Database Desktop (DBD) е класически инструмент, предоставен от Borland, за създаване и управление на локални таблици (най-често в форматите **Paradox** или **dBase**).

Стъпки за създаване на нова база данни (таблица):

---

1. Стартиране и избор на формат

1. Отворете **Database Desktop** (обикновено се намира в папката на Borland C++ Builder в Start менюто).
2. Отидете на **File -> New -> Table...**
3. Ще се появи прозорец **Table Type**. Изберете **Paradox 7** (стандарт за Windows приложения по това време) и натиснете **OK**.

---

2. Дефиниране на структурата (Полета)

Вижадате таблица, в която трябва да се опишат колоните на вашата база данни. Нека създадем база за "Клиенти":

Field Name	**Type**	**Size**	**Key**
ID	+ (Autoincrement)	-	* (Double click)
Name	A (Alpha)	50	
Phone	A (Alpha)	15	
Balance	$ (Money)	-	

Важни означения за Type:

- **A (Alpha):** Текст. Изисква размер (Size).
- **N (Number):** Числа с плаваща запетая.
- **$ (Money):** Валута.
- **D (Date):** Дата.
- **+ (Autoincrement):** Автоматично нарастващо число (идеално за уникален номер на запис).
  Забележка: За да зададете **Primary Key** (първичен ключ), щракнете два пъти в колоната **Key**. Ключовите полета трябва да са в началото на списъка.

---

3. Запазване на таблицата

1. След като сте готови, натиснете бутона **Save As**.
2. Изберете папка, в която ще се намира проектът ви (напр. C:\MyProject\Data\).
3. Дайте име на файла, например customers.db.

---

4. Конфигуриране на Alias (Препоръчително)

За да може C++ Builder да намира лесно вашата база, е добре да създадете **Alias** (псевдоним):

1. В Database Desktop отидете на **Tools -> Alias Manager**.
2. Натиснете **New**.
3. В полето **Database Alias** напишете MyProjectData.
4. В полето **Path** посочете пътя до папката с вашия файл customers.db.
5. Натиснете **Keep New** и след това **OK**.

Как да я използвате в C++ Builder 6?

Сега в компонента TTable, вместо да търсите файл по целия диск, просто:

1. Изберете MyProjectData в свойството **DatabaseName**.
2. Изберете customers.db в свойството **TableName**.
3. Поставете **Active = true**.

Така вашето приложение е готово да чете и записва в току-що създадената таблица!

Използване на бази данни създадени с Ms Access

Свързването на Microsoft Access база данни (`.mdb` или `.accdb`) с Borland C++ Builder 6 се различава от работата с Paradox, тъй като Access изисква ADO (ActiveX Data Objects) технология.

1. Необходими компоненти

В палитрата на C++ Builder отидете на таб ADO и поставете върху формата следните компоненти:

- **TADOConnection: Управлява физическата връзка с файла на Access.**
- **TADOTable: Служи за избор на конкретна таблица от базата.**
- **TDataSource (от таб *Data Access*): Свързва ADO компонентите с визуалните контроли (като `DBGrid`).**

---

2. Конфигуриране на връзката (Connection String)

Това е най-важната стъпка. Трябва да укажете на програмата кой драйвер да използва.

1. **Маркирайте ADOConnection1.**
2. **В *Object Inspector* щракнете върху трите точки `...` на свойството ConnectionString.**
3. **Изберете Build....**
4. **В таб Provider изберете:**
    - **`Microsoft Jet 4.0 OLE DB Provider` (за `.mdb` файлове).**
    - ***Ако ползвате по-нов `.accdb` формат, ще ви трябва `Microsoft Office 12.0 (или по-нов) Access Database Engine`.***
5. **Натиснете Next и в таб Connection изберете вашия Access файл чрез бутона `...`.**
6. **Натиснете Test Connection, за да се уверите, че всичко е наред.**
7. **Накрая, в *Object Inspector* задайте свойството LoginPrompt = false, за да не ви пита за парола при всяко стартиране.**

---

3. Свързване на веригата

След като връзката е готова, трябва да "навържете" компонентите:

1. **ADOTable1:**
    - **`Connection` -> изберете `ADOConnection1`.**
    - **`TableName` -> изберете името на таблицата от падащото меню.**
    - **`Active` -> задайте `true`.**
2. **DataSource1:**
    - **`DataSet` -> изберете `ADOTable1`.**
3. **DBGrid1:**
    - **`DataSource` -> изберете `DataSource1`.**

