## CS 61: Database Systems

### Joins

Adapted from Silberschatz, Korth, and Sundarshan unless otherwise noted

## Agenda

## 📫 1. Joins

- 2. nyc\_inspections schema
- 3. Joins on nyc\_inspections
- 4. Conditional evaluation

# With JOIN store data one time in multiple tables; combine to form larger table

#### instructor table

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000

#### teaches table

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-315	1	Spring	2018

## Teaches table lists courses and sections that are taught by instructors

Book's schema also has additional table for courses (not shown)

# With JOIN store data one time in multiple tables; combine to form larger table

#### instructor table

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
			` <b>T:</b> * + *

teaches table

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-315	1	Spring	2018

#### SELECT i.\*, t.\*

**FROM** *instructor i, teaches t -- cartesian product* WHERE *i.ID* = *t.ID; -- filter cartesian product* 

instructor.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017

4

# If we kept data in one large table, there are several anomalies that can occur

Problems keeping one large table with many attributes

instructor.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year	Anomalies if
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017	keep one large
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018	table instead of
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017	multiple tables
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018	a lacort
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018	• insert
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017	Update
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018	Delete
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018	
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018	
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017	
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018	
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017	
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017	
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018	

## Insert anomalies

### Problems keeping one large table with many attributes

r.ID   na	ame	dept_name	salary	teaches.ID	course_id	sec_id	semester	year	Insert anomalies
Sr	rinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017	• If hire a new
Sr	rinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018	instructor.
Sr	rinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017	they do not
W	<sup>v</sup> u	Finance	90000	12121	FIN-201	1	Spring	2018	chow up in
M	lozart	Music	40000	15151	MU-199	1	Spring	2018	snow up in
e Ei	instein	Physics	95000	22222	PHY-101	1	Fall	2017	database
6 E1	l Said	History	60000	32343	HIS-351	1	Spring	2018	until they
i Ka	atz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018	teach a
Ka	atz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018	course
	rick	Biology	72000	76766	BIO-101	1	Summer	2017	
C	rick	Biology	72000	76766	<b>BIO-301</b>	1	Summer	2018	
Br	randt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017	
Br	randt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017	
Br	randt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018	

## Update anomalies

#### Problems keeping one large table with many attributes

pdate	U	year	semester	sec_id	course_id	teaches.ID	salary	dept_name	name	instructor.ID
nomalies	ar	2017	Fall	1	CS-101	10101	65000	Comp. Sci.	Srinivasan	10101
If instructor	•	2018	Spring	1	CS-315	10101	65000	Comp. Sci.	Srinivasan	10101
gots a raiso		2017	Fall	1	CS-347	10101	65000	Comp. Sci	Srinivasan	10101
gets a raise,		2018	Spring	1	FIN-201	12121	90000	Finance	Wu	12121
must update		2018	Spring	1	MU-199	15151	40000	Music	Mozart	15151
salary in <u>all</u>		2017	Fall	1	PHY-101	22222	95000	Physics	Einstein	22222
rows for that		2018	Spring	1	HIS-351	32343	60000	History	El Said	32343
instructor		2018	Spring	1	CS-101	45565	75000	Comp. Sci.	Katz	45565
Can lead to	•	2018	Spring	1	CS-319	45565	75000	Comp. Sci.	Katz	45565
inconsistent		2017	Summer	1	BIO-101	76766	72000	Biology	Crick	76766
literi		2018	Summer	1	BIO-301	76766	72000	Biology	Crick	76766
data!		2017	Spring	1	CS-190	83821	92000	Comp. Sci.	Brandt	83821
What is the	•	2017	Spring	2	CS-190	83821	92000	Comp. Sci.	Brandt	83821
instructor's		2018	Spring	2	CS-319	83821	92000	Comp. Sci.	Brandt	83821
true salary?	I	. 1	1 -		1	1		1 –	1	1

