Курс лекцій «Основи WEB програмування»
НТУУ «КПІ» ім. Сікорського ФІОТ ОТ Стешин В.В.
www.vv-steshyn.edu.kpi.ua v.steshyn@kpi.ua

Лекція 10
Структура БД, права доступу, phpMyAdmin, SQL запити, вибір типу даних для сайту

Реляционные базы данных

1.БД у вибляді двовимірного масиву
2. Існування ключів, первичный ключ – уникальный идентификатор строки, внешний ключ – связи между таблицами
3. Таблиця – стовпчик – первичний ключ

Индексы и индексация таблиц

Представьте себе, что ваш приятель загадал число между 1 и 1000 и просит вас угадать его за минимальное число попыток, сообщая лишь о том, в большую или меньшую сторону вы ошиблись. Как вы поступите? Очевидно, предложите при первой попытке версию 500 (то есть начнете с середины). Если он ответит: «меньше», — предложите 250. Если «больше» — 750. Так, разбивая интервалы пополам, вы уложитесь в 10 попыток (ведь 210 > 103). Если бы приятель загадал число в пределах миллиарда, то количество попыток уложилось бы в 30 (230 > 109).

Типы запросов данных

Есть четыре основных типа запросов данных в SQL, которые относятся к так называемому языку манипулирования данными (Data Manipulation Language или DML):

SELECT – выбрать строки из таблиц;
INSERT – добавить строки в таблицу;
UPDATE – изменить строки в таблице;
DELETE – удалить строки в таблице;

Использование запроса SELECT для выборки нужных данных

```
SELECT column1, column2 FROM table_name;
```

Также, можно получить все столбцы из таблицы, используя подстановочный знак «*»:

```
SELECT * FROM table_name;
```

Условные операторы

=	Равно
<>	Не равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно

BITWEEN и LIKE IN EXISTS

Оператор	Описание	Примеры
IS NULL, IS NOT NULL	Выражение **IS NULL** проверяет, равно ли выражение в левой части оператора - `Null`, то есть пустому значению. Значение равное `Null` имеют незаданные поля.	`Улица IS NULL` (все записи в которых не задано поле `Улица`) `Area IS Not NULL` (все записи в которых задано поле `Area`)

Оператор	Описание	Примеры
	Выражение **IS NOT NULL** соответственно проверяет неравенство значения `Null`.	
=	Проверяется равенство значений. Проверка равенства для строковых значений с учетом регистра значений, т.е. значение `Самолет` и `самолет` будут признаны разными значениями	`Квартал='105'` (Здания относящиеся к 105 кварталу)
>, <	Операторы «строгого» неравенства, проверяется что значение в левой части выражения строго больше, либо строго меньше значения в правой.	`perimeter>50` (все здания с периметром более 50 метров) `area<300` (все здания с площадью меньше 300 кв.м.
>=, <=	Операторы «нестрогого» неравенства, проверяется что значение в левой части выражения больше либо равно, или меньше либо равно значению в правой части.	`perimeter>=50` (все здания периметр которых равен, либо более 50 метров) `area<=300` (все здания с площадью меньше, либо равной 300 кв.м.)
!<, !>	Альтернативная запись операторов нестрого неравенства. Проверяется что значение в левой части выражения не меньше, либо не больше значения в правой части.	`perimeter!<50` (все здания с периметром равным, либо превышающим 50 метров) `area!>300` (все здания с площадью не больше 300 кв.м.)
<>, !=	Операторы неравенства. Проверяется, неравенство значений.	`Улица<>'2й Южный пер.'` `Квартал!='105'`
BETWEEN	Операция сравнения, проверяющая расположение значения в левой части выражения в заданном диапазоне значений в правой части выражения. Оператор задается в формате *<выражение 1>***BETWEEN***<1>***AND***<2>*, где *<1>* и *<2>* нижняя и верхняя границы допустимых значений.	`[Номер дома] BETWEEN 1 AND 5` (все здания с номерами от 1 до 5)
LIKE	Проверяется соответствие строкового значения шаблону, заданному в параметре оператора **LIKE**. В шаблоне могут использоваться буквы алфавита и знаки препинания, а также различные символы и выражения подстановки: • `%` - вместо данного символа в искомой строке может располагаться любое количество произвольных символов. Для поиска в искомой строке непосредственно символа `%`следует его продублировать; • `_`- один произвольный символ; • Квадратные скобки `[]` - на место квадратных скобок символ указанный в данных скобках. В скобках могут либо перечисляться допустимые символы (например `[abc]` соответствует символам `a`, или `b`, или с , либо указываться диапазон допустимых символов, (например `[a-z]` соответствует любому символу от `a` до `z`). Если после открывающей скобки стоит знак `^`, допустимы все символы кроме заданных в скобках. Например для выражения `[^135]` допустимы любые символы, кроме `1`, `3`, `5`.	`Улица LIKE '%Южный пер.'` (все записи с текстом заканчивающимся на `Южный пер.`) `Улица LIKE '_й Южный пер.'` (все строки начинающиеся с произвольного символа и заканчивающиеся на `й Южный пер.`) `Улица LIKE '[0-9]%'` (строки начинающиеся с цифры) `Улица LIKE '[авдй]_'` (строки из двух символов - первый символ один из а, в, д, й и второй - произвольный) `Улица LIKE '[^0-9]%'` (любая строка не начинающаяся с цифры) `Улица LIKE '[[]%]'` (строки заключенные в квадратные скобки) Обратите внимание, что знак] не является сам по себе символом подстановки и может указываться напрямую.

