O'REILLY®

Mastering the Basics of Relational SQL Querying

Welcome to O`reilly Live Online Training

- 6 Hours of training over 2 days
- Prerequisites
  - Familiar with IT terminology
  - Some programming background
- For beginners, and experienced SQL developers alike

Goals

- Glimpse of Background - what are SQL and RDM?
- Understand SQL query processing phases
- Get into a ‘set mindset’
- Write basic SQL queries, and fully understand them
- Solid foundation, less focus on detail and syntax
- Passion for SQL

Day 1 Agenda

- What is SQL?
- SQL, RDM, and RDBMS
- Development environment
- SQL language constructs
- The data query language (DQL)
- SELECT query clauses and Query logical processing
- FROM and JOIN

Day 2 Agenda

- WHERE filtering
- GROUP BY and HAVING
- ORDER BY and LIMIT / FETCH
- Subqueries
- Set operators
- Conclusions

Administrative

- Hours
- Breaks
- Hands-on exercises
- Safari platform
  - Questions
  - Help
- Course evaluation

Questions

?

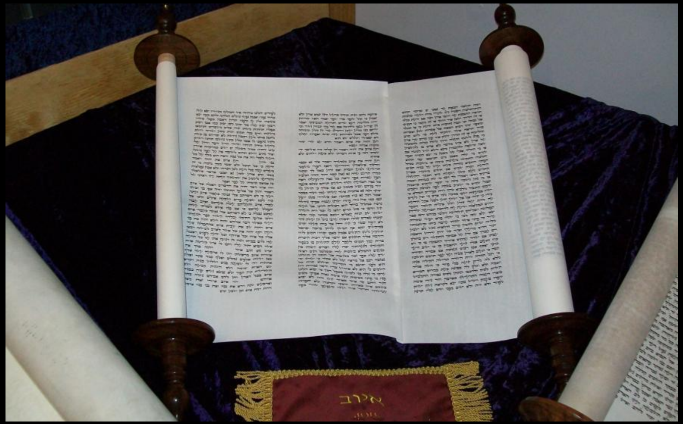
Next

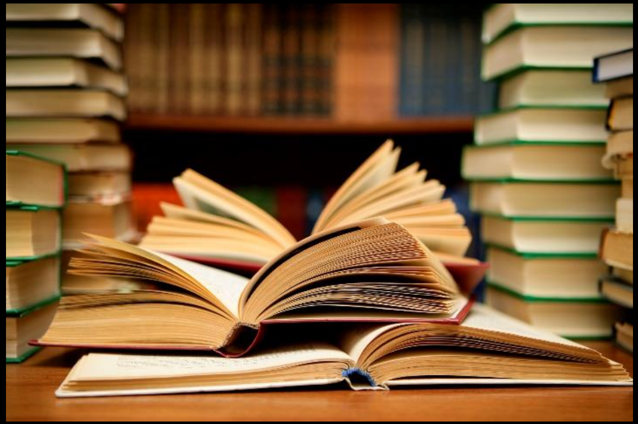
- What is SQL?

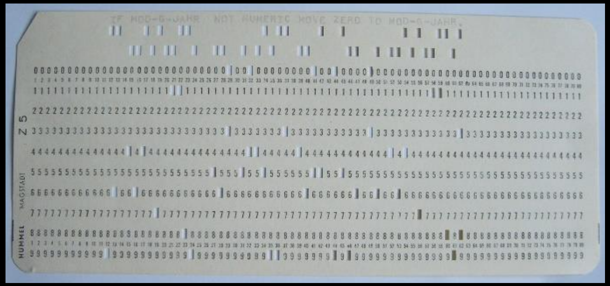
Mastering Relational SQL Querying

What is SQL?

A brief history of data storage







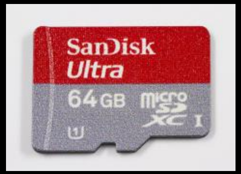


Photo attribution:
https://commons.wikimedia.org/w/index.php?curid=18997280
https://commons.wikimedia.org/w/index.php?curid=57651005
https://commons.wikimedia.org/w/index.php?curid=7799579

Data access in the “good old days”

- Data was stored and accessed based on physical order
  - ➢ Books had page numbers
  - ➢ File cabinet drawers had index labels
  - ➢ Punched cards had a physical order
  - ➢ Magnetic tapes had physical address pointers

Unfortunately, most still see it that way…


NO
CORRECTION FLUID

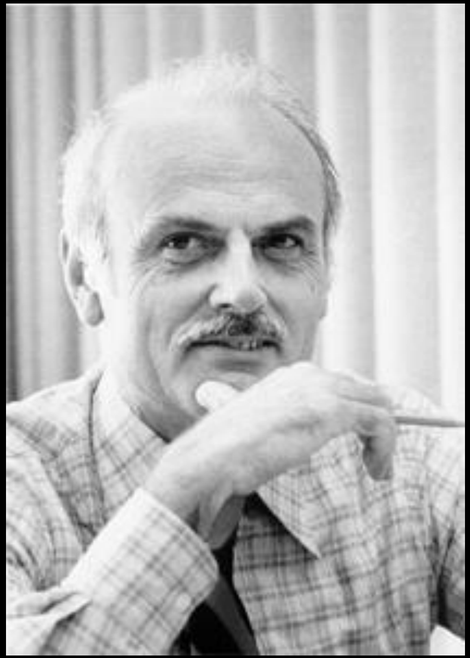
Flat Files

- CSV: Row Number, Customer, Date, Price, Item…

```
1, Ami , 1/1/1965, $3,  apple
2, John, 1/2/1965, $4,  sock
3, Ami , 1/3/1965, $10, pencil
4, Ed  , 2/3/1965, $1,  gum
```

- Challenges
  - Sequential access / ISAM
  - Data consistency
  - Update, Delete, and Insertion anomalies

Dr. Edgar F. “Ted” Codd



- Born 1923 in England
- Studied Mathematics and chemistry at Oxford
- RAF Coastal Command pilot in WW2
- Moved to the USA in 1948
- 1953 - 1957, moved to Canada, angered by Sen. McCarthy
- In 1965 received doctorate in computer science
- 1967 Moved to IBM Research in San Jose, CA
- Received Turing award in 1981
- Died in 2003 from heart failure

“A Relational Model of Data for Large Shared Data Banks”

- 1969 internal at IBM, 1970 public
- Codd’s relational Alpha sub-language
- System R used non-relational SEQUEL
  - Developed by Chamberlin and Boyce
- SEQUEL was renamed to SQL
- Competitors were early to adopt

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the “connection trap”).

