

Курс 2 в 1: SQL

+ подготовка к
собеседованиям

Неделя 3

Екатерина Рехерт

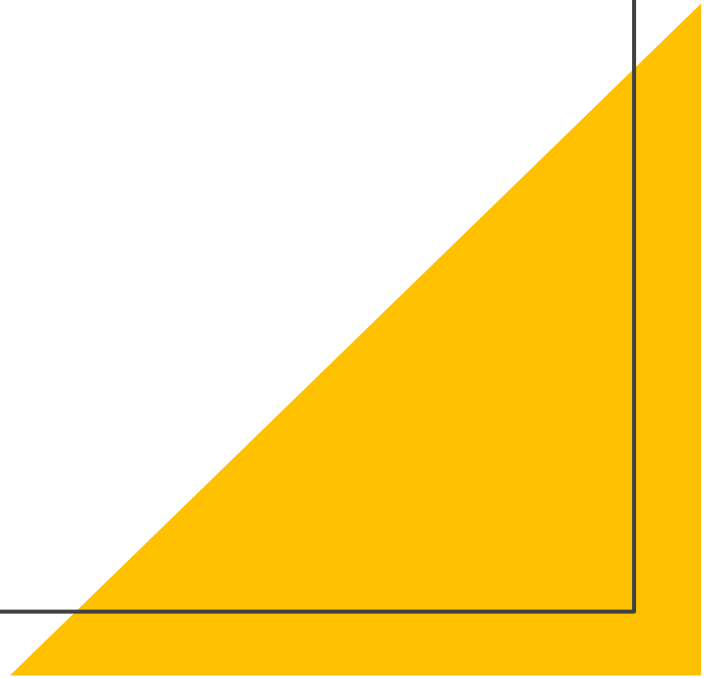


Спасибо за
обратную связь!

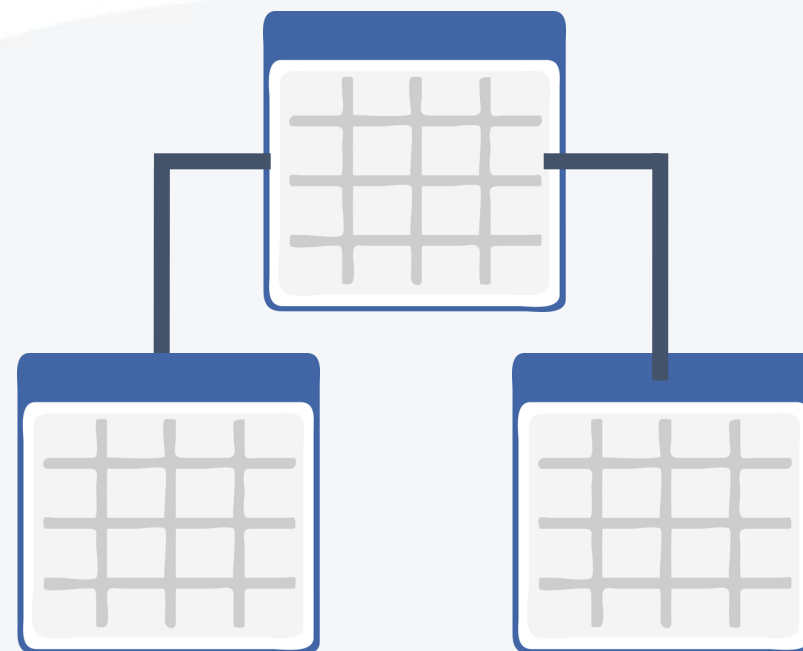
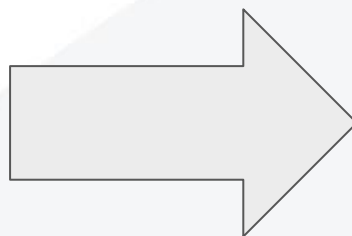
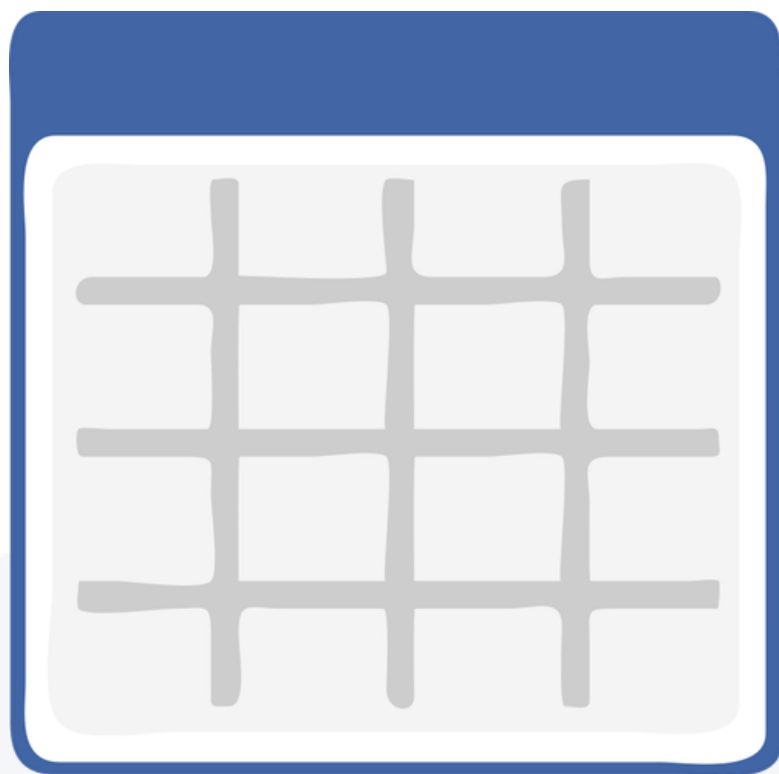


План:

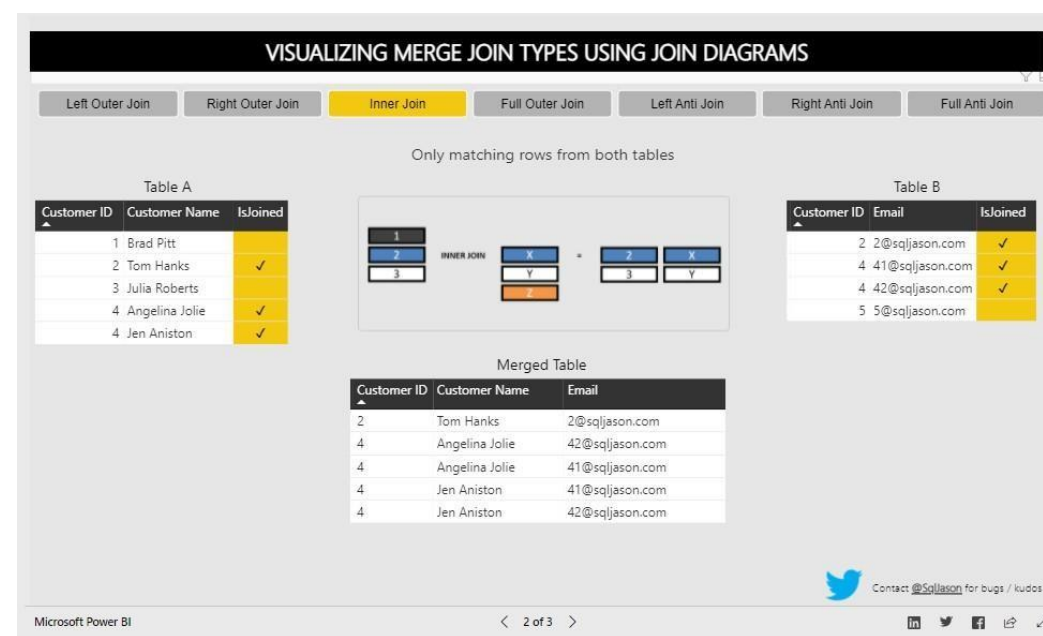
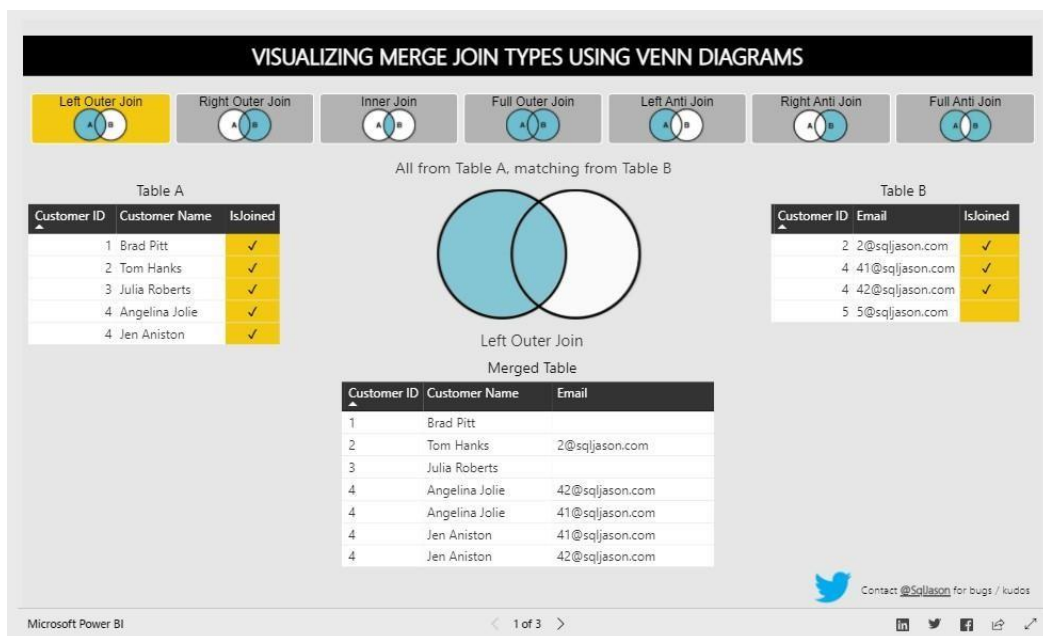
- DML, DDL, TCL
- Работа с несколькими таблицами: виды JOINS
- вопросы с моих реальных собеседований



