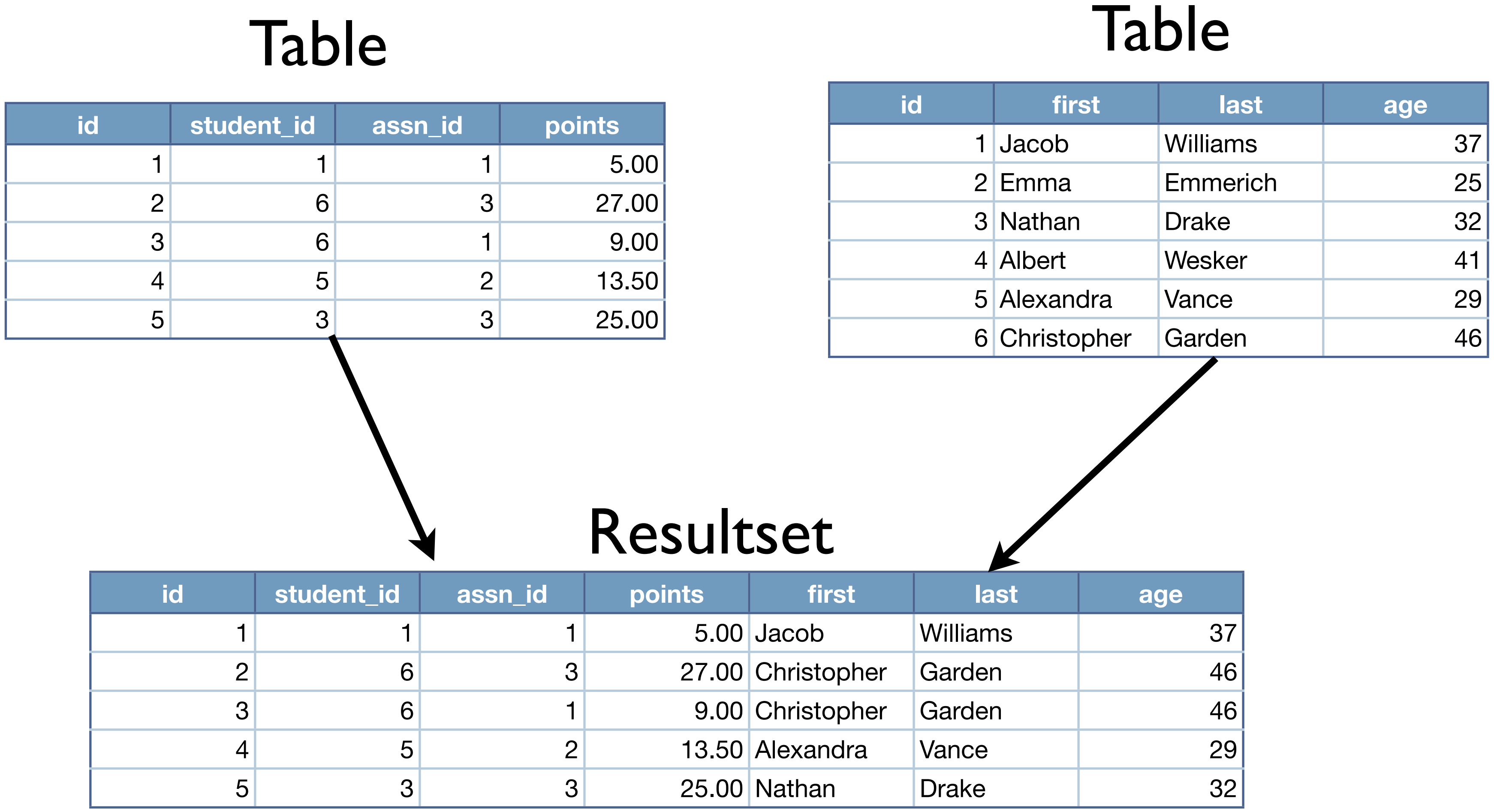
SQL Unit 6 Views and Temporary Tables

Ryan Nixon

Warning

- So far we have only been working with SELECT statements which do not modify the database
- This changes with the introduction of Views. Views are stored in the database alongside the other tables
- Temporary tables will also stick around, to an extent

