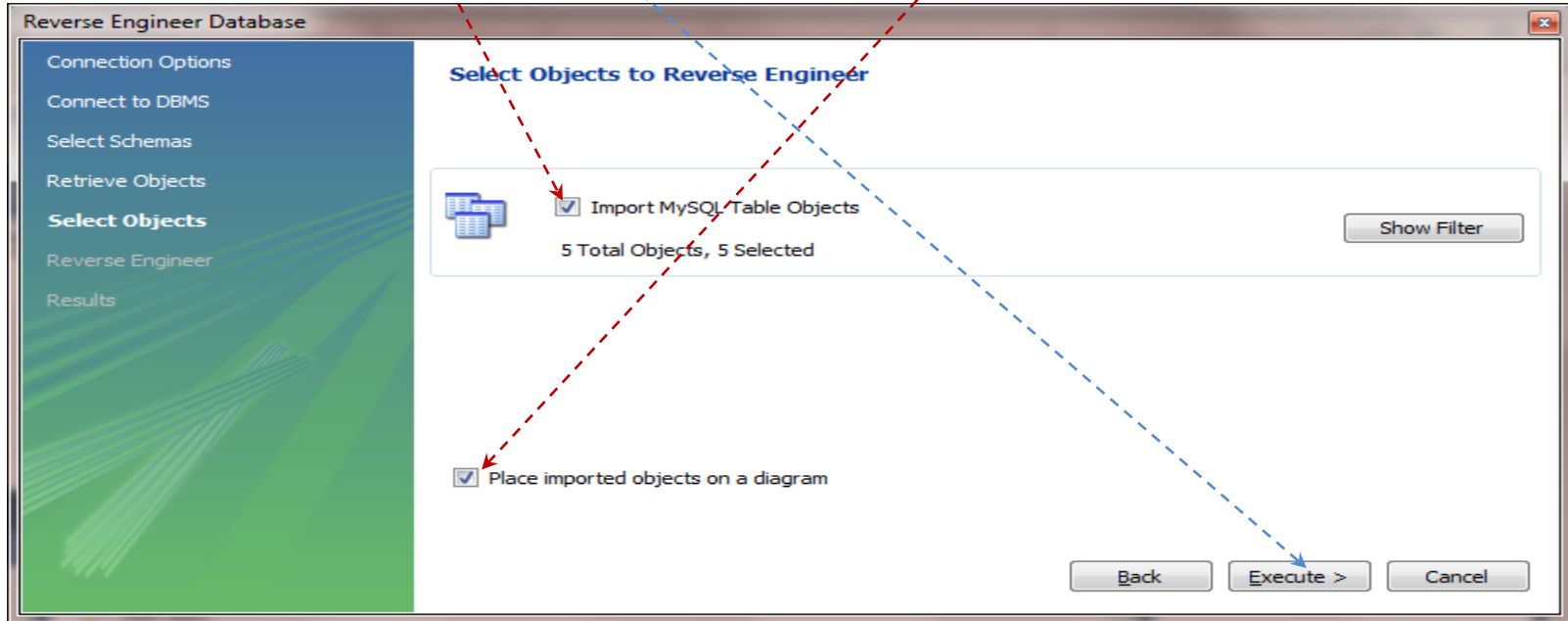
Select database **election** and **click** button **Next**.



