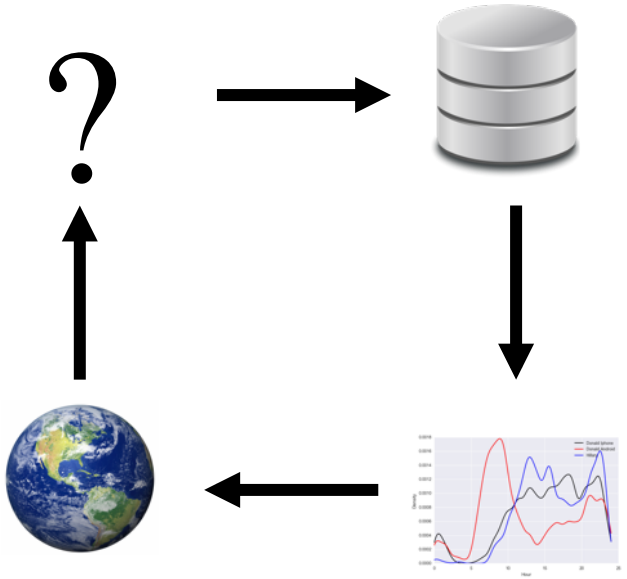
Data Science 100
Databases Part 2
(The SQL)

Slides by:

Joseph E. Gonzalez & Joseph Hellerstein,

jegonzal@berkeley.edu

jhellerstein@berkeley.edu



Previously …

Database Management Systems

A **database management systems (DBMS)** is a software system that **stores**, **manages**, and **facilitates access** to one or more databases.

- **Relational** database management systems

- *Logically* organize data in **relations** (tables)
- Structured Query Language **(SQL)** to define, manipulate and compute on data.

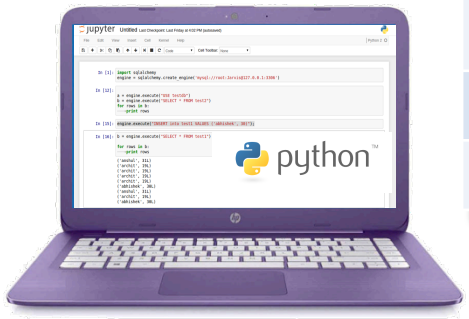
Physical Data Independence

Relations (Tables)

Name	Prod	Price
Sue	iPod	$200.00
Joey		
Alice		

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58			

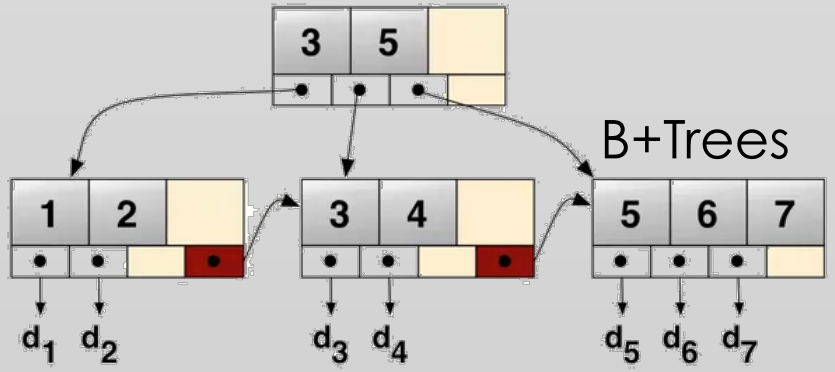
bid	bname	color
101	Interlake	blue
102	Interlake	red
104	Marine	red
103	Clipper	green