## Delete anomalies

### Problems keeping one large table with many attributes

elete	year	semester	sec_id	course_id	teaches.ID	salary	dept_name	name	instructor.ID
omalies	2017	Fall	1	CS-101	10101	65000	Comp. Sci.	Srinivasan	10101
If course is	2018	Spring	1	CS-315	10101	65000	Comp. Sci.	Srinivasan	10101
only taught	2017	Fall	1	CS-347	10101	65000	Comp. Sci.	Srinivasan	10101
one time and	2018	Spring	1	FIN-201	12121	90000	Finance	Wu	12121
one time and	2018	Spring	1	MU-199	15151	40000	Music	Mozart	15151
instructor	2017	Fall	1	PHY-101	22222	95000	Physics	Einstein	22222
taught only	2018	Spring	1	HIS-351	32343	60000	History	El Said	32343
one course, if	2018	Spring	1	CS-101	45565	75000	Comp. Sci.	Katz	45565
delete	2018	Spring	1	CS-319	45565	75000	Comp. Sci.	Katz	45565
	2017	Summer	1	BIO-101	76766	72000	Biology	Crick	76766
instructor	2018	Summer	1	BIO-301	76766	72000	Biology	Crick	76766
Instructor	2017	Spring	1	CS-190	83821	92000	Comp. Sci.	Brandt	83821
too!	2017	Spring	2	CS-190	83821	92000	Comp. Sci.	Brandt	83821
If delete	2018	Spring	2	CS-319	83821	92000	Comp. Sci.	Brandt	83821
DUV 101							1		

If delete PHY-101, loose Einstein!<sup>8</sup>

# We can avoid these anomalies by keeping data in multiple tables

#### instructor table

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000

#### teaches table

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-315	1	Spring	2018

#### Better to store data in multiple tables

**Insert:** new instructor can be added to database without teaching a class **Update:** instructor gets a raise, only update one row in instructor table **Delete:** can delete course, instructor will still exist in instructor table

## Agenda

### 1. Joins

## 2. nyc\_inspections schema

### 3. Joins on nyc\_inspections

## 4. Conditional evaluation

# The old single table is now multiple related tables in nyc\_inspections

#### use nyc\_inspections;



# The old single table is now multiple related tables in nyc\_inspections

**One entry for** 

### use nyc\_inspections;

		each resta	urant	InspectionViolations V	
🔲 Restaurants 🔹 🔻	One entry	inspection	fk_InspectionViolation	s InspectionID INT(11)	
RestaurantID INT(11)	for each	Inspections	▼	ViolationCode VARCHAR(10)	
Hestaurantivame VARCHAR(100)     Boro VARCHAR(20)     D    it is a VARCHAR(20)	restaurant	InspectionID INT(1 RestaurantID INT(11)	1)	¥	
Building VARCHAR(20)     Street VARCHAR(100)     ZipCode INT(11)     Phone BIGINT(20)	fk_Inspections_Restaurants	InspectionTypeID II  InspectionDate DA  ActionID INT(11)	NT(11) TE	fk_ViolationCodes	
♦ Latitude DOUBLE		<ul> <li>Score INT(11)</li> <li>Grade VARCHAR(4)</li> </ul>	4)	ViolationCodes V	
<ul> <li>Longitude DOUBLE</li> <li>CuisineID INT(11)</li> <li>InspectionCount INT(11)</li> </ul>	CuisineID FK	GradeDate DATE	•	<ul> <li>ViolationCode VARCHAR(10)</li> <li>ViolationCodeDescription VARCHAR(400</li> <li>CriticalFlag VARCHAR(4)</li> </ul>	
♦ InspectionAvgScore DOUBLE Indexes	means	<u>¥</u> ¥		Indexes	
fk_Restaurants_Cuisine	must be in <sup>fk_Inspections_</sup>	Actions	fk_Inspections_InspectionTy Action and Inspect	<sup>pes</sup> tion	
		Type muse	InspectionTypes	<b>v</b>	
Cuisine 🔻	ActionID INT(11)		InspectionTypeID INT(11)		
CuisineID INT(11) CuisineDescription VARCHAR(100) Indexes	ActionDescription VAR	CHAR(150)	InspectionTypeDescription VARCH ndexes	HAR(100)	

# The old single table is now multiple related tables in nyc\_inspections

**One entry for** 

**One inspection may lead** 

### use nyc\_inspections;



## Agenda

### 1. Joins

2. nyc\_inspections schema

## 3. Joins on nyc\_inspections

## 4. Conditional evaluation

## Recommended way to join is not in the WHERE clause, but with JOIN command

#### JOIN

Thus far we have joined relations by matching in the WHERE clause, but JOIN is preferred

```
Format:
                                                         Joins store results in a temporary
SELECT A_1, A_2, \dots A_n
                                                         table in the database
FROM r_1 {type} JOIN r_2 {type} JOIN .. {type} JOIN r_n
where P :
{type} = [NATURAL | INNER | OUTER [LEFT RIGHT FULL]]. If type not specified, INNER
```

