



# HOWTO SQL: SIERRA - PART 1

Ray Voelker

[ray.voelker@cincinnati library.org](mailto:ray.voelker@cincinnati library.org)



# LINKS FOR THIS PRESENTATION:

- Slides:  
<https://rayvoelker.github.io/iug2021/>
- PDF Slides:  
<https://rayvoelker.github.io/iug2021/slides.pdf>
- SQL Sandbox:  
<https://howtosql.cincy.pl/iug2021/>

# MEET JEREMY AND RAY

...and Rufus and Audrey



# OVERVIEW OF RELATIONAL DATABASES

- Data in these types of databases are stored in collections, or **tables**
- Tables have a number of rows and columns
- Each row is (often) represented with a **unique key**.

# OVERVIEW OF RELATIONAL DATABASES: KEYS

Keys (typically called ID's in the Sierra Database) come in two varieties, and they define the relationship between tables.

- Primary Key
- Foreign Key

sierra\_view  
record\_metadata

	id bigint	record_type_code character(1)	record_num integer	creation_date_gmt timestamp with time zone
1	420907795009	b	1000001	2012-06-19 18:48:06-04
2	420907795010	b	1000002	2012-06-19 18:48:07-04
3	420907795011	b	1000003	2012-06-19 18:48:07-04
4	420907795012	b	1000004	2012-06-19 18:48:07-04
5	420907795013	b	1000005	2012-06-19 18:48:08-04

sierra\_view  
record\_metadata

	id bigint	record_type_code character(1)	record_num integer	creation_date_gmt timestamp with time zone
1	420907795009	b	1000001	2012-06-19 18:48:06-04
2	420907795010	b	1000002	2012-06-19 18:48:07-04
3	420907795011	b	1000003	2012-06-19 18:48:07-04
4	420907795012	b	1000004	2012-06-19 18:48:07-04
5	420907795013	b	1000005	2012-06-19 18:48:08-04

sierra\_view  
bib\_record\_property

	id integer	bib_record_id bigint	best_title character varying(1000)	publish_year integer
1	357762	420907795009	Water monsters : opposing viewpoints	1991
2	357763	420907795010	Seeking the old paths, and other sermons;	1899
3	357764	420907795011	The Foundation grants index.	1971
4	357765	420907795012	The religion of tomorrow	1899
5	357766	420907795013	Upward steps	1899

sierra\_view  
record\_metadata

 primary key

	id bigint	record_type_code character(1)	record_num integer	creation_date_gmt timestamp with time zone
1	420907795009	b	1000001	2012-06-19 18:48:06-04
2	420907795010	b	1000002	2012-06-19 18:48:07-04
3	420907795011	b	1000003	2012-06-19 18:48:07-04
4	420907795012	b	1000004	2012-06-19 18:48:07-04
5	420907795013	b	1000005	2012-06-19 18:48:08-04

sierra\_view  
bib\_record\_property

	id integer	bib_record_id bigint	best_title character varying(1000)	publish_year integer
1	357762	420907795009	Water monsters : opposing viewpoints	1991
2	357763	420907795010	Seeking the old paths, and other sermons;	1899
3	357764	420907795011	The Foundation grants index.	1971
4	357765	420907795012	The religion of tomorrow	1899
5	357766	420907795013	Upward steps	1899



sierra\_view  
record\_metadata

primary key  
foreign key

	id bigint	record_type_code character(1)	record_num integer	creation_date_gmt timestamp with time zone
1	420907795009	b	1000001	2012-06-19 18:48:06-04
2	420907795010	b	1000002	2012-06-19 18:48:07-04
3	420907795011	b	1000003	2012-06-19 18:48:07-04
4	420907795012	b	1000004	2012-06-19 18:48:07-04
5	420907795013	b	1000005	2012-06-19 18:48:08-04

