Hacker School SQL Talk

Spring Batch 2013

By Dom Roselli

Agenda

1. What is SQL?
2. PostgreSQL Intro and Terminology
3. Basic Queries
4. Indexes

What is SQL?

Structured Query Language. Based on relational algebra and tuple relational calculus

Two parts:

- DDL (Data Definition Language)
- DML (Data Manipulation Language)

* We're focusing on DML

What is SQL? (con't)

- Adopted by ANSI in 86 (current is SQL: 2011), but different vendors do not perfectly follow the standard and they extend the standard in proprietary ways.

- This is important to understand because not all SQL works across products and some have features that others don't, or different keywords for similar behavior.

PostgreSQL Intro & Terminology

Two tools that I know of

- CLI: psql
- GUI: pgAdmin III

Basic Terminology

- Database
  a logical container for tables, views, functions, indexes, etc...
- Table
  stores data
- View
  a virtual table that is used to provide an interface to complex wqueries or controll access to underlying data

Basic Terminology (con't)

- Function/Stored Procedure
  Some RDBMS have stored procedures, which are objects that contain a series of statements that are (typically) executed in order. They can return a value or set of values, but don't necessarily have to.

  Functions are objects that return a single value (scalar) or a list of records (dataset). I believe Postgres only has Functions

Logical Query Process - Overview

(7) SELECT
(1) FROM <left_table>
(3) <join_type> JOIN <right_table>
(2) ON <join_condition>
(4) WHERE <where_condition>
(5) GROUP BY <group_by_list>
(6) HAVING <having_condition>
(8) ORDER BY <order_by_list>

Basic Logical Query Processing

(3) SELECT <select_list>
(1) FROM <left_table>
(2) WHERE <where_condition>

(1) FROM

Performs a Cartesian product on tables in the FROM clause.

(2) WHERE

The WHERE filter is applied to all rows in the virtual table coming from the previous step.

Only rows where the filter evaluates to true become part of the virtual table created in this step.

Three Valued Logic

- In SQL, the possible values of a logic expression are TRUE, FALSE and UNKNOWN.
- UNKNOWN logical value occurs in expressions that involve NULL
- **Most important thing to realize**
  The only thing evaluated with UNKNOWN that is TRUE, is TRUE OR UNKNOWN. Everything else is either FALSE or UNKNOWN

Kleene Logic Table

A	B	A and B	A or B
False	False	False	False
False	Unknown	False	Unknown
False	True	False	True
Unknown	**False**	**False**	**Unknown**
Unknown	**Unknown**	**Unknown**	**Unknown**
Unknown	**True**	**Unknown**	**True**
True	False	False	True
True	Unknown	Unknown	True
True	True	True	True

source: http://en.wikipedia.org/wiki/Three-valued_logic

(3) SELECT

This is the virtual table that will be eventually returned to the caller. This can be base columns or manipulation of base columns from the virtual table obtained in the previous step.

Filter Comparison Operators

Let's look at some code...

Aggregate Functions

(5) SELECT <select_list>
(1) FROM <left_table>
(2) WHERE <where_condition>
(3) GROUP BY <group_by_list>
(4) HAVING <having_condition>

(3) GROUP BY

- The GROUP BY list contains columns that constitute how you want to group the data of the results from the previous step.
- The only things that can show up in your SELECT list are:
  1. the columns that are in your GROUP BY list
  2. any aggregate function you apply to the raw rows in that group

(4) HAVING

HAVING is a filter, like WHERE. But it applies to the groups you create in GROUP BY.

JOINS

(7) SELECT
(1) FROM <left_table>
(3) <join_type> JOIN <right_table>
(2) ON <join_condition>
(4) WHERE <where_condition>
(5) GROUP BY <group_by_list>
(6) HAVING <having_condition>

(2) ON

- ON is a filter, like WHERE and HAVING, except that it's applied first.
- It's applied to each row in the virtual table from the previous step (in this case FROM).
- But, wait!! You just said that ON comes before JOIN? How so?

  Short answer:
  The behavior varies based on whether it's a INNER or OUTER JOIN. Explanation to follow.