Переход от работы с одной таблицей к нескольким



Визуализация видов JOIN в PowerBI



Что такое НОРМАЛИЗАЦИЯ ДАННЫХ?



The Relational Data Model

- **данные хранятся в множестве связанных между собой таблиц – таблицы и связи между ними**
- **Святой Грааль - Третья Форма Нормальности, данные приведены в такой вид, когда целостность данных наивысшая, а различные аномалии, неполные и лишние данные отсутствуют**

The holy grail of DB design is getting your DB into 3NF 3rd normal form – highest in terms of data integrity, reduced anomalies like redundancy and incomplete data

NORMALIZATION - НОРМАЛИЗАЦИЯ

Зачем нужна нормализация данных?



BENEFITS OF NORMALIZATION

- **Целостные данных и отсутствие «лишних» данных (improved data integrity, no redundant data)**
- **Меньший размер и лучше производительность (smaller databases - better performance)**
- **Меньше индексов, меньше менеджмента (fewer indexes, less management)**
- **Предупреждает аномалии в обновлении/изменении данных (prevents data modification anomalies)**

**Сколько
существует форм
нормальности
(normal forms)?**



Normal Forms

```
graph TD; NF[Normal Forms] --- 1NF[1NF]; NF --- 2NF[2NF]; NF --- 3NF[3NF]; NF --- BCNF[BCNF]; NF --- 4NF[4NF]; NF --- 5NF[5NF];
```

1NF

2NF

3NF

BCNF

4NF

5NF

First Normal Form(1NF)

1NF

- В каждой колонке только один вид данных
- Каждая строка имеет уникальный идентификатор

Second Normal Form(2NF)

2NF

- Удовлетворены условия 1 формы нормальности
- Частичные зависимости удалены

Third Normal Form(3NF)

- Удовлетворены условия 2 формы нормальности
- отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых

Пояснение к 3NF: Все колонки в таблице зависят от первичного ключа и не зависят друг от друга

1st Normal Form

- Удаляем повторяющиеся группы колонок, делаем так, чтобы в одной колонке был только один вид данных и одно значение, создаем/проверяем наличие уникального идентификатора каждой строки - PRIMARY KEY (PK)

(ELIMINATE REPEATING GROUPS OF DATA OR REPEATING COLUMNS + PRIMARY KEY)

1	NAME	DOB	CERTIFICATIONS	CERTIFICATION 1	CERTIFICATION 2	CERTIFICATION 3
2	Yekaterina	8/12/1993	MVP, MTA: Databases & SQL, PowerBI	PowerBI	MTA: Databases & SQL	MVP
3	Aigerim	3/1/1992	PowerBI	PowerBI	NULL	NULL
4	Meruert	10/28/1992	PowerBI, Excel, MSCA	PowerBI	Excel	MSCA
5	Aleksandra	7/15/1996	PowerBI	PowerBI	NULL	NULL

1st Normal Form

1	NAME	DOB	CERTIFICATIONS	CERTIFICATION 1	CERTIFICATION 2	CERTIFICATION 3
2	Yekaterina	8/12/1993	MVP, MTA: Databases & SQL, PowerBI	PowerBI	MTA: Databases & SQL	MVP
3	Aigerim	3/1/1992	PowerBI	PowerBI	NULL	NULL
4	Meruert	10/28/1992	PowerBI, Excel, MSCA	PowerBI	Excel	MSCA
5	Aleksandra	7/15/1996	PowerBI	PowerBI	NULL	NULL



LearnerID	LearnerName	DOB	CertificationID	CertificationName	VendorName	DateAcquired
1	Yekaterina	8/12/1993	1	MTA: Databases & SQL	Microsoft	9/12/2017
1	Yekaterina	8/12/1993	2	PowerBI	Microsoft	11/12/2019
1	Yekaterina	8/12/1993	3	Most Valuable Professional	Microsoft	2/1/2021
2	Aigerim	3/1/1992	2	PowerBI	Microsoft	12/12/2020
3	Meruert	10/28/1992	2	PowerBI	Microsoft	1/12/2021
3	Meruert	10/28/1992	3	Excel	Microsoft	1/15/2021
3	Meruert	10/28/1992	4	MSCA	Microsoft	1/15/2021
4	Aleksandra	7/15/1996	2	PowerBI	Microsoft	1/15/2021

First Normal Form(1NF)

1NF

- В каждой колонке только один вид данных
- Каждая строка имеет уникальный идентификатор

Second Normal Form(2NF)

2NF

- Удовлетворены условия 1 формы нормальности
- Частичные зависимости удалены

Third Normal Form(3NF)

- Удовлетворены условия 2 формы нормальности
- отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых

2nd Normal Form

- **1NF + Частичные зависимости от РК удалены (в случае с композитными ключами)**

1NF + NO PARTIAL DEPENDENCIES ON THE PK (in case with composite key).

***Если РК задан одной колонкой, то ничего не нужно делать, таблица УЖЕ в 2NF**

2nd Normal Form

- 1NF + Частичные зависимости от РК удалены

LearnerID	LearnerName	DOB
1	Yekaterina	8/12/1993
2	Aigerim	3/1/1992
3	Meruert	10/28/1992
4	Aleksandra	7/15/1996

CertificationID	CertificationName	VendorName
1	MTA: Databases & SQL	Microsoft
2	PowerBI	Microsoft
3	Most Valuable Professional	Microsoft
4	MSCA	Microsoft
5	Excel	Microsoft

LearnerID	CertificationID	DateAcquired
1	1	9/12/2017
1	2	11/12/2019
1	3	2/1/2021
2	2	12/12/2020
3	2	1/12/2021
3	5	1/15/2021
3	4	1/15/2021
4	2	1/15/2021

3rd Normal Form

- 1NF + 2NF + каждая колонка должна зависеть полностью от РК (применимо только к таблицам с ключем состоящим из одного поля)

CertificationID	CertificationName	VendorName
1	MTA: Databases & SQL	Microsoft
2	PowerBI	Microsoft
3	Most Valuable Professional	Microsoft
4	MSCA	Microsoft
5	Excel	Microsoft



CertificationID	CertificationName	VendorID
1	MTA: Databases & SQL	1
2	PowerBI	1
3	Most Valuable Professional	1
4	MSCA	1
5	Excel	1

VendorID	VendorName
1	Microsoft
2	Amazon

First Normal Form(1NF)

1NF

- В каждой колонке только один вид данных
- Каждая строка имеет уникальный идентификатор

Second Normal Form(2NF)

2NF

- Удовлетворены условия 1 формы нормальности
- Частичные зависимости удалены

Third Normal Form(3NF)

- Удовлетворены условия 2 формы нормальности
- отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых

3rd Normal Form

LearnerID	LearnerName	DOB
1	Yekaterina	8/12/1993
2	Aigerim	3/1/1992
3	Meruert	10/28/1992
4	Aleksandra	7/15/1996

