DDL COMMANDS AND SQL BASICS

BY VISHWA PATEL

SHIFTKEY LABS

JANUARY EVENTS

JAN 6
ACADEMY UP: SQL FOUNDATIONS
6:00PM-8:00PM
Goldberg Computer Science Building

JAN 10
INDUSTRY SHOWCASE: UBISOFT
4:00PM-5:30PM
Ubisoft Halifax

JAN 13
ACADEMY UP: SQL FOUNDATIONS
6:00PM-8:00PM
Goldberg Computer Science Building

JAN 15
INDUSTRY SHOWCASE: CIBC
Cancelled

JAN 16
CYBERSECURITY IN NETWORKS
6:00PM-8:00PM
Goldberg Computer Science Building

JAN 17-19
GEN AI AND MENTAL WELLBEING HACKATHON
Goldberg Computer Science Building

JAN 20
ACADEMY UP: SQL FOUNDATIONS
6:00PM-8:00PM
Goldberg Computer Science Building

JAN 23
INDUSTRY SHOWCASE: XEROX CANADA
4:00PM-5:30PM
Goldberg Computer Science Building

JAN 24-26
GLOBAL GAME JAM
CHEB and Goldberg Computer Science Building

JAN 25
GAME JAM NEXT
10:00AM-3:00PM
The PIER

JAN 27
INDUSTRY SHOWCASE: IBM
4:00PM-5:30PM
IBM Client Innovation Centre Nova Scotia

JAN 27
ACADEMY UP: SQL FOUNDATIONS
6:00PM-8:00PM
Goldberg Computer Science Building

JAN 29
PROMPT ENGINEERING
6:00PM-7:00PM
Goldberg Computer Science Building

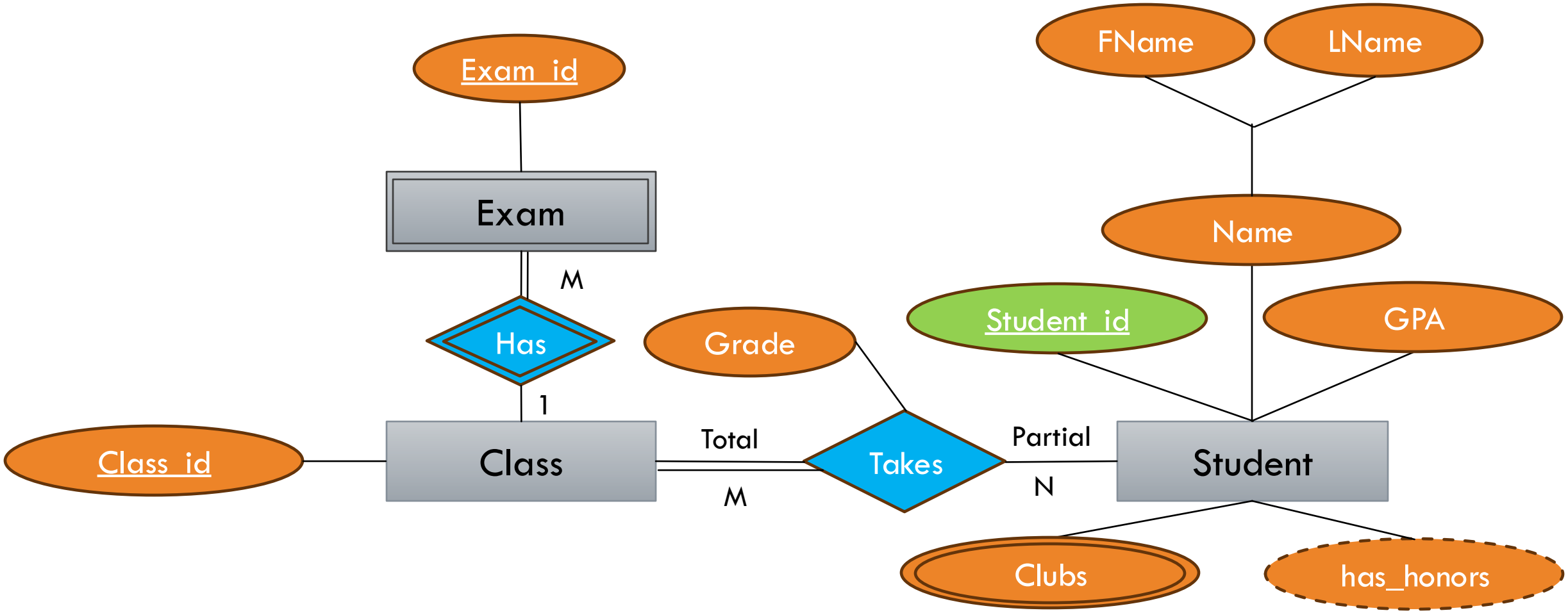
JAN 30
SCRUM AGILE WORKSHOP
5:30-7:30PM
Goldberg Computer Science Building