Оператор	Описание	Примеры
	Для поиска в исходной строке самих символов подстановки можно заключать их в квадратные скобки (например `[%]` для поиска знака `%`, или `[[]` для поиска символа `[`), либо задать так называемый escape символ (задается выражением **ESCAPE** *<символ>* после команды **LIKE**). В шаблоне подстановки escape символ указывается перед символом подстановки который требуется искать в строке.	`Улица LIKE '!%%!^' ESCAPE '!'`(любая строка, начинающаяся со знака `%` и заканчивающаяся знаком `^`).
IN	Проверяется соответствие значения одному из значений перечисленному после ключевого слова **IN** в формате *<Выражение>***IN** (*<Значение1>, <Значение2>...,<Значение N>*), где *<Значение1>* - *<Значение N>*, список допустимых значений записи. Список допустимых значений может быть результатом выполнения подзапроса («Подзапросы»).	`Улица IN ('Нахимова', '1й Южный пер.')` (все здания располагающиеся на улицах Нахимова и 1й Южный пер.)
EXISTS	Операция сравнения, которая возвращает TRUE, если подзапрос (subquery) возвращает по крайней мере одну строку («Подзапросы»).	EXISTS (subquery)

SELECT * FROM table_name WHERE ((Age >= 18) AND (LastName BETWEEN ‘Иванов’ AND ‘Сидоров’)) OR Company LIKE ‘%Motorola%’;

Использование запроса INSERT для вставки новых данных

Запрос INSERT используется для создания новой строки данных. Для обновления уже существующих данных или пустых полей строки нужно использовать запрос UPDATE.

INSERT INTO table_name (column1, column2, column3) VALUES (‘data1’, ‘data2’, ‘data3’);

Запрос UPDATE и условие WHERE

UPDATE table_name SET column1 = ‘data1’, column2 = ‘data2’ WHERE column3 = ‘data3’;

UPDATE table_name SET FirstName = ‘Василий’ WHERE FirstName = ‘Василий’ AND LastName = ‘Пупкин’;

Будьте осторожны! Запрос DELETE удаляет целые строки

DELETE FROM table_name WHERE column1 = ‘data1’;

Использование псевдонимов AS

Для создания псевдонима для слоя, во FROM части запроса, после названия слоя следует указать псевдоним для этого слоя (перед псевдонимом можно также добавить ключевое слово **AS).**

Сортировка итоговой таблицы

Формат использования: **ORDER BY** <данные для сортировки>, где в качестве данных для сортировки могут использоваться названия полей данных по которым сортируется таблица, либо номера столбцов таблицы выводимой в результате запроса (нумерация столбцов идет с 1). Данные для сортировки перечисляются через запятую.