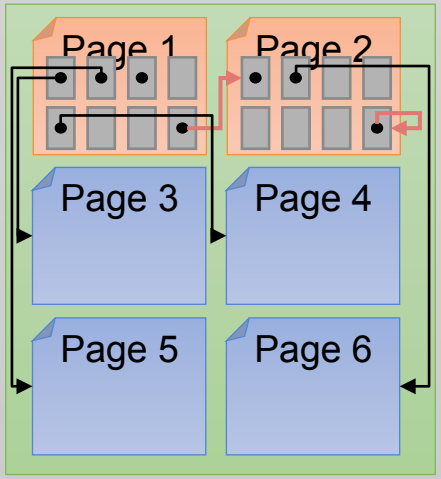


Abstraction

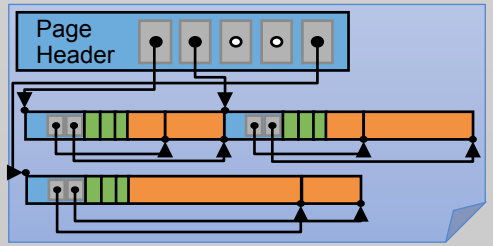
Database Management System

Optimized Data Structures





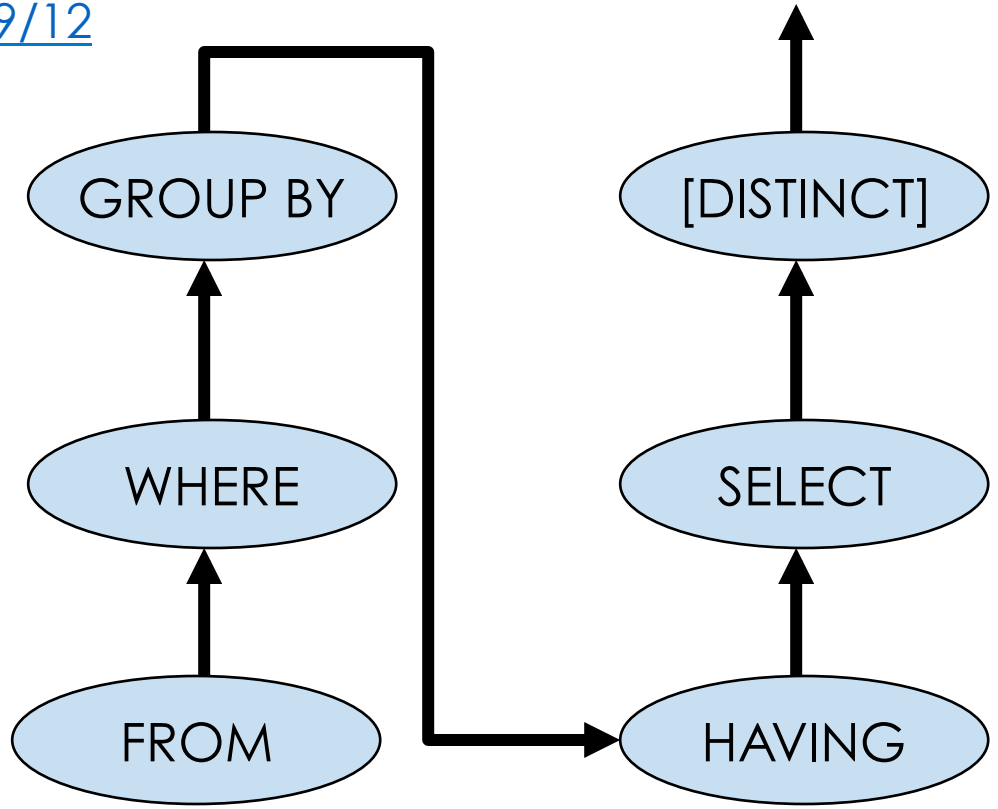
Optimized Storage



Conceptual SQL Evaluation

```
SELECT      [DISTINCT]  target-list
FROM        relation-list
WHERE       qualification
GROUP BY    grouping-list
HAVING      group-qualification
```

Try Queries Here
http://sqlfiddle.com/#!17/67109/12



Form groups & aggregate

Apply selections (eliminate rows)

One or more tables to use (outer product …)

Eliminate duplicates

Project away columns (just keep those used in SELECT, GBY, HAVING)

Eliminate groups

How do you interact with a database?

What is the DBMS?

- Server
- Software
- A library

Answer: It can be all of these.