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. Data Dependencies in Present Systems

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

Technology Survival

The RDM and SQL survived unscathed for nearly 5 decades…

Questions

?

Next

- SQL, RDM, and RDBMS

Mastering Relational SQL Querying

SQL, RDM, and RDBMS

The Relational Model

Declarative method
for specifying data and queries

- Set Theory
- First Order Predicate Logic (FOPL)
- Relational Algebra
- Tuple Relational Calculus

Relations and Tuples

- A tuple is a finite ordered list (sequence) of elements
- A relation is a set of tuples, where each set of corresponding elements (keys / attributes), are of a data domain

Order ID	Customer	Date	Price	Item
1	Ami	1/1/1965	$3	Apple
2	John	1/2/1965	$4	Sock
3	Ami	1/3/1965	$10	Pencil
4	Ed	2/3/1965	$1	Gum

Sets in the Relational Model

- Unique tuples
- Uniquely referenceable attributes
- No tuple or attribute order
- Atomic values
- “All at once” operations

SQL Tables

An engineering approximation to the relational model

- A SQL table (definition) corresponds to the predicate variable
- A single SQL row corresponds to a tuple
- A set of SQL table rows correspond to a relation
- Constraints and queries correspond to predicates

Orders

Order ID	Customer	Date	Price	Item
1	Ami	1/1/1965	$3	Apple
2	John	1/2/1965	$4	Sock
3	Ami	1/3/1965	$10	Pencil
4	Ed	2/3/1965	$1	Gum

SQL != Relational

SQL is not truly relational, but close enough

- Duplicate rows
- Column order significance
- Unnamed columns
- Column-less tables
- NULL
- Physical Data Independence (PDI)

SQL Tables

- Constraints
- Candidate and Primary Keys
- Declarative Referential Integrity (DRI)

Orders

Order ID	Customer	Date	Price	Item
1	Ami	1/1/1965	$3	Apple
2	John	1/2/1965	$4	Sock
3	Ami	1/3/1965	$10	Pencil
4	Ed	2/3/1965	$1	Gum

Customers

Customer
Ami
John
Ed

Normalization - An oft-abused term

- Normal forms are out of scope for this training
- Normalization Rules' goals:
  - *To free the collection of relations from undesirable insertion, update and deletion dependencies;*
  - *To reduce the need for restructuring the collection of relations, as new types of data are introduced*
  - *To make the relational model more informative to users;*
  - *To make the collection of relations neutral to the query statistics*

A Normal Model

Customers

Customer	Country
Ami	USA
John	USA
Ed	Canada
Tim	NULL

Items

Item
Apple
Sock
Pencil
Gum

Orders

Order#	Customer	Order Date
1	Ami	1/1/1965
2	John	1/2/1965
3	Ami	1/3/1965
4	Ed	2/3/1965

Order Items

Order#	Item	Qty	Price
1	Apple	1	$3
2	Sock	1	$4
3	Pencil	1	$10
4	Gum	1	$1

Relational Database Management Systems

ANSI/ISO SQL Standard

- First ANSI standard published in 1986 (SQL:86)
- Latest (9th generation) published in 2016 (SQL:2016)
  - Adds JSON, pattern recognition, polymorphic table functions
- SQL conformance levels: Entry, Intermediate, and Full
- Every vendor uses custom ‘extensions’ to the standard to form SQL ‘dialects’
  - T-SQL, PL/SQL, PL/pgSQL, SPL, SQL PL etc.

Questions

?

Next

- Development Environment

Mastering Relational SQL Querying

Development Environment

Free and Open Source

Platform	Tools	O/S
SQLite	DB Browser	
MySQL Community	MySQL Workbench	
PostgreSQL	pgAdmin	
SQL Server Express	Azure Data Studio SSMS	
Oracle Express	Oracle Developer	
Online resources	Browser	

Our Demo Database

Customers

Customer	Country
Ami	USA
John	USA
Ed	Canada
Tim	NULL

Items

Item
Pencil
Pen
Marker
Notebook
Ruler

Orders

Order#	Customer	Order Date
1	Ami	1/1/1965
2	John	1/2/1965
3	Ami	1/3/1965
4	Ed	2/3/1965

Order Items

Order#	Item	Qty	Price
1	Apple	1	$3
2	Sock	1	$4
3	Pencil	1	$10
4	Gum	1	$1

SQLite

- Download and install SQLite DB Browser
  - http://sqlitebrowser.org/
- Download demo DB from [TBD]
- Launch DB Browser
- File -> Open Database -> [downloaded file]
- In ‘Execute SQL’, type SELECT ‘Hello World’ and execute

Next

- Hands-on exercise: Install or configure development environment
- Break (10 minutes)
- Q&A
- SQL Language Constructs

Mastering Relational SQL Querying

SQL Language Constructs

SQL Language Constructs

- DDL - Data Definition Language
- DCL - Data Control Language
- DML - Data Manipulation Language
- DQL – Data Query Language
- Vendor specific extensions
- Imperative constructs

Data Definition Language

Create and modify objects

- CREATE
- ALTER
- DROP

CREATE TABLE

```
CREATE TABLE Customers
(
Customer        VARCHAR(20) NOT NULL PRIMARY KEY,
Country         VARCHAR(20) NULL
);
```

ALTER TABLE

```
ALTER TABLE Customers
ADD StateCode CHAR(2) NULL;

ALTER TABLE Users
ADD CONSTRAINT CheckEmailFormat
CHECK (Email LIKE '%@%’); *

ALTER TABLE Customers DROP StateCode *
```

** Not supported in SQLite*

DROP TABLE

DROP TABLE Users

DROP TABLE Users, Orders

Data Control Language

Manage access to objects

- GRANT
- REVOKE
- *DENY (SQL Server)*

GRANT

GRANT SELECT ON Users TO Ami

GRANT DELETE ON Orders TO Guest

GRANT ALL ON Customers TO Administrator

REVOKE

REVOKE SELECT ON Users FROM Ami

REVOKE DELETE ON Orders FROM Guest

REVOKE ALL ON CreditCards FROM Administrators

Data Manipulation Language

Add, modify, and delete data

- INSERT INTO
- UPDATE
- DELETE FROM

Data Query Language

- SELECT
- *OUTPUT / RETURNING*

Vendor Extensions

- Additional / enhanced SQL functionality
- Additional Languages
- Data formats
- Controlling session and default behavior
- Flow control, variables, performance objects
- Server administration

Questions

?