A Naming Reminder



Table

id	student_id	assn_id	points
1	1	1	5.00
2	6	3	27.00
3	6	1	9.00
4	5	2	13.50
5	3	3	25.00

Table

id	first	last	age
1	Jacob	Williams	37
2	Emma	Emmerich	25
3	Nathan	Drake	32
4	Albert	Wesker	41
5	Alexandra	Vance	29
6	Christopher	Garden	46

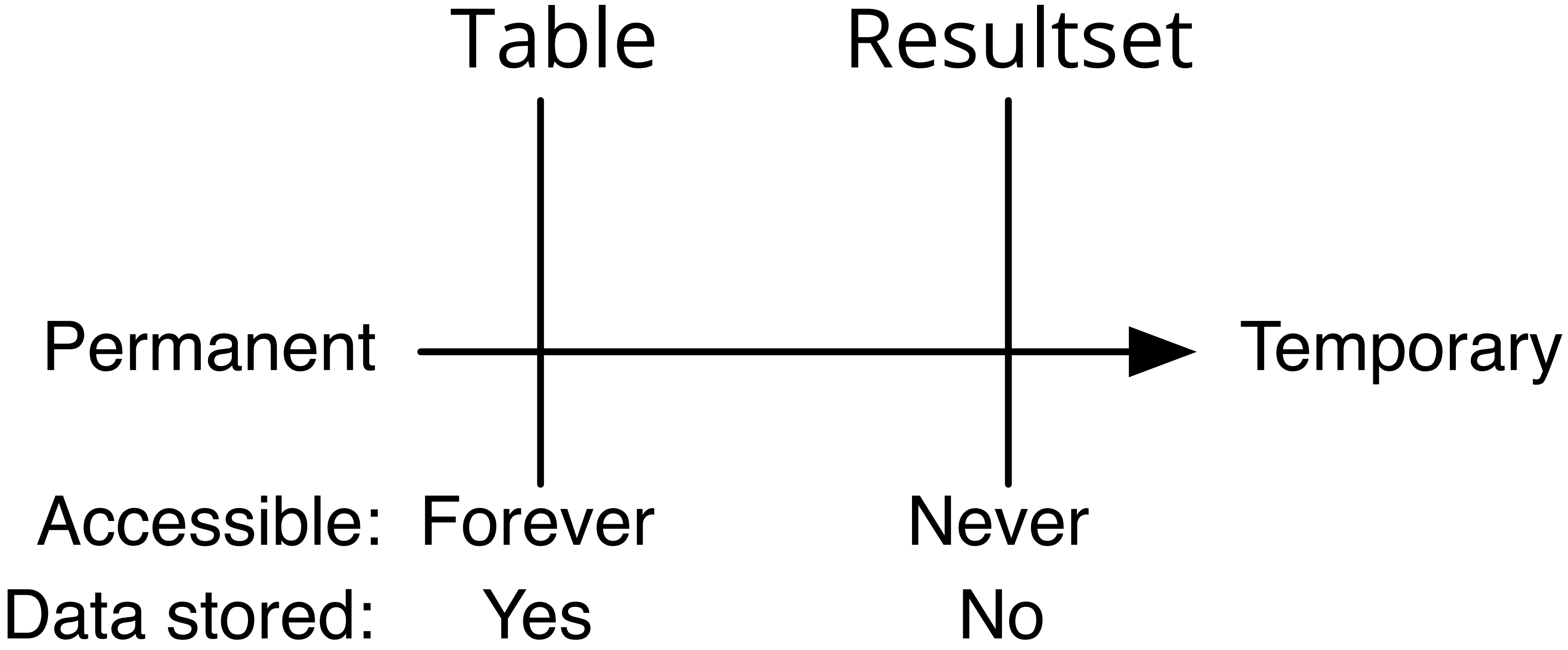
Resultset

id	student_id	assn_id	points	first	last	age
1	1	1	5.00	Jacob	Williams	37
2	6	3	27.00	Christopher	Garden	46
3	6	1	9.00	Christopher	Garden	46
4	5	2	13.50	Alexandra	Vance	29
5	3	3	25.00	Nathan	Drake	32

Tables vs Resultsets

- Tables are "persistent". They are a permanent part of the database and will not be deleted unless specific action is taken
- Resultsets are just the data collected from the tables and go away the moment the query completes
- But what if we want to keep this data around longer?