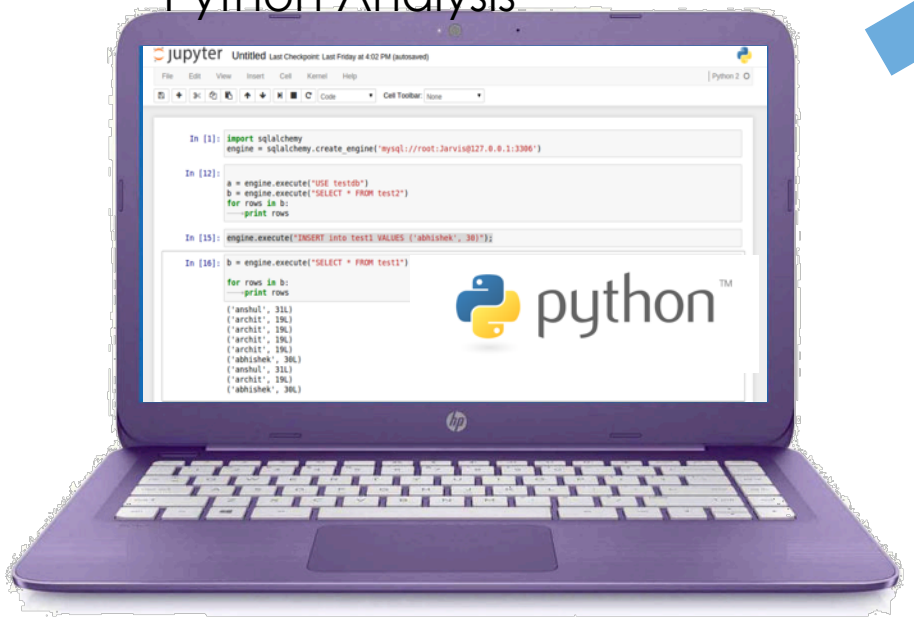
Interacting with a DBMS

DBMS Server

Query

```
SELECT * FROM sales
WHERE price > 100.0
```

Python Analysis



Response

Date	Purchase ID	Name	Price
9/20/2012	1234	Sue	$200.00
8/21/2012	3453	Joe	$333.99

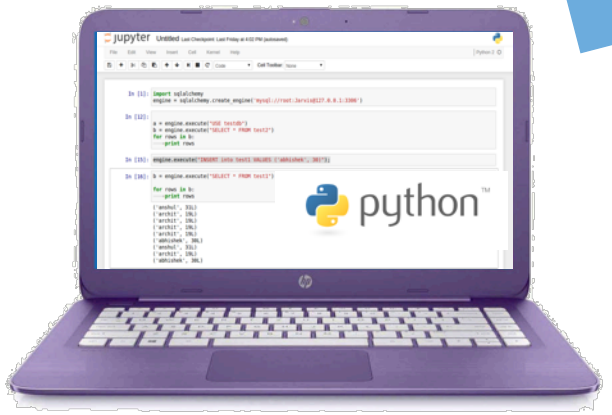
Interacting with a DBMS

DBMS Server

HTTP

Web Applications

Python Analysis



Visualization

Often many systems will connect to a DBMS **concurrently**.

Data in the Organization

A little bit of buzzword bingo!

Multidimensional Data Model

Sales **Fact Table**

pid	timeid	locid	sales
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
12	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35
11	2	2	22
11	3	2	10
12	1	2	26

Locations

locid	city	state	country
1	Omaha	Nebraska	USA
2	Seoul		Korea
5	Richmond	Virginia	USA