sierra\_view  
bib\_record\_property

	id integer	bib_record_id bigint	best_title character varying(1000)	publish_year integer
1	357762	420907795009	Water monsters : opposing viewpoints	1991
2	357763	420907795010	Seeking the old paths, and other sermons;	1899
3	357764	420907795011	The Foundation grants index.	1971
4	357765	420907795012	The religion of tomorrow	1899
5	357766	420907795013	Upward steps	1899

sierra\_view  
record\_metadata

primary key  
foreign key

	id bigint	record_type_code character(1)	record_num integer	creation_date_gmt timestamp with time zone
1	420907795009	b	1000001	2012-06-19 18:48:06-04
2	420907795010	b	1000002	2012-06-19 18:48:07-04
3	420907795011	b	1000003	2012-06-19 18:48:07-04
4	420907795012	b	1000004	2012-06-19 18:48:07-04
5	420907795013	b	1000005	2012-06-19 18:48:08-04



sierra\_view  
bib\_record\_property

	id integer	bib_record_id bigint	best_title character varying(1000)	publish_year integer
1	357762	420907795009	Water monsters : opposing viewpoints	1991
2	357763	420907795010	Seeking the old paths, and other sermons;	1899
3	357764	420907795011	The Foundation grants index.	1971
4	357765	420907795012	The religion of tomorrow	1899
5	357766	420907795013	Upward steps	1899

sierra\_view  
record\_metadata

primary key  
foreign key

	id bigint	record_type_code character(1)	record_num integer	creation_date_gmt timestamp with time zone
1	420907795009	b	1000001	2012-06-19 18:48:06-04
2	420907795010	b	1000002	2012-06-19 18:48:07-04
3	420907795011	b	1000003	2012-06-19 18:48:07-04
4	420907795012	b	1000004	2012-06-19 18:48:07-04
5	420907795013	b	1000005	2012-06-19 18:48:08-04



sierra\_view  
bib\_record\_property

	id integer	bib_record_id bigint	best_title character varying(1000)	publish_year integer
1	357762	420907795009	Water monsters : opposing viewpoints	1991
2	357763	420907795010	Seeking the old paths, and other sermons;	1899
3	357764	420907795011	The Foundation grants index.	1971
4	357765	420907795012	The religion of tomorrow	1899
5	357766	420907795013	Upward steps	1899

sierra\_view  
record\_metadata

primary key  
foreign key

	id bigint	record_type_code character(1)	record_num integer	creation_date_gmt timestamp with time zone
1	420907795009	b	1000001	2012-06-19 18:48:06-04
2	420907795010	b	1000002	2012-06-19 18:48:07-04
3	420907795011	b	1000003	2012-06-19 18:48:07-04
4	420907795012	b	1000004	2012-06-19 18:48:07-04
5	420907795013	b	1000005	2012-06-19 18:48:08-04



sierra\_view  
bib\_record\_property

	id integer	bib_record_id bigint	best_title character varying(1000)	publish_year integer
1	357762	420907795009	Water monsters : opposing viewpoints	1991
2	357763	420907795010	Seeking the old paths, and other sermons;	1899
3	357764	420907795011	The Foundation grants index.	1971
4	357765	420907795012	The religion of tomorrow	1899
5	357766	420907795013	Upward steps	1899

sierra\_view  
record\_metadata

primary key  
foreign key

	id bigint	record_type_code character(1)	record_num integer	creation_date_gmt timestamp with time zone
1	420907795009	b	1000001	2012-06-19 18:48:06-04
2	420907795010	b	1000002	2012-06-19 18:48:07-04
3	420907795011	b	1000003	2012-06-19 18:48:07-04
4	420907795012	b	1000004	2012-06-19 18:48:07-04
5	420907795013	b	1000005	2012-06-19 18:48:08-04



sierra\_view  
bib\_record\_property

	id integer	bib_record_id bigint	best_title character varying(1000)	publish_year integer
1	357762	420907795009	Water monsters : opposing viewpoints	1991
2	357763	420907795010	Seeking the old paths, and other sermons;	1899
3	357764	420907795011	The Foundation grants index.	1971
4	357765	420907795012	The religion of tomorrow	1899
5	357766	420907795013	Upward steps	1899