The Persistence Scale



Table

Resultset

Permanent → Temporary

Accessible: Forever Never

Data stored: Yes No

Persistence

- Some of our queries have gotten very complex or have imposed limitations on what we can do
- GROUP BY, for example, causes trouble when you want to add many fields to your SELECT statement
- Temporary tables & views provide a method of splitting these queries into multiple steps

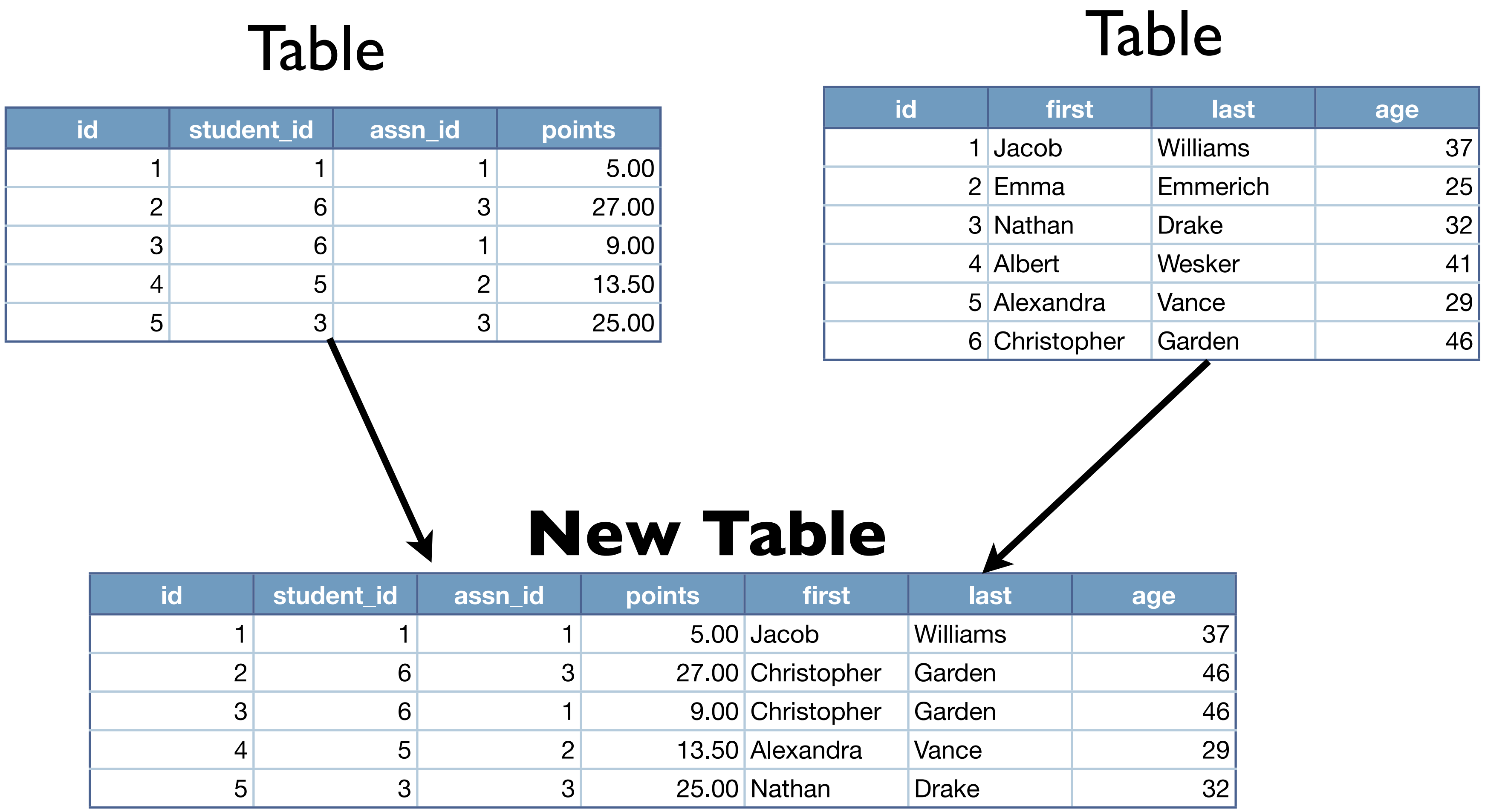
Temporary Tables

- Temporary tables live at the halfway point between tables and resultsets
- They are persistent...but only for the duration of the connection
- In the context of this class, connection refers to your open window looking at the database. Close the window and the connection is closed

Temporary Tables

- Temporary tables are stored in another location on the same server (the machine running the database). They won't be visible from the tables list
- These tables are created from a resultset. Because of this, changing the data in a temporary table will not affect the table that it originated from

Temporary Tables



Table

id	student_id	assn_id	points
1	1	1	5.00
2	6	3	27.00
3	6	1	9.00