LearnerID	CertificationID	DateAcquired
1	1	9/12/2017
1	2	11/12/2019
1	3	2/1/2021
2	2	12/12/2020
3	2	1/12/2021
3	5	1/15/2021
3	4	1/15/2021
4	2	1/15/2021

CertificationID	CertificationName	VendorID
1	MTA: Databases & SQL	1
2	PowerBI	1
3	Most Valuable Professional	1
4	MSCA	1
5	Excel	1

VendorID	VendorName
1	Microsoft
2	Amazon

Практика

Цель: 3.15



SQL Language Statements

```
graph TD; A[SQL Language Statements] --> B[DML]; A --> C[DDL]; A --> D[DCL]; A --> E[TCL]; B --> B1[SELECT]; B --> B2[INSERT]; B --> B3[UPDATE]; B --> B4[DELETE]; C --> C1[CREATE]; C --> C2[ALTER]; C --> C3[DROP]; D --> D1[GRANT]; D --> D2[REVOKE]; E --> E1[BEGIN TRAN]; E --> E2[COMMIT TRAN]; E --> E3[ROLLBACK];
```

DML

SELECT
INSERT
UPDATE
DELETE

DDL

CREATE
ALTER
DROP

DCL

GRANT
REVOKE

TCL

BEGIN
TRAN
COMMIT
TRAN
ROLLBACK

SQL Language Statements

```
graph TD; A[SQL Language Statements] --> B[DML]; A --> C[DDL]; A --> D[DCL]; A --> E[TCL]; B --> B1[SELECT]; B --> B2[INSERT]; B --> B3[UPDATE]; B --> B4[DELETE]; C --> C1[CREATE]; C --> C2[ALTER]; C --> C3[DROP]; D --> D1[GRANT]; D --> D2[REVOKE]; E --> E1[BEGIN TRAN]; E --> E2[COMMIT TRAN]; E --> E3[ROLLBACK];
```

DML

SELECT
INSERT
UPDATE
DELETE

DDL

CREATE
ALTER
DROP


DCL

GRANT
REVOKE

TCL

BEGIN
TRAN
COMMIT
TRAN
ROLLBACK

Нюанс с NULL

A photograph of a railway track receding into a dense fog. The tracks are made of metal rails and wooden sleepers, leading the viewer's eye towards the horizon. The surrounding landscape is covered in dry, brownish grass. In the lower right foreground, a rectangular road sign on a metal post reads 'ПУСТОТА' in Cyrillic, which translates to 'The Void' or 'Emptiness'. The overall atmosphere is mysterious and contemplative.

ПУСТОТА

CONSTRAINTS (ограничения)

- **NOT NULL** – ограничение, которое говорит, что эта колонка не может быть пустой (NULL), обязательно должна иметь значение
- **PRIMARY KEY** добавляет primary key constraint к колонке, это хорошая практика таким образом указывать в каждой таблице ту колонку, которая содержит первичный ключ.
- **IDENTITY** – ограничение, которое автоматически увеличивает на 1 и вставляет новое значение когда добавляется новая строка в таблицу. IDENTITY = IDENTITY(1,1) также означает что счет начинается с единицы и при вставке новой строки, значение будет увеличиваться на 1 (1,2,3,4...), но можно это изменить, и указать например IDENTITY(2,4) – тогда счет начнется с 2, и увеличиваться каждый раз на 4 (2,6,10, 14...) – в этом случае 2 это seed (старт), а 4 – это increment (увеличитель)
- **DEFAULT** – можно задать значение колонки по умолчанию, когда другое значение не было указано

WHERE и операторы сравнения

OPERATOR	MEANING
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to
!=	Not equal to (not ISO standard)
!<	Not less than (not ISO standard)
!>	Not greater than (not ISO standard)

NULL – это пустышка, поле БЕЗ
значения



NULL – это пустышка, поле БЕЗ значения

- **NULL** – это НЕ ноль, это НЕ пробел(ы)
- **Операторы сравнения =, <, <> не работают с NULL**
- **IS NULL и IS NOT NULL** используются для проверки не пустых значений, вместо операторов сравнения

```
SELECT column_names FROM table_name  
WHERE column_name IS NULL
```

```
SELECT column_names FROM table_name  
WHERE column_name IS NOT NULL
```

- Колонка может содержать пустые записи, если отсутствует ограничение **NOT NULL** constraint

SQL Language Statements

```
graph TD; A[SQL Language Statements] --> B[DML]; A --> C[DDL]; A --> D[DCL]; A --> E[TCL]; B --> B1[SELECT]; B --> B2[INSERT]; B --> B3[UPDATE]; B --> B4[DELETE]; C --> C1[CREATE]; C --> C2[ALTER]; C --> C3[DROP]; D --> D1[GRANT]; D --> D2[REVOKE]; E --> E1[BEGIN TRAN]; E --> E2[COMMIT TRAN]; E --> E3[ROLLBACK];
```

DML

SELECT
INSERT
UPDATE
DELETE

DDL

CREATE
ALTER
DROP

DCL

GRANT
REVOKE

TCL

BEGIN
TRAN
COMMIT
TRAN
ROLLBACK

UPDATE

```
UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

(!) Если WHERE отсутствует, то будут обновлены ВСЕ записи в таблице. Обязательно указывайте WHERE

SQL Language Statements