sierra\_view  
record\_metadata

primary key  
foreign key

	id bigint	record_type_code character(1)	record_num integer	creation_date_gmt timestamp with time zone
1	420907795009	b	1000001	2012-06-19 18:48:06-04
2	420907795010	b	1000002	2012-06-19 18:48:07-04
3	420907795011	b	1000003	2012-06-19 18:48:07-04
4	420907795012	b	1000004	2012-06-19 18:48:07-04
5	420907795013	b	1000005	2012-06-19 18:48:08-04



sierra\_view  
bib\_record\_property

	id integer	bib_record_id bigint	best_title character varying(1000)	publish_year integer
1	357762	420907795009	Water monsters : opposing viewpoints	1991
2	357763	420907795010	Seeking the old paths, and other sermons;	1899
3	357764	420907795011	The Foundation grants index.	1971
4	357765	420907795012	The religion of tomorrow	1899
5	357766	420907795013	Upward steps	1899

# OVERVIEW OF DATABASE ENTITY- RELATIONSHIP MODEL (ERM VIEW)

- Defines the types of relationships that can exist between entities (tables)
  - One-to-One
  - One-to-Many (most common relationship)
  - Many-to-Many

# **DATABASE ENTITY- RELATIONSHIP MODEL**

## **ONE-TO-ONE**

- A Country can have one (and only one) Capital City
- A Capital City can have one (and only one) Country



# DATABASE ENTITY- RELATIONSHIP MODEL

ONE-TO-ONE



# **DATABASE ENTITY- RELATIONSHIP MODEL**

## **ONE-TO-MANY**

- A Mother may have many Children
- A Child has only one (biological) Mother

# DATABASE ENTITY- RELATIONSHIP MODEL

## ONE-TO-MANY



# **DATABASE ENTITY- RELATIONSHIP MODEL**

## **MANY-TO-MANY**

- Authors can write several Books
- Books can be written by several Authors

# DATABASE ENTITY- RELATIONSHIP MODEL

MANY-TO-MANY



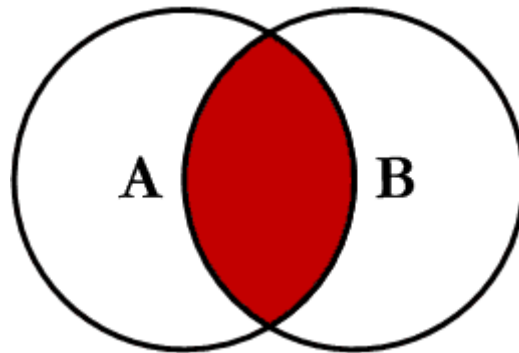
## Relational Databases: Relationships (JOINS)

- Sets of data can be derived from Relational Operators from traditional math sets.
- We'll cover two of the more common JOIN operations
  - JOIN (or INNER JOIN)
  - LEFT JOIN (or LEFT OUTER JOIN)

## Relational Databases: Relationships (JOINS) cont.

### **JOIN (OR INNER JOIN)**

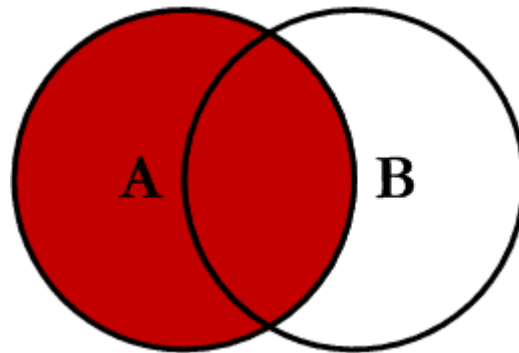
- Most common type of join that there is
- Given two sets, A (left) and B (right), performing this join will return a set containing all elements of A that also belong to B.



Relational Databases: Relationships (JOINS) cont.