Допускается выполнять сортировку по полям БД, не выводимым в итоговой таблице. Ключевое слово ORDER BY располагается в запросе после ключевых слов **FROM, WHERE, GROUP BY, HAVING.**

Примеры сортировки данных

Сортировка по одному полю

SELECT area, perimeter FROM Здания ORDER BY Sys

Сортирует итоговую таблицу по значениям поля Sys при том, что поле Sys не выводится в таблице.

Сортировка по нескольким полям

SELECT area, perimeter FROM Здания ORDER BY [Количество этажей], Area

Сортирует таблицу сначала по значению поля Количество этажей, а затем - по Area.

Сортировка по столбцу итоговой таблицы

SELECT area/perimeter FROM Здания ORDER BY 1

Сортирует таблицу по первому ее столбцу (частному от деления площадей зданий на их периметры).

Сортировка по убыванию значений

SELECT area, perimeter FROM Здания ORDER BY Area DESC

Сортирует таблицу по убыванию значения поля Area.

Сортировка по возрастанию и по убыванию значений

SELECT area, perimeter FROM Здания ORDER BY Area DESC, Sys

Сортирует таблицу по убыванию значения поля Area, записи с равными значениями данного поля сортируются по возрастанию значения поля Sys.

Агрегирование и группировка данных

В языке SQL используются следующие агрегирующие функции:

SUM([DISTINCT] <выражение>)

Выводит в итоговой таблице сумму значений для выражения по полям выборки. Выражение должно возвращать числовое значение.

AVG([DISTINCT] <выражение>)

Среднее значение для выражения. Выражение должно возвращать числовое значение.

COUNT([DISTINCT] <выражение> |*)

Подсчитывает число записей, в который выражение не имеет значение Null (поля имеют значение Null, когда никакое значение для них не задано). Выражение может возвращать произвольное значение.

При используемом формате функции COUNT(*) возвращает общее количество записей в БД слоя.

MAX(<выражение>)

Возвращает максимальное значение выражения для выборки.

MIN(<выражение>)

Возвращает минимальное значение выражения из выборки.

Применение агрегирующих функций

Простой пример

```
SELECT SUM(Perimeter) FROM Здания
```

Выводит сумму периметров зданий.

Одновременное применение нескольких функций

```
SELECT AVG(Area), Count(*) FROM Здания
```

Выводит среднюю площадь здания и общее количество зданий.

Применение функций совместно с условиями отбора

```
SELECT SUM(Area) FROM Здания WHERE Улица='Нахимова'
```

Возвращает сумму площадей зданий расположенных на улице Нахимова.

Применение выражений в качестве аргументов агрегирующих функций

```
SELECT SUM(Area/Perimeter*2) FROM Здания
```

Для каждого здания рассчитывается величина равная Площадь/Периметр*2 и суммируется.

Применение агрегирующих функций в составе выражений

```
SELECT SQRT(SUM(Area)), "Общий периметр" + SUM(Perimeter) FROM Здания
```

Возвращает квадратный корень от суммарной площади всех зданий и фразу вида «Общий периметр XXX», где XXX - суммарный периметр всех зданий.

Использование ключевого слова DISTINCT

```
SELECT COUNT(DISTINCT Улица) FROM Здания
```

Возвращает количество разных названий улиц в БД слоя.

Группировка записей

Группировка по полю данны

```
SELECT Улица, COUNT(*) FROM Здания GROUP BY Улица
```

Выводит список улиц и для каждой улицы количества записей.

Группировка по номеру столбца

```
SELECT FLOOR(Area/10)*10, SUM(Area) FROM Здания GROUP BY 1
```

Группирует записи по площади домов (с шагом 10, в первой группе с 0 до 10, во второй с 10 до 20 и т.д.) и выводит список групп площадей суммарную площадь для каждой группы.

Группировка по нескольким полям

```
SELECT COUNT(*) FROM Здания GROUP BY Квартал, Улица
```

Группирует записи по кварталам, а в кварталах по улицам и выводит количество записей для каждой подгруппы.