```
graph TD; A[SQL Language Statements] --> B[DML]; A --> C[DDL]; A --> D[DCL]; A --> E[TCL]; B --> B1[SELECT]; B --> B2[INSERT]; B --> B3[UPDATE]; B --> B4[DELETE]; C --> C1[CREATE]; C --> C2[ALTER]; C --> C3[DROP]; D --> D1[GRANT]; D --> D2[REVOKE]; E --> E1[BEGIN TRAN]; E --> E2[COMMIT TRAN]; E --> E3[ROLLBACK];
```

DML

SELECT
INSERT
UPDATE
DELETE

DDL

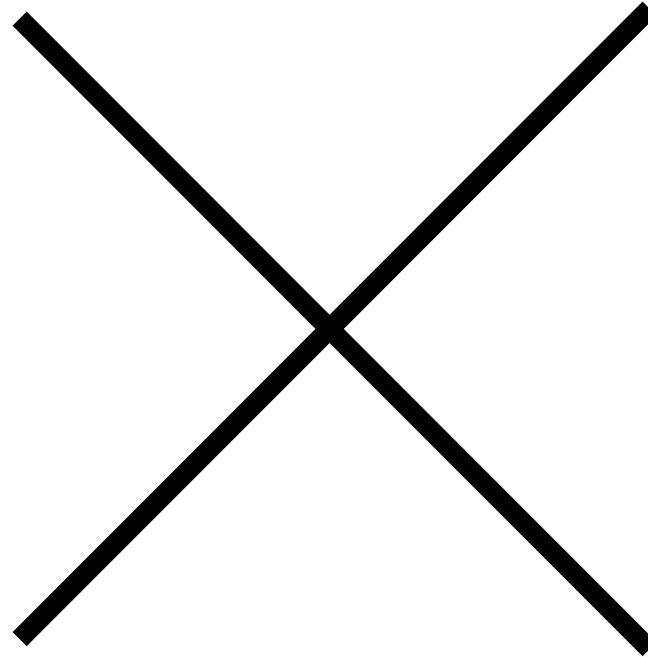
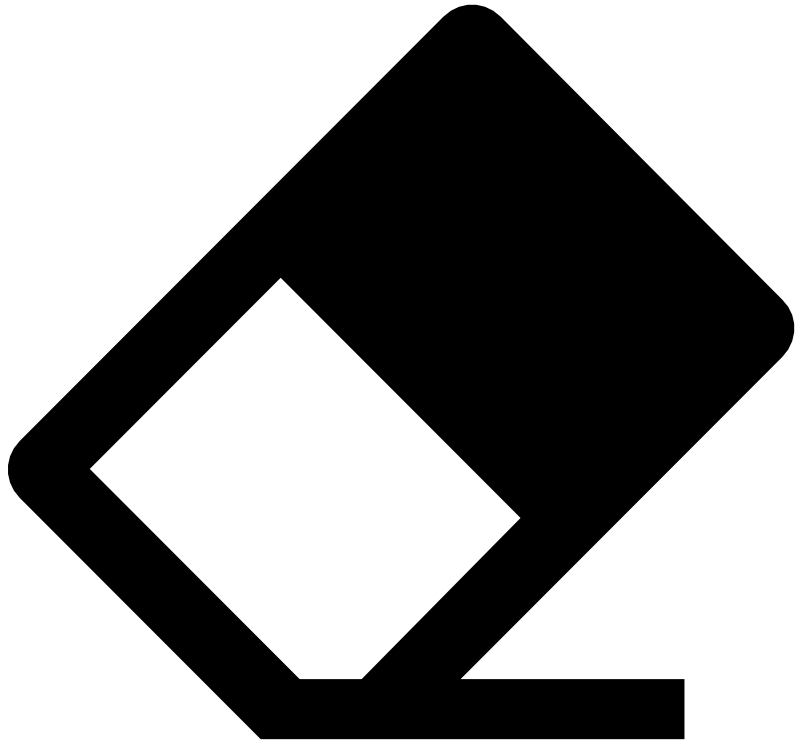
CREATE
ALTER
DROP

DCL

GRANT
REVOKE

TCL

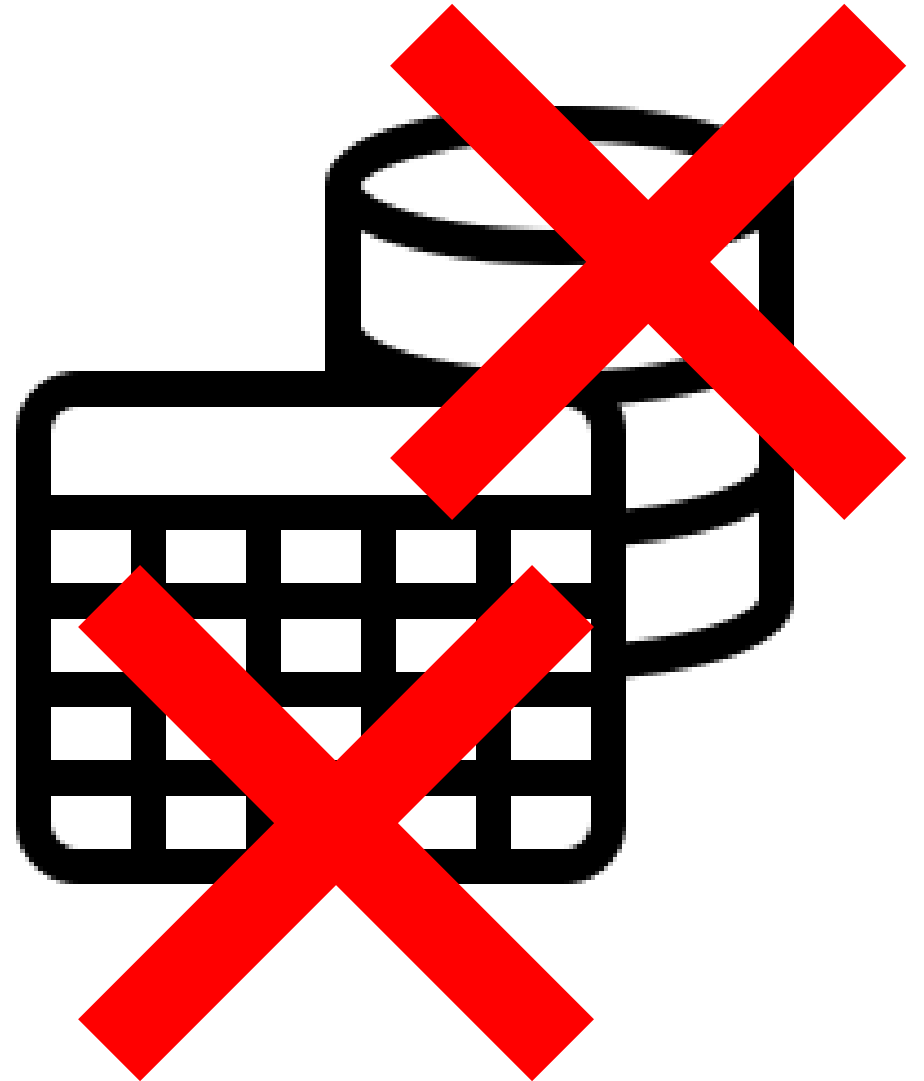
BEGIN
TRAN
COMMIT
TRAN
ROLLBACK



DROP vs **DELETE** vs **TRUNCATE**

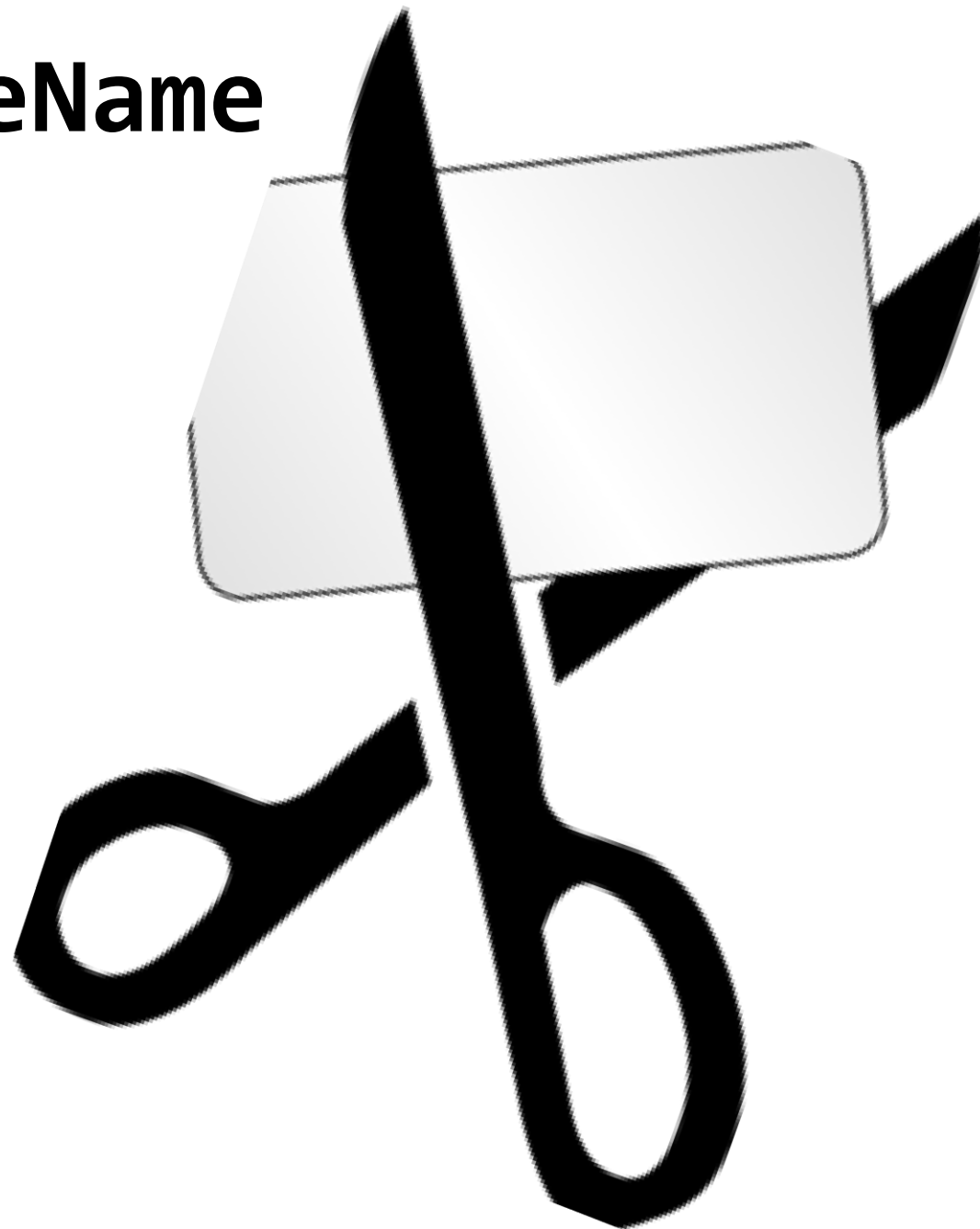
DROP OBJECT ObjectName

- удаляет объект
- **DROP DATABASE** DatabaseName - удаляет базу данных
- **DROP TABLE** TableName - удаляет таблицу
- DROP TABLE не работает, если вы пытаетесь удалить им таблицу, в которой есть хотя бы одно поле на которое ссылается другая таблица с помощью FOREIGN KEY constraint. Сначала нужно будет удалить все referencing FOREIGN KEY constraint в других таблицах, и только потом вы сможете удалить таблицу.



TRUNCATE TABLE TableName

- удаляет все строки из таблицы, а также место, занимаемое этими строками
- делает то же самое, что и DELETE без WHERE, но не логирует каждую удаленную строку, потому что быстрее, чем DELETE
- не работает, если на таблицу ссылаются другие через foreign key constraint

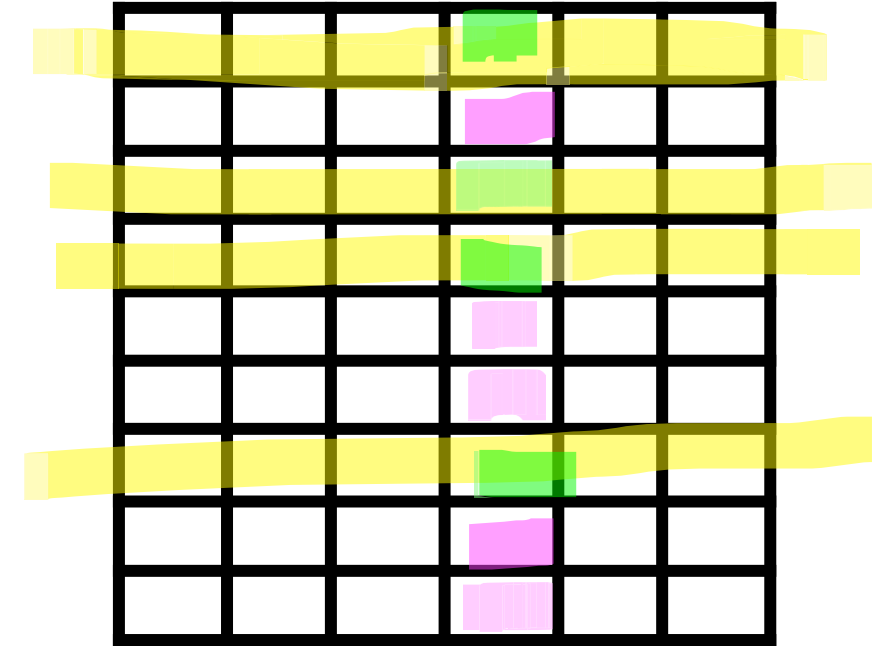


DELETE

DELETE FROM table_name

DELETE FROM table_name **WHERE** condition

- можно использовать вместе с WHERE
- удаляет строки из таблицы, НО НЕ удаляет место, занимаемое этими строками
- DELETE без WHERE, делает то же самое, что и TRUNCATE, но логирует каждую удаленную строку, потому медленнее, чем TRUNCATE



В чем разница между операторами DELETE, DROP и TRUNCATE?

DELETE	TRUNCATE	DROP
Используется для удаления строки в таблице	Используется для удаления ВСЕХ строк из таблицы	Используется для удаления таблицы/объекта
Вы можете восстановить данные после удаления	Вы не можете восстановить данные (операции логируются по разному, но в SQL Server есть возможность сделать rollback - откат транзакции)	Вы не можете восстановить данные (операции логируются по разному, но в SQL Server есть возможность сделать rollback - откат транзакции)
DML-команда	DDL-команда	DDL-команда
Медленнее, чем TRUNCATE	Быстрее	Быстрее

SQL Language Statements

