Изпитна тема № 9: Бази от данни

# План-тезис:

Въведение в базите от данни. Типове данни. Основни команди. Моделиране на релационни бази от данни. Заявки за извличане и промяна на данни. Сложни заявки за извличане на данни. Съединения на таблици (SQL JOIN). Агрегиращи функции. Групиране на данни. Скаларни функции, транзакции, съхранени процедури, тригери.

## [Типове данни в MSSQL](https://www.promotic.eu/en/pmdoc/Subsystems/Db/MsSQL/DataTypes.htm)

## [Основните SQL команди](https://www.bitdegree.org/learn/sql-commands-list)

Select, Insert, Alter, Drop/ Delete, Truncate и др.

## [MSSQL свойства на колоните на таблицата](https://docs.microsoft.com/en-us/sql/relational-databases/tables/table-column-properties-sql-server-management-studio?view=sql-server-ver15)

Например auto increment, unique, primary key и др.

## Моделиране на релационни бази от данни

Стъпките за проектиране на бази от данни са следните:

* **Идентифициране на видовете данни** – таблиците обикновено представят обекти от реалния живот (ученици, курсове, градове и др.)
* **Идентифициране на колоните в таблицата** – колоните са пояснения към данните (например: учениците имат име , снимка, дата на записване и др.)
* **Задаване на основен ключ за всяка таблица** – дефинирането на колона за основен ключ е задължително. Трябва да е цяло число или GUID декларирано като основен ключ. Ако е цяло число може да се използва auto increment свойстовото за автоматично нарастване с всеки следващ запис. Основния ключ е хубаво да е първата колона в таблицата. Изключения от тези правила могат да са данни, които имат известна идентификация, например държави (BG, DE, US) или валути (USD, EUR, BGN).
* **Идентифициране и моделиране на отношенията –** взаимоотношенията са зависимости между данните.
* **Задаване на други ограничения**
* **Наливане на информация в таблицата**

## Заявки за извличане и промяна на данни

### Псевдоними/ Pseudos

Преименуват таблица или заглавие на колона. Може да се скъсяват полета или да се разгръщат абревиатури. Ако името на псевдонима съдържа интервали в MySQL се огражда с ` `.

### Заявки/ Queries

Това са команди към базата от данни за връщане на определени данни. Има взаимосвързани заявки, при които чрез псевдоними споменаваме таблици от външна заявка в друга заявка, вложена в първата. Ако няма такава връзка чрез псевдоними то значи под заявката и заявката са необвързани.

### Подзаявки/ Subqueries

Заявките могат да бъдат вложени една в друга. Има няколко вида подзаявки:

* **Скаларни/ Scalar subqueries** – връщат точно една стойност
* **Колонни/ Column subqueries** – връщат една колона с нула или повече стойности
* **Таблични/ Table subqueries** – връщат таблица с редове и колони

### Обединяване на заявки/ Queries union

Заявки от РАЗЛИЧНИ таблици могат да бъдат обединени чрез съответния оператор (UNION в MySQL). Броят на исканите колони в двете заявки трябва да е един и същ, а за имена на колоните се вземат имената на колоните от първата заявка. Типът на колоните от двете таблици не е задължително да не е един и същ. Може да дадете друго име на колоните с помощта на псевдоними! В резултата няма да присъстват повтарящи се редове. Ако се изисква да има и повтарящи трябва да се използва друг оператор (UNION ALL в MySQL).

## Съединения на таблици

### Декартово произведение/ Cartesian product/ Cross-join

Резултатът на едно Декартово произведение (сливане на две таблици) е всеки ред от първата таблица да се комбинира с всеки ред от втората. При 200 реда в първата и 300 във втората ще има 60 000 реда в таблицата с резултата! При декартово произведение или Липсва join условието, или то е невалидно или е така формулирано, че всички редове от първата таблица да са свързани с всички от втората. За да се избегне декартовото произведение е нужно винаги валидно join условие. При сливане на таблици с валидно join условие връзките между таблиците са много полезни, защото така можем да извлечем информация и от двете таблици коректно.

### INNER JOIN

Връзката между таблици по подразбиране. Тя връща само редовете, отговарящи на условието за свързване.

### LEFT OUTER JOIN

Тази връзка връща записите, отговарящи на свързващото условие и също така несъвпадащите записи от лявата таблица.

### RIGHT OUTER JOIN

Същото като LEFT OUTER JOIN, но несъвпадащите записи са от дясната таблица

### FULL JOIN

Тази връзка връща записите, отговарящи на свързващото условие заедно с несъвпадащите записи от лявата и от дясната таблици.

## Агрегация и групиране на данните

### Групиране/ Grouping

Групирането позволява връщане на данни в отделни групи, базирано на общо свойство.

### Агрегиращи функции/ Aggregate functions

Използват се за опериране върху една или повече групи (съдържащи елементи в себе си), анализирайки данните във всяка една от тези групи. Обикновено такива функции прескачат празните стойности. (максимална, минимална стойност, средно-аритметично, брой на стойности и др. агр. функции)

### HAVING

Прилича много на WHERE, но разликата е че отново извършва филтриране, но след като се случи агрегирането.

## Скаларни функции и работа с дати транзакции

### Потребителски дефинирани функции/ User defined functions

Винаги връщат стойност. Могат да приемат параметри и могат да са вложени

### [Транзакции/ Transaction](https://blog.cloudboost.io/how-to-use-transaction-in-sql-server-aaaaa23e491a)

Поредица от действия (операции в базата данни), изпълнявани като цялост т.е. или всички се изпълняват заедно успешно или нито едно от тях не се изпълнява (пример банков превод от една сметка към друга – банкова транзакция – ако изтеглянето или внасянето се провалят, цялата операция се отменя). Транзакциите гарантират пълнотата и целостта на базата от данни, защото всички промени в една транзакция са временни. Промените се изпълняват едва след COMMIT. По всяко време всички операции могат да се отменят чрез ROLLBACK (всички операции се изпълняват като едно цяло)

### [Съхранени процедури/ Stored procedures](https://www.w3schools.com/sql/sql_stored_procedures.asp)

Капсулират повтаряща се програмна логика. Могат да приемат входни параметри. Могат да връщат изходни резултати. Споделят логика. Подобряват производителността. Намалят мрежовия трафик

### [Тригери/ Triggers](https://www.sqlshack.com/triggers-in-sql-server/)

Много приличат на съхранените процедури с разликата, че се извикват в случай на дадено събитие. Тригерите не се извикват изрично. Те се прикрепят към таблицата. Изпълняват се, когато определена SQL заявка се изпълнява върху съдържанието на таблицата (например при добавяне на нов ред в таблицата)

### Събитие/ Event

Има три различни събития, които могат да се изпълнят в рамките на триге:

* **Вмъкване/ Insert**
* **Промяна/ Update**
* **Изтриване/ Delete**