CONVERTING TO DATABASE TABLES

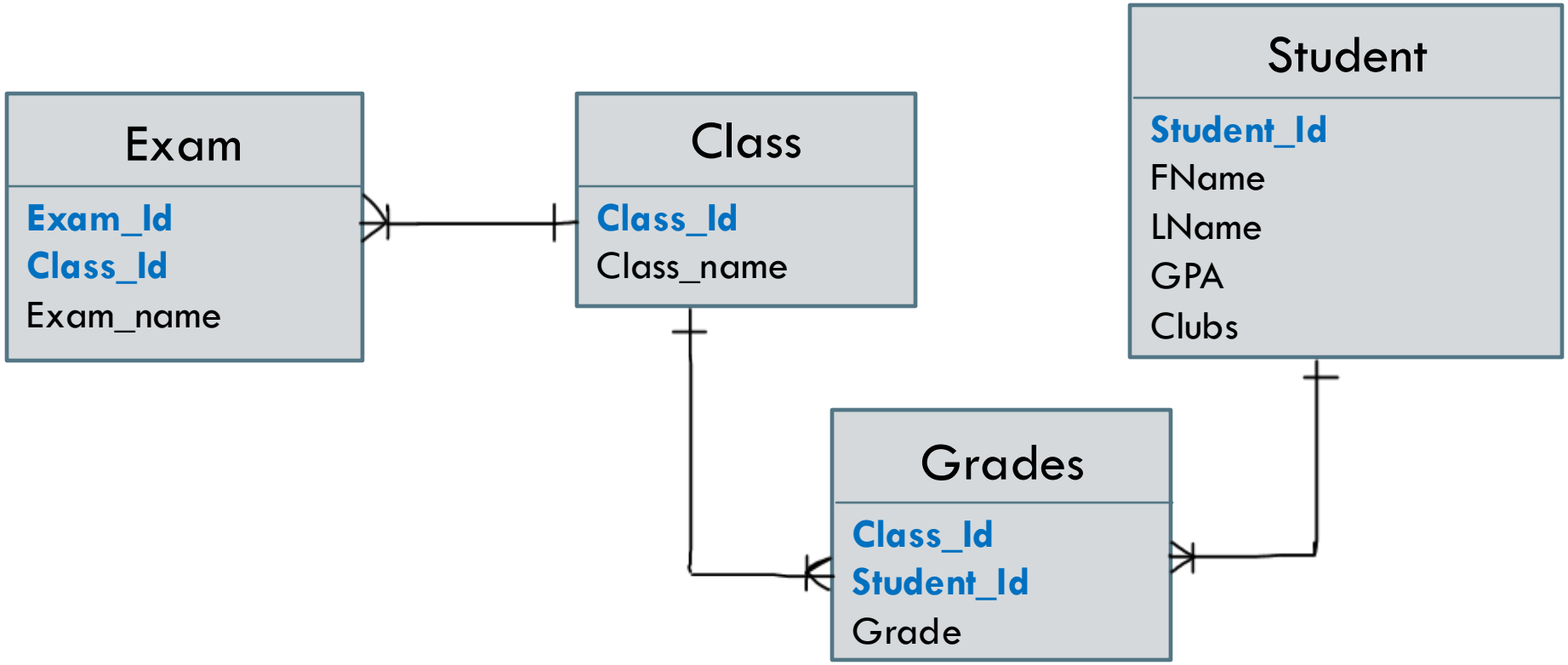
ER DIAGRAM TO DATABASE TABLES

CONVERTING TO DATABASE TABLES

- Using the ER model, we are going to convert the entities into tables and then design the database schema
- Step 1: Creating Entities
- Step 2: Mapping Weak Entity
- Step 3: Mapping relationships(1:1, 1:N, N:M)


Exam_id
Exam
M
Has
1
Class_id
Class
Total
M
Grade
Takes
Partial
N
Student
FName
LName
Name
Student_id
GPA
Clubs
has_honors

CONVERTING TO ER DIAGRAM



STEP 1: CREATING ENTITIES

- Take all the simple attributes for the entity and create a table header for them
- We have Student and Class as entites.
- Composite attributes are broken down.