```
graph TD; A[SQL Language Statements] --> B[DML]; A --> C[DDL]; A --> D[DCL]; A --> E[TCL]; B --> B1[SELECT]; B --> B2[INSERT]; B --> B3[UPDATE]; B --> B4[DELETE]; C --> C1[CREATE]; C --> C2[ALTER]; C --> C3[DROP]; D --> D1[GRANT]; D --> D2[REVOKE]; E --> E1[BEGIN TRAN]; E --> E2[COMMIT TRAN]; E --> E3[ROLLBACK];
```

DML

SELECT
INSERT
UPDATE
DELETE

DDL

CREATE
ALTER
DROP

DCL

GRANT
REVOKE

TCL

BEGIN
TRAN
COMMIT
TRAN
ROLLBACK

BEGIN TRANSACTION tran_name

BEGIN TRAN tran_name

- начало транзакции

ROLLBACK TRANSACTION tran_name

ROLLBACK TRAN tran_name

- отмена транзакции

COMMIT

-завершение транзакции, после которого данные уже не
ВОССТАНОВИТЬ

DML команды

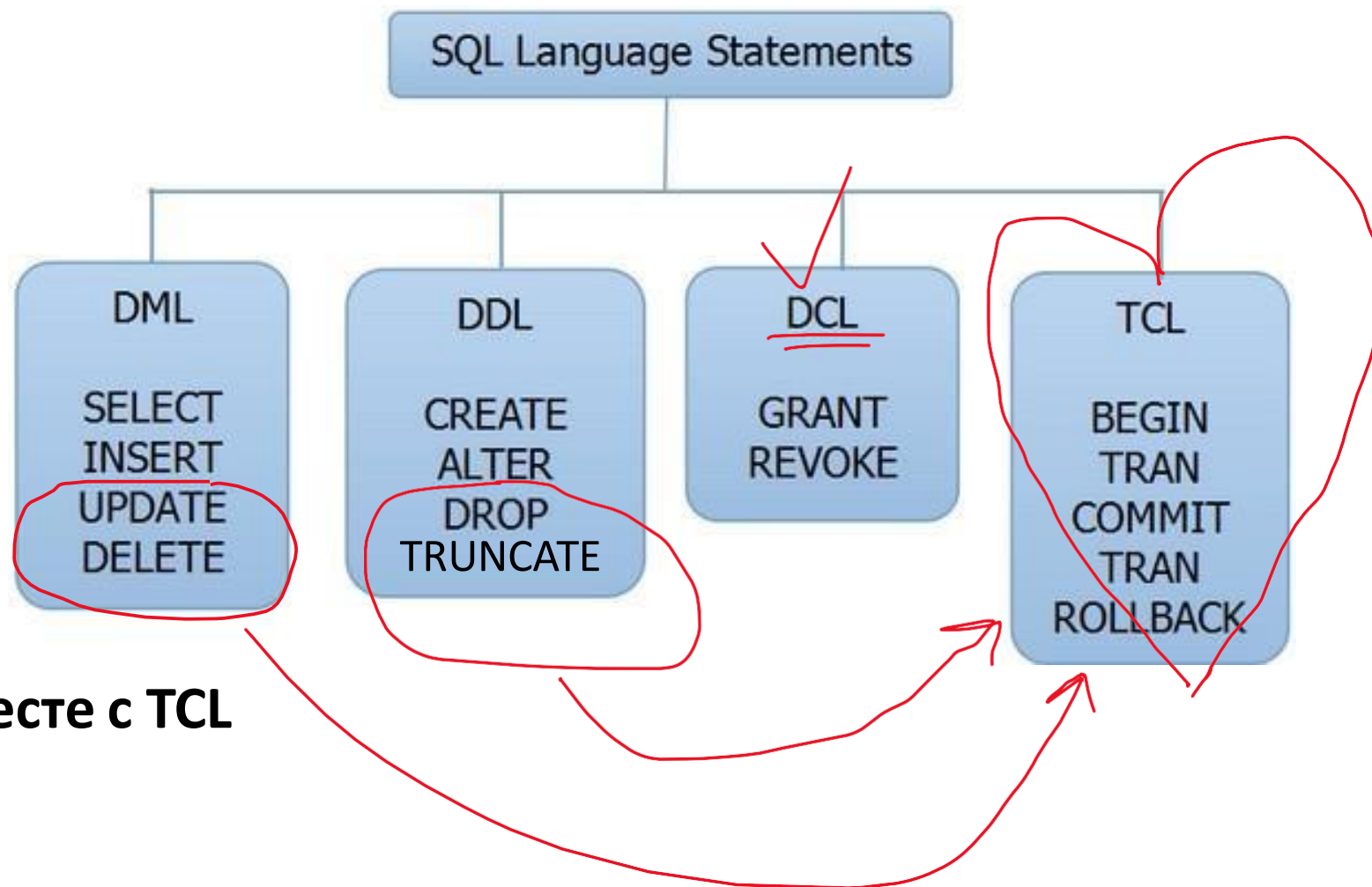
- UPDATE
- DELETE

а также DDL команды

- DROP
- TRUNCATE

• Лучше использовать вместе с TCL

- BEGIN TRAN
- ROLLBACK TRAN
- COMMIT



SQL Language Statements

