**Тема:**

**Основни понятия в база данни**

**От**:

Симеон Марков

**Клас:** 11А

1. **База данни**
2. Определение - Базата данни е инструмент за **събиране и организиране** на информация. Базите данни могат да съхраняват информация за хората, продуктите, поръчките или каквото и да било друго. Много бази данни се стартират като списък в програма за текстообработка или електронна таблица. Тъй като списъкът става по-голям, съкращенията и несъответствията започват да се показват в данните. Данните стават все по-трудни за разбиране във формуляра за списък и има ограничени начини за търсене или издърпване на поднабори от данни за преглед. След като започнат да се показват тези проблеми, е добра идея да прехвърлите данните в база данни, създадена от система за управление на бази данни (СУБД), като например Access.
3. Система за управление на база данни - представлява **набор от инструменти** за управление на данни в определен формат. Бихте могли да създадете свой-собствен набор от инструменти или да закупите съществуващ такъв. Има едно нещо, което всички системи за управление на бази данни включват — машината за бази данни. Машината за бази данни (database engine) е приложението, като например Microsoft Access или Oracle, което обработва данните в базата данни. Тя е сърцето на вашата система за управление на бази данни. Обикновено тази система е или обектно-ориентирана, или релационна.

1. **Релационен модел**
2. Определение **-** Релационната база данни(RDB) е колекция от информация, която организира точки от данни с дефинирани релации за лесен достъп**. В релационния модел на база данни структурите от данни - включително таблици с данни, индекси и изгледи - остават отделни от физическите структури за съхранение**, което позволява на администраторите на бази данни да редактират физическото съхранение на данни, без да засягат логическата структура от данни. В предприятието релационните бази данни се използват за организиране на данни и идентифициране на релации между ключови точки от данни. Те улесняват сортирането и намирането на информация. работят добре със структурирани данни.
3. **Система за управление на релационнa базa данни**
4. Определение **-** Софтуерът, използван за съхраняване, управление, подаване заявки и извличане на данни, съхранявани в релационна база данни, се нарича **система за управление на релационни бази данни (RDBMS)**. Тя осигурява интерфейс между потребителите и приложенията и базата данни, както и административни функции за управление на съхранението, достъпа и производителността на данните.
5. Видове RDBMS
6. [MySQL](https://www.google.com/search?sca_esv=575623720&cs=0&sxsrf=AM9HkKmZLEfKOPQk5pGVDf2e-dLfPKt8GQ:1697995917482&q=mysql&stick=H4sIAAAAAAAAAOMwVGI0iOIrqSxILVbIT1MIcnHyDf7FKO1WlJ-rUJxfWpQMFE9MLsovLlYoyUhVKE9N-sXEEe_qF-IZEhm_gYXxFQsXF4d-rr6BSaVx9isWbi5OEMfQyKCsCMErK7esgis0rcwrR1ZYUobgZacXWr5i4eHiAvEs05ONDIoQ-izMq5BUFhllIXhFZoUFcIUpeZaGSIakl5jkIlQWJqUYwSWNTEtSkkzgXONkw5yUqkWsrLmVxYU5t9gkGQ5cyVzyRl3qd-jjsxLx7Mxr2r3l3DOY2G8DAIAjOGc3AQAA&sa=X&ved=2ahUKEwiQsvXbl4qCAxVBhP0HHfq-BlsQ7fAIegQIABBo)/MariaDB