Student	Student_Id	FName	LName	GPA	Clubs

Class	Class_Id	Class_name

STEP 2: MAPPING WEAK ENTITY TYPES

- Create a new relational(table) that has all the simple attributes of the table.
- Primary key will be the primary key of itself and the owner relation, i.e. here it will be Exam_Id and Class_Id acting together as a primary key for the Exam table called Compound Key.

Student	Student_Id	FName	LName	GPA	Clubs

Class	Class_Id	Class_name

Exam	Exam_Id	Class_Id	Exam_name

STEP 3A: MAPPING ONE-TO-ONE RELATION

- All relationships are called binary cause 2 relational tables are involved
- There needs to be a foreign key of entity which has total participation with another entity.
- If both are partial or are total participation, then you can choose to add a foreign key in whichever entity you prefer.
- No 1:1 relation in the High School database! So, no changes to the database table structure.

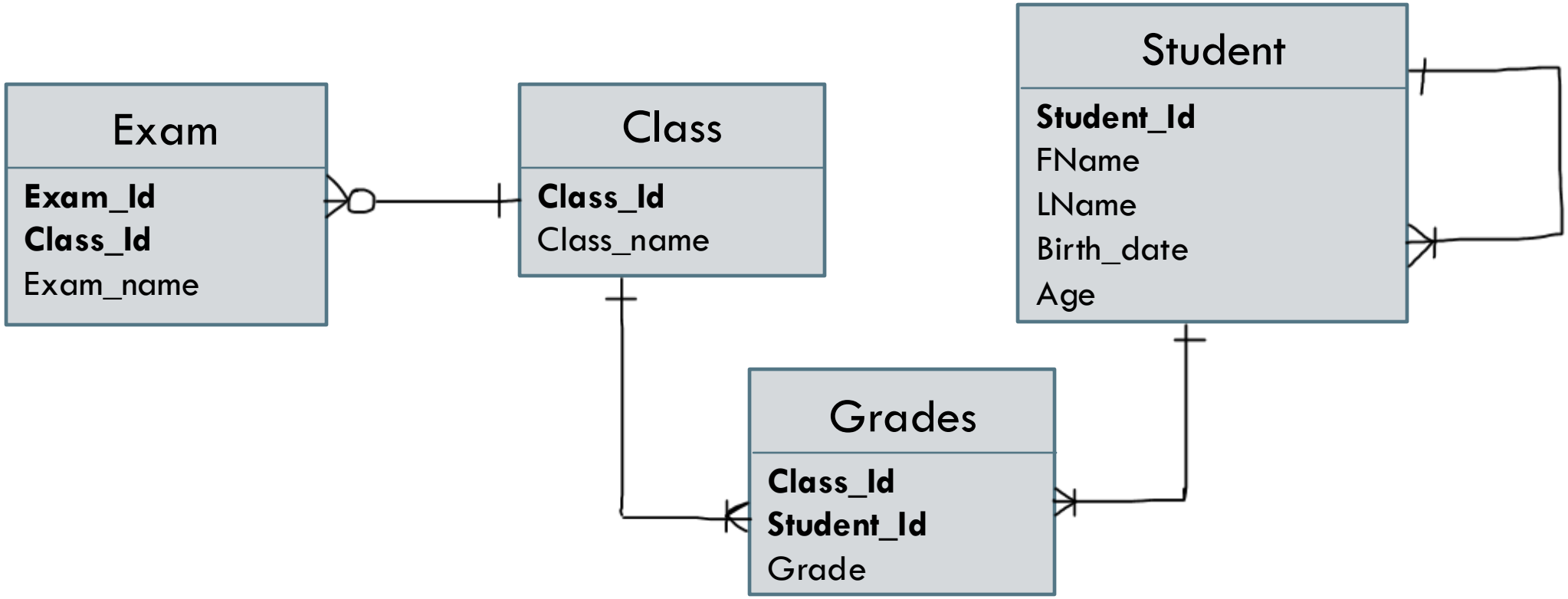
STEP 3B: MAPPING ONE-TO-MANY RELATION

- Include the ONE side's primary key to MANY side's entity table
- Here, Exam is in 1:N relation with Class, so we will include Class_Id in Exam relational table
- Since it is a weak entity, it is already present.

