Some Comments about RA

- Natural join

$$Result \leftarrow Employee * Department$$

- AS and renaming

$$Result \leftarrow (\pi_{ESSN\ AS\ MGRSSN,FName,LName} Employee) * Department$$

$$R1(MSGRSSN, FName, LName) \leftarrow \pi_{ESSN,FName,LName} Employee)$$
$$Result \leftarrow R1 * Department$$

4. **Works_ON / Project.PNUMBER**

Why PK Must Be Minimal

Student(SID, Name, GPA)

- (SID)
  - (01345, John, 67)
  - (22234, Kate, 88)
  - (01345, Jane, 78)
  - … …
- (SID, Name)
  - (01345, John,67)
  - (22234, Kate, 88)
  - (01345, Jane, 78)
  - … …

What is Wrong?

- Find the name of the employee with the highest salary

$$Result \leftarrow \pi_{FName,LName}(\mathcal{G}_{max(Salary)}\ (Employee))$$

$$R1 \leftarrow \mathcal{G}_{max(Salary)\ AS\ MS}\ (Employee)$$
$$Result \leftarrow \pi_{FName,LName}\ (\sigma_{Salary=MS}(R1))$$

Review of Last Lecture

- SQL basics
  - IBM, pronunciation, features
- SQL Data Manipulation:

  SELECT

  FROM

  WHERE
- Which one can be skipped?

Today's Plan

- Extension to Select
  - DISTINCT, AS, expression, aggregated
- Extension to From
  - Nested query
  - Join

In-class Exercise #1

- Find the name of the supervisor of John Smith' supervisor.

```
SELECT   SS.FName, SS.LName
FROM
WHERE
```