7. [PostgreSQL](https://www.google.com/search?sca_esv=575623720&cs=0&sxsrf=AM9HkKmZLEfKOPQk5pGVDf2e-dLfPKt8GQ:1697995917482&q=postgresql&stick=H4sIAAAAAAAAAOMwVGI0juIrqSxILVbIT1MIcnHyDf7FKO1WlJ-rUJxfWpQMFE9MLsovLlYoyUhVKE9N-sXEEe_qF-IZEhm_gYXxFQsXF4d-rr6BSaVx9isWbi5OEMfQyKCsCMErK7esgis0rcwrR1ZYUobgZacXWr5i4eHiAvEs05ONDIoQ-izMq5BUFhllIXhFZoUFcIUpeZaGSIakl5jkIlQWJqUYwSWNTEtSkkzgXONkw5yUqkWsXAX5xSXpRanFhTm32CQZDlzJXPJGXep36OOzEvHszGvaveXcM5jYbwMASPLVhzwBAAA&sa=X&ved=2ahUKEwiQsvXbl4qCAxVBhP0HHfq-BlsQ7fAIegUIABChAQ)
8. [Firebird](https://www.google.com/search?sca_esv=575623720&cs=0&sxsrf=AM9HkKmZLEfKOPQk5pGVDf2e-dLfPKt8GQ:1697995917482&q=firebird+(database+server)&stick=H4sIAAAAAAAAAOMwVGIyNIriK6ksSC1WyE9TCHJx8g3-xSjtVpSfq1CcX1qUDBRPTC7KLy5WKMlIVShPTfrFxBHv6hfiGRIZv4GF8RULFxeHfq6-gUmlcfYrFm4uThDH0MigrAjBKyu3rIIrNK3MK0dWWFKG4GWnF1q-YuHh4gLxLNOTjQyKEPoszKuQVBYZZSF4RWaFBXCFKXmWhkiGpJeY5CJUFialGMEljUxLUpJM4FzjZMOclKpFrFJpmUWpSZlFKQoaKYkliUmJxakKxalFZalFmrfYJBkOXMlc8kZd6nfo47MS8ezMa9q95dwzmNhvAwBqFfA9TQEAAA&sa=X&ved=2ahUKEwiQsvXbl4qCAxVBhP0HHfq-BlsQ7fAIegQIABA3)
9. [IBM Db2](https://www.google.com/search?sca_esv=575623720&cs=0&sxsrf=AM9HkKmZLEfKOPQk5pGVDf2e-dLfPKt8GQ:1697995917482&q=ibm+db2&stick=H4sIAAAAAAAAAOMwVGI0ieIrqSxILVbIT1MIcnHyDf7FKO1WlJ-rUJxfWpQMFE9MLsovLlYoyUhVKE9N-sXEEe_qF-IZEhm_gYXxFQsXF4d-rr6BSaVx9isWbi5OEMfQyKCsCMErK7esgis0rcwrR1ZYUobgZacXWr5i4eHiAvEs05ONDIoQ-izMq5BUFhllIXhFZoUFcIUpeZaGSIakl5jkIlQWJqUYwSWNTEtSkkzgXONkw5yUqkWs7JlJuQopSUa32CQZDlzJXPJGXep36OOzEvHszGvaveXcM5jYbwMAPG8sBjkBAAA&sa=X&ved=2ahUKEwiQsvXbl4qCAxVBhP0HHfq-BlsQ7fAIegUIABC0AQ)
10. [Oracle Database](https://www.google.com/search?sca_esv=575623720&cs=0&sxsrf=AM9HkKmZLEfKOPQk5pGVDf2e-dLfPKt8GQ:1697995917482&q=oracle+database&stick=H4sIAAAAAAAAAOMwVGI0iuIrqSxILVbIT1MIcnHyDf7FKO1WlJ-rUJxfWpQMFE9MLsovLlYoyUhVKE9N-sXEEe_qF-IZEhm_gYXxFQsXF4d-rr6BSaVx9isWbi5OEMfQyKCsCMErK7esgis0rcwrR1ZYUobgZacXWr5i4eHiAvEs05ONDIoQ-izMq5BUFhllIXhFZoUFcIUpeZaGSIakl5jkIlQWJqUYwSWNTEtSkkzgXONkw5yUqkWs_PlFick5qQopiSWJSYnFqbfYJBkOXMlc8kZd6nfo47MS8ezMa9q95dwzmNhvAwBjzPtIQQEAAA&sa=X&ved=2ahUKEwiQsvXbl4qCAxVBhP0HHfq-BlsQ7fAIegUIABCOAQ)

1. **Какво знаем за MySQL/MariaDB**

* MySQL е система за управление на релационни бази данни с open source (RDBMS), която позволява на потребителите **да съхраняват, управляват и извличат структурирани данни ефективно.** Той се използва широко за различни приложения, от малки проекти до мащабни уебсайтове и решения на корпоративно ниво. Относно MySQL, базата данни е структурирана колекция от данни, организирани и съхранявани в таблици. Той служи като централно хранилище, където информацията се управлява ефективно, което позволява на потребителите **да актуализират и изтриват данни**. MySQL осигурява софтуерната рамка за създаване, поддържане и взаимодействие с тези бази данни, което прави съхранението и извличането на данни безпроблемно и надеждно.