Next

- Data Query Language

Mastering Relational SQL Querying

Data Query & Manipulation Language

SELECT

SELECT ‘Hello World’;

SELECT * FROM Customers;

SELECT Customer, Country FROM Customers;

INSERT INTO

```
INSERT INTO Customers (Customer, Country)
VALUES ('Dave', 'USA'), ('John', 'USA');

INSERT INTO CustomersBackup (Customer, Country)
SELECT * FROM Customers
```

CREATE TABLE AS SELECT

DDL + DML in one statement

```
CREATE TABLE CustomersBackup AS
SELECT Customer, Country
FROM Customers
```

UPDATE

UPDATE Customers
SET Country = ‘Costa Rica’
WHERE Customer = ‘Dave’

DELETE FROM

```
DELETE FROM Customers
WHERE Customer = ‘Dave’;
```

Questions

?

Next

- Hands-on exercise: INSERT, UPDATE, DELETE and RETRIEVE data
- SELECT query logical processing

Mastering Relational SQL Querying

SELECT Query Logical Processing

Query Logical Processing

6

5 SELECT [DISTINCT] *Customer, COUNT(*) AS NumOrders*

1 FROM *Orders*

2 WHERE OrderDate > ‘20180101’

3 GROUP BY Customer

4 HAVING COUNT(*) > 1

7 ORDER BY *NumOrders DESC*

8 *LIMIT 10 OFFSET 0*

Orders

Order ID	Customer	Order Date
4	John	9/1/2018
2		2/1/2018

FROM Clause

- Defines the source set for the query
- May be a single table, multiple tables, or nested queries that produce ***a valid set***

WHERE Clause

Predicate filters eliminate rows from the source set

```
SELECT    Customer AS Client,
          Country AS CountryOfOrigin
FROM      Customers
WHERE     Country = 'USA'
```

GROUP BY Clause

Combine rows from the filtered source set into groups based on the grouping expression(s) value

Grouping expressions will determine the shape of the row, and the details that we 'lose'

```
SELECT    Country, COUNT(*)
FROM      Customers
GROUP BY Country
```

HAVING Clause

- Uses predicate filters to eliminate whole groups
- Can use aggregate functions on ‘lost detail’ columns

```
SELECT    Country, COUNT(*)
FROM      Customers
GROUP BY Country
HAVING COUNT(*) > 2
```

SELECT List

- Aliasing and removing duplicate rows

```
SELECT    ALL | DISTINCT
          Customer AS Client,
          Country AS CountryOfOrigin
FROM      Customers
```

“All at Once” Principal

```
SELECT   Customer AS Client,
         ~~Client + ‘ From ‘ + Country AS  ExtendedName~~
FROM     Customers

UPDATE   Table
SET      Column1 = Column2, Column2 = Column1
```

ORDER BY Clause

Guarantees row order for presentation purposes
Not necessarily deterministic…

```
SELECT    Customer AS Client,
          Country AS CountryOfOrigin
FROM      Customers
ORDER BY Country
```

LIMIT / OFFSET

Limits the number of rows returned from the SELECT list, based on the ORDER BY clause

No support in SQL yet for a presentation order other than the LIMIT order…

A full SQL Statement

```
SELECT      Country, COUNT(*) AS NumCustomers
FROM        Customers
WHERE       Country <> ‘USA’
GROUP BY    Country
HAVING      COUNT(*) > 2
ORDER BY    COUNT(*) DESC
LIMIT 10    OFFSET 0
```

Questions

?

Next

- Break (10 minutes)
- Q&A
- The FROM clause and JOINs

O'REILLY

Mastering Relational SQL Querying

FROM and JOIN

ANSI 89 “Old” Join Syntax

FROM *<Table Source 1>,*

<Table Source 2>,

<Table Source 3>

WHERE *<Join Predicate 1> AND <Join Predicate 2>…*

Recommended Join Syntax

<Table Source 1>

[CROSS / INNER / [LEFT/RIGHT/FULL] OUTER] JOIN

<Table Source 2>

ON/USING <Join Predicate 1>

[CROSS / INNER / [LEFT/RIGHT/FULL] OUTER] JOIN

<Table Source 3> …

Join Processing

Phase 1: Cartesian product

Phase 2: Matching rows ON/USING join condition

Phase 3: Add rows from reserved (outer) table

Join Processing – Cartesian Product

Customers **JOIN** **Orders**

Customer	Country
Dave	USA
John	USA
Gerald	Canada
Jose	Peru

Order ID	Customer	Order Date
1	Dave	1/1/2018
2	John	2/1/2018
3	Gerald	3/1/2018
4	John	9/1/2018

Customer	Order ID	Customer	Order Date
Dave	1	Dave	1/1/2018
Dave	2	John	2/1/2018
Dave	3	Gerald	3/1/2018
Dave	4	John	9/1/2018
John	1	Dave	1/1/2018
John	2	John	2/1/2018
John	3	Gerald	3/1/2018
John	4	John	9/1/2018
Gerald	1	Dave	1/1/2018
Gerald	2	John	2/1/2018
Gerald	3	Gerald	3/1/2018
Gerald	4	John	9/1/2018
Jose	1	Dave	1/1/2018
Jose	2	John	2/1/2018
Jose	3	Gerald	3/1/2018
Jose	4	John	9/1/2018