```
graph TD; A[SQL Language Statements] --> B[DML]; A --> C[DDL]; A --> D[DCL]; A --> E[TCL]; B --> B1[SELECT]; B --> B2[INSERT]; B --> B3[UPDATE]; B --> B4[DELETE]; C --> C1[CREATE]; C --> C2[ALTER]; C --> C3[DROP]; D --> D1[GRANT]; D --> D2[REVOKE]; E --> E1[BEGIN]; E --> E2[TRAN]; E --> E3[COMMIT]; E --> E4[TRAN]; E --> E5[ROLLBACK];
```

DML

SELECT
INSERT
UPDATE
DELETE

DDL

CREATE
ALTER
DROP

DCL

GRANT
REVOKE

TCL

BEGIN
TRAN
COMMIT
TRAN
ROLLBACK

Aggregation functions - Функции агрегации

Count() - количество

Sum() - сумма

Avg() - среднее

Min() - минимум

Max() - максимум

ALIASES (псевдонимы)

- присваивают таблице или столбцу в таблице временное имя
- используют для облегчения чтения названий колонок или таблиц

```
SELECT column_name AS alias_name  
FROM table_name;
```

GROUP BY

- Используется для группировки нескольких строк по одной или нескольким колонкам
- Всегда используется вместе с функциями агрегации

```
SELECT column_name(s), aggregation_function() as Alias
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
```

GROUP BY

Employee

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

```
SELECT DeptID, AVG(Salary)  
FROM Employee  
GROUP BY DeptID;
```

GROUP BY
Employee Table
using DeptID

DeptID	AVG(Salary)
1	3000.00
2	4000.00
3	4250.00

WHERE vs HAVING

- WHERE фильтрует запрос ДО группировки
- WHERE используется вместе с Select, Insert, Update
- Функции агрегации НЕ могут быть использованы в WHERE clause, за исключением случая, когда используется subquery (подзапрос)

- HAVING фильтрует запрос ПОСЛЕ группировки
- HAVING используется только вместе с Select
- Функции агрегации МОГУТ быть использованы в HAVING clause

SQL Language Statements

```
graph TD; A[SQL Language Statements] --> B[DML]; A --> C[DDL]; A --> D[DCL]; A --> E[TCL]; B --> B1[SELECT]; B --> B2[INSERT]; B --> B3[UPDATE]; B --> B4[DELETE]; C --> C1[CREATE]; C --> C2[ALTER]; C --> C3[DROP]; D --> D1[GRANT]; D --> D2[REVOKE]; E --> E1[BEGIN]; E --> E2[TRAN]; E --> E3[COMMIT]; E --> E4[TRAN]; E --> E5[ROLLBACK];
```

DML

SELECT
INSERT
UPDATE
DELETE

DDL

CREATE
ALTER
DROP

DCL

GRANT
REVOKE

TCL

BEGIN
TRAN
COMMIT
TRAN
ROLLBACK

DISTINCT

- Уникальные (неповторяющиеся) значения из колонки

```
SELECT DISTINCT column_name  
FROM table_name
```


TOP N

- Верхние N строк

```
SELECT TOP N column_name(s)  
FROM table_name
```

В MySQL используется оператор **limit N**, который ставится в конце запроса

ORDER BY

Сортировка по возрастанию

```
SELECT column_name(s)  
FROM table_name  
ORDER BY column_name(s) ASC
```

Сортировка по убыванию