Products

pid	pname	category	price
11	Corn	Food	25
12	Galaxy 1	Phones	18
13	Peanuts	Food	2

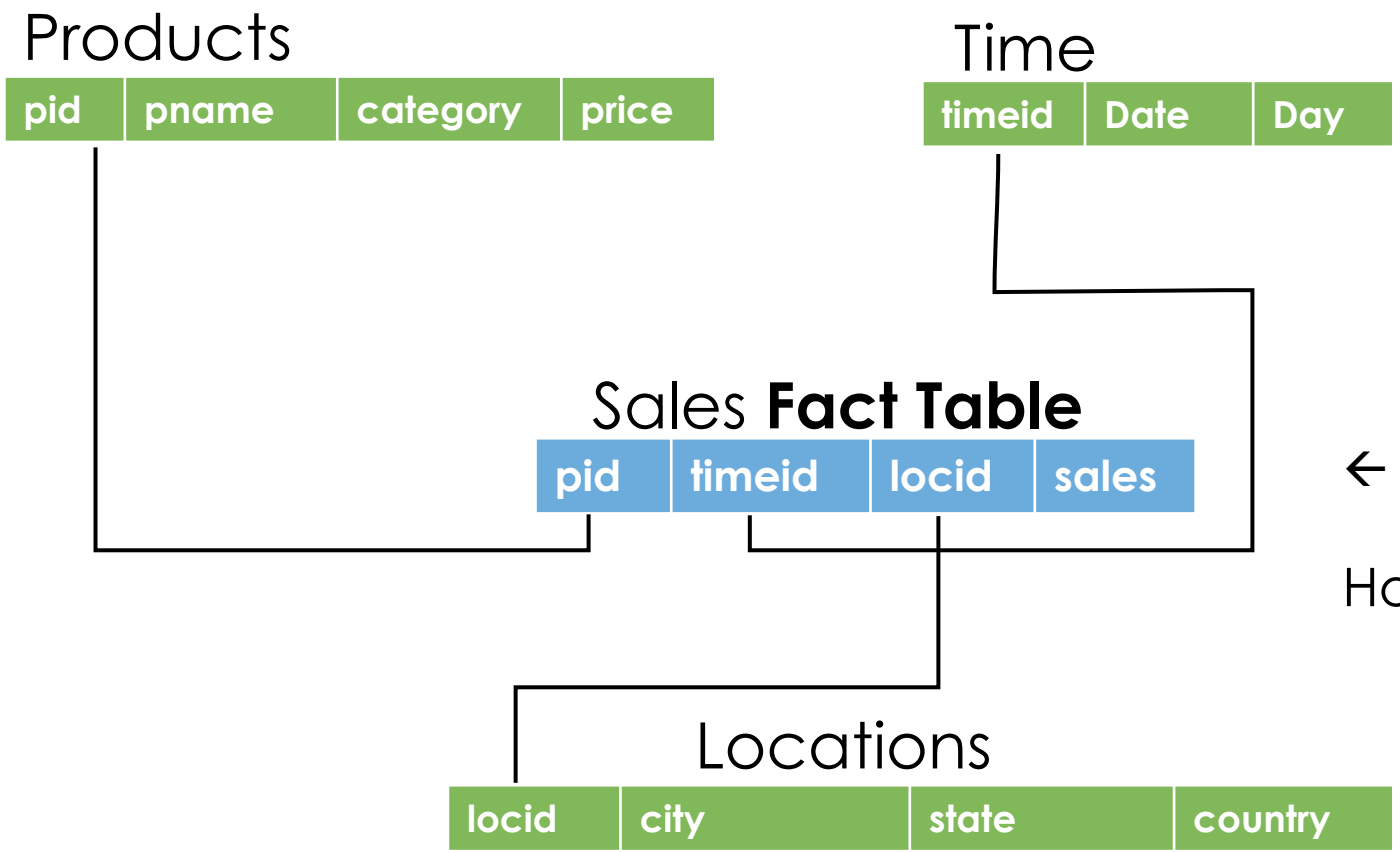
Time

timeid	Date	Day
1	3/30/16	Wed.
2	3/31/16	Thu.
3	4/1/16	Fri.

Dimension Tables

- Normalized Representation
- Fact Table
  - minimizes redundant info.
  - Reduces data errors
- Dimensions
  - easy to manage and summarize
  - Rename: Galaxy1 → Phablet

Connections between table



← This looks like a star …

How do we do analysis?

Joins!!!!!

Joins!

Bringing tables together for decades.