STEP 3B: MAPPING ONE-TO-MANY RELATION

- If there was a student leader for every grade, then how to represent that in ER model?

STEP 3B: MAPPING ONE-TO-MANY RELATION



STEP 3B: MAPPING ONE-TO-MANY RELATION

- There would be a one-to-many relationship within the student table, what key would you add to the student table?

STEP 3B: MAPPING ONE-TO-MANY RELATION

- You would add a leader_Id which would be a student_Id value.

Student	Student_Id	FName	LName	GPA	Clubs	Leader_Id

Class	Class_Id	Class_name

Exam	Exam_Id	Class_Id	Exam_name

STEP 3C: MAPPING MANY-TO-MANY RELATION

- Create a new relation table whose primary key is a combination of both the primary keys of tables.
- Include all relation attributes in this table

STEP 3C: MAPPING MANY-TO-MANY RELATION

- Here, Student table has Student_Id and Class table has Class_Id and both together acting as a primary key of new table, called a Compound Key for Grades table.

Student	Student_Id	FName	LName	GPA	Clubs	Leader_Id

Class	Class_Id	Class_name

Exam	Exam_Id	Class_Id	Exam_name

Grades	Class_Id	Student_Id	Grade

FINAL DATABASE SCHEMA STRUCTURE

Student	Student_Id	FName	LName	GPA	Clubs	Leader_Id

Class	Class_Id	Class_name

Exam	Exam_Id	Class_Id	Exam_name

Grades	Class_Id	Student_Id	Grade

HOW WOULD DATABASE LOOK?

High School Database

Students table

Student_Id	FName	LName	GPA	Clubs	Leader_Id
1	John	Doe	7	Art	4
2	Mark	Smith	8	Art, Dance	4
3	Julia	Robert	8	Dance	4
4	John	Wick	7.5	Music	NULL

Class table

Class_Id	Class_name
1	Math
2	Biology
3	Physics

Exam table

Exam_Id	Class_Id	Exam_name
1	1	Math Final
2	2	Biology Test
3	2	Biology Final

Grades table

Class_Id	Student_Id	Grade
1	1	8
1	2	9
2	2	10
2	3	8

SUMMARY

- ER Model contains, entities, attributes and relationships.
- ER Diagram/Model is the middleman between business requirement and the actual implementation of the database schema.
- Primary Key is the unique attribute to identify every row in the table.
- Foreign key is an attribute that links the table values to another table using their primary key in their own table.

EXECUTING DATABASE SCHEMA STRUCTURE

- So now, we know what all columns a table would have to execute the business requirement
- Next, will learn how to build the database tables using **SQL!**

COMMANDS IN SQL

DDL, DML, DCL AND TCL COMMANDS

TYPES OF COMMANDS

- DDL – Data Definition Language commands
- DML – Data Manipulation Language Commands
- DCL – Data Control Language Commands
- TCL – Transaction Control Language Commands

DDL COMMANDS

- Modifying the structure of a table, such as creating a new table, deleting an existing table, or altering a table.
- auto-committed commands, changes are saved permanently
- CREATE, DROP, ALTER, TRUNCATE, RENAME

DML COMMANDS

- Used to make modifications to the database, handling all types of changes within it.
- Not auto-committed changes
- INSERT, UPDATE, DELETE, SELECT

DCL COMMANDS

- Used to grant and take back authority (revoke) from any database user.
- GRANT, REVOKE

TCL COMMANDS

- Not all transactions will be passed
- They ensure data consistency
- Can only be used with DML commands
- COMMIT, ROLLBACK, SAVE AGAIN

ANY QUESTIONS?