```
SELECT column_name(s)  
FROM table_name  
ORDER BY column_name(s) DESC
```



JOINS

Primary Key and Foreign Key – нужны в таблицах для создания связей (relations)

Artists			Albums		
ArtistId	ArtistName	Desc	AlbumId	AlbumName	ArtistId
1	AC/DC	One of t...	1	NellyVille	3
2	U2	Another	2	Black Ice	1
3	Nelly	When y...	3	Ballbreaker	1
4	Lorde	From N...	4	October	2

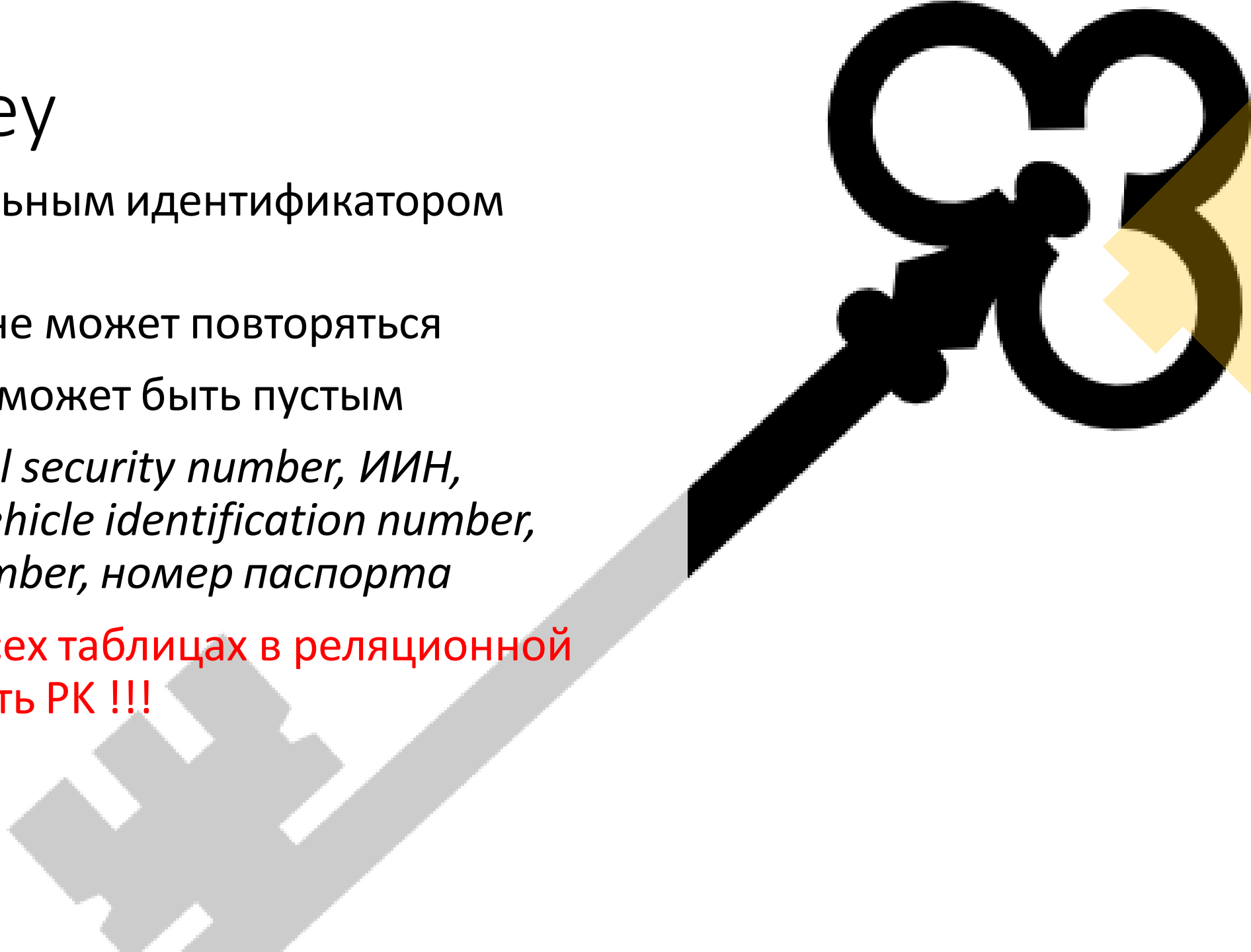


Primary Key

- Служит уникальным идентификатором каждой строки
- Уникальный - не может повторяться
- NOT NULL - Не может быть пустым

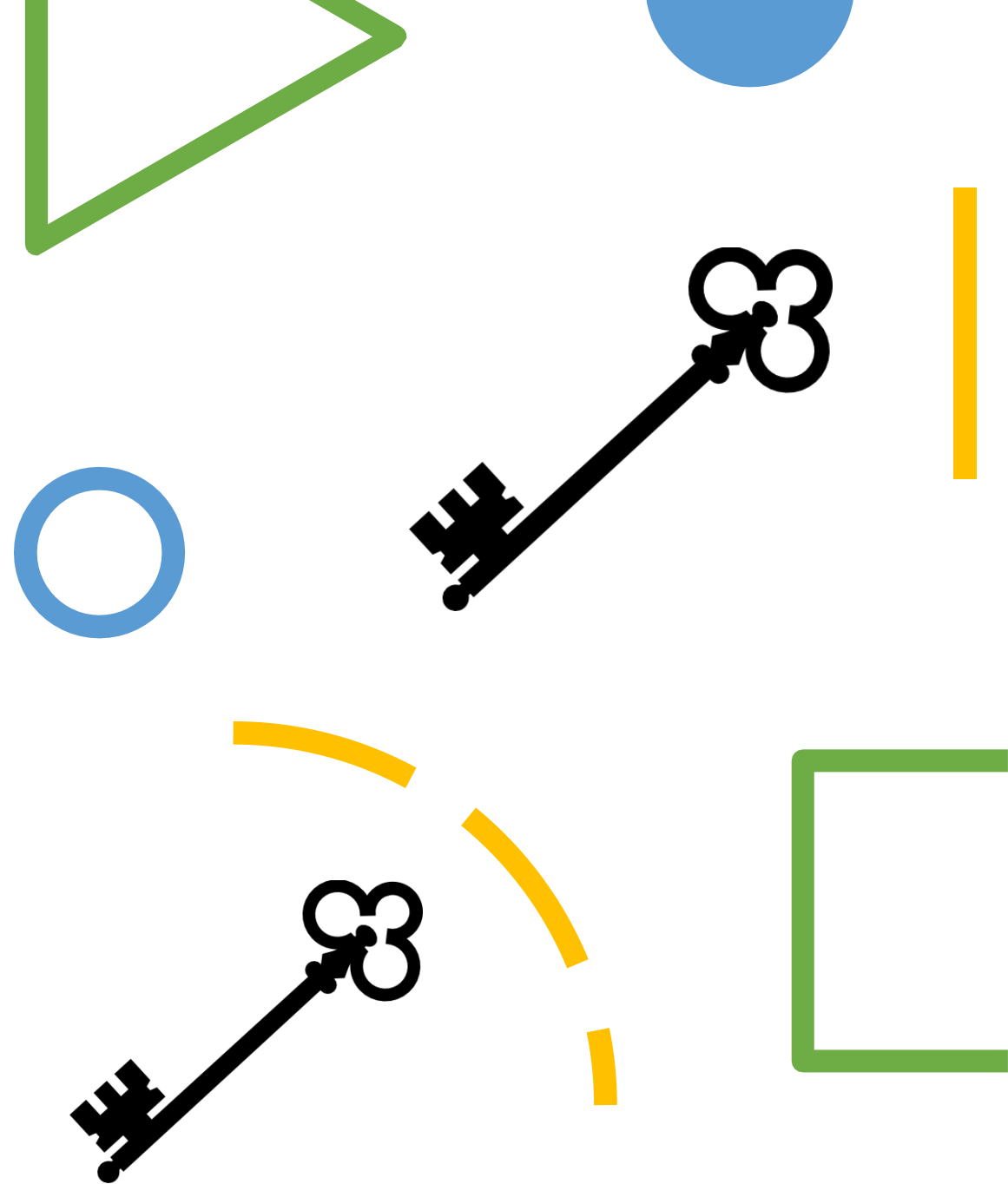
Например: social security number, ИИН, email address, vehicle identification number, driver licence number, номер паспорта

Все записи во всех таблицах в реляционной БД должны иметь РК !!!



Foreign Key

- это когда РК находится «в гостях» у другой таблицы, чтобы мы могли по нему вернуться «домой» и выяснить все о том, кто пришел к нам «в гости»
- является «ссылкой» на РК
- Может быть НЕ УНИКАЛЬНЫМ, повторяться в табл



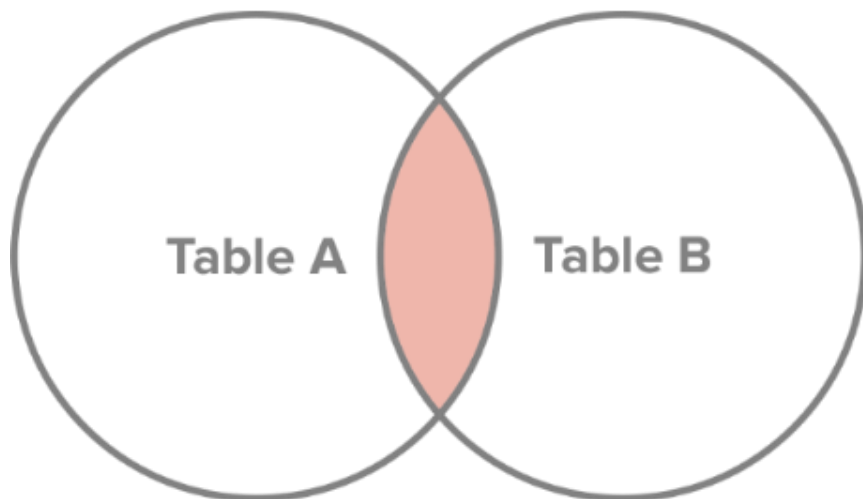
Primary Key and Foreign Key – нужны в таблицах для создания связей (relations)



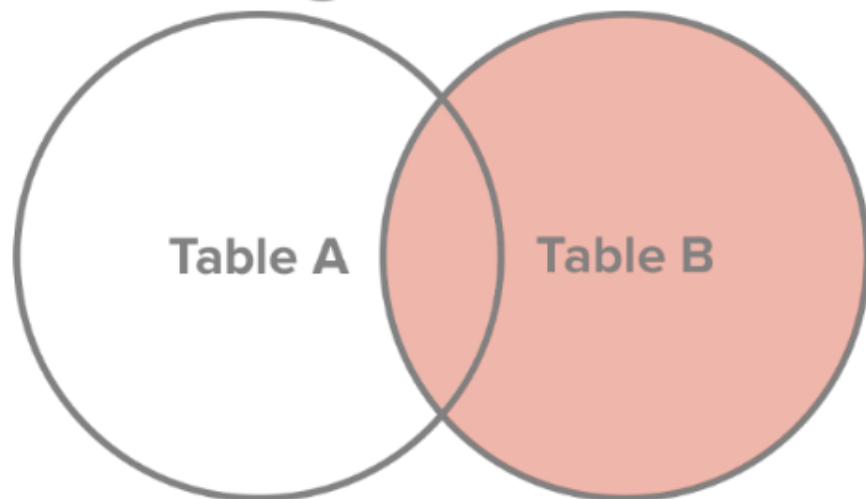