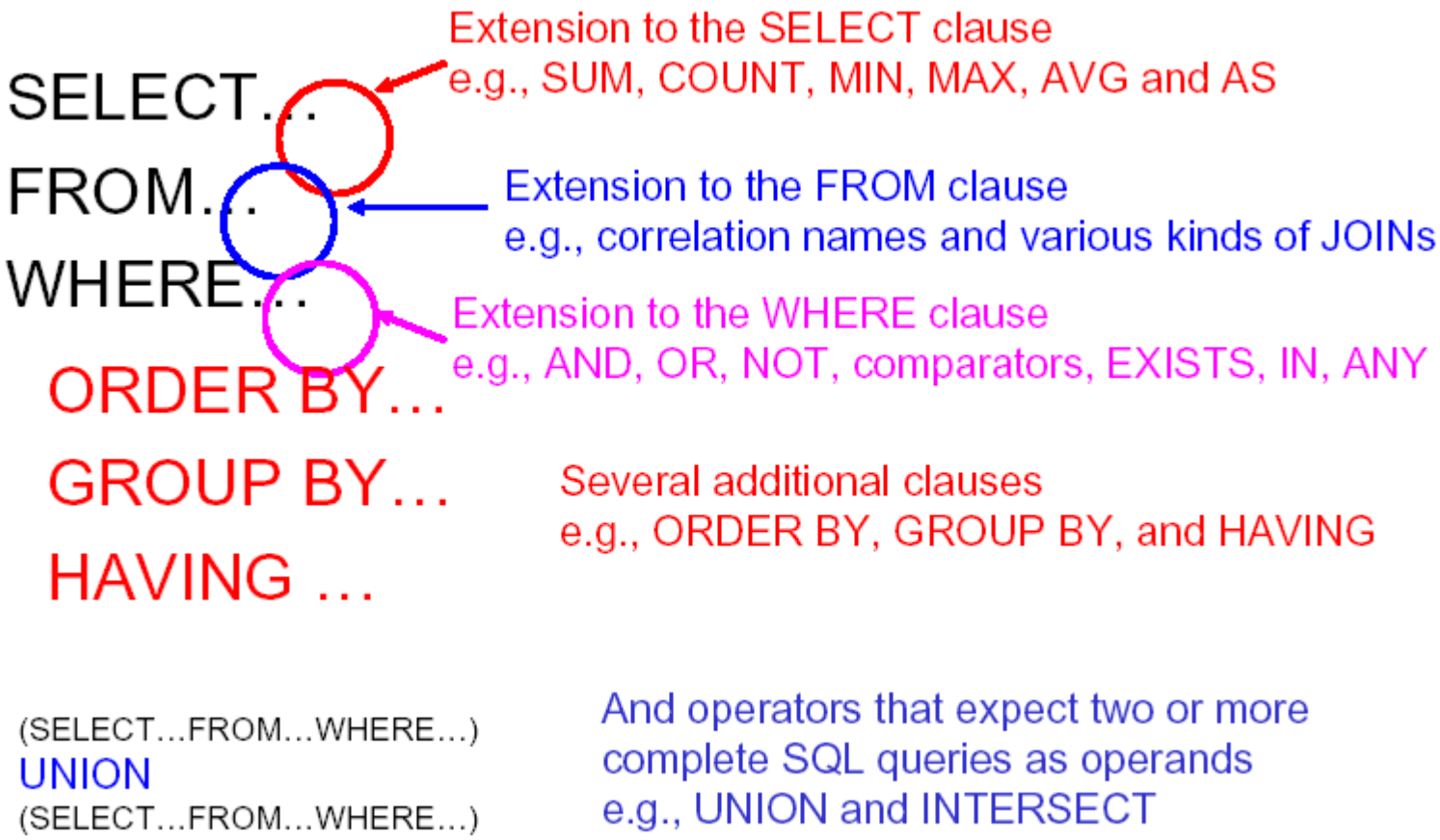
E

Fr	Bda
J	1965-0
F	1955-1
A	1968-0
J	1941-0
R	1962-0
J	1972-0
A	1969-0
J	1937-1

S

Fr									
J									
A									
J									
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

SQL Extension



Extension to SELECT

```
SELECT    DISTINCT FName
FROM      Employee;

SELECT    SSN AS Sunshine_List
FROM      Employee
WHERE     Salary >= 100,000;

SELECT    SSN, (Salary * 1.05) AS Raise_Projection
FROM      Employee;

SELECT    MIN(Salary), MAX(Salary), AVG(Salary)
FROM      Employee;
```

Distinct

Consider the following two queries:

```
SELECT    DISTINCT    FName
FROM                  Employee;

SELECT                FName
FROM                  Employee;
```

The first query eliminates duplicate rows from the answer.
The second query does not eliminate duplicate rows.

- Relational DBMS are said to operate on *multisets (bags)* though the relational model is based on sets!

The query writer gets to choose whether duplicates are eliminated!

Distinct: Exercise

- What is the output for the following query?

```
SELECT    DISTINCT    LName, Salary
FROM                  Emp
```

- What is the output for the following query?

```
SELECT    DISTINCT    LName
FROM                  Emp
```

Emp

EID	FNAME	LNAME	Salary
1	John	Liu	80,000
2	John	King	90,000
3	Kate	King	80,000

AS

- The following two queries are equivalent except for the attribute name used in query answer: “Sunshine_List” is used in the second query.

```
SELECT    SSN
FROM      Employee
WHERE     Salary >= 100,000;

SELECT    SSN AS Sunshine_List
FROM      Employee
WHERE     Salary >= 100,000;
```

- We can assign new names to the attributes (columns) that appear in a query result with the AS <new attribute name> clause.

Expression

We can augment attributes in the SELECT clause with formulas

```
SELECT    SSN, (Salary * 1.05) AS Raise_Projection
FROM      Employee;
```

Use the AS operator to name the result attributes for formulas. When the AS operator is missing, the DBMS may use a proprietary default name (exprX), or no name at all.

What happens when the formula doesn't work with the attribute's domain (e.g. "LastName+1")?

Aggregation

We can use aggregate operators in the SELECT clause: COUNT, SUM, MIN, MAX, and AVG

```
SELECT    MIN(Salary), MAX(Salary), AVG(Salary)
FROM      Employee;

SELECT    MIN(Salary), MAX(Salary), AVG(Salary)
FROM      Employee
WHERE     BDate < 1960
```

If one aggregate operator appears in the SELECT clause, then ALL of the entries in the select clause must be an aggregate operator (unless the query includes a GROUP BY clause (covered later)).'