Фильтрация сгруппированных данных

Применение конструкции HAVING

Применение аналогично WHERE

```
SELECT Улица, COUNT(*) FROM Здания GROUP BY Улица HAVING Улица Like '%пер.'
```

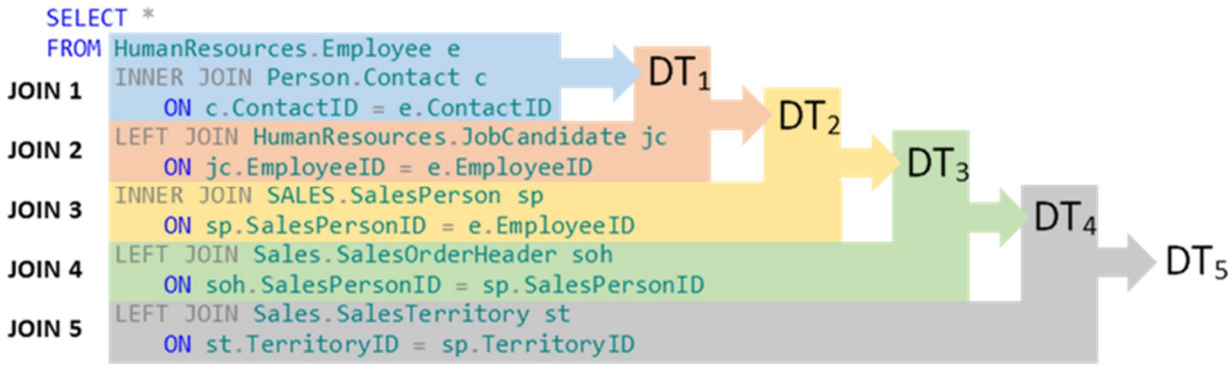
Группировка всех записей по полю Улица, и вывод значений для групп в которых название улицы заканчивается символами «пер.».

Отбор групп

```
SELECT Улица, AVG(Area) FROM Здания GROUP BY Улица HAVING COUNT(*)>3
```

Группировка записей по названию улицы, и вывод названий улиц и средних площадей зданий для групп в которых более трех записей.

Выборка данных из нескольких слоев



Как уже упоминалось выше, в системе Zulu допускается выборка данных из нескольких слоев карты. При этом в результате такой выборки будет выведена таблица с декартовым пересечением запрошенных полей перечисленных слоев. Дополнительные возможности по управлению выборками из нескольких слоев предоставляет конструкция JOIN, располагающаяся в команде выборки после ключевого слоя FROM, но перед ключевыми словами WHERE, GROUP BY, HAVING и ORDER BY. В конструкции задаются условия, по которым объединяются и выводятся поля БД слоев.

В системе Zulu предусмотрено несколько вариантов использования данной конструкции, каждый из которых имеет свои особенности и область применения:

INNER JOIN (внутреннее соединение)

Каждая запись данных первого слоя сопоставляется с каждой записью другого слоя на предмет выполнения условия соединения (например, выполнения условия пространственного соответствия для объектов соединяемых слоев) и выводятся все соответствующие условию записи.

Конструкция имеет следующий синтаксис: [INNER] JOIN <Слой> ON <Условие>, где <Слой> - слой добавляемый к выборке, а <Условие> - логическое выражение по которому проводится отбор полей. Ключевое слово INNER необязательно и может быть опущено в команде выборки.

По результату, конструкция внутреннего соединения аналогична применению условия (с помощью ключевого слова WHERE) к выборке по нескольким слоям.

Пример применения INNER JOIN

```
SELECT b.Sys AS Здание, k.Sys AS Квартал FROM Здания AS b INNER JOIN Кварталы AS k ON b.Geometry.STWithin(k.geometry) ORDER BY 2
```

В результате данной команды выборки (конструкция b.Geometry.STWithin(k.geometry) проверяет не находится ли объект слоя Здания геометрически внутри объекта слоя Кварталы, (см. "Работа с пространственными данными в запросах") будут выведены значения полей Sys для всех пар объектов слоев Здание и Квартал в которых объект слоя Здание находится внутри объекта слоя Кварталы. Результаты сортируются по второму столбцу таблицы.