SQL ONLINE COMPILER

- Use this link to access the website for online SQL compiler that we will use in the class, https://sqliteonline.com/
- Will create a new database by clicking on the plus icon, name the database "University".

LET'S CREATE THE TABLE BELOW

Students table

Student_Id	FName	LName	GPA	Leader_Id
1	John	Doe	7	4
2	Mark	Smith	8	4
3	Julia	Robert	8	4
4	John	Wick	7.5	NULL

SQL DATATYPES

- INT – Represents whole numbers
- DECIMAL(M,N) – Like Float, decimal numbers with exact values
- VARCHAR(1) – String with length 1
- BLOB – Binary large Object, images or files
- DATE – Stores date in the format “YYYY-MM-DD”
- TIMESTAMP – Date and time stored as “YYYY-MM-DD HH:MI:SS”

CREATING TABLES

- Before querying any information in the database, we need to build the database
- SQL command CREATE is used, to define attributes, table names and constraints on those columns defined.

```
CREATE TABLE table_name (
        column1 datatype,
        column2 datatype,
        ....
)
```

STUDENT TABLE CREATE COMMAND

- CREATE TABLE – reserved words/ keyword, written in capital usually but not necessary
- students – student table name
- student_id – unique id (primary key)
- fname – student first name
- lname – student last name
- gpa – current GPA of student
- leader_id – student id who is acting as a leader for some number of students

STUDENT TABLE CREATE COMMAND

```
CREATE TABLE student (
    student_id INT PRIMARY KEY,
    fname VARCHAR(25),
    lname VARCHAR(25),
    gpa DECIMAL(4,2),
    leader_id INT
);
```

STUDENT TABLE CREATE COMMAND

```
CREATE TABLE student (
    student_id INT,
    fname VARCHAR(25),
    lname VARCHAR(25),
    gpa DECIMAL(4,2),
    leader_id INT,
    PRIMARY KEY (student_id)
);
```

CONSTRAINTS – SPECIFY RULES FOR DATA

- NOT NULL – the column needs to store a value, CAN'T store NULL values
- PRIMARY KEY – uniquely identifies a record in the table
- AUTOINCREMENT – a unique number is generated when new record entered in table
- UNIQUE – all values in the column are different
- DEFAULT – sets default value for the column
- FOREIGN KEY – column that refers to the primary key in another table, to create links

IMPORTANT NOTE

PRIMARY KEY is already NOT NULL and UNIQUE.

Means all the columns that are defined as PRIMARY KEY, the values stored in them should not be NULL and all of them needs to be unique in value.

INSERTING DATA INTO TABLES

- After your table is created, you can insert data values into the table using INSERT INTO command

```
INSERT INTO table_name VALUES (value1, value2,….);
```

OR

```
INSERT INTO table_name (column1, column2,…) VALUES (value1, value2,…);
```

SELECT

- SELECT Query is used for creating temporary views of the data in the database
- These views aren't stored anywhere
- View contents of a table
- * used to refer to all the columns in the table

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

DROP

- To delete the table and all its content, you use the DROP command

```
DROP TABLE table_name;
```

PRACTICE – CREATE THE TABLE BELOW

Id	Title	Director	Minutes	Year
1	Toy Story	John Lasseter	81	1995
2	Cars	John Lasseter	117	2006
3	Finding Nemo	Andrew Stanton	107	2003
4	Toy Story 2	John Lasseter	93	1999
5	WALL-E	Andrew Stanton	104	2008

Note: Minutes should be 100 by default and Id should be auto incrementing.

WHERE CLAUSE

- WHERE clause is used with SELECT query to be able to filter through data based on any column present in the table

```
SELECT column1, column2,...
FROM table_name WHERE
column1 = value;
```

DISTINCT

- Returns only distinct values, i.e., all different values.

```
SELECT DISTINCT column1 FROM
table_name;
```

ORDER BY

- Sort the result set into ascending or descending order
- By default, it sorts in ascending order
- DESC added after column name to sort in descending order

```
SELECT column1,column2,…
FROM table_name
ORDER BY column1 ASC|DESC;
```

AND OPERATOR

- Used with WHERE clause
- One or more AND operators can be used in one WHERE condition
- Any record which satisfies **all** the conditions.

```
SELECT column1,column2,…
FROM table_name
WHERE column1 = value1 AND
column2 = value2;
```