Join Queries

```
SELECT [DISTINCT] <column expression list>
  FROM <table1 [AS t1], ... , tableN [AS tn]>
[WHERE <predicate>]
[GROUP BY <column list>
 [HAVING <predicate>] ]
[ORDER BY <column list>];
```

1. FROM : compute ***outer product*** of tables.
2. WHERE : Check conditions, discard tuples that fail.
3. SELECT : Specify desired fields in output.

- Note: likely a terribly inefficient strategy!
  - Query optimizer will find more efficient plans.

The Outer-Product (×)

R1 × S1: *Each row of **R1** paired with each row of **S1***

R1:

sid	bid	day
22	101	10/10/96
58	103	11/12/96

× **S1:**

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

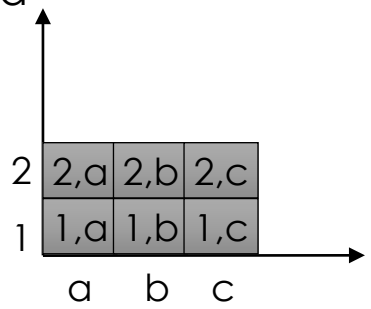
=

R1 × S1

sid	bid	day	sid	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
22	101	10/10/96	31	lubber	8	55.5
22	101	10/10/96	58	rusty	10	35.0
58	103	11/12/96	22	dustin	7	45.0
58	103	11/12/96	31	lubber	8	55.5
58	103	11/12/96	58	rusty	10	35.0

How many rows in the result? | |R1| * |R2| |

Sometimes also called
Cartesian Product:



Return Sailors (S) and the dates of their Reservations (R)

```
SELECT S.sname, R.day
FROM Reserves AS R, Sailors AS S
WHERE S.sid = R.sid
```

Symbol for join (Rel. Alg.)

$R1 \bowtie S1$

R:

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

sid	bid	day	sid	sname	rating	age
22	101	10/10/96	22	dustin	7	45.0
~~22~~	~~101~~	~~10/10/96~~	~~31~~	~~lubber~~	~~8~~	~~55.5~~
~~22~~	~~101~~	~~10/10/96~~	~~58~~	~~rusty~~	~~10~~	~~35.0~~
~~58~~	~~103~~	~~11/12/96~~	~~22~~	~~dustin~~	~~7~~	~~45.0~~
~~58~~	~~103~~	~~11/12/96~~	~~31~~	~~lubber~~	~~8~~	~~55.5~~
58	103	11/12/96	58	rusty	10	35.0

http://sqlfiddle.com/#!17/53815/1140/0

About Range Variables

- Needed when ambiguity could arise.
  - e.g., same table used multiple times in FROM ("self-join")

```
SELECT  *
FROM    Sailors AS S1, Sailors AS S2
WHERE   S1.age > S2.age
```

S1:

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1				S2			
sid	sname	rating	age	sid	sname	rating	age
22	dustin	7	45.0	58	rusty	10	35.0
31	lubber	8	55.5	22	dustin	7	45.0
31	lubber	8	55.5	58	rusty	10	35.0

http://sqlfiddle.com/#!17/53815/4

Join Variants

```
SELECT (column_list)
FROM  table_name
  [INNER | {LEFT | RIGHT | FULL } {OUTER}] JOIN table_name
    ON qualification_list
WHERE …
```

- INNER is default
- Inner join is akin to what we have seen so far.
- The term Outer is optional for Left, Right, and Full joins
  - For example: LEFT OUTER = LEFT

Inner/Natural Joins

```
SELECT s.sid, s.sname, r.bid
    FROM Sailors s, Reserves r
   WHERE s.sid = r.sid
     AND s.age > 20;

SELECT s.sid, s.sname, r.bid
    FROM Sailors s INNER JOIN Reserves r
      ON s.sid = r.sid
   WHERE s.age > 20;

  SELECT s.sid, s.sname, r.bid
    FROM Sailors s NATURAL JOIN Reserves r
   WHERE s.age > 20;
```

all 3 are equivalent!