(3) JOIN

There are 5 basic types of JOINS that we'll discuss:

1. CROSS JOIN
2. INNER JOIN
3. LEFT OUTER JOIN
4. RIGHT OUTER JOIN
5. FULL OUTER JOIN

These can be classified into two categories:

1) JOINS that preserve one or both tables
2) JOINS that don't

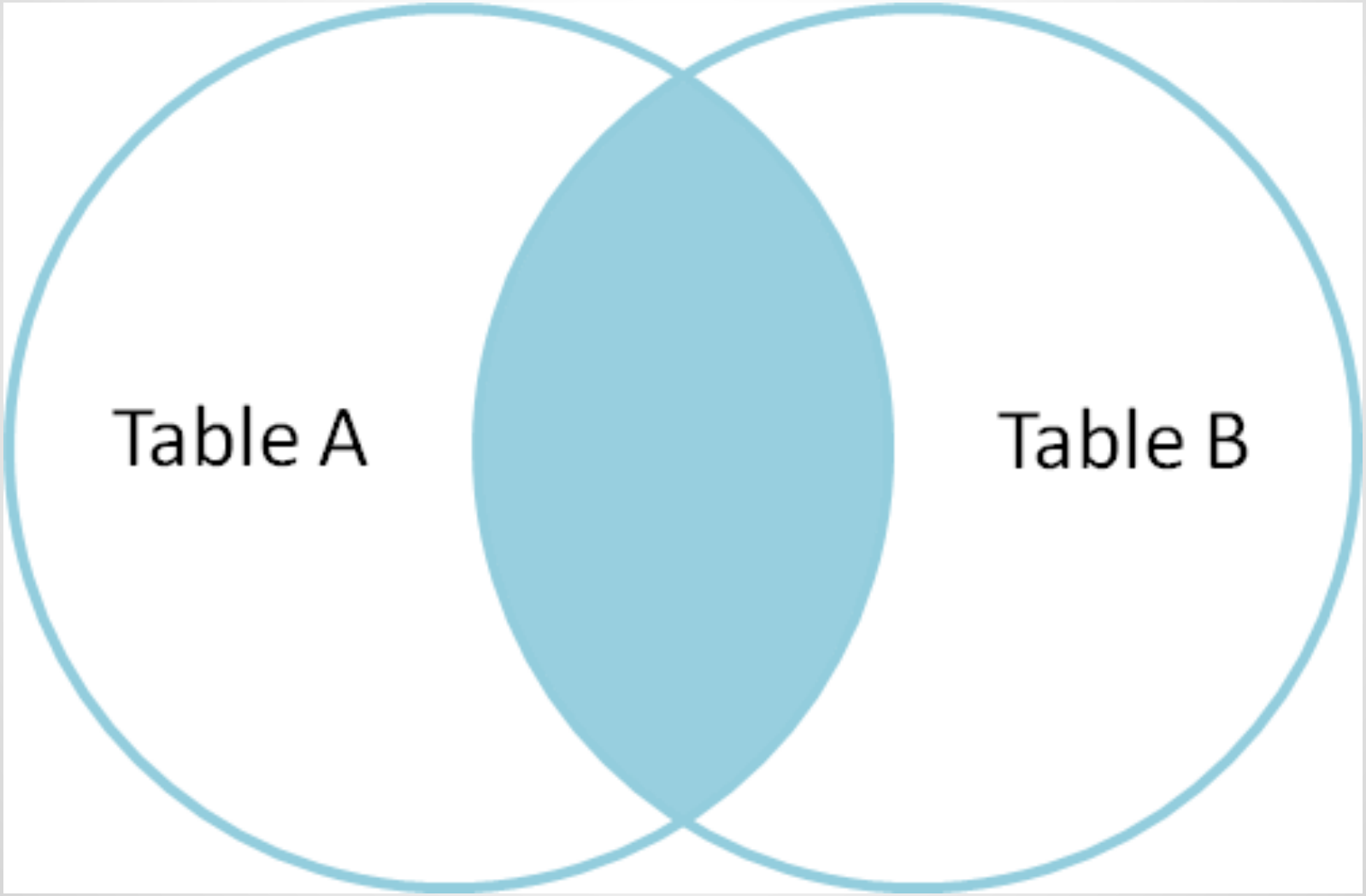
CROSS JOIN

- Performs the Cartesian product of the right and left tables
- Does not preserve either table

INNER JOIN

- An INNER JOIN returns a virtual table that contains all the rows from the LEFT table and RIGHT table, where the ON clause evaluates to TRUE
- Canonical way to relate all the entities (rows) in one table to the entities in another
- Does not preserve either table