1. Основни типове данни:

|  |  |
| --- | --- |
| **Типове данни** | **Стойности** |
| CHAR | 0 - 255 |
| VARCHAR | 0 – 65535 |
| TEXT | До 65535 bytes |
| LONGTEXT | 0 - 4294967295 |
| BOOL | True/false |
| INT | -2147483648 - 2147483647 |
| FLOAT | 32 bits |
| Decimal  DATE  DATETIME | 64 bits  Format: YYYY-MM-DD  Format: YYYY-MM-DD hh:mm:ss |

A logo of a server

Description automatically generated

1. **Нормализация при база данни**
   1. **Определение -** Нормализацията е един от основновните процеси при проектиране на бази данни. Най общо казано нормализацията премахва повтаряемостта и минимизира излишъка от данни. Резултатът е по добра организация,по добра използваемост на паммета и премахване на аномалиите в базата данни. Нормализацията не винаги е най-доброто решение за дадена база данни, например при data Warehouse бази данни се прилага съвсем различен подход(използват се силно денормализирани бази данни). Или иначе казано сляпото прилагане на всички нормални форми може да доведе до катастрофални последици, изключително голям брой таблици, сложни заявки за сливане и най-важното много лоша производителност. Нормализирането е най-полезно, след като сме представили всички информационни елементи и сме стигнали до предварителен проект. То има за цел да ни помогне да се уверим, че сме разделили информационните елементи в подходящите таблици. Това, което нормализирането не може да гарантира, е че разполагаме с всички правилни данни, с които да започнем.
   2. **Видове нормализации:**
2. Първа нормална форма (1 NF)
3. Втора нормална форма (2 NF)
4. Трета нормална форма (3 NF)
5. Boyce Codd нормална форма или четвърта нормална форма (BCNF или 4 NF
6. Пета нормална форма (5 NF)
7. Шеста нормална форма (6 NF)

Първа нормална форма (1 NF)

* 1. **Първaтa нормална форма(първо правило)** за нормализация гласи, че при всяко пресичане на ред и колона в таблицата трябва да има една-единствена стойност и никога – списък със стойности.

**Пример:**

A screenshot of a phone number

Description automatically generatedИмаме следната таблица:

Таблицата " *Employee*" не е в първата си нормална форма, защото колоната **Phnone No. съдържа множество стойности**. Превеждайки я получаваме:

A green and black text on a table

Description automatically generated

Втора нормална форма(2 NF)

* 1. **Второто правило** за нормализация изисква всяка неключова колона да е напълно зависима от целия първичен ключ, а не само от част от него. Това правило се прилага, когато имате първичен ключ, който се състои от повече от една колона.

**Пример:**

A blue and white table with black text

Description automatically generatedРазполагаме със следната таблица:

Атрибута **Age** в таблицата е зависим от **Teacher ID,** подходящо подмножество на кандидат ключ. Поради това той нарушава правилото 2NF.

Затова разделяме таблицата на две:

A table with numbers and text

Description automatically generated

A yellow and white table with black text

Description automatically generated

Трета нормална форма(3 NF)

* 1. **Третото правило** за нормализация изисква не само всяка неключова колона да е зависима от целия първичен ключ, но и неключовите колони да са независими една от друга, т.е. всяка неключова колона трябва да е зависима само и единствено от първичния ключ.

**Пример:**

Да вземем следната таблица:

A table with text on it

Description automatically generated

A green and white table with text

Description automatically generatedТаблицата се разпада на:

A close-up of a name

Description automatically generated

1. **Primary и Foreign Key**
   1. **Първичен(Primary) ключ** - Първичният ключ е атрибут или набор от атрибути, които помагат за еднозначното идентифициране на записите в релационната таблица. Първичният ключ осигурява средствата за разграничаване на един запис от всички останали в релацията. Той помага на потребителя да идентифицира местоположението, а също и системата от бази данни, за да идентифицира, локализира и се позовава на един конкретен запис в релацията. Главните му черти са:
2. трябва да съдържа уникална стойност за всеки ред с данни.
3. не може да съдържа празни стойности.
4. Всеки ред трябва да има стойност на първичен ключ.
   1. **Синтаксис в MySQL:**