Sailors

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

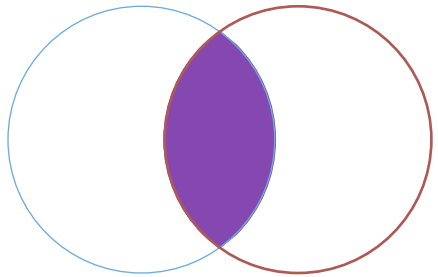
Boats

bid	bname	color
101	Nina	red
102	Pinta	blue
103	Santa Maria	red

Reserves

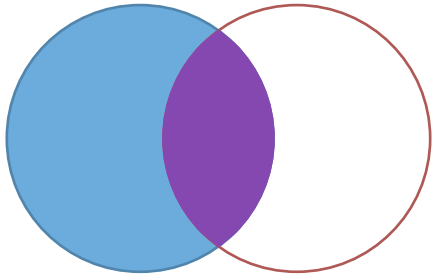
sid	**bid**	**day**
1	102	9/12
2	102	9/13

- "NATURAL" means equi-join for each pair of attributes with the same name



http://sqlfiddle.com/#!17/4215a/10

Left Join



Returns all matched rows, and *preserves* all unmatched rows from the table on the left of the join clause

(use nulls in fields of non-matching tuples)

```
SELECT s.sid, s.sname, r.bid
FROM Sailors2 s LEFT JOIN Reserves2 r
ON s.sid = r.sid;
```

Returns all sailors & bid for boat in any of their reservations

Note: If there is a sailor without a boat reservation then the sailor is matched with the NULL bid.

```
SELECT s.sid, s.sname, r.bid
FROM Sailors2 s LEFT JOIN Reserves2 r
ON s.sid = r.sid;
```

Sailors2

sid	sname	rating	age
22	Dustin	7	45
31	Lubber	8	55.5
95	Bob	3	63.5

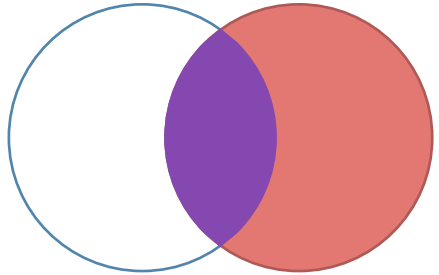
Reserves2

sid	bid	day
22	101	1996-10-10
95	103	1996-11-12

sid	sname	bid
22	Dustin	101
95	Bob	103
31	Lubber	**(null)**

http://sqlfiddle.com/#!17/54a88/2

Right Join



- Right join returns all matched rows, **and *preserves* all unmatched rows from the table on the right** of the join clause

```
SELECT r.sid, b.bid, b.bname
FROM Reserves2 r RIGHT JOIN Boats2 b
ON r.bid = b.bid;
```

- Returns all boats & information on which ones are reserved.
- No match for b.bid? r.sid IS NULL!

```
SELECT r.sid, b.bid, b.bname
FROM Reserves2 r RIGHT JOIN Boats2 b
ON r.bid = b.bid;
```

Reserves2

sid	bid	day
22	101	1996-10-10
95	103	1996-11-12

Boats2

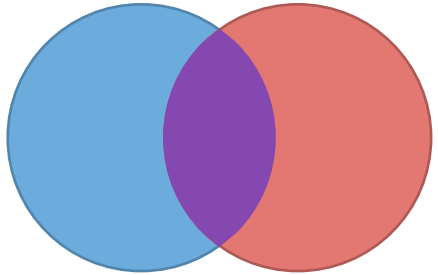
bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Result:

sid	bid	bname
22	101	Interlake
95	103	Clipper
(null)	104	Marine
(null)	102	Interlake

http://sqlfiddle.com/#!17/a7b2f/1

Full Outer Join



- Full Outer Join returns all (matched or unmatched) rows from the tables on both sides of the join clause

```
SELECT r.sid, b.bid, b.bname
FROM Reserves2 r FULL JOIN Boats2 b
ON r.bid = b.bid
```

