Lecture 2:

Last Day: Relational Data Model

Today: Query Languages

- Relational Algebra (Formal)
- SQL (commercial)

Korth + Silberschatz: § 3.1, 3.2, 4.1

The Relational Model:

Data is Represented as Tables (Relations)

eg.

PERSON		
NAME	AGE	SALARY
Joe	19	15 K
John	25	27 K
Tony	34	50 K

← entry, tuple, row, record.

↑ Field, attribute, column.

Note: A table is treated as a set of tuples.

- Each tuple appears only once.
- The order of tuples is unimportant

Observation

Database tables are large.
Sometimes millions of records,
with hundreds of fields.

eg. Income Tax dept. has a record
for each income tax return.

$\therefore \approx 8,000,000$ records/year in Canada.
$\approx 80,000,000$ records/year in USA.

$\therefore \approx 80,000,000$ records in 10 years in Canada
$\approx 800,000,000$ records in 10 years in USA.

Maybe have one table for each tax year.
$\therefore$ 20 years $\Rightarrow$ 20 tables (at least).

Relational Algebra:

Operations on Tables.

Projection: $\Pi_{A_1 \cdots A_n} R$

Project Table R onto Attributes $A_1 \cdots A_n$.

$\Pi_{NAME, AGE}$ PERSON

NAME	AGE
Joe	19
John	25
Tony	34

② Selection: $\sigma_F R$

Select those rows in table R that make Formula F true.

e.g.: $\sigma_{AGE \geq 21}$ PERSON

NAME	AGE	SALARY
John	25	27K
Tony	34	50K

$\sigma_{(AGE \geq 21) \wedge (SALARY \geq 30K)}$ PERSON

NAME	AGE	SALARY
Tony	34	50K

Can use: $< = > \leq \geq$, $\wedge \vee \neg$

Remaining Algebra Operators

③ Union : R ∪ S

④ Difference: R − S

⑤ Cartesian Product: R × S

} relatively rare operations

7:

R

A	B	C
a	b	c
d	a	f
c	b	d

S

A	B	c
b	3	a
d	a	f

R ∪ S

a	b	c
d	a	f
c	b	d
b	g	a

R − S

a	b	c
c	b	d

ote: Duplicate rows are eliminated

R

A	B	C
a	b	c
d	a	f
c	b	d

S

D	E	F
b	g	a
d	a	f

R x S

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

(6) Natural Join: A frequent operation.

"Join" tables R & S on their common fields.

eg.

R

A	B
a_1	b_1
a_2	b_2
a_3	b_3

S

B	C
b_1	c_1
b_2	c_2
b_3	c_3

Natural Join of R and S

$R \bowtie S$

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

Motivation For Joins

Information that is scattered over several tables must be joined into a single table so that it can be queried.

eg, Given these tables:

R (Employee, Dept.)

S (Dept, Manager)

- Retrieve the name of each employee and his manager.

"Retrieve the name of each employee and his manager."

9.

R

Emp	Dept
Tony	Sales
Tom	Admin
Joe	Research

S

Dept	Mgr
Sales	George
Research	Jack
Admin	Sue

$R \bowtie S$

Emp	Dept	Mgr
Tony	Sales	George
Tom	Admin	Sue
Joe	Research	Jack

intermediate result

$\Pi_{Emp,Mgr}(R \bowtie S)$

Emp	Mgr
Tony	George
Tom	Sue
Joe	Jack

final result

In general if $(a, b) \in R$
and $(b, c) \in S$
then $(a, b, c) \in R \bowtie S$

} join definition

eg.

R

A	B
a_1	b_1
a_2	b_1
a_3	b_2
a_4	b_3

S

B	C
b_1	c_1
b_2	c_2
b_2	c_3
b_4	c_4

$R \bowtie S$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_1
a_3	b_2	c_2
a_3	b_2	c_3

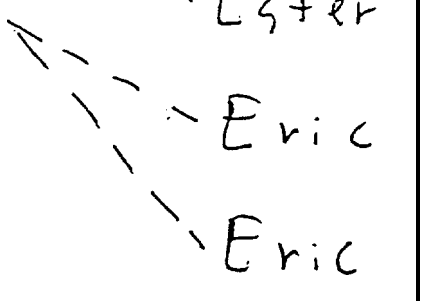
Example

R

Emp	City
Tony	NY
Ester	Paris
Eric	Paris
Pat	NY

S

Emp	Child
Tony	Vaclav
Ester	Vaclav
Ester	Lucy
Eric	Lucy
Eric	Bill



R ⋈ S

Emp	City	Child
Tony	NY	Vaclav
Ester	Paris	Vaclav
Ester	Paris	Lucy
Eric	Paris	Lucy
Eric	Paris	Bill

joined relations may share several attributes.

eg.

R

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

S

B	C	D
b_1	c_1	d_1
b_2	c_2	d_2
b_3	c_3	d_3

$R \bowtie S$

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3	b_3	c_3	d_3

In general, if $(a,b,c) \in R$

and $(b,c,d) \in S$

then $(a,b,c,d) \in R \bowtie S$

Example

R

Name	Address	Age
Tony	Bloor	31
Ester	Bloor	23
Tony	Jarvis	65
Lucy	Yonge	18

S

Name	Address	Child
Tony	Bloor	Vaclav
Ester	Bloor	Vaclav
Tony	Jarvis	Ron
Tony	Jarvis	George

Query: Retrieve the ages of Vaclav's parents.

R ⋈ S

Name	Address	Age	Child
Tony	Bloor	31	Vaclav
Ester	Bloor	23	Vaclav
Tony	Jarvis	65	Ron
Tony	Jarvis	65	George

$\sigma_{child = Vaclav}(R \bowtie S)$

Name	Address	Age	Child
Tony	Bloor	31	Vaclav
Ester	Bloor	23	Vaclav

$\pi_{Age}\ \sigma_{child=Vaclav}