A screenshot of a computer

Description automatically generated

* 1. **Примери при употребата на първичен ключ:**

1. ИД на служител: В таблица на служител първичният ключ може да бъде уникалният ИД на служител, присвоен на всеки служител
2. ИД на студент: В студентска таблица първичният ключ може да бъде уникалният ИД на ученик, присвоен на всеки ученик.
3. ИД на клиент: В таблица клиент първичният ключ може да бъде уникалният ИД на клиент, присвоен на всеки клиент.
4. ИД на продукт: В продуктова таблица първичният ключ може да бъде уникалният ИД на продукт, присвоен на всеки продукт.
5. "Номер на сметка": В таблица на акаунта първичният ключ може да бъде уникалният номер на сметка, присвоен на всяка сметка.
   1. A screenshot of a computer

      Description automatically generated**Външен ключ -** Външен ключ (FK) е колона или комбинация от колони, която се използва **за установяване и налагане на връзка между данните** в две таблици, за да се контролират данните, които могат да се съхраняват в таблицата с външен ключ. В препратка към външен ключ се създава връзка между две таблици, когато колоната или колоните, които съдържат стойността на първичния ключ за една таблица, са реферирани от колоната или колоните в друга таблица. Тази колона става външен ключ във втората таблица.
   2. A screenshot of a computer code

      Description automatically generated **Синтаксис:**
   3. **Употреба -** Използването на тези външни ключове ни спестява необходимостта да съхраняваме едни и същи данни многократно – не е нужно да съхраняваме името на потребителя в таблицата с поръчки, защото можем да използваме orders.user\_id, за да посочим уникалния ред на този потребител, users.user\_id да получим неговото име и друга информация за него. Но истинската цел на външните ключове е, че те добавят ограничение, което помага да се установят и наложат последователни взаимоотношения в нашите данни. Записите в таблицата с външен ключ трябва да имат стойност, която съответства на колоната "чужда" таблица. Това ограничение се нарича **ограничение(constraint) за външен ключ**.
   4. **Композитен(Composite) ключ -** Композитният ключ, в контекста на релационни бази данни, е комбинация от две или повече колони в таблица, която може да се използва за еднозначно идентифициране на всеки ред в таблицата. Уникалността е гарантирана само когато колоните са комбинирани; Когато се разглеждат поотделно, колоните не гарантират уникалност.
6. **Unique key**

* Уникален ключ е набор от едно или повече полета/колони на таблица, които еднозначно идентифицират запис в таблица на база данни.

A screenshot of a computer code

Description automatically generated**Пример:**

**Имаме ID**, което уникално и не може да има дублирани стойности

1. **Каскадна операция**

* Каскада в SQL се използва за едновременно изтриване или актуализиране на запис както от child, така и от parent таблица. Ключовата дума CASCADE се използва като връзка при писане на заявката ON DELETE или ON UPDATE. Да предположим, че сме създали две таблици с ВЪНШЕН КЛЮЧ в релация с външен ключ, което прави и двете таблици родител и дете. След това дефинираме клауза ON DELETE CASCADE за един ВЪНШЕН КЛЮЧ, която трябва да бъде зададена, за да може другият да успее в каскадните операции. Ако каскадата ON DELETE CASCADE е дефинирана само за една клауза FOREIGN KEY, тогава каскадните операции ще хвърлят грешка.

A diagram of a number

Description automatically generated**Пример:**

1. **Случай 1** - Ако ученикът иска да промени класа си, той трябва да изтрие името си от всички таблици, т.е. T1, T2 и T3. Тъй като ученикът ще бъде преместен в друг клас, информацията му трябва да бъде изтрита. S
2. **Случай 2** - Ако студентът иска да промени своето име / номер на записване, той трябва да актуализира и трите таблици, т.е. T1, T2 и T3.

Ако се опитаме да изпълним сценариите, предоставени в горните случаи, по традиционен начин, промените ще трябва да се извършват ръчно и в трите таблици. Използваме ключовата дума CASCADE, за да преодолеем този проблем в SQL заявките. С една единствена команда можем да променим и трите таблици едновременно.

1. **Връзки между таблици**
2. **Видове връзки**
   1. **Един-към-един (one-to-one) -** В релация "един към един" запис в една таблица може да съответства само на един запис в друга таблица (или в някои случаи да няма записи).