Example: count how many time each bakery has been inspected Our previous way (old style join): **SELECT** r.RestaurantID, RestaurantName, count(\*) Join is done in the WHERE clause **FROM** Restaurants r, Inspections i Must specify which table attribute from **WHERE** r.RestaurantID = i.RestaurantID Both do the same thing! **AND** r.CuisineID = 5 - - look up bakery ID in Cuisine table **Recommended way: GROUP BY** RestaurantID; Join is done in the FROM clause

#### **Preferred way:**

using JOIN; implicitly an INNER join **SELECT** r.RestaurantID, RestaurantName, count(\*) **FROM** Restaurants r **JOIN** Inspections i **ON** r.RestaurantID = i.RestaurantID WHERE r.CuisineID = 5 Can still use WHERE to limit results **GROUP BY** RestaurantID;

# Recommended way to join is not in the WHERE clause, but with JOIN command

### JOIN

Thus far we have joined relations by matching in the WHERE clause, but JOIN is preferred

```
Format:

SELECT A_1, A_2, ... A_n

FROM r_1 {type} JOIN r_2 {type} JOIN .. {type} JOIN r_n

where P;

{type} = [NATURAL | INNER | OUTER [LEFT RIGHT FULL]]. If type not specified, INNER
```

Example: count how many time each bakery has been inspected SELECT r.RestaurantID, RestaurantName, count(\*) FROM Restaurants r, Inspections i WHERE r.RestaurantID = i.RestaurantID AND r.CuisineID = 5 - - look up bakery ID in Cuisine table GROUP BY RestaurantID; No du

#### Preferred way:

```
SELECT RestaurantID, RestaurantName, count(*)
FROM Restaurants r NATURAL JOIN Inspections i
WHERE CuisineID = 5
GROUP BY RestaurantID;
```

Could also do a NATURAL JOIN

- No need to tell which attributes to match for join (uses attributes with same name to join)
- No duplicate attributes in result

I prefer using JOIN ON (last slide) I know for sure what attributes are used for join (or at least use JOIN USING)

# INNER join only returns rows if comparison attribute is in both tables

### **INNER JOIN**

TableA

ID	<b>A</b> <sub>1</sub>
1	m
2	n
4	0

Rows returned with attributes from both tables if match between values in comparison columns

SELECT \* FROM TableA a JOIN TableB b ON a.ID=b.ID

#### TableB

ID	A <sub>2</sub>
2	р
3	q
5	r

ID of 2 is in both tables so it is returned, others are not

Result has attributes from both tables (A<sub>1</sub> and A<sub>2</sub>) and duplicate ID

Rows 1 and 4 from TableA and rows 3 and 5 from TableB not returned





#### **Result (temp table)**

ID	A <sub>1</sub>	ID	<b>A</b> <sub>2</sub>
2	n	2	р

NATURAL JOIN omits duplicate attributes (could also pick in SELECT)

ID	A <sub>1</sub>	A <sub>2</sub>	
2	n	р	1

# LEFT OUTER JOIN returns all rows from the left table



## RIGHT OUTER JOIN returns all rows from the right table



1 and 4 in TableA not returned because those keys not in TableB

# FULL OUTER JOIN returns all rows from both tables



Adapted from: https://www.w3resource.com/slides/sql-joins-slide-presentation.php

## Practice

### Use nyc\_inspections

You've opened a new fruit/vegetable restaurant in Manhattan called 'Tim's Tasty Treats' (keep the apostrophe in the name!):

- Insert a new row in your Restaurants table for this restaurant
  - $\circ~$  Set the RestaurantID to 1111
  - Set the CuisineID to the proper value for a fruits/vegetables restaurant
  - You can set address, phone, lat/long to NULL (or another value)
- See how many other fruit/vegetable restaurants there are (your competition)
- Count how many times each fruit/vegetable restaurant has been inspected, include:
  - RestaurantID
  - RestaurantName
  - Count of inspections
  - Make sure Tim's restaurant is on the list and shows zero inspections! (note: this is tricky!)