4	5	2	13.50
5	3	3	25.00

Table

id	first	last	age
1	Jacob	Williams	37
2	Emma	Emmerich	25
3	Nathan	Drake	32
4	Albert	Wesker	41
5	Alexandra	Vance	29
6	Christopher	Garden	46

New Table

id	student_id	assn_id	points	first	last	age
1	1	1	5.00	Jacob	Williams	37
2	6	3	27.00	Christopher	Garden	46
3	6	1	9.00	Christopher	Garden	46
4	5	2	13.50	Alexandra	Vance	29
5	3	3	25.00	Nathan	Drake	32

Temporary Tables

- Most SQL:

  -
    ```
    CREATE TEMPORARY TABLE tbl AS
      SELECT column1, column2
      FROM table
    ```
- Microsoft SQL Server:

  -
    ```
    SELECT column1, column2
    INTO #tbl
    FROM table
    ```

Temporary Tables

- The results of the SELECT will be inserted into the new table with the exact column names and row order as the query
- Once created, you may run queries against the temporary table exactly like you would a normal table

An Example

```
CREATE TEMPORARY TABLE last_name AS
  SELECT last, COUNT(id) AS count
  FROM person
  GROUP BY last
  ORDER BY last DESC;

SELECT *
FROM last_name
```

An Example (MS SQL)

```
SELECT last, COUNT(id)
INTO #last_name
FROM person
GROUP BY last
ORDER BY last DESC;

SELECT *
FROM #last_name
```