A computer flowchart with text

Description automatically generated

* 1. **Един-към-много(one-to-many) -** В релация "един към много" (понякога наричана "много към едно") запис в една таблица съответства на нула, един или много записи в друга таблица.

A screenshot of a computer

Description automatically generated

* 1. **Много-към-много(many-to-many) -** Релация "много към много" показва, че множество записи в една таблица са свързани с множество записи в друга таблица.

A screenshot of a diagram

Description automatically generated

**Източници:**

* [**https://support.microsoft.com/bg-bg/office/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B8-%D0%BD%D0%B0-%D0%B1%D0%B0%D0%B7%D0%B8%D1%82%D0%B5-%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8-a849ac16-07c7-4a31-9948-3c8c94a7c204**](https://support.microsoft.com/bg-bg/office/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B8-%D0%BD%D0%B0-%D0%B1%D0%B0%D0%B7%D0%B8%D1%82%D0%B5-%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8-a849ac16-07c7-4a31-9948-3c8c94a7c204)
* [**https://www-it.fmi.uni-sofia.bg/courses/WDB/db.htm**](https://www-it.fmi.uni-sofia.bg/courses/WDB/db.htm)
* [**https://www.techtarget.com/searchdatamanagement/definition/relational-database**](https://www.techtarget.com/searchdatamanagement/definition/relational-database)
* [**https://www.oracle.com/database/what-is-a-relational-database/#link8**](https://www.oracle.com/database/what-is-a-relational-database/#link8)
* [**https://www.hostinger.com/tutorials/what-is-mysql#What\_is\_MySQL**](https://www.hostinger.com/tutorials/what-is-mysql#What_is_MySQL)
* [**https://www.w3schools.com/mysql/mysql\_datatypes.asp**](https://www.w3schools.com/mysql/mysql_datatypes.asp)
* [**https://www.cphpvb.net/db/1876-how-much-to-normalize-databases/**](https://www.cphpvb.net/db/1876-how-much-to-normalize-databases/)
* [**https://support.microsoft.com/bg-bg/office/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B8-%D0%BD%D0%B0-%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%B0%D0%BD%D0%B5%D1%82%D0%BE-%D0%BD%D0%B0-%D0%B1%D0%B0%D0%B7%D0%B8-%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5#bmnormalization**](https://support.microsoft.com/bg-bg/office/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B8-%D0%BD%D0%B0-%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%B0%D0%BD%D0%B5%D1%82%D0%BE-%D0%BD%D0%B0-%D0%B1%D0%B0%D0%B7%D0%B8-%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5#bmnormalization)
* [**https://www.analyticsvidhya.com/blog/2022/07/different-types-of-normalization-techniques/**](https://www.analyticsvidhya.com/blog/2022/07/different-types-of-normalization-techniques/)
* [**https://www.geeksforgeeks.org/primary-key-in-dbms/**](https://www.geeksforgeeks.org/primary-key-in-dbms/)
* [**https://www.techopedia.com/definition/5547/primary-key**](https://www.techopedia.com/definition/5547/primary-key)
* [**https://www.cockroachlabs.com/blog/what-is-a-foreign-key/#what-is-a-foreign-key-tldr**](https://www.cockroachlabs.com/blog/what-is-a-foreign-key/#what-is-a-foreign-key-tldr)
* [**https://www.techopedia.com/definition/6572/composite-key**](https://www.techopedia.com/definition/6572/composite-key)
* [**https://www.w3schools.com/sql/sql\_ref\_foreign\_key.asp**](https://www.w3schools.com/sql/sql_ref_foreign_key.asp)
* [**https://www.tutorialspoint.com/sql/sql-unique-key.htm**](https://www.tutorialspoint.com/sql/sql-unique-key.htm)
* [**https://www.tutorialspoint.com/sql/sql-unique-key.htm**](https://www.tutorialspoint.com/sql/sql-unique-key.htm)
* [**https://www.javatpoint.com/mysql-on-delete-cascade**](https://www.javatpoint.com/mysql-on-delete-cascade)
* [**https://www.scaler.com/topics/sql/cascade-in-sql/**](https://www.scaler.com/topics/sql/cascade-in-sql/)
* [**https://www.metabase.com/learn/databases/table-relationships**](https://www.metabase.com/learn/databases/table-relationships)