Join Processing – Matching Rows

Customer	Country	Order ID	Customer	Order Date
Dave	USA	1	Dave	1/1/2018
~~Dave~~	~~USA~~	~~2~~	~~John~~	~~2/1/2018~~
~~Dave~~	~~USA~~	~~3~~	~~Gerald~~	~~3/1/2018~~
~~Dave~~	~~USA~~	~~4~~	~~John~~	~~9/1/2018~~
~~John~~	~~USA~~	~~1~~	~~Dave~~	~~1/1/2018~~
John	USA	2	John	2/1/2018
~~John~~	~~USA~~	~~3~~	~~Gerald~~	~~3/1/2018~~
John	USA	4	John	9/1/2018
~~Gerald~~	~~Canada~~	~~1~~	~~Dave~~	~~1/1/2018~~
~~Gerald~~	~~Canada~~	~~2~~	~~John~~	~~2/1/2018~~
Gerald	Canada	3	Gerald	3/1/2018
~~Gerald~~	~~Canada~~	~~4~~	~~John~~	~~9/1/2018~~
~~Jose~~	~~Peru~~	~~1~~	~~Dave~~	~~1/1/2018~~
~~Jose~~	~~Peru~~	~~2~~	~~John~~	~~2/1/2018~~
~~Jose~~	~~Peru~~	~~3~~	~~Gerald~~	~~3/1/2018~~
~~Jose~~	~~Peru~~	~~4~~	~~John~~	~~9/1/2018~~

Customers INNER JOIN Orders
ON
Customers.Customer = Orders.Customer

Join Processing – Reserved Tables

Customer	Country	Order ID	Customer	Order Date
Dave	USA	1	Dave	1/1/2018
John	USA	2	John	2/1/2018
John	USA	4	John	9/1/2018
Gerald	Canada	3	Gerald	3/1/2018
Jose	Peru	NULL	NULL	NULL
Tim	NULL	NULL	NULL	NULL

Customers LEFT OUTER JOIN Orders
ON Customers.Customer = Orders.Customer

Customer	Country	Order ID	Customer	Order Date
NULL	NULL	1	Dave	1/1/2018
John	USA	2	John	2/1/2018
John	USA	4	John	9/1/2018
Gerald	Canada	3	Gerald	3/1/2018
Jose	Peru	NULL	NULL	NULL
Tim	NULL	NULL	NULL	NULL
Dave	USA	NULL	NULL	NULL

Customers FULL OUTER JOIN Orders
ON Customers.Customer = Orders.Customer
AND OrderDate > '20180101' *

** Not supported by SQLite*

Join Order

A
JOIN B ON …
JOIN C ON …
JOIN D ON …

(
(
(A JOIN B ON …)
JOIN C ON …)
JOIN D ON …)

Questions

?

Next

- Hands-on exercise: Using JOIN
- Q&A
- See you tomorrow morning…

Mastering Relational SQL Querying

Filtering with WHERE

NULL - A very controversial concept…

- Codd introduced the concept to support missing and inapplicable data points first in 1975 and argued for having 2 distinct types of NULLs (4VL)
- NULL is a state, not a value!

WHERE Clause

- SQL implements ternary Logic
  - 3 Value Logic – TRUE, FALSE, UNKNOWN
- WHERE filters out rows for which the predicate ***does not evaluate to*** *TRUE*
- NULL comparison is always UNKNOWN
  - IS NULL is a state predicate, not a comparison

IN predicates

```
SELECT *
FROM Customers
WHERE Customer IN ('Dave', 'John');

SELECT *
FROM Customers
WHERE Customer IN (SELECT Customer FROM Orders);
```

Questions

?

Next

- Hands-on exercise: Filtering Rows
- GROUP BY and HAVING

Mastering Relational SQL Querying

GROUP BY and HAVING

GROUP BY

GROUP BY Customer

Orders

Order ID	Customer	Order Date
1	Dave	1/1/2018
2	John	2/1/2018
3	Gerald	3/1/2018
4	John	9/1/2018

Orders

Order ID	Customer	Order Date
1	Dave	1/1/2018
3	Gerald	3/1/2018
2 4	John	2/1/2018 9/1/2018

GROUP BY

- Items in following phases can only refer to:
  - The GROUP BY Columns
  - Aggregate function on any other column
- SQLite allows ‘bare columns’, a mortal sin
- Can group by multiple columns
- Can group by expressions based on these columns

Aggregate Functions

- MIN(), MAX(), AVG()
- SUM()
- COUNT
  - COUNT (*)
  - COUNT (Expression)
  - COUNT (DISTINCT Expression)
- NULL Handling

HAVING

HAVING COUNT(*) > 1

Orders

Order ID	Customer	Order Date
1	Dave	1/1/2018
2	John	2/1/2018
3	Gerald	3/1/2018
4	John	9/1/2018

Orders

Order ID	Customer	Order Date
1	Dave	1/1/2018
3	Gerald	3/1/2018
2 4	John John	2/1/2018 9/1/2018

Questions

?

Next

- Hands-on exercise: Grouping data
- Break (10 minutes)
- ORDER BY and LIMIT / OFFSET

Mastering Relational SQL Querying

ORDER BY and LIMIT

ORDER BY

- Used for presentation order
- Performed after SELECT list is evaluated
  - Therefor, can use aliases
- Returns a CURSOR; no longer a set

LIMIT / OFFSET

- Typically used for pagination
- Performed after ORDER BY
- Filters out rows based on order

Questions

?

Next

- Hands-on exercise: Order and page data
- Q&A
- Sub queries

O'REILLY

Mastering Relational SQL Querying

Subqueries

Expression Subquery

- Allowed instead of an expression
- Must return a set of one row, one column
  - Some RDBMS provide “discounts” – beware!

```
SELECT    Item, Quantity,
          (SELECT MAX(Quantity) FROM OrderItems)
FROM      OrderItems AS OI
```

Derived Tables and IN Lists

Can be used as table source in FROM clause

- Join aggregates to source rows

A list for IN predicates

- Returns one column, except for row value constructors

Correlated Subqueries

- Not self-contained
- Typically used for expression sub queries, IN, and EXISTS
- Subquery references a column from the outer query

Questions

?