- If no boat for a sailor? → b.bid IS NULL AND b.bname IS NULL!
- If no sailor for a boat? → r.sid IS NULL!

```
SELECT r.sid, b.bid, b.bname
FROM Reserves3 r FULL JOIN Boats2 b
ON r.bid = b.bid
```

Reserves3

sid	bid	day
22	101	1996-10-10
95	103	1996-11-12
38	42	2010-08-21

Boats2

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Result:

sid	bid	bname
22	101	Interlake
95	103	Clipper
38	**(null)**	**(null)**
(null)	104	Marine
(null)	102	Interlake

http://sqlfiddle.com/#!17/e1f3a/3/0

Brief Detour: Null Values

- Field values are sometimes unknown
  - SQL provides a special value NULL for such situations.
  - Every data type can be NULL
- The presence of null complicates many issues. E.g.:
  - Selection predicates (WHERE)
  - Aggregation
- But NULLs are common after outer joins

NULL in the WHERE clause

- Consider a tuple where `rating` IS NULL.

```
INSERT INTO sailors VALUES
 (11, 'Jack Sparrow', NULL, 35);
```

- If we run the following query

```
SELECT * FROM sailors
WHERE rating > 8;
```

- Jack Sparrow will not be included in the output.

http://sqlfiddle.com/#!17/36ca9/2

NULL in comparators

What entries are in the output of all these queries?

```
SELECT rating = NULL FROM sailors;

SELECT rating < NULL FROM sailors;

SELECT rating >= NULL FROM sailors;
```

All of these queries evaluate to null!

Even this one!

```
SELECT * FROM sailors WHERE rating = NULL;
```

Rule: (x *op* NULL) evaluates to … NULL!

http://sqlfiddle.com/#!17/f35aa/6

Explicit NULL Checks

- To check if a value is NULL you must use explicit NULL checks

```
SELECT * FROM sailors WHERE rating IS NULL;
```

```
SELECT * FROM sailors WHERE rating IS NOT NULL;
```

http://sqlfiddle.com/#!17/f35aa/4

NULL in Boolean Logic

Three-valued logic:

Not

T	F
F	T
N	N

And

	T	F	N
T	T	F	N
F	F	F	F
N	N	F	N

Or

	T	F	N
T	T	T	T
F	T	F	N
N	T	N	N

```
SELECT * FROM sailors WHERE rating > 8 AND TRUE;

SELECT * FROM sailors WHERE rating > 8 OR TRUE;

SELECT * FROM sailors WHERE NOT (rating > 8);
```

http://sqlfiddle.com/#!17/f35aa/2

NULL and Aggregation

```
SELECT count(rating) FROM sailors;
```
- 4

```
SELECT sum(rating) FROM sailors;
```
- 27

```
SELECT avg(rating) FROM sailors;
```
- ??

```
SELECT count(*) FROM sailors;
```
- ??

sid	sname	rating	age
1	Popeye	10	22
2	OliveOyl	11	39
3	Garfield	1	27
4	Bob	5	19
11	Jack Sparrow	(null)	35

http://bit.ly/ds100-sp18-null

http://sqlfiddle.com/#!17/f35aa/7

NULL and Aggregation

```
SELECT count(rating) FROM sailors;
```
- 4

```
SELECT sum(rating) FROM sailors;
```
- 27

```
SELECT avg(rating) FROM sailors;
```
- (10+11+1+5) / ***4*** = 6.75

```
SELECT count(*) FROM sailors;
```
- 5

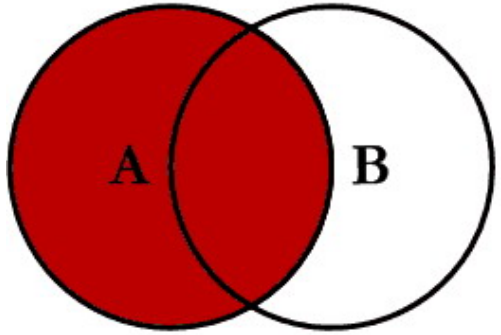
sid	sname	rating	age
1	Popeye	10	22
2	OliveOyl	11	39
3	Garfield	1	27
4	Bob	5	19
11	Jack Sparrow	(null)	35