INNER JOIN



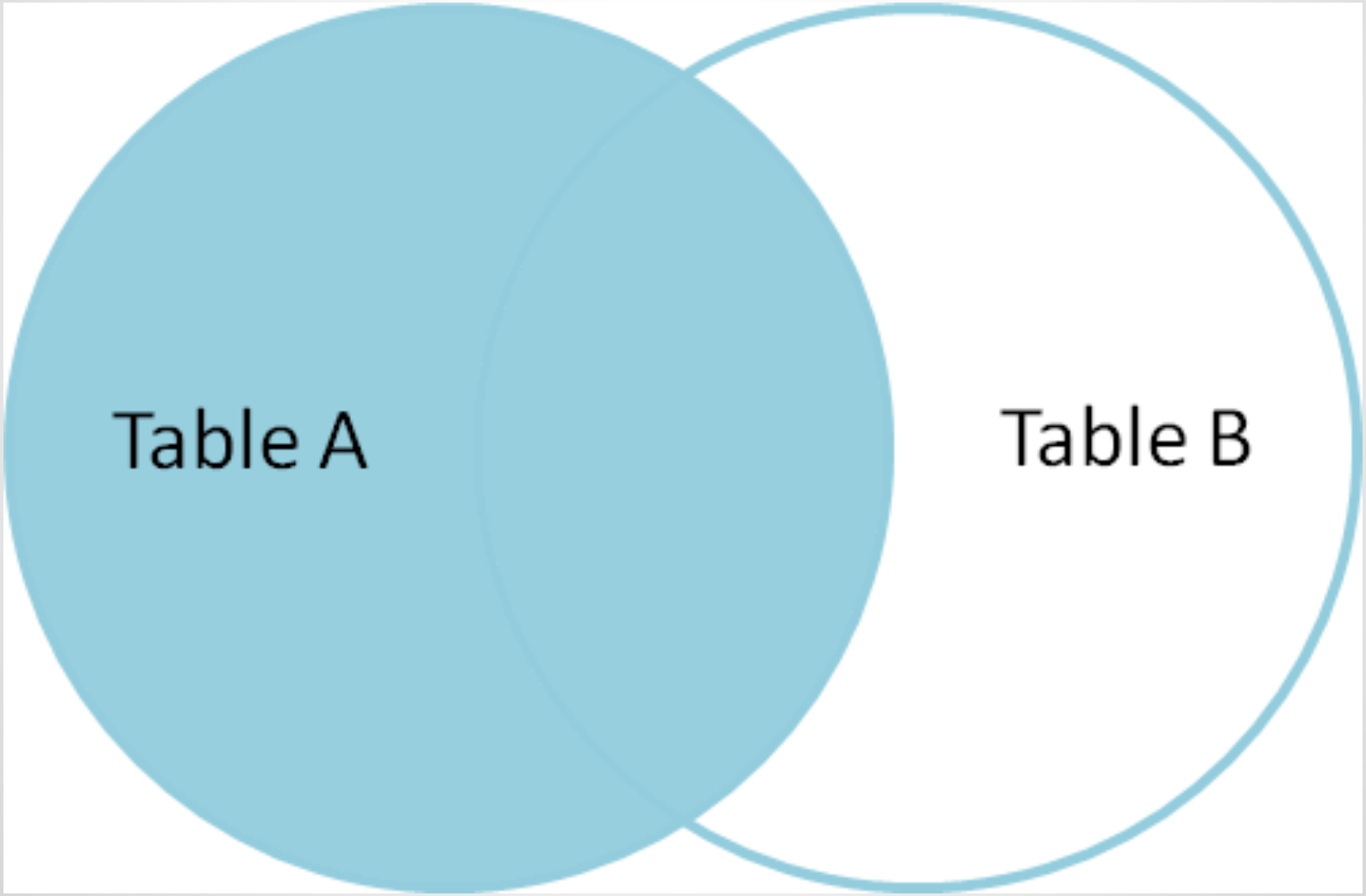
OUTER JOINs

- These do preserve at least one table
- This is logically where JOIN is evaluated after ON.
- The ON clause filters out the rows that don't result in a true evaluation. However, the rows from the preserved table(s) that did not evaluate to true in the ON clause will be added back into the resulting virtual table.