## **LEFT JOIN (OR LEFT OUTER JOIN)**

- Given two sets, A (left) and B (right) performing this join will return a set containing all elements of table A, as well as the elements of A that also belong to B





# SQL OVERVIEW

- Structured Query Language
- Standardized language that allows a user to interface with a relational database

## SQL Overview cont.

- SQL statements are groups of clauses or other statements that define the operation on the database
- Some of the more common statements include the following... (SQL is picky about the order in which these statements appear in so they're presented in the order that they can appear in the statement.)

## SQL Overview cont.

- SELECT
  - Retrieves data from tables
  - Most commonly used statement
- UPDATE and SET
  - Modifies a set of existing table rows
- DELETE
  - Remove set of existing rows from the table

## SQL Overview cont.

- CREATE (TEMPORARY TABLE)
  - Typically used to create a table in the database
  - Can be used to create TEMPORARY table for use in subsequent queries  
(exists for the duration of a database session, and is dropped when the connection or session ends)

## SQL Overview cont.

- JOIN / LEFT JOIN / etc
  - Performing a join will combine the data with that of another table
- FROM
  - Indicates which table to retrieve data from

## SQL Overview cont.

- WHERE / WHERE IN
  - Include or exclude data based on comparisons
- GROUP BY
  - Reduces sets into common values
- HAVING
  - Allows for filtering of the GROUP BY statement

## SQL Overview cont.

- ORDER BY
  - Sort the set  
in ascending order (ASC)  
or descending order (DESC)
- LIMIT / OFFSET
  - Returns specific numbers of rows from given starting point
  - Usually used in conjunction with the ORDER BY clause to ensure the set is always presented in the same order

# SQL SELECT STATEMENTS

- FINALLY SOME EXAMPLES!
- Let us say we wanted a list ...
  - Bib Record info
  - 10 Oldest (earliest dates created)
- The following query would give us a very basic view of those records:



```
1 SELECT
2 r.id, r.record_type_code,
3 r.record_num, r.creation_date_gmt,
4 r.deletion_date_gmt, r.num_revisions
5
6 FROM
7 sierra_view.record_metadata AS r
8
9 WHERE
10 r.record_type_code = 'b'
11
12 ORDER BY
13 r.creation_date_gmt ASC
```



## Filter by specific bib numbers ...

```
1 SELECT
2 r.id, r.record_type_code,
3 r.record_num, r.creation_date_gmt,
4 r.deletion_date_gmt, r.num_revisions
5
6 FROM
7 record_metadata AS r
8
9 WHERE
10 r.record_type_code = 'b'
11 AND r.record_num IN (
12 1000001, 1000032, 1000041, 1000045, 1000056,
13 1000139, 1000303, 1000345, 1000410, 1000426
14 )
15
```



## SQL SELECT Statement cont.

- This is ok, but maybe we also wanted "Title" from the bib record
- We'll use a JOIN!
- Looks like table "bib\_record\_property" has what we need

bib\_record\_property

Each row of bib\_record\_property contains primary descriptive data from a specific bib record.

Column	Data Type	Not NULL?	Comment
id	int	false	System-generated sequential ID.
bib_record_id	bigint	false	Foreign key to bib_record.
best_title	varchar	false	<div>The most authoritative available version of the title.</div> <div>The system uses the following criteria to determine the best title, in order of precedence:</div> <div><div>1. The first t-tagged MARC 245 field, subfields a, b, g, h, n, p.</div><div>2. The first t-tagged non-MARC field.</div><div>3. The first t-tagged MARC field other than 245, any subfields indexed for the t index display.</div></div>