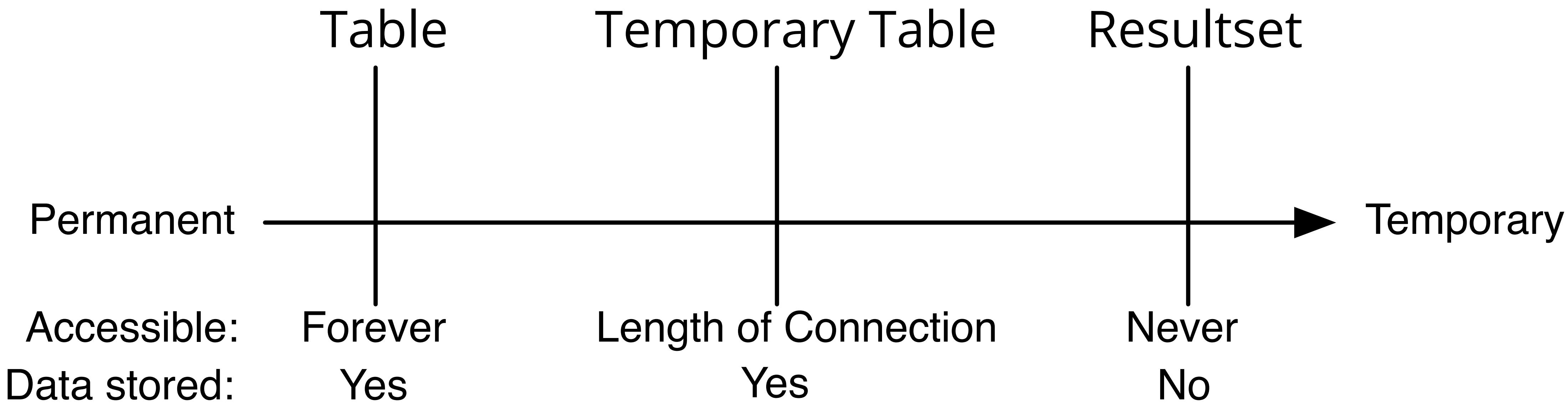
Remember the Semicolon?

- The two examples above were run in a transaction. This allowed both queries (separated by semicolons) to be run as a group and still return the last query's resultset
- Unfortunately SQLite does not play well with transactions that involve creating tables & views. The second query would not return a resultset

Potential Uses

- Reducing long or complex queries to a series of smaller queries, especially ones involving GROUP BY
- Joining two or more tables into one temporary table before querying for information
- Reusing the same resultset over the course of many queries

The Persistence Scale



Permanent → Temporary

	Table	Temporary Table	Resultset
Accessible:	Forever	Length of Connection	Never
Data stored:	Yes	Yes	No

A Couple of Database Examples

Back to Persistence

- Temporary tables are great, but the moment you disconnect they are removed
- That works well for a short while, but what if we wanted the data around for even longer?

Views

- Views solve the persistence issue, placed between temporary tables and the actual permanent tables
- They are like tables in that they are permanent, but also like resultsets in that their data is temporary

Views

- If a temporary table is like a "saved resultset", then a view is like a "saved query"
- Each time you use a view, the database will run that view like a query and return it as a resultset
- In this way, a view is no different than a query except that it can be rerun without typing it in again

Views

```
• CREATE VIEW vw AS
        SELECT column1, column2
        FROM table
```

Views

- After creating the view, it will appear in the database with or near the table list and will stay there until dropped
- You may then use the view just as you would a temporary table or normal table
- Each time you run the query it will pull the data from the original table(s) again

Views

```
CREATE VIEW last_name_v AS
  SELECT last, COUNT(id)
  FROM person
  GROUP BY last
  ORDER BY last DESC;

SELECT *
FROM last_name_v
```

Views

- Very similar to temporary tables, right?
- Careful not to confuse the two, their levels of persistence are very different
- Best practice: When naming views, append '_v' to the end. This will make absolutely sure that the next person to use the database knows it is a view

Views

- Views also keep track of the tables that their fields come from
- Down the road, we will see how to run UPDATE queries on data
- These queries may be applied to views just as if they were real tables. The UPDATE will map back to the original table & row when changing the data