OR OPERATOR

- Used with WHERE clause
- One or more OR operators can be used in one WHERE condition
- Any record which satisfies **any** of the conditions.

```
SELECT column1,column2,…
FROM table_name
WHERE column1 = value1 OR
column2 = value2;
```

NOT OPERATOR

- Used in combination with other operators
- Gives opposite results
- Can use it with WHERE, BETWEEN, LIKE, IN, >, <

```
SELECT column1,column2,…
FROM table_name
WHERE NOT column1 = value1;
```

GREATER THAN, LESS THAN

- Used in conditions
- Comparing values in the column

```
SELECT column1,column2,...
FROM table_name
WHERE NOT column1 = value1;
```

IN

- Search in the list of values you provide
- Used with WHERE clause

```
SELECT column1,column2,…
FROM table_name
WHERE column1 IN (value1,
value2, value3);
```

BETWEEN

- Selects values within a given range
- Inclusive of the ranges
- Can be used with text, numbers and dates

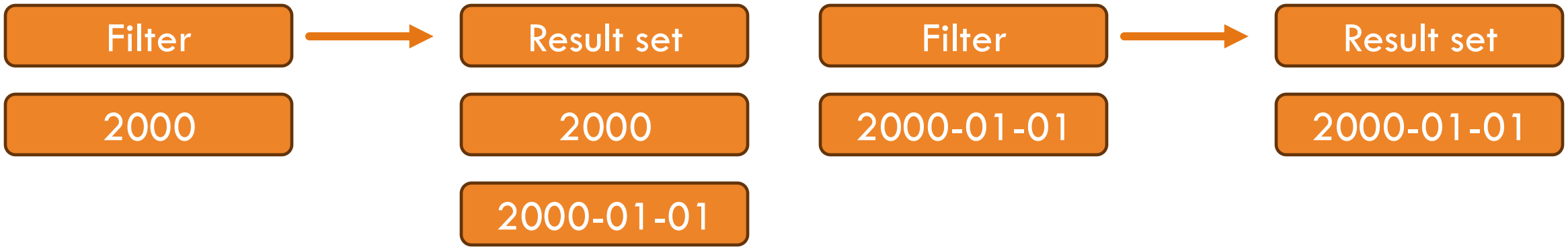
```
SELECT column1,column2,…
FROM table_name
WHERE column1 BETWEEN
value1 AND value2;
```

DATE

- Since we know the format of dates stored in database, we can use it in different ways to filter through the data
- We can only use year, or the entire date
- If you are using only year, use just the numerical value
- If you are using the whole date, then use it as string in the query, i.e., '2000-01-01'

DATE

- Numerical year value can lookup for numerical year and date formats
- Date formats can only look for date formatted value and not numerical years because it has no month and day portion.



LIMIT OFFSET

- The **LIMIT** will reduce the number of rows to return, and the optional **OFFSET** will specify where to begin counting the number rows from.

```
SELECT column1,column2,…
FROM table_name
LIMIT value OFFSET offset_value;
```

QUESTIONS

1. Find the **Title** of each film
2. Find **Director** of each film
3. Find **Title** and **Director** of each film
4. Find the movie with **Id** equal 8
5. Find movies released between **year** 2000 and 2008
6. Find all the Movies with duration more than **100 minutes.**

QUESTIONS

7. Find all the movies released after **date** 9th Jan 2018.
8. Find the list of **different directors** for movies released after **year** 2004.
9. Find the list of movies **directed** by Brad Bird, Pete Docter and Andrew Stanton.
10. Find all the **Directors** who have made movies of **duration** more than 1.5 hours.
11. Sort the movies by **Minutes** from shortest to longest movies.
12. View the last 5 **movies** in the table.

QUESTIONS

13. Find all the **movies** directed by Pete Doctor and Brad Bird.
14. Find all the movies with **id** more than 6, **directed** by Dan Scanlon and duration being more than 95 **minutes**.
15. View all the movies released after **year** 2006 but in reverse alphabetical order.

ORDER OF EXECUTION

SELECT DISTINCT column

FROM table_name

WHERE constraint_expression

ORDER BY column_name ASC/DESC

LIMIT count1 OFFSET count2;

Note: Constraint expression would contain and, or, between and other conditional statements.