Аналогичных результатов можно добиться с использованием ключевого слова WHERE

```
SELECT b.Sys AS Здание, k.Sys AS Квартал FROM Здания AS b, Кварталы AS k WHERE b.Geometry.STWithin(k.geometry) ORDER BY 2
```

CROSS JOIN (перекрестное соединение)

Результаты применения данной конструкции в команде полностью аналогичны перечислению названий двух слоев после ключевого слова FROM. В таблице, отображаемой в результате выполнения выборки выводится декартово пересечение записей, в запросе будет набор записей со всеми возможными комбинациями полей из записей первого и второго слоя, т.е., например при запросе поля A из слоя содержащего 2 записи и запросе поля B из слоя также содержащего две записи, в итоговой таблице будет четыре записи со следующими данными: A1+B1, A1+B2, A2+B1, A2+B2.

Конструкция имеет синтаксис CROSS JOIN <Слой>.

Пример применения CROSS JOIN

```
SELECT * FROM Здания CROSS JOIN Кварталы
```

В результате выполнения команды выборки будут выведено декартово пересечение полей БД слоев Здания и Кварталы.

Такой же результат будет при выполнении следующей команды выборки:

```
SELECT * FROM Здания, Кварталы
```

OUTER JOIN (внешнее соединение)

Как и в других вариантах использования конструкции JOIN в команде выборки задаются два слоя. Один после ключевого слова FROM и еще один, - в конструкции JOIN.

В результате выполнения команды выборки для одного из заданных слоев (назовем его основным) выводятся значения для всех его записей, а для другого слоя (назовем его дополнительным) выводятся только значения для записей соответствующих записям основного слоя по условию заданному в конструкции JOIN.

В системе Zulu предусмотрено два варианта использования конструкции OUTER JOIN : LEFT OUTER JOIN (левое соединение) и RIGHT OUTER JOIN (правое соединение). В первом случае основным слоем считается слой задаваемый после ключевого слоя FROM, а во-втором - задаваемый в конструкции JOIN.

Конструкция имеет синтаксис LEFT|RIGHT [OUTER] JOIN <Слой> ON <Условие>, где LEFT|RIGHT - вид используемого соединения, <Слой> - слой добавляемый к выборке, а <Условие> - логическое выражение по которому проводится отбор полей. Ключевое слово OUTER необязательно и может быть опущено в команде выборки.

Пример применения OUTER JOIN

```
SELECT b.sys AS Здание, k.sys AS Квартал FROM Здания AS b LEFT JOIN Кварталы AS k ON b.Geometry.STOverlaps(k.Geometry)
```