http://sqlfiddle.com/#!17/f35aa/7

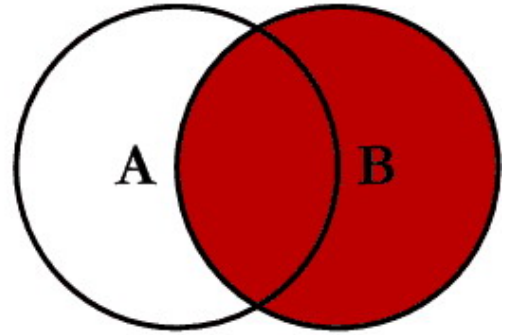
NULLs: Summary

- NULL *op* NULL is NULL
- WHERE NULL: do not send to output
- Boolean connectives: 3-valued logic
- Aggregates ignore NULL-valued inputs

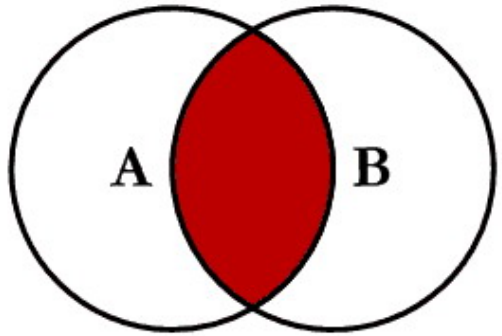
SQL JOINS



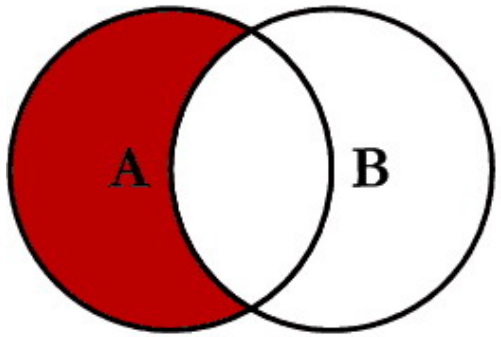

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



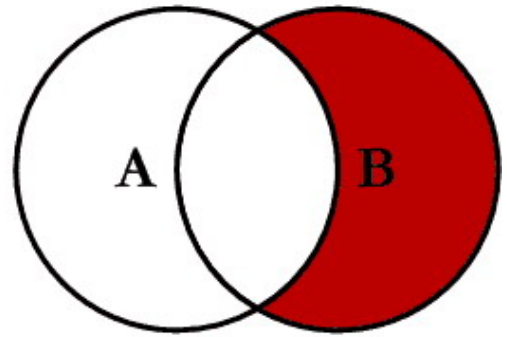

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



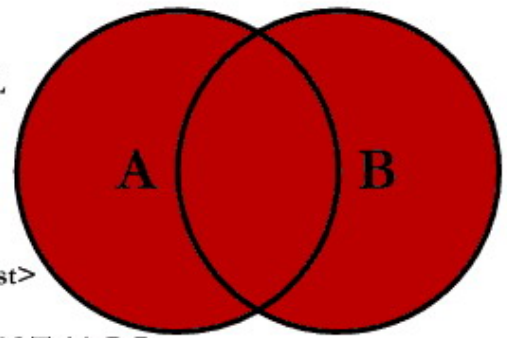

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



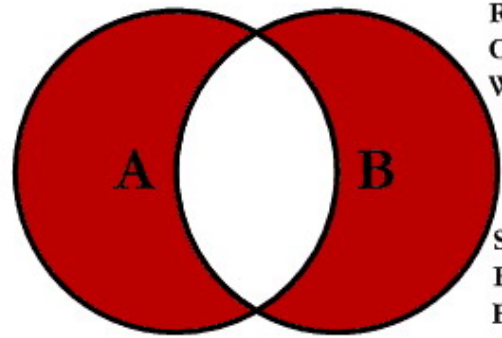

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

SQL Query Demo

Returning to Notebook