Example: Aggregation

- From Employee table, find the average salary in department #5

```
SELECT    AVG(Salary)
FROM      Employee
WHERE     DNO = 5;
```

```
SELECT    AVG(Salary) AS Average-salary
FROM      Employee
WHERE     DNO = 5;
```

In-class Exercise #2

- From Employee table, find the minimal salary and maximal salary in the Research department

DISTINCT Inside COUNT()

- What is the difference between these two queries?

```
SELECT    COUNT(FName)
FROM      Emp

SELECT    COUNT(DISTINCT FName)
FROM      Emp
```

Emp

EID	FNAME	LNAME	Salary
1	John	Liu	80,000
2	John	King	90,000
3	Kate	King	80,000

DISTINCT Inside Aggregation

What is the implication of using DISTINCT when computing the SUM or AVG of an attribute?

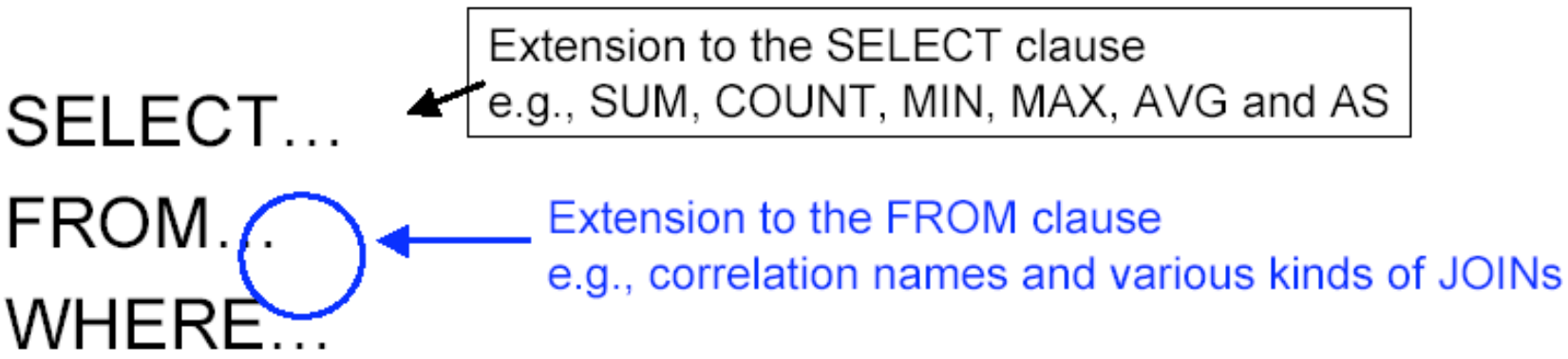
What is the implication of using DISTINCT when computing the MIN or MAX of an attribute?

In-class Exercise #3

1. Find the number of supervisors in the company.

2. Project the total cost if the company gives a 5% raise to all managers.

Extension to From



Nested SQL

- The FROM clause takes a relation, but results from SOL queries are themselves relations, so we can use them in the FROM clause, too!

Customer (Number, Name, Address, CRating,
CAmount, CBalance,SalespersonNum)

```
SELECT    (N.CRating+1) AS ClncrRating
FROM      (SELECT    *
           FROM      Customer
           WHERE     CRating = 0) AS N
WHERE     N.CBalance = 0;
```

Nested Query: Example

- List the names of employees who have the lowest salary.

```
SELECT    FName, LName, min(Salary)
FROM      Employee;
```

```
SELECT    FName, LName, Salary = min(Salary)
FROM      Employee;
```

```
SELECT    FName, LName
FROM      Employee
WHERE     Salary > min(Salary);
```

Nested Query: Example

- List the names of employees who have the lowest salary.

```
SELECT    E.FName, E.LName
FROM      Employee E, M
WHERE
```

Employee

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

M

MinS
25000

Nested Query: Example

- List the names of employees who have the lowest salary
  - .Find the lowest salary (produce M).

```
SELECT
FROM        Employee;
```

Employee

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

M

MinS
25000

Nested Query: Example

- List the names of employees who have the lowest salary.

```
SELECT    E.FName, E.LName
FROM      Employee E,
          
WHERE     E.Salary = M.MinS;
```