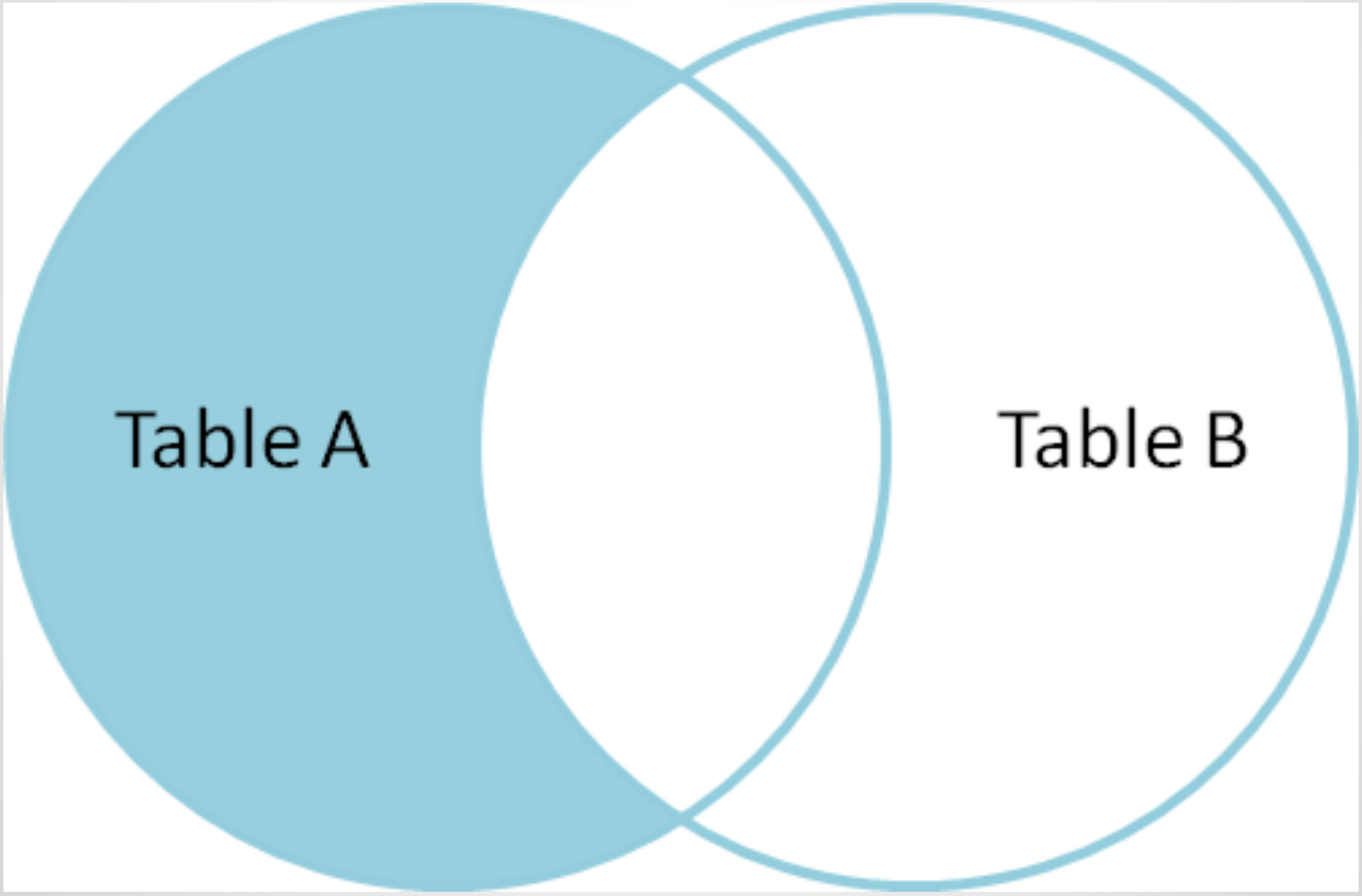
LEFT OUTER JOIN

- Initially, logically evaluates like an INNER JOIN. But the rows from the LEFT table that did not evaluate to TRUE with the ON clause are added back into the virtual table produced for the query.
- The corresponding values for rows that did not match in the RIGHT table are NULL

LEFT OUTER JOIN VENN DIAGRAM



LEFT OUTER JOIN VENN DIAGRAM (NULL FILTERED)