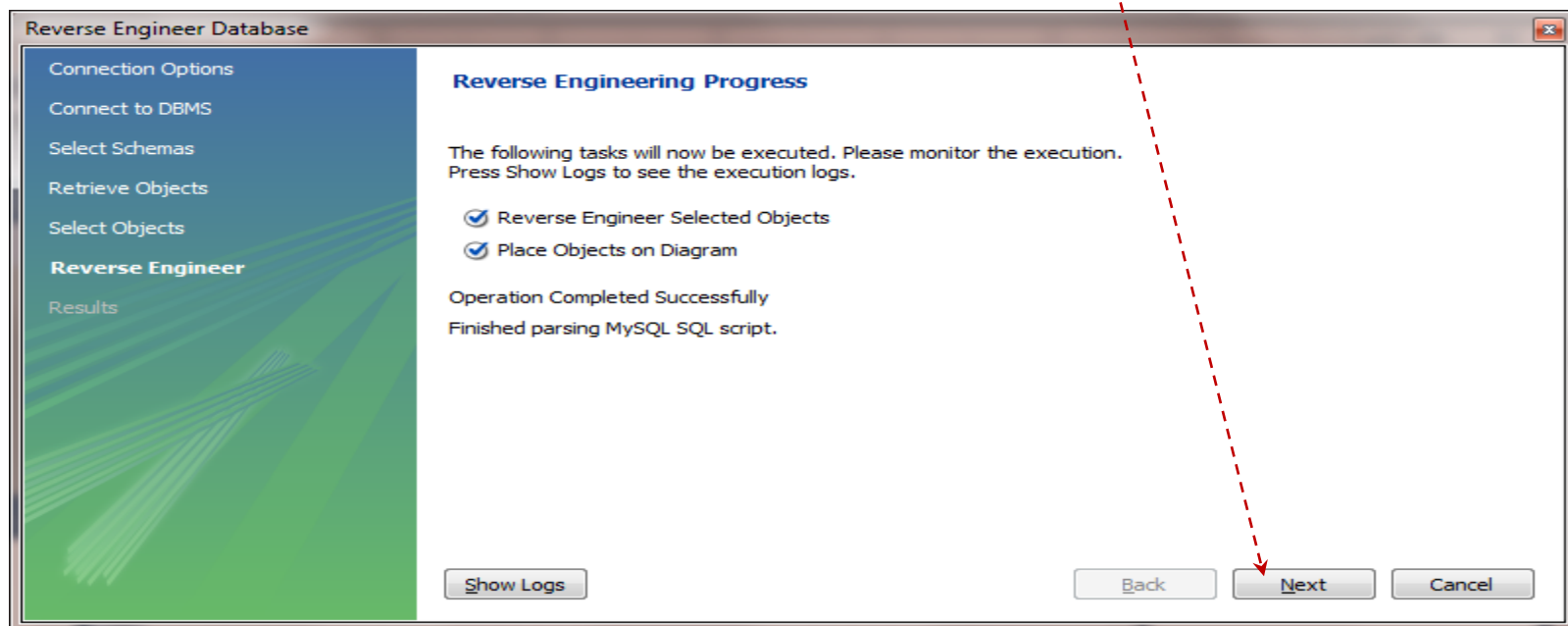
Click button **Next**.



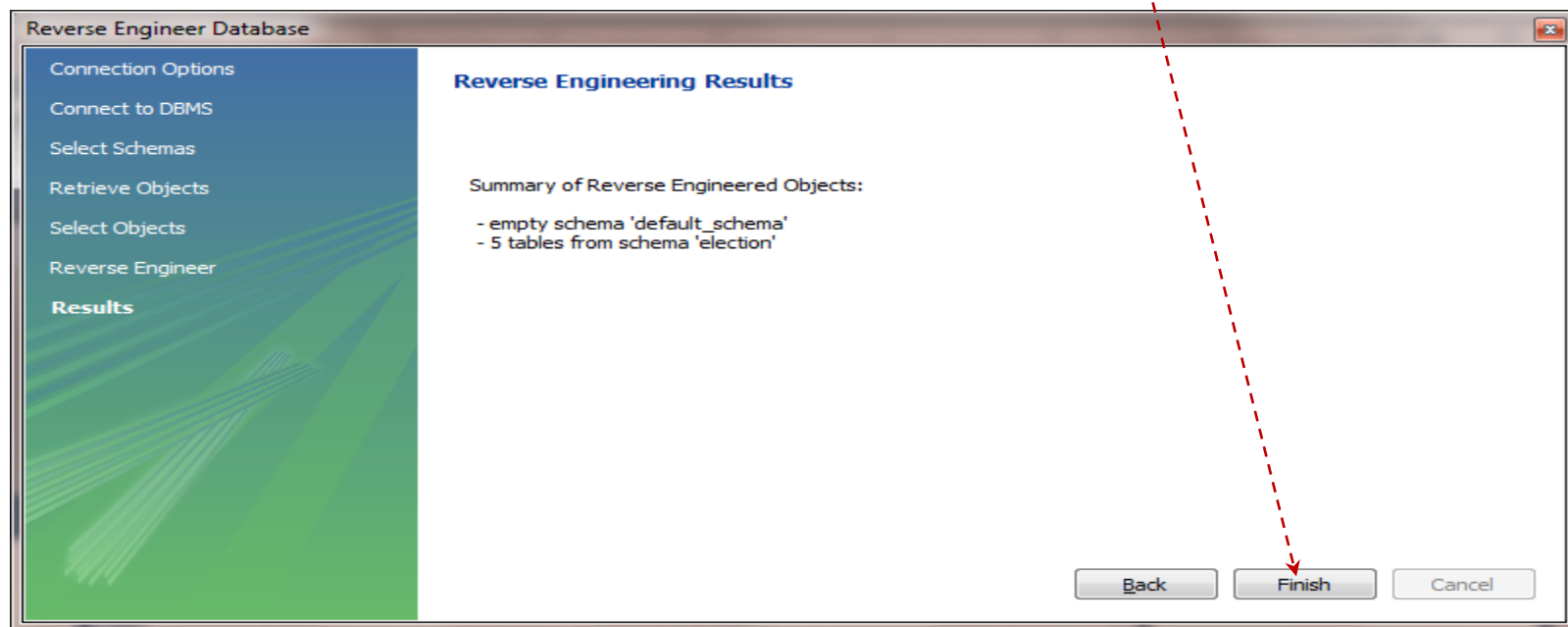
Make sure options **Import MySQL Table Objects** and **Place imported objects on a diagram** are selected and **click** button **Execute**.