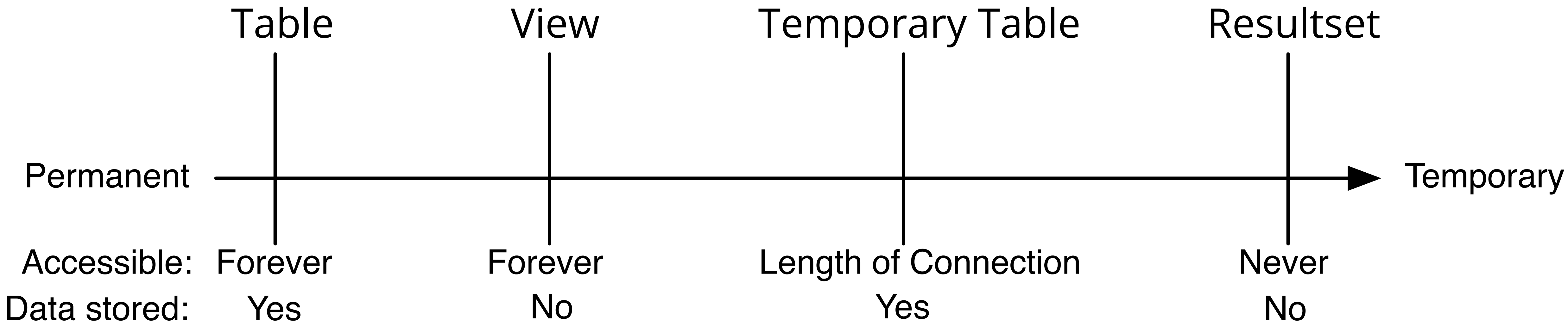
Another View of Views

- Views do not actually need to be defined to be useful. There is a concept called an “inline view” that allows the view to be defined and executed in a single query
- These inline views are also called “subqueries”. They will be covered in the next unit

Potential Uses

- Pre-filtering and processing results based off of specific conditions. For example:
  - How many times have I asked "How many students are registered for a course?" Set up a view and you'll never need to write that query again
- Limiting displayed data for security or organization purposes. For example:
  - A view, "person_public_v" that removes all SSN & ADL information from a person record

The Persistence Scale



Permanent → Temporary

	Table	View	Temporary Table	Resultset
Accessible:	Forever	Forever	Length of Connection	Never
Data stored:	Yes	No	Yes	No

A Couple More Database Examples

A Persistence Caveat

- Temporary tables and views, once defined, cannot be created again unless they are first removed
- As shown in the example, if you do not remove the item before running the CREATE query again you will get an error
- To recreate the table or view you need to first "drop" it

Dropping Tables & Views

- Removing items is easy!
- DROP VIEW for views
- DROP TABLE for tables
- These will remove the named table or view if it exists. If it doesn't it will display an error saying it can't find the table to remove

Dropping Tables & Views

```
CREATE VIEW last_name_v AS
  SELECT last, COUNT(id)
  FROM person
  GROUP BY last
  ORDER BY last DESC;

SELECT *
FROM last_name_v;

DROP VIEW last_name_v
```

Dropping Tables & Views

```
CREATE TEMPORARY TABLE last_name AS
  SELECT last, COUNT(id)
  FROM person
  GROUP BY last
  ORDER BY last DESC;

SELECT *
FROM last_name;

DROP TABLE last_name
```

Dropping Tables & Views

- Be careful, DROP TABLE works for all tables. You can remove your original source tables if you use the wrong name
- In MySQL and SQLite you may also use IF EXISTS with your drop to skip the errors
- `DROP TABLE IF EXISTS last_name`
- This does not have an easy equivalent in MS SQL and so will not be included in the course

The Microsoft Conundrum

- If you’re curious, the alternative would be:

  ```
  IF EXISTS (SELECT * FROM sys.objects WHERE name =
  'last_name' AND type = 'u')
       DROP TABLE last_name
  ```

- Or in some cases:

  ```
  IF EXISTS(SELECT * FROM last_name)
       DROP TABLE last_name
  ```

The Last Couple Database Examples

Warning

- Remember the aforementioned persistence issues when working on your homework
- If you create a temporary table or view, you must remove it before creating it again
- If you drop a table that is a critical part of the database, you will have to re-download the database

Reminders

- Next Monday is the test. Expect it to contain questions based off of the assignments such as:
    - Queries involving any of the functions on the reference sheet which may not be used
    - Complex queries that require Joins, Grouping and filtering with HAVING
    - The creation and use of a temporary table and/or view over multiple questions

Reminders

- Assignment 6 up tonight. Due 10/22
- Lab time Wednesday in SSB 172
- Extra lab time Sunday 2pm-4pm in SSB 172
- Test Monday, 10/22
    - Note: A single 3”x”5 index card will be allowed for this test