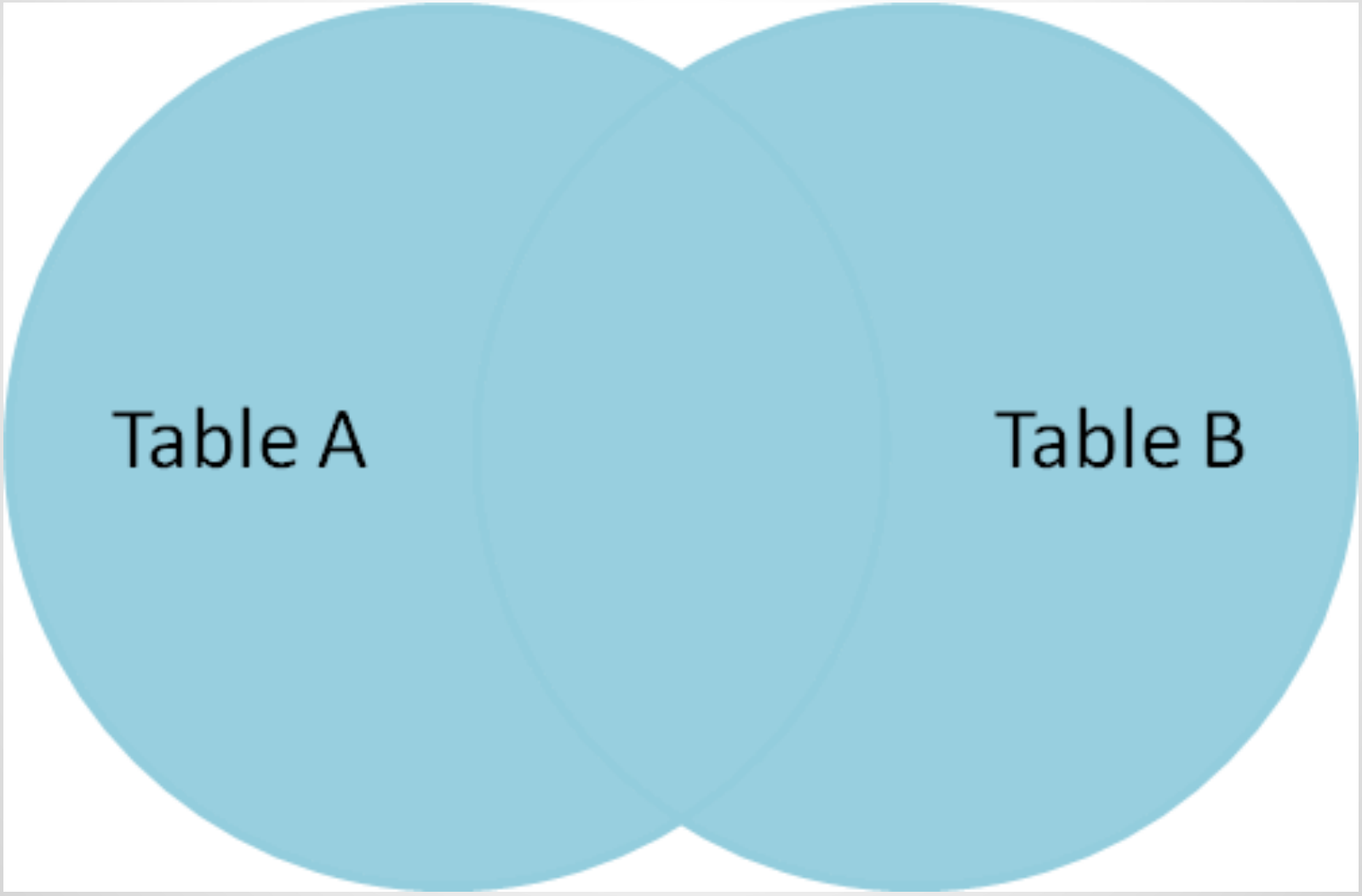
RIGHT OUTER JOIN

- Just like a LEFT OUTER JOIN, except it does it from the RIGHT tables perspective. (i.e. the RIGHT table is preserved, and rows are added back into the resulting virtual table from it, instead of the LEFT table)