Click button **Next**.

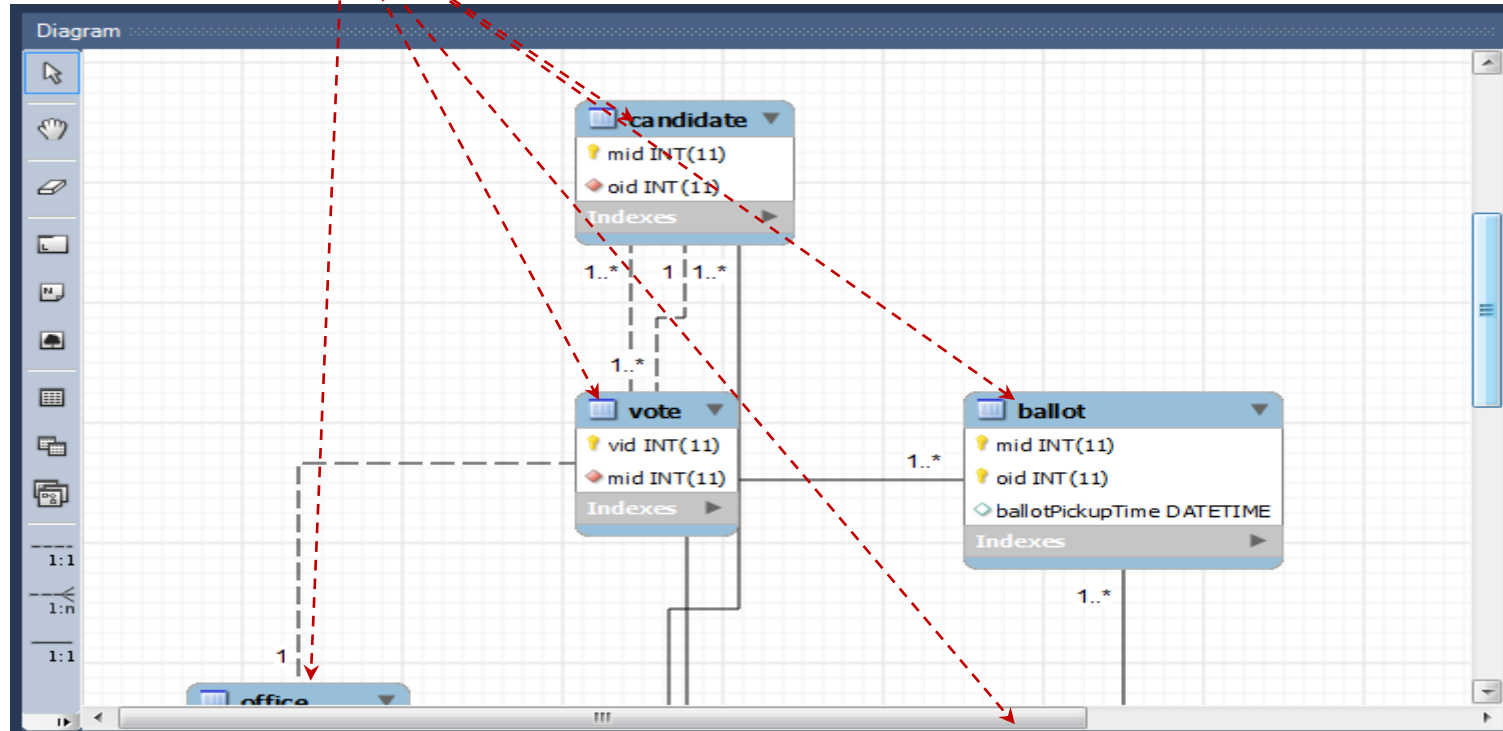


Click button **Finish**.