## JOIN table **bib\_record\_property** ...

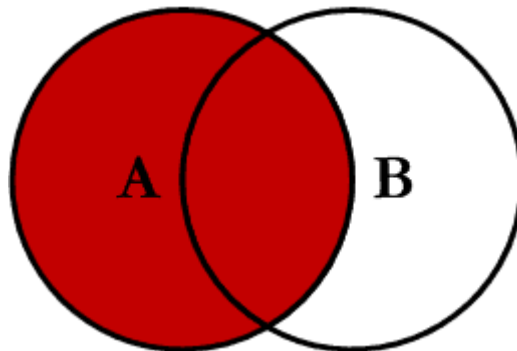
```
1 SELECT  
2 r.id, r.record_type_code,  
3 r.record_num, r.creation_date_gmt,  
4 r.deletion_date_gmt, r.num_revisions,  
5 p.best_title  
6  
7 FROM  
8 record_metadata AS r  
9  
10 JOIN bib_record_property AS p  
11     ON p.bib_record_id = r.id  
12  
13 WHERE  
14 r.record_type_code = 'b'  
15 AND r.record_num IN (
```





## SQL SELECT Statement cont.

- Table, "**bib\_record\_property**", has no foreign key for the deleted record and therefore won't be joined in our results
- We can fix this!
- ... with a LEFT JOIN (or LEFT OUTER JOIN)!



~~JOIN~~ LEFT OUTER JOIN table bib\_record\_property ...

```
1  SELECT
2  r.id, r.record_type_code,
3  r.record_num, r.creation_date_gmt,
4  r.deletion_date_gmt, r.num_revisions,
5  p.best_title
6
7  FROM
8  record_metadata AS r
9
10 --          JOIN bib_record_property as p
11 LEFT OUTER JOIN bib_record_property as p
12     ON p.bib_record_id = r.id
13
14 WHERE
15 r.record_type_code = 'b'
```



# SQL GROUP BY / HAVING AND AGGREGATE FUNCTIONS

- **GROUP BY**
  - Reduces sets into common values, or groups results of one or more column together
  - Often used along with aggregate functions
    - COUNT()
    - SUM()
    - MAX()

## SQL GROUP BY / HAVING and Aggregate Functions cont.

- How many Titles are there in our database that have
  - **publish\_year** of **1977**?

# SQL GROUP BY / HAVING and Aggregate Functions cont.

```
1 SELECT
2   p.publish_year,
3   count(*) as count
4
5 FROM
6   bib_record_property as p
7
8 WHERE
9   p.publish_year = 1977
10
11 GROUP BY
12   p.publish_year
```



## SQL GROUP BY / HAVING and Aggregate Functions cont.

- When grouped by `publish_year` How many `publish_year` values are there having a count of exactly 200 titles?



# SQL GROUP BY / HAVING and Aggregate Functions cont.

```
1 SELECT
2   p.publish_year,
3   count(*) as count
4
5 FROM
6   bib_record_property as p
7
8 GROUP BY
9   p.publish_year
10
11 HAVING
12   count(*) = 200
```



## SQL GROUP BY / HAVING and Aggregate Functions cont.

- What are the top 5 most popular titles (by item checkout\_total + renewal\_total) for items currently in the location "Blue Ash Documentaries"?

# SQL GROUP BY / HAVING and Aggregate Functions cont.

```
1 SELECT
2 p.best_title,
3 r.record_num AS bib_record_num,
4 sum( i.checkout_total + i.renewal_total) AS total_circ
5
6 FROM
7 item_record AS i
8
9 JOIN
10 bib_record_item_record_link as l
11     ON l.item_record_id = i.record_id
12
13 JOIN
14 bib_record_property AS p
15     ON p.bib_record_id = l.bib_record_id
```





## Subquery

- Subqueries are simply nested queries
- Can occur in the SELECT, FROM, or WHERE clauses
- Useful to use to use as a filter in the WHERE clause, or limiting to a single value in the SELECT clause (unexpected one-to-many situations)

## Subquery cont.

- Are there multiple barcodes associated with single item records?
  - Start by getting a set of item record ID values where the `item_record_id` appears multiple times when joining to the barcode varfield value