## Agenda

## 1. Joins

- 2. nyc\_inspections schema
- 3. Joins on nyc\_inspections
- **4**. Conditional evaluation

# IF allows conditional evaluation, giving one of two values

IF command

Format:

If expr results in true or not null, return true value else return false value

SELECT IF (expr, true value, false value) AS name

Like a lambda expression in many programming languages

**Practice**: Some restaurant inspection grades are NULL If Grade is null, return 'N/A' else return Grade

Note: if attribute could come from more

than relation, identify which one

**SELECT** i.RestaurantID, RestaurantName, InspectionDate, Score,

IF(Grade IS NULL,'N/A',Grade) AS Grade, GradeDate FROM Inspections i JOIN Restaurants r on i.RestaurantID = r.RestaurantID WHERE r.RestaurantID = 30075445

**ORDER BY** InspectionDate;

use nyc inspections;

See day5.sql

# COALESCE returns the first non-null value in a list of arguments

### COALESCE

Format: SELECT COALESCE(value<sub>1</sub>, value<sub>2</sub>, ... value<sub>n</sub>) AS name

value<sub>x</sub> can be an attribute or literal If value<sub>1</sub> is NULL, SQL tries value<sub>2</sub>, ... Returns first non-null value If all values are NULL, returns NULL

### Given a table of customer orders

- Try using State as Location
- If State is NULL, try Country
- If Country is NULL, use 'N/A'

Example:

**SELECT** OrderID, **COALESCE**(State, Country, 'N/A') **AS** Location **FROM** Orders

# CASE provides if-then-else logic in one of two formats

CASE	Value of attribute	No attribute here,
SELECT CASE attribute	S S	ELECT CASE 🖌 it is in expression
WHEN value1 THEN re	esult1 OR	WHEN expr1 THEN result1
WHEN value2 THEN re	esult2	WHEN expr2 THEN result2
[ <b>ELSE</b> else result]		[ <b>ELSE</b> else result]
END AS AttributeName	E E	ND AS AttributeName
Don't fo	rget END!	
Practice: Categorize rest	aurants as having	infrequent (<10), moderate (10-15),
or frequent (>10) inspec	tions See day5	.sql
SELECT i.RestaurantID, Restau	irantName, count(*) A	S `Total Inspections`,
CASE		
<b>WHEN</b> count(*) < 1	0 <b>THEN</b> 'Infrequent ins	pections '
WHEN count(*) BET	IWEEN 10 AND 15 THI	<b>N</b> 'Moderate inspections'
ELSE 'Frequent insp	ections'	
END AS Frequency	Can use BETV	VEEN x and y; or
FROM Inspections i, Restaura	<sup>nts r</sup> Could have u	sed count(*) >10 AND count(*) > 15
WHERE i.RestaurantID = r.Res	taurantID	25
GROUP BY i.RestaurantID;		20

## Practice

### **CASE practice**

We can combine CASE with aggregate operations. For each restaurant provide the number of times an inspection resulted in each possible letter Grade

- First determine the grades possible (or at least those than have been given)
- Then on one line provide the number of times each restaurant received a distinct grade (e.g. RestaurantID, RestaurantName, A, B, C..., NULL, Total)
- Output should look like this:

RestaurantID	RestaurantName	А	В	С	G	Ν	Ρ	Z	NULL	Total
30075445	MORRIS PARK BAKE SHOP	4	0	0	0	0	0	0	2	6
30112340	WENDY'S	5	0	0	0	0	0	0	4	9
30191841	DJ REYNOLDS PUB AND RESTAU	3	0	0	0	0	0	0	1	4
40356018	RIVIERA CATERERS	3	0	0	0	0	0	0	1	4
40356151	BRUNOS ON THE BOULEVARD	2	0	0	0	0	0	0	2	4
40356483	WILKEN'S FINE FOOD	3	0	0	0	0	0	0	0	3
40356731	TASTE THE TROPICS ICE CREAM	3	0	0	0	0	0	0	0	3
40357217	WILD ASIA	3	0	0	0	0	0	0	0	3
40359480	<b>1 EAST 66TH STREET KITCHEN</b>	3	0	0	0	0	0	0	0	3