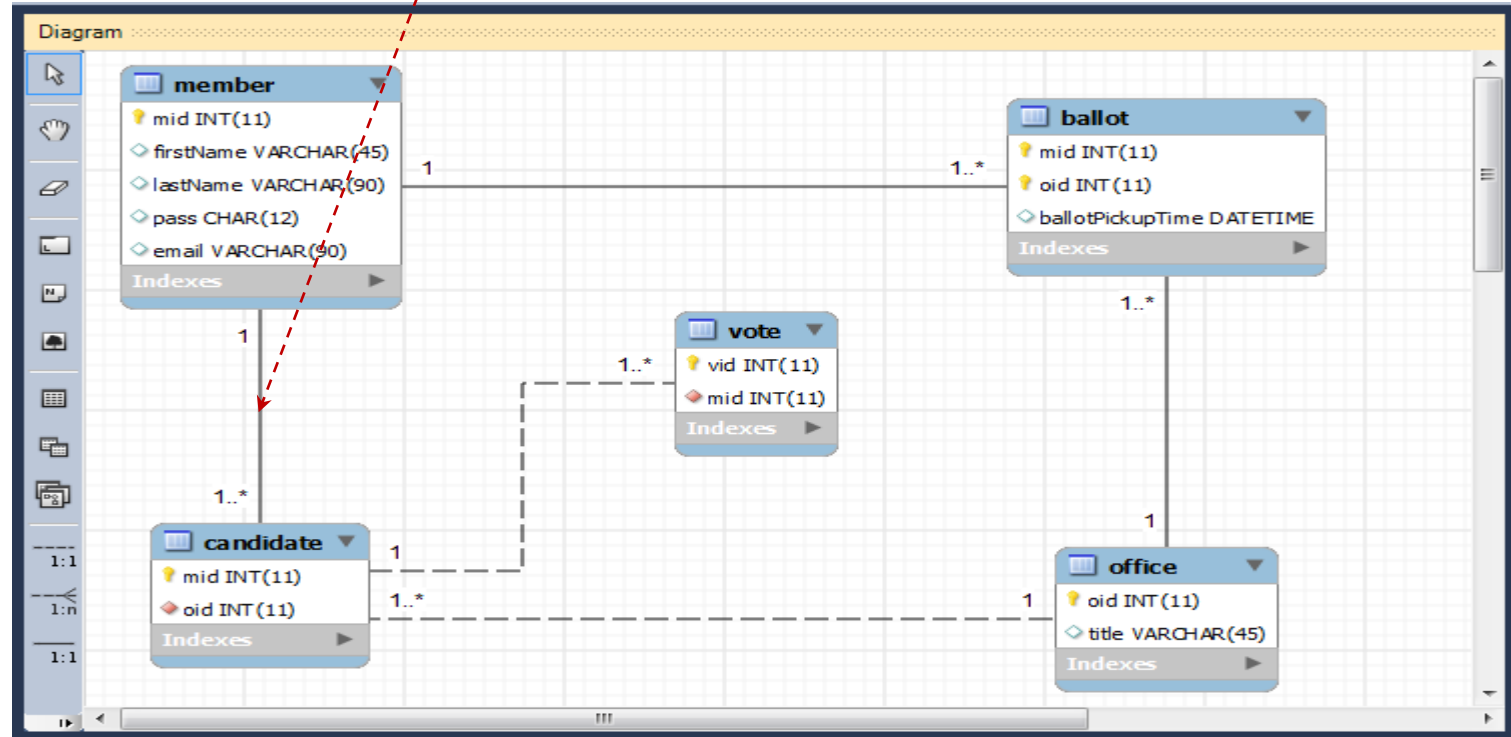
The table objects are not arranged in a friendly way.