Next

- Hands-on exercise: Sub queries
- Break
- Set Operators

O'REILLY

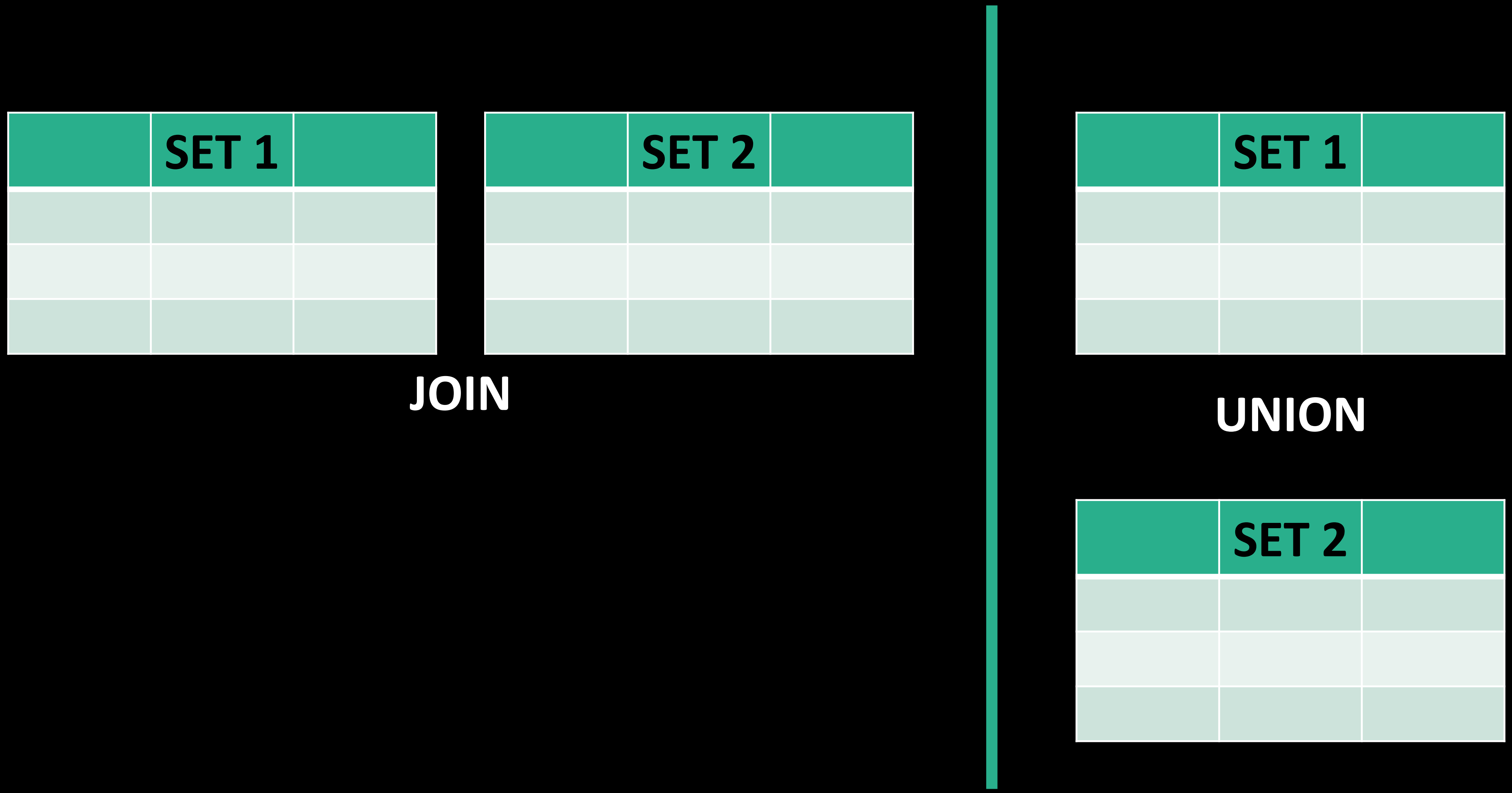
Mastering Relational SQL Querying