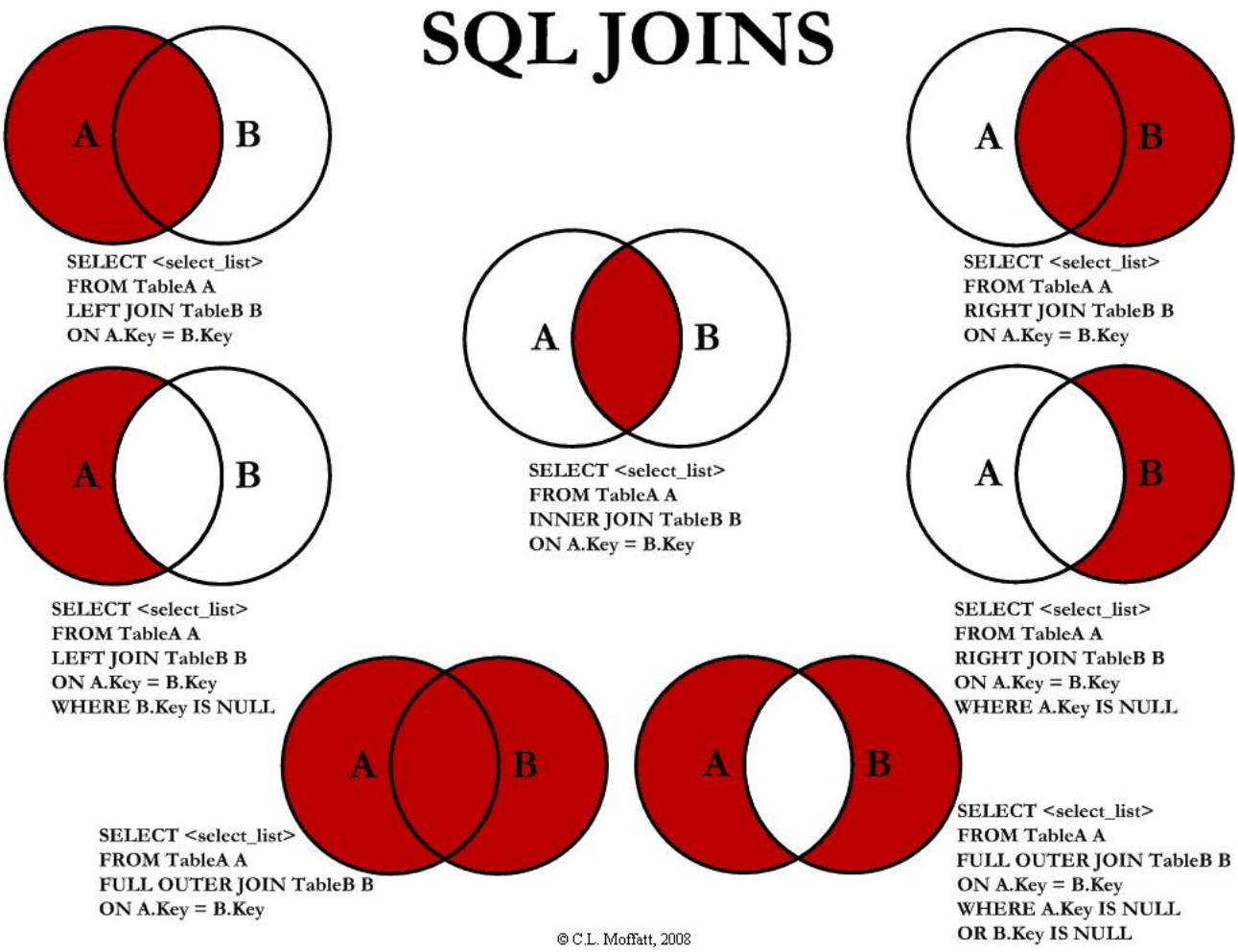
В результате выполнения команды выборки (конструкция b.Geometry.STOverlaps(k.Geometry) проверяет, не пересекается ли геометрически объект слоя Здания с объектом слоя Кварталы) будут выведены поля Sys для всех объектов слоя Здания и поля Sys объектов слоя Кварталы пересекаемых объектами слоя Здания.

Если же выполнить команду:

```
SELECT b.sys AS Здание, k.sys AS Квартал FROM Здания AS b RIGHT JOIN Кварталы AS k ON b.Geometry.STOverlaps(k.geometry)
```

То будут выведены поля Sys для всех объектов слоя Кварталы, а для слоя Здания будут выведены только Sys объектов пересекающих границы объектов слоя Кварталы.

В команде выборки может последовательно использоваться несколько конструкций JOIN, в результате чего будет выполнено соединение полей из нескольких заданных слоев. Например, команда выборки SELECT * FROM Здания CROSS JOIN Кварталы CROSS JOIN Надписи формирует таблицу с декартовым пересечением всех трех перечисленных слоев.



Шифрування

La fonction s'utilise dans une requête SQL de la manière suivante :

```
SELECT MD5('test');
```

Une telle requête retournera la chaîne suivante :

```
098f6bcd4621d373cade4e832627b4f6
```

Cet exemple concret montre bien l'intérêt d'utiliser MD5() pour créer une clé de

```
UPDATE utilisateur SET hashage = MD5('345_2013-10-04 22:16:45_moutarde') WHERE id = 345
```

```
SELECT idFROM utilisateur WHERE login = 'alphonse' AND password = MD5('secret');
```

https://www.politerm.com/zuludoc/spatial_query_sql.htm