---

4. Програмен код

Кодът за добавяне, редактиране и изтриване остава почти идентичен с този за **`TTable`,** но използваме методите на **`TADOTable`:**

```
// Добавяне на запис в Access база данни
void __fastcall TForm1::BtnAddClick(TObject *Sender)
{
  ADOTable1->Append();
```

```
}

// Записване на промените
void __fastcall TForm1::BtnSaveClick(TObject *Sender)
{
  if (ADOTable1->State == dsEdit || ADOTable1->State == dsInsert)
  {
    ADOTable1->Post();
  }
}

// Изтриване
void __fastcall TForm1::BtnDeleteClick(TObject *Sender)
{
  if (!ADOTable1->IsEmpty())
  {
    ADOTable1->Delete();
  }
}
```

Важни съвети при работа с Access:

- **Драйвери:** Ако пренасяте програмата на друг компютър, той трябва да има инсталиран съответния **OLE DB Provider (обикновено идва с MS Office или Microsoft Access Runtime).**
- **Път до базата:** В професионалните приложения пътят до базата данни не се "твърдо" задава (hardcoded), а се извлича динамично по време на изпълнение (runtime), за да работи програмата, независимо в коя папка е инсталирана:

```
String DBPath = ExtractFilePath(Application->ExeName) + "mybase.mdb";
ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + DBPath + ";Persist Security Info=False";
ADOConnection1->Connected = true;
ADOTable1->Active = true;
```

Използването на TADOQuery е по-гъвкавият начин за работа с база данни, тъй като ви позволява да изпълнявате специфични SQL команди (**SELECT, INSERT, UPDATE, DELETE**), вместо да разчитате само на автоматизацията на `TADOTable`.

Настройване и използване в Borland C++ Builder 6:

1. Конфигурация на компонентите

   1. Поставете **TADOQuery** (от таб ADO) върху формата.
   2. В свойството `Connection` на `ADOQuery1` изберете вашия вече съществуващ **TADOConnection1**.
   3. Ако искате да виждате резултата в таблица, променете свойството `DataSet` на вашия **TDataSource** да сочи към `ADOQuery1` (вместо към `ADOTable1`).

---

2. Изпълнение на SELECT заявка (Извличане на данни)

Заявките, които връщат резултат (таблица), се активират с метода `Open()` или чрез свойството `Active = true`.

```
void __fastcall TForm1::BtnSearchClick(TObject *Sender)
{
   ADOQuery1->Close(); // Винаги затваряйте преди промяна на SQL
   ADOQuery1->SQL->Clear(); // Изчиства старата заявка

   // Добавяне на нова SQL команда
   ADOQuery1->SQL->Add("SELECT * FROM Customers WHERE City = 'Plovdiv'");

   ADOQuery1->Open(); // Използва се Open за SELECT заявки
}
```

3. Използване на параметри (Сигурен и по-добър начин)

За да избегнете грешки при въвеждане на специални символи и да защитите базата, използвайте параметри (означени с `:` в SQL текста).

```
void __fastcall TForm1::BtnFilterClick(TObject *Sender)
{
   String searchName = Edit1->Text;

   ADOQuery1->Close();
   ADOQuery1->SQL->Clear();
            ADOQuery1->SQL->Add("SELECT * FROM Customers WHERE Name LIKE :clientName");

   // Задаване на стойност на параметъра
    ADOQuery1->Parameters->ParamByName("clientName")->Value = "%" + searchName + "%";

   ADOQuery1->Open();
}
```

4. Изпълнение на команди (INSERT, UPDATE, DELETE)

За команди, които променят данни, но **не връщат** таблица, се използва методът `ExecSQL()`.

```
void __fastcall TForm1::BtnUpdateSalaryClick(TObject *Sender)
{
   ADOQuery1->Close();
   ADOQuery1->SQL->Clear();
     ADOQuery1->SQL->Add("UPDATE Employees SET Salary = Salary * 1.1 WHERE Dept = 'IT'");

   try {
      ADOQuery1->ExecSQL(); // Използва се за променящи заявки
      ShowMessage("Заплатите бяха актуализирани!");
   }
   catch (const Exception &e) {
```

```
      ShowMessage("Грешка при изпълнение: " + e.Message);
    }
}
```

Метод	**Кога се използва?**	**Пример за SQL**
Open()	За четене на данни.	SELECT ...
ExecSQL()	За промяна, добавяне или триене.	INSERT, UPDATE, DELETE

Когато се работи с `TADOQuery`, **винаги поставяйте кода в блок `try...catch`**. Access често "заключва" файловете или извежда грешки при неправилен синтаксис на SQL, които могат да сринат приложението ви, ако не са обработени.