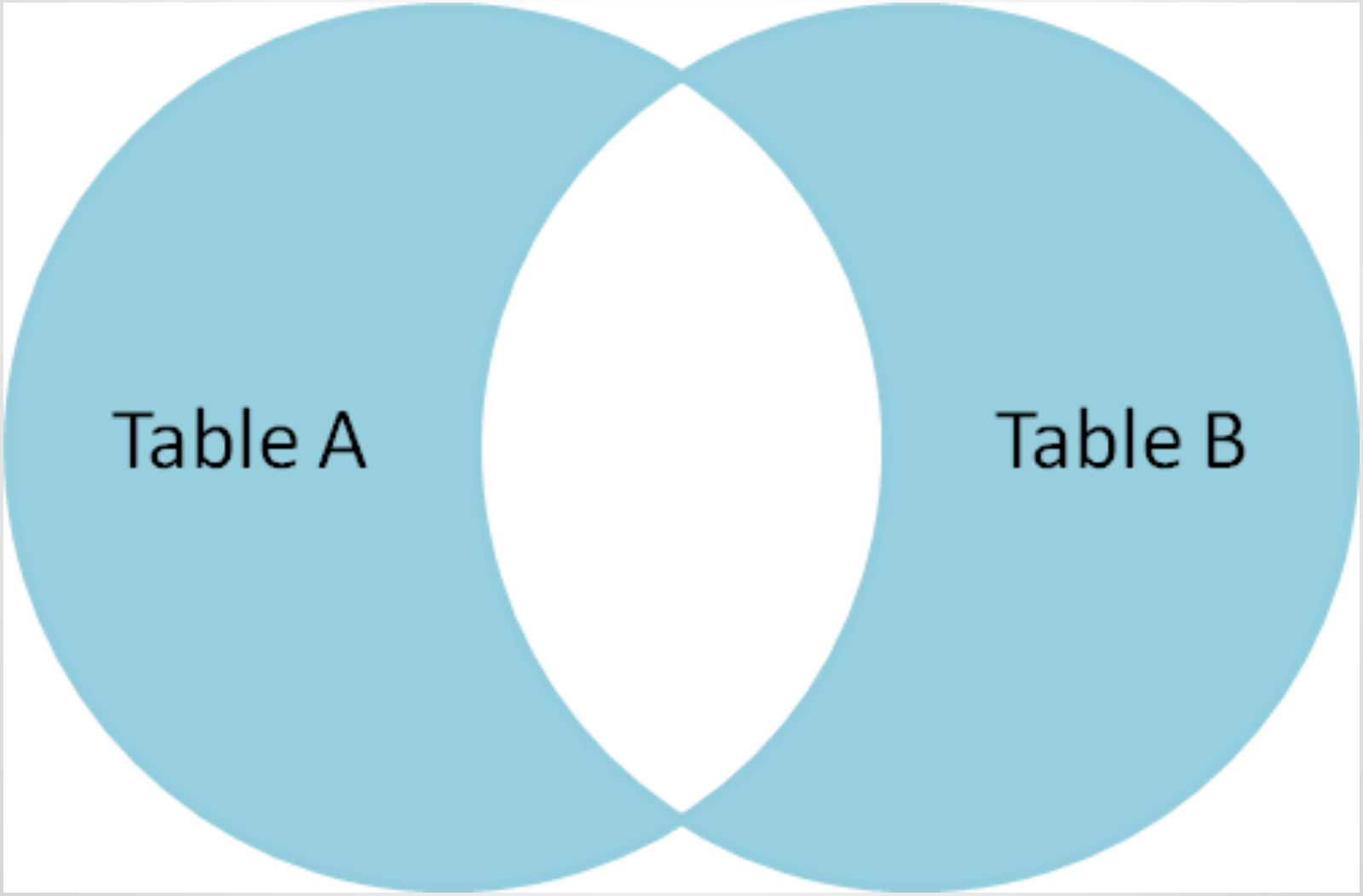
FULL OUTER JOIN

- Both the RIGHT and LEFT table are preserved.
- The rows that evaluate to FALSE in the ON clause for both tables are added back into the resulting virtual table during the JOIN.
- The rows that are FALSE in ON from both tables have corresponding NULL values..

FULL OUTER JOIN VENN DIAGRAM



FULL OUTER JOIN VENN DIAGRAM (NULL FILTERED)



ON vs. WHERE
Which should I use?

- Only really matters when you do OUTER JOINs
- Recall:
  ON filters before the OUTER JOIN
  WHERE filters after the OUTER JOIN
- The OUTER JOIN may add results back into the resulting virtual table that were filtered out in the ON filter
- WHERE will filter out anything that does not evaluate to TRUE

Questions?

Acknowledgements

- Inside Microsoft® SQL ServerTM 2005 T-SQL Querying, By Itzik Ben-Gan
- Database Tutorial http://michaelmclaughlin.info/db1/lesson-5-querying-data/single-table-query/
- Three Valued Logic http://en.wikipedia.org/wiki/Three-valued_logic
- Paglia Sample Database http://pgfoundry.org/projects/dbsamples/
- SQL JOIN Venn Diagrams http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html