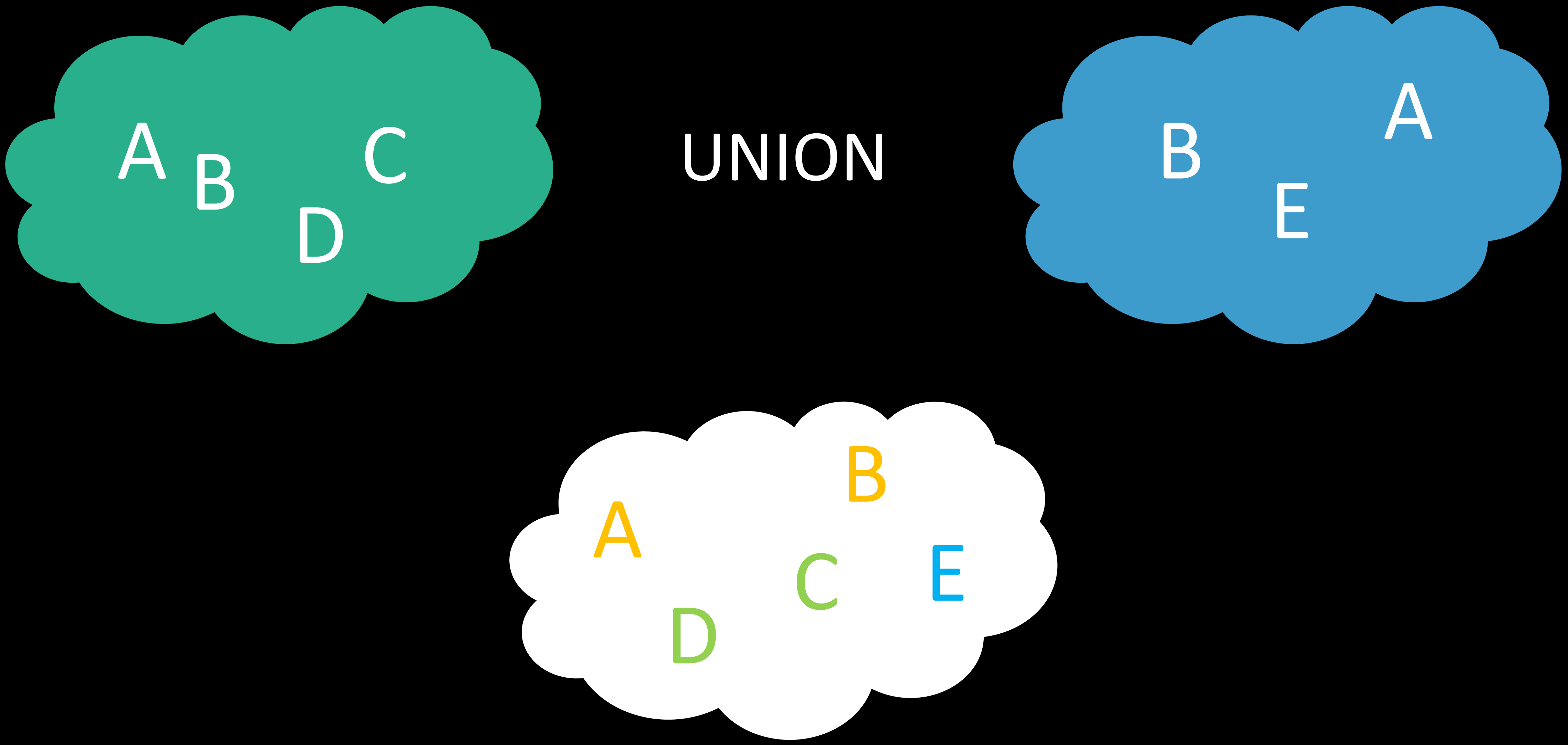
Set Operators

Set Operators

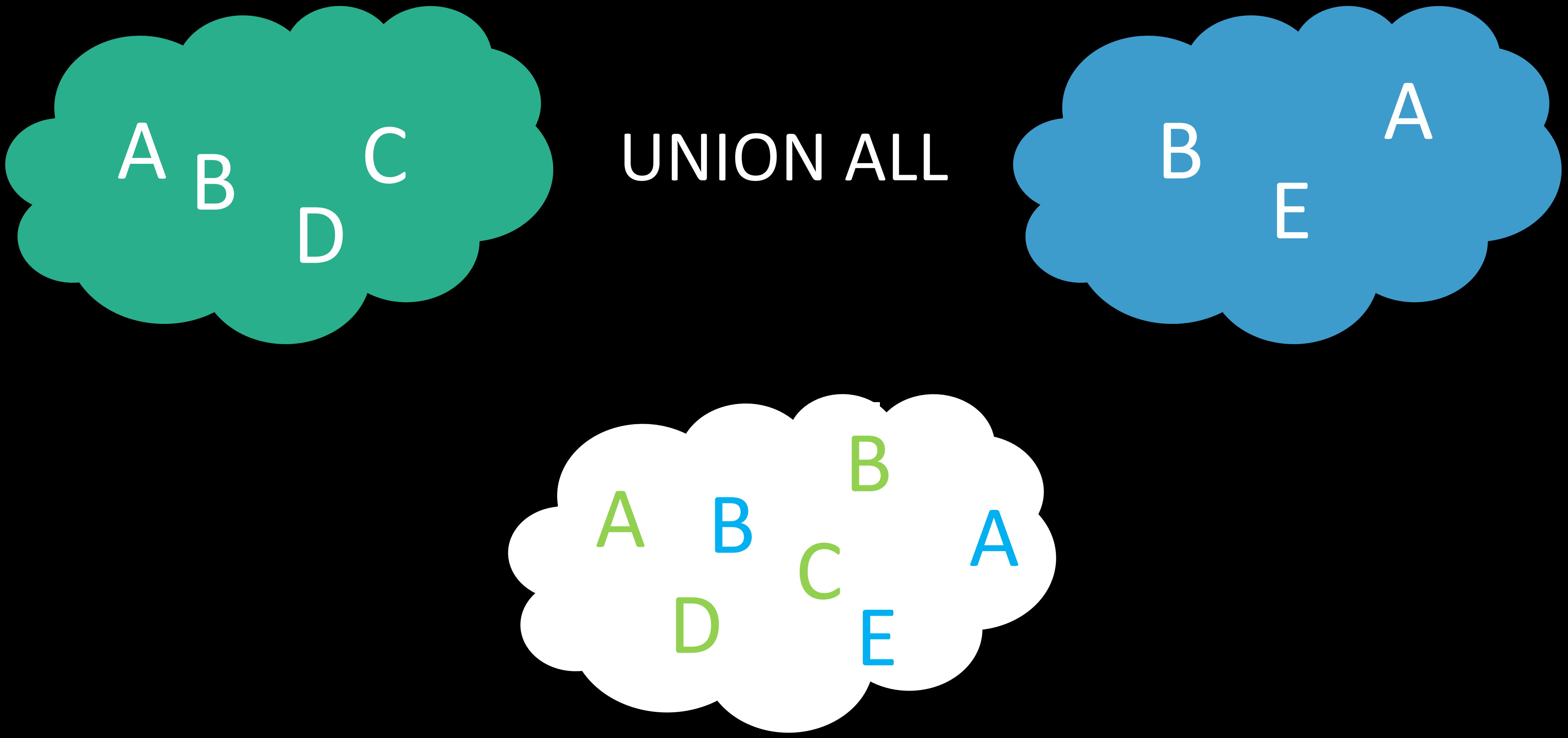
- UNION
- UNION ALL
- INTERSECT
- EXCEPT