Drag the **tables** around in order to better organize the diagram.



Notice that the diagram is almost identical to the original one (developed manually). The critical difference is that the relationship between **Member** and **Candidate** is shown as One-to-Many (1:1..*).

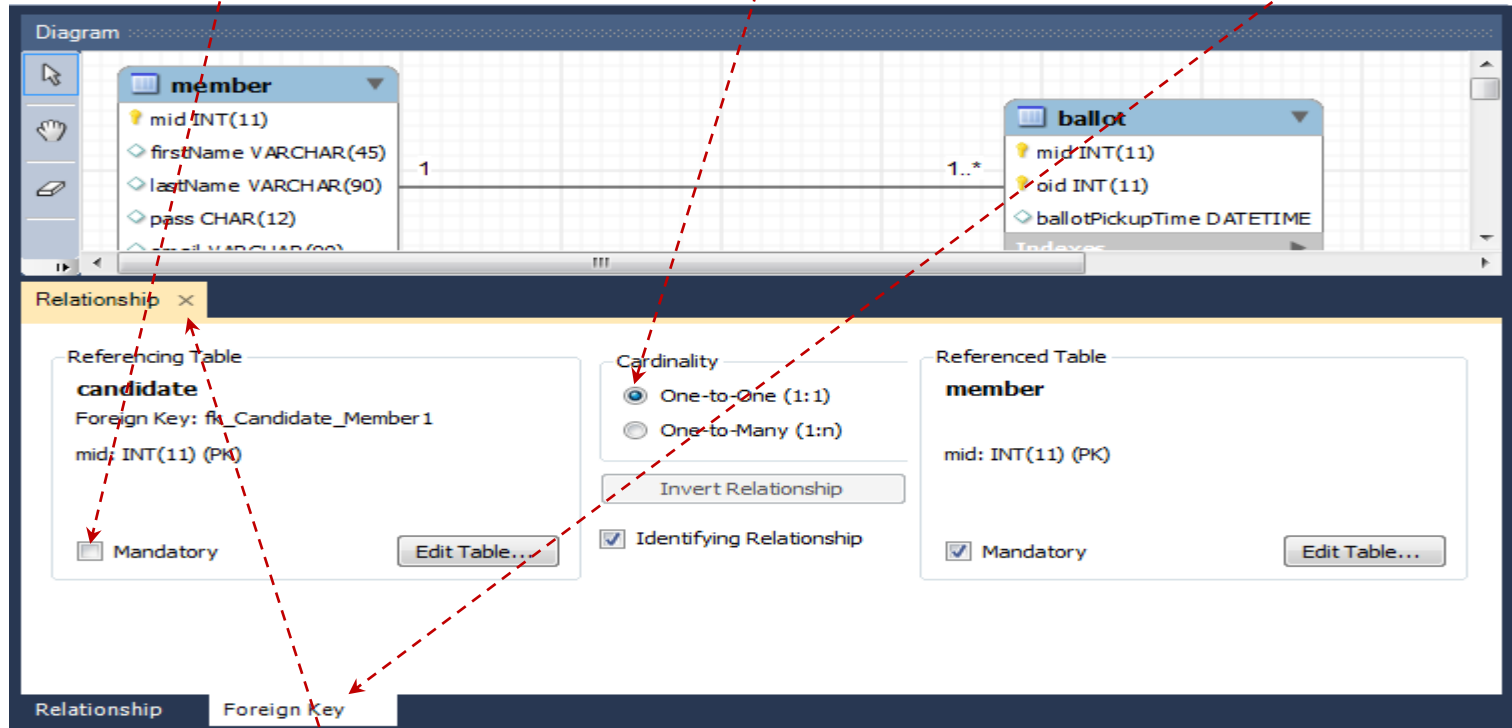
Double-click the **Member - Candidate** relationship link.