## Subquery cont.

```
1 SELECT
2   ir.record_id
3
4 FROM
5   item_record as ir
6
7 JOIN varfield AS vf
8     ON vf.record_id = ir.record_id
9     AND vf.varfield_type_code = 'b'
10
11 GROUP BY
12   ir.record_id
13
14 HAVING
15   count(*) > 1
```



## Subquery cont.

```
1 SELECT
2 i.record_id as item_record_id,
3 r.record_num as item_record_num,
4 i.location_code,
5 v.field_content,
6 v.occ_num
7
8 FROM
9 item_record as i
10
11 JOIN
12 record_metadata as r
13     ON r.id = i.record_id
14
15 JOIN varfield AS v
```



## Subquery cont.

```
1  SELECT
2  i.record_id,
3  i.item_status_code,
4  i.location_code,
5  v.field_content
6
7  FROM
8  item_record AS i
9
10 JOIN
11 varfield AS v
12     ON v.record_id = i.record_id
13     AND v.varfield_type_code = 'b'
14
15 -- say we were targeting these two item records.
```



## Subquery cont.

```
1  SELECT
2  i.record_id,
3  i.item_status_code,
4  i.location_code,
5  (
6      SELECT
7      v.field_content
8      FROM
9      varfield as v
10     WHERE
11     v.record_id = i.record_id
12     AND v.varfield_type_code = 'b'
13     ORDER BY
14     v.occ_num
15     LIMIT 1
```





## CASE Statement

- Returns a value when a condition is met
- Helpful for
  - formatting one value into another value
  - creating values to aggregate on

## CASE Statement cont.

- Produce a list of item record numbers and give a nicer name for the 'item\_status\_code' value

## CASE Statement cont.

```
1 SELECT
2 r.record_num AS item_record_num, i.item_status_code,
3 CASE
4 WHEN item_status_code = '-' THEN 'AVAILABLE'
5 WHEN item_status_code = 'o' THEN 'LIB USE ONLY'
6 ELSE item_status_code
7 END as our_item_status
8
9 FROM
10 item_record as i
11
12 JOIN
13 record_metadata as r
14     ON r.id = i.record_id
15
```



## CASE Statement cont.

- From the location "Main Children's Library" (1cj), produce a count of items based on their last circulation dates
  - '1-recent checkouts' = 2020-01-01 to Present
  - '2-kind-a-recent checkouts' = 2018-01-01 to 2020-01-01
  - '3-long-ago checkouts' = OLDER THAN '2018-01-01'
  - '4-NO CHECKOUTS!' = no last checkout value

## CASE Statement cont.

```
1  SELECT
2
3  r.record_num as item_record_num,
4  i.last_checkout_gmt,
5  CASE
6  WHEN i.last_checkout_gmt >= '2020-01-01' THEN '1-recent che
7  WHEN i.last_checkout_gmt >= '2018-01-01' THEN '2-kinda-rece
8  WHEN i.last_checkout_gmt < '2018-01-01' THEN '3-long-ago c
9  WHEN i.last_checkout_gmt is null THEN '4-NO CHECKOUTS!'
10 END as 'last_checkout_groups'
11
12 FROM
13 item_record as i
14
15 JOIN
```



## CASE Statement cont.

```
1 SELECT
2 CASE
3 WHEN i.last_checkout_gmt >= '2020-01-01' THEN '1-recent che
4 WHEN i.last_checkout_gmt >= '2018-01-01' THEN '2-kinda-rece
5 WHEN i.last_checkout_gmt < '2018-01-01' THEN '3-long-ago c
6 WHEN i.last_checkout_gmt is null THEN '4-NO CHECKOUTS!'
7 END as 'last_checkout_groups',
8 count(*) as count_items
9
10 FROM
11 item_record as i
12
13 WHERE
14 i.location_code = '1cj'
15
```





# THANK YOU!

## Other Useful Resources

- IUG 2019 - Sierra SQL presentation material:  
<https://iug2019-sql.github.io/>
- HOWTO SQL Sandbox (Datasette):  
<https://howtosql.cincy.pl/iug2021/>