Employee

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

M

MinS
25000

Typical Student Question

- Why do we have to combine the two instead of step by step like in RA?

```
/* Find employees with the lowest salary */
(SELECT min(Salary) AS MinS
FROM    Employee) AS M;

SELECT  E.FName, E.LName
FROM    Employee AS E, M
WHERE   E.Salary = MinS;
```

$$M \leftarrow g_{\min(Salary)\ AS\ MinS}\ (Employee)$$

$$EM \leftarrow M \bowtie_{Salary = MinS} Employee$$

$$Result \leftarrow \pi_{FName,\ LName}\ (EM)$$

Join

- There are a number of join that can be expressed in the FROM clause:
  - Cross Join
  - Join (AKA., inner join, basic join)
  - Natural join
  - Left outer join
  - Right outer join
  - Full outer join

Cross Join

- A "CROSS JOIN" is simply a cross product
- The following queries are equivalent:

```
SELECT      *
FROM        Employee, Department;

SELECT      *
FROM        Employee CROSS JOIN Department;
```

On Clause for Join

- JOIN with ON: cross product followed by a selection
- The selection condition is specified using ON

```
SELECT    LName, FName, DName
FROM      Employee JOIN Department ON DNO = DNumber;
```

- Join condition in the ON clause vs. WHERE clause
- These are the same:

```
SELECT    LName, FName, DName
FROM      Employee,Department
WHERE     DNO = DNumber;
```

Equijoin and Natural Join

Equijoin: Join condition has only equality

- Result will contain two attributes with identical values, perhaps different names

Natural join: Equijoin on attributes with same names

- No need to specify join attributes
- Result will contain each join attribute only once

What if there are no common attribute names?

- In that case, Natural Join ≡ Cross Product

No Join, Join, Natural Join

equivalent

```
SELECT      *
FROM        Customer C, Salesperson S
WHERE       C.SalespersonNum = S.Number;

SELECT      *
FROM        Customer C JOIN Salesperson S on
                        SalespersonNum;
```

```
SELECT      *
FROM        Customer NATURAL JOIN Salesperson ON
                        SalespersonNum;
```

this query is not equivalent to above two queries

Natural Join Equivalence

Customer (Number, Name, Address, CRating,
CAmount, CBalance,SalespersonNum)
Salesperson (Number, Name, Address, Office)

```
SELECT      *
FROM        Customer NATURAL JOIN Salesperson;
```

Left Outer Join

Customer

Number	Name	Address	CRating	CAmount	CBalance	SalespersonNum
1	smith	xxx	5	1,000	1,000	101
2	jones	yyy	7	5,000	4,000	101
3	wei	zzz	10	10,000	10,000	<null>

Salesperson

Number	Name	Address	Office
101	johnson	aaa	23
102	miller	bbb	26

INNER JOIN on C.SalespersonNum = S.Number gives us:

"smith" with "johnson" and "jones" with "johnson"

LEFT OUTER JOIN on C.SalespersonNum = S.Number gives us:

INNER JOIN plus "wei" with "<null>" salesperson

- Lists all customers, and their salesperson if any

Right Outer Join

Customer

Number	Name	Address	CRating	CAmount	CBalance	SalespersonNum
1	smith	xxx	5	1,000	1,000	101
2	jones	yyy	7	5,000	4,000	101
3	wei	zzz	10	10,000	10,000	<null>

Salesperson

Number	Name	Address	Office
101	johnson	aaa	23
102	miller	bbb	26

INNER JOIN on C.SalespersonNum = S.Number gives us:

"smith" with "johnson" and "jones" with "johnson"

RIGHT OUTER JOIN on C.SalespersonNum = S.Number gives us:

INNER JOIN plus "<null>" customer with "miller"

- Lists customers that have a salesperson, and salespersons that do not have a customer

Full Outer Join

FULL OUTER JOIN = LEFT OUTER JOIN ∪ RIGHT OUTER JOIN

FULL OUTER JOIN on C.SalespersonNum = S.Number gives us:

INNER JOIN plus “wei” with “<null>” salesperson plus “<null>” customer with “miller”

- Lists all customer-salesperson pairs, and customers that do not have a salesperson, and salespersons that do not have a customer

NOTE: You could also have NATURAL <left, right, full> OUTER JOIN

Summary of Today's Lecture

- Extension to Select
  - DISTINCT, AS, expression, aggregated
- Extension to From
  - Nested query
  - Join