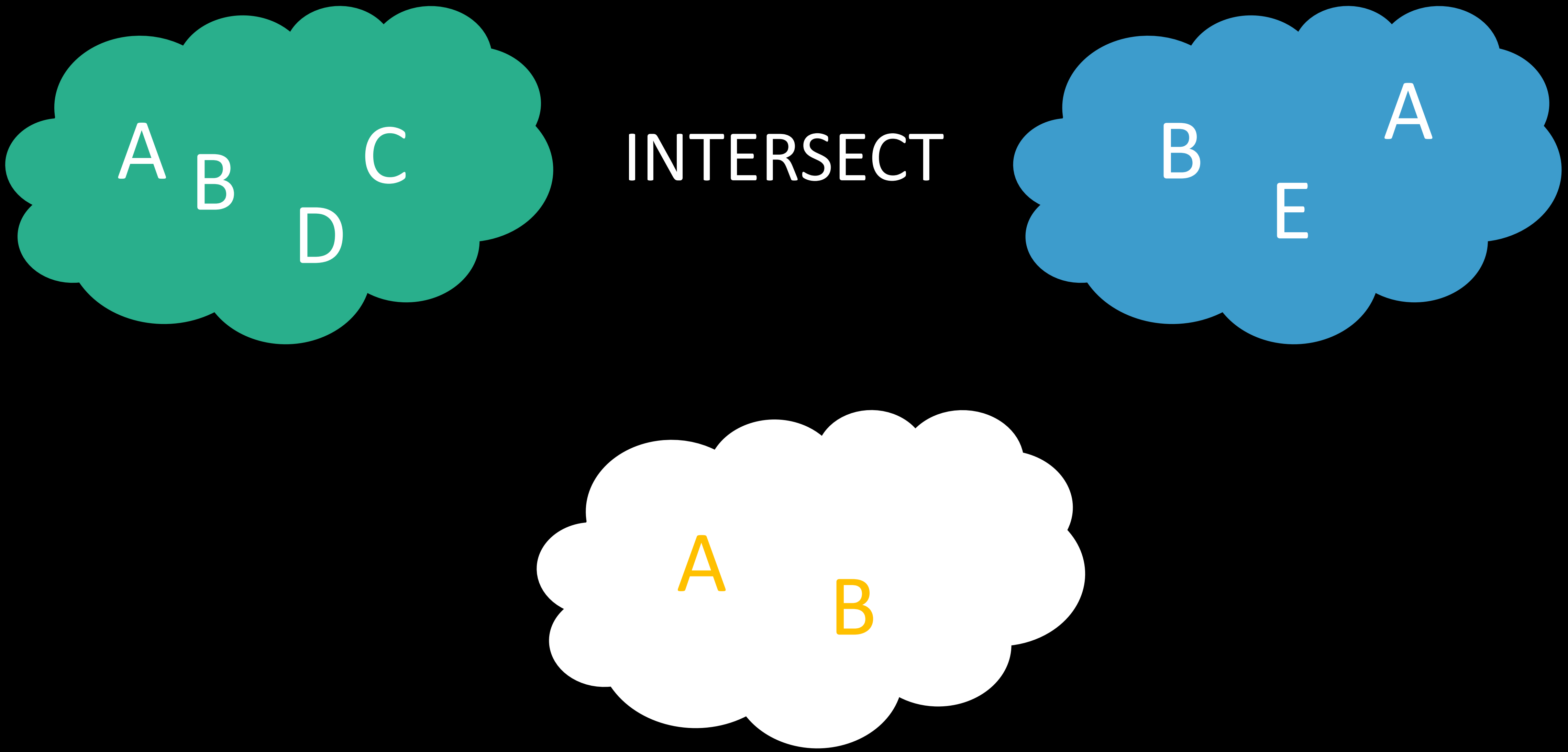
Union vs. Join

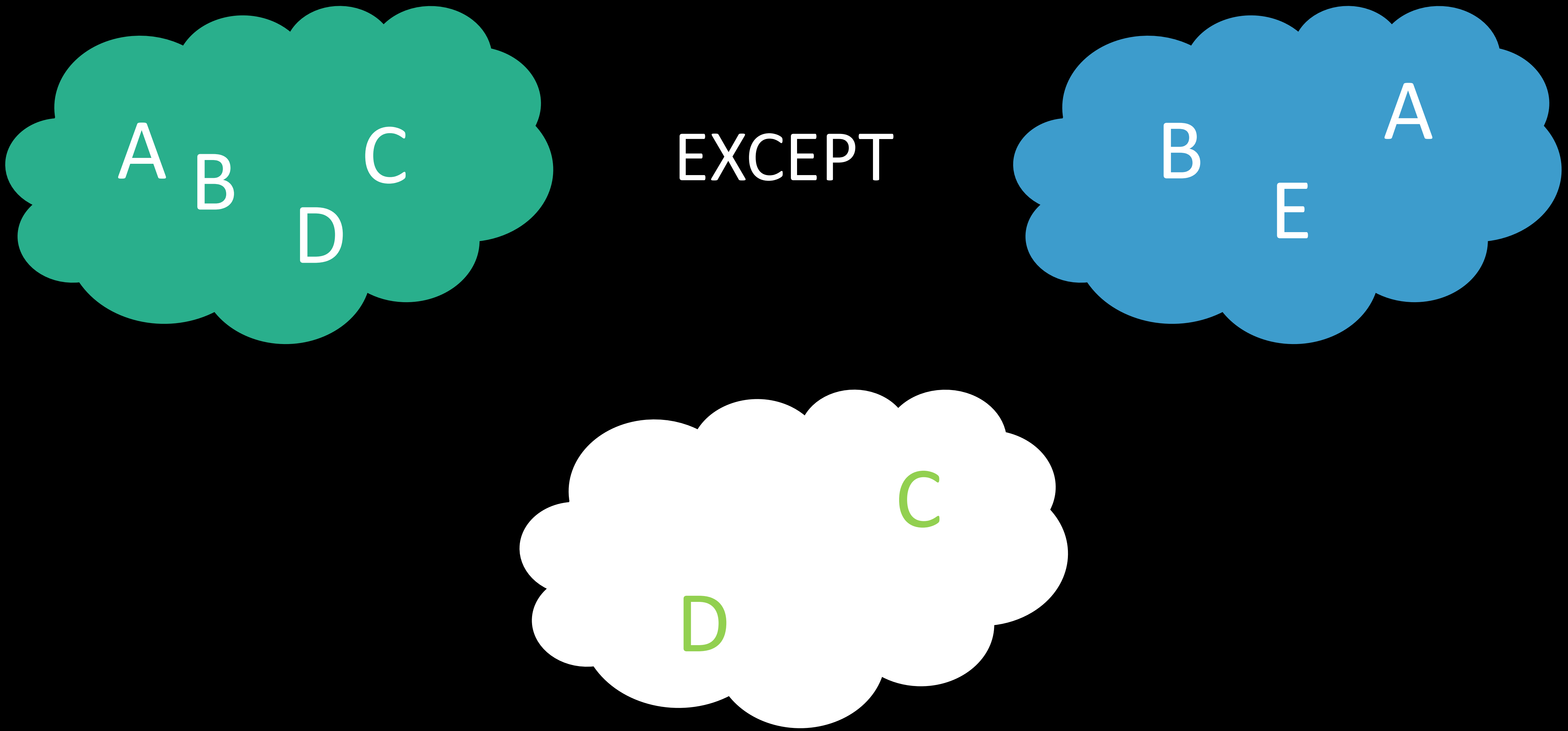




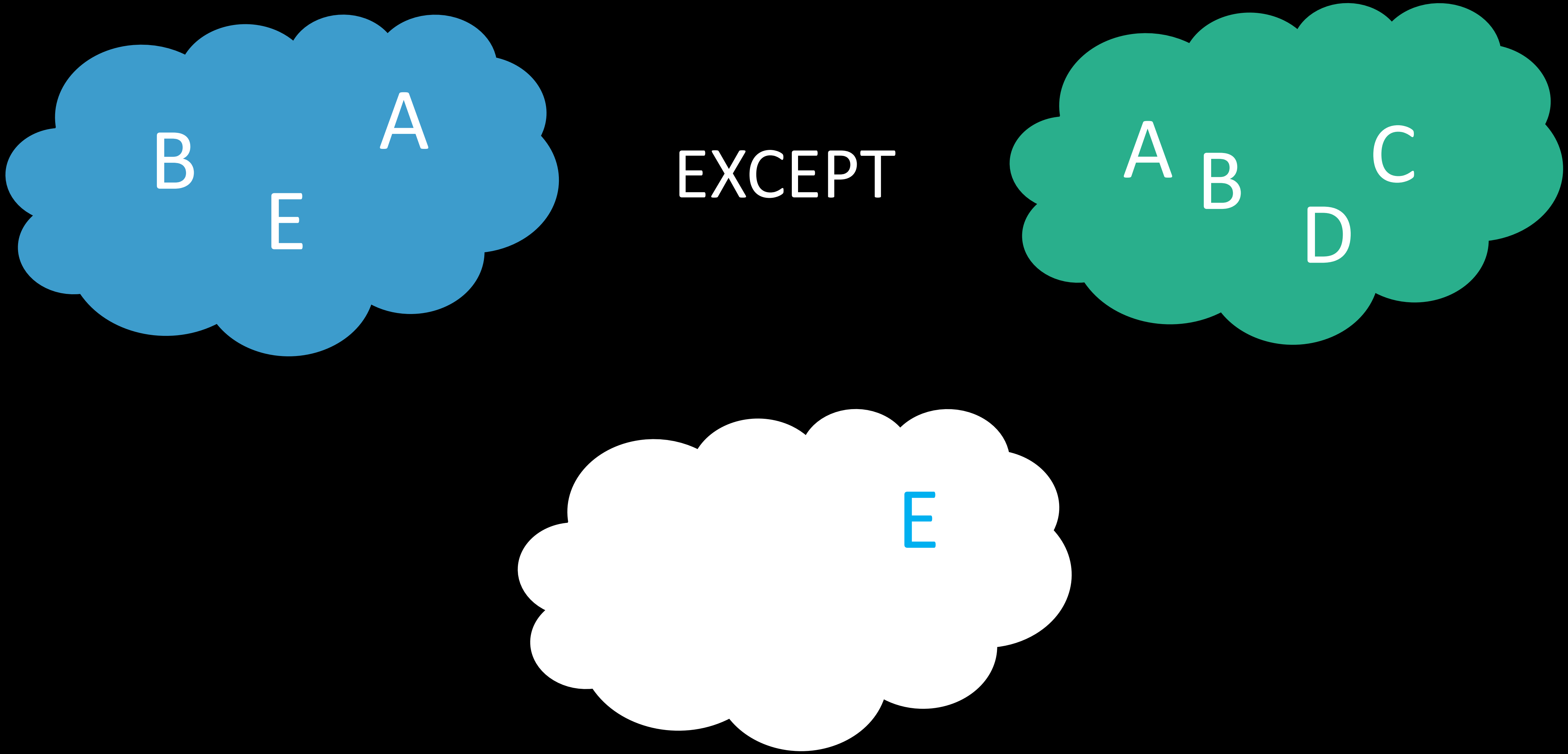
A B C D
UNION
B E A
A B C D E


A
B
D
C
UNION ALL
B
E
A
A
B
B
C
D
E
A


A B C D
INTERSECT
B E A
A B



A
B
D
C
EXCEPT
B
E
A
C
D


B
E
A
EXCEPT
A
B
D
C
E

Questions

?

Next

- Hands-on exercise: Using set operators
- Q&A
- Conclusions

Mastering Relational SQL Querying

Course Conclusion

Day 1 Agenda

- What is SQL?
- SQL, RDM, and RDBMS
- Installing the development environment
- SQL language constructs
- The data query language (DQL)
- SELECT query clauses and Query logical processing
- FROM and JOIN

Day 2 Agenda

- WHERE filtering
- GROUP BY and HAVING
- ORDER BY and LIMIT / FETCH
- Subqueries
- Set operators
- Conclusions

Goals

- Background - what are SQL and RDM?
- Understand SQL query processing phases
- Think sets, not procedural
- Write basic SQL queries, and fully understand them
- Solid foundation; less focus on detail and syntax
- Passion for SQL

What Next?

- Master the relational model
- Advanced Courses
- Advanced Books
- Product Documentation
- Use www.wikipedia.org
  - And please consider donating annually

Get involved!

- Local User Groups and meetups
- Online Communities
- Trade Conferences
- Blogs
- Twitter #sql #sqlhelp …

Course Evaluations

Questions

?

Take Home Exercise

Write a query that returns

- All USA customers with all the different items they have ordered
- The average quantity per item they have ordered across all orders
- The average quantity ordered for that item, for all ***other*** customers (US or not) across all orders
- Exclude customers who ordered less than 3 distinct items total

Send me your solutions at www.amilevin.com or on GitHub