ArtistId	ArtistName	Desc
1	AC/DC	One of t...
2	U2	Another
3	Nelly	When y...
4	Lorde	From N...

AlbumId	AlbumName	ArtistId
1	NellyVille	3
2	Black Ice	1
3	Ballbreaker	1
4	October	2

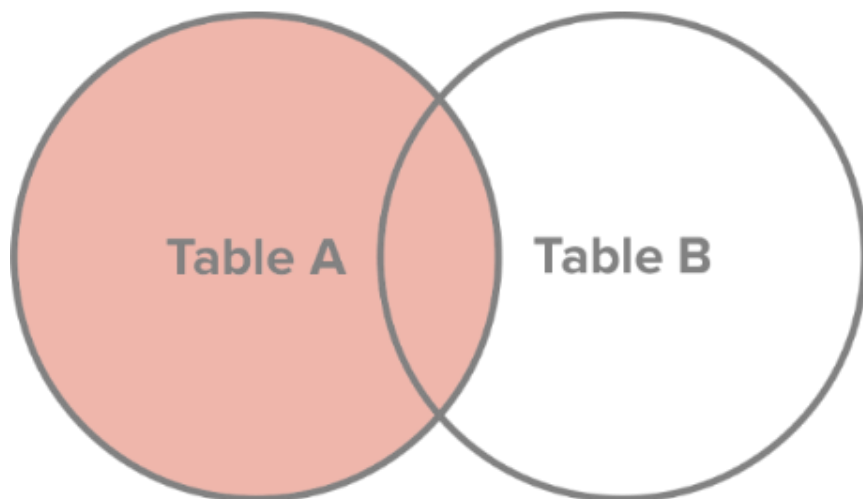
Inner Join



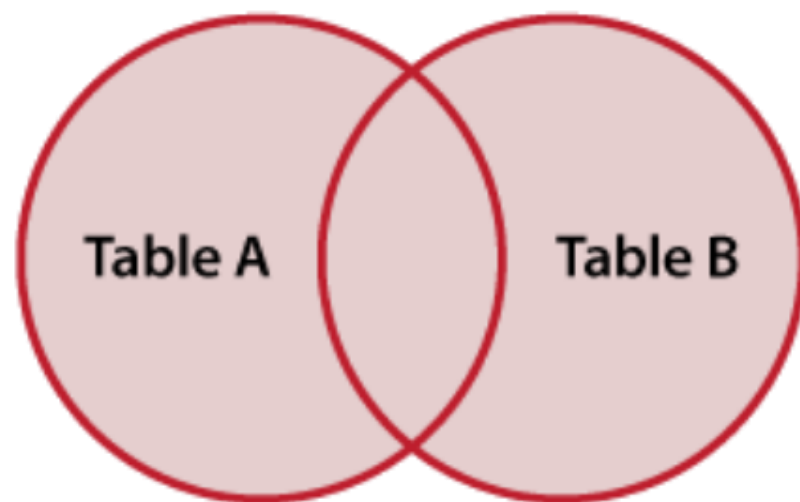
Right Join



Left Join

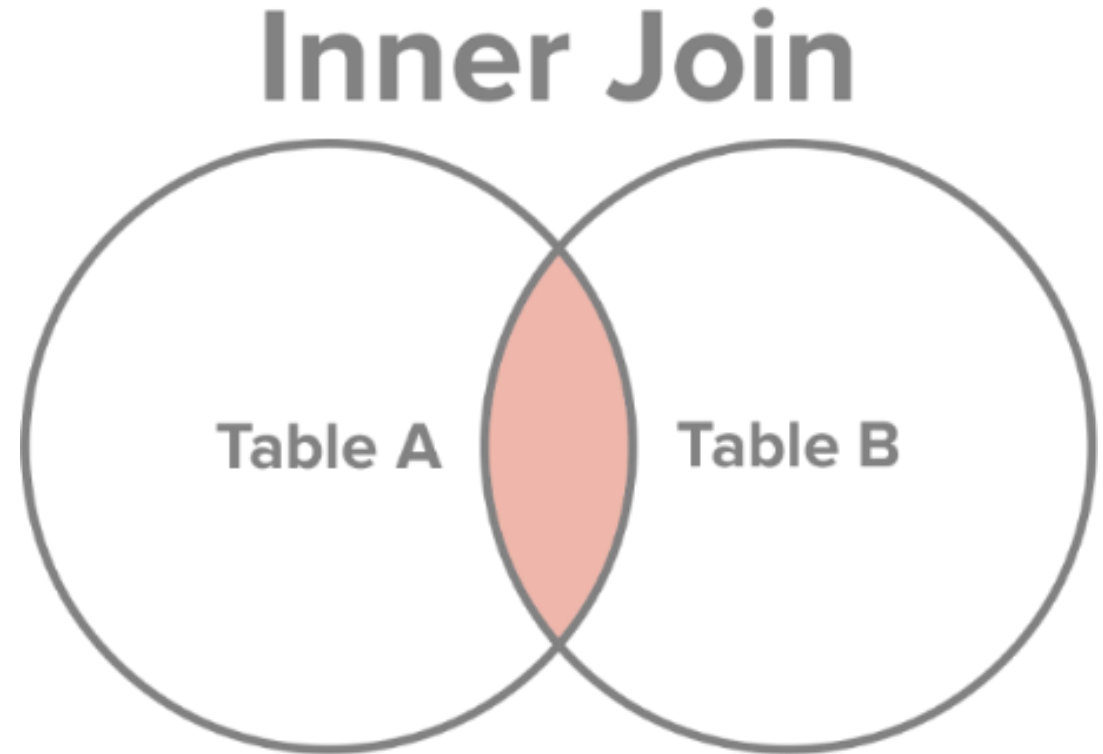


FULL OUTER JOIN



(INNER) JOIN

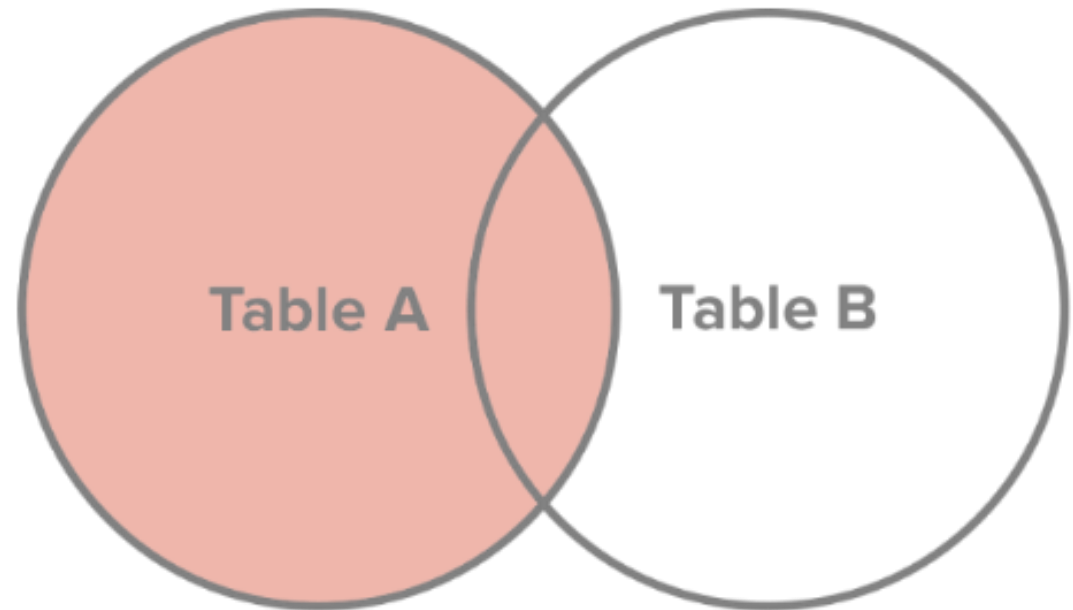
- Возвращает только совпадения из 2 таблиц
- Используется чаще других видов JOINS
- JOIN это то же самое, что и INNER JOIN



LEFT (OUTER) JOIN

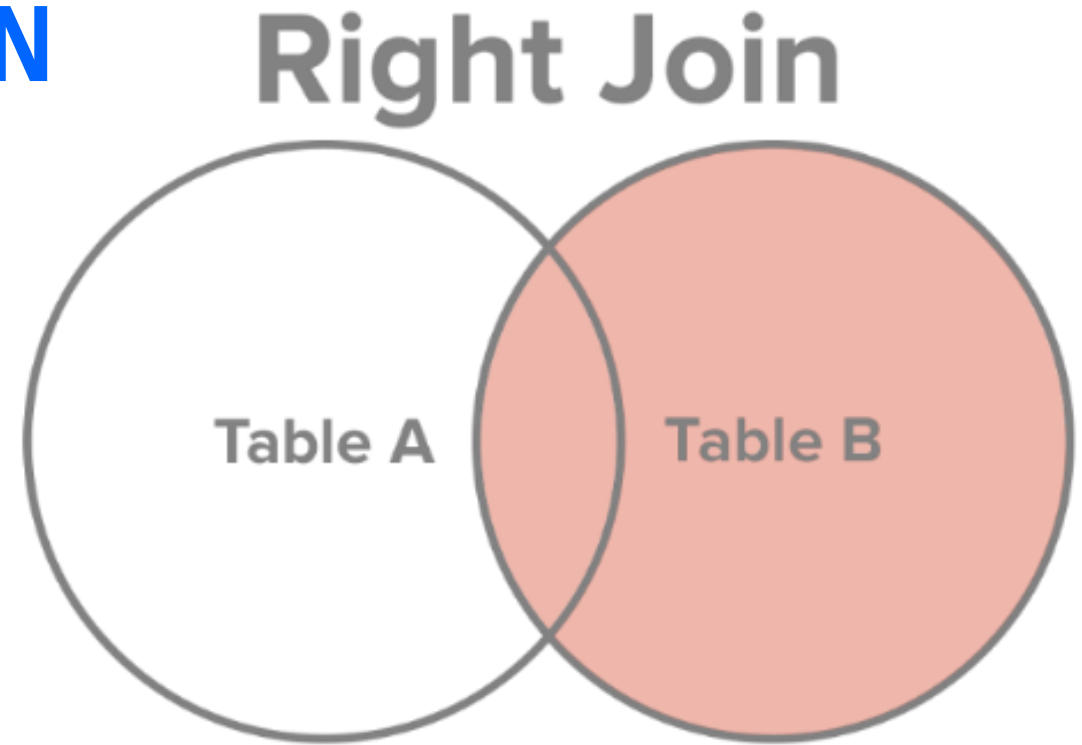
- Возвращает все строки из левой таблицы И совпадения из правой таблицы
- LEFT JOIN и LEFT OUTER JOIN это одно и то же
- Можно поменять местами таблицы и сделать RIGHT JOIN чтобы получить тот же результат

Left Join



RIGHT (OUTER) JOIN

- Возвращает все строки из правой таблицы И совпадения из левой таблицы
- RIGHT JOIN и RIGHT OUTER JOIN это одно и то же
- Можно поменять местами таблицы и сделать LEFT JOIN чтобы получить тот же результат



FULL (OUTER) JOIN

- Возвращает все строки из обеих таблиц
- FULL JOIN и FULL OUTER JOIN это одно и то же

FULL OUTER JOIN