Click the **Foreign Key** tab.

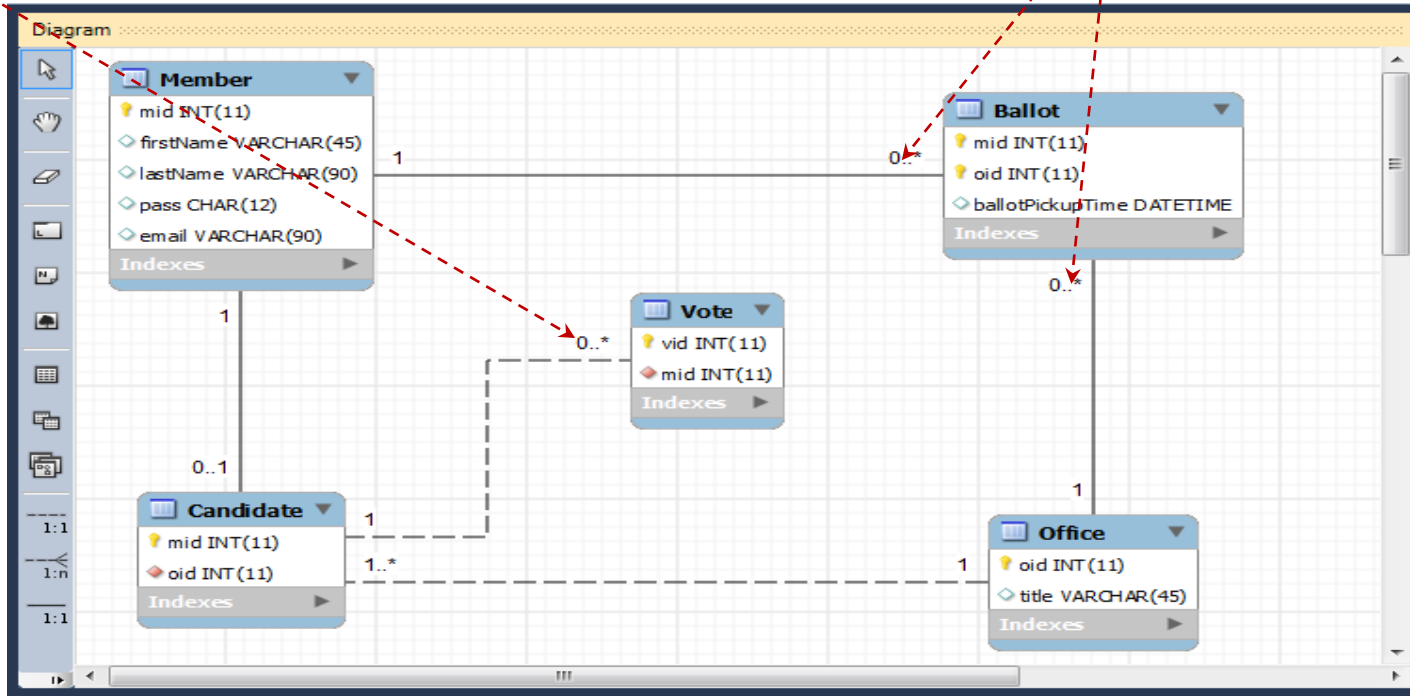
Uncheck the **Mandatory** option in the **Candidate** panel.

Turn on the **One-to-One (1:1)** cardinality option.



When done, close the **Relationship** panel.

Using similar operations, modify the **Member – Ballot**, **Ballot – Office** and **Candidate – Vote** relationships. Make sure that the **Mandatory** participation of entities **Ballot** and **Vote** are turned **off** so that they will all show up as **0..*** (Optional-Many).



A perfectionist would also show the table names in the title case. Finally, save the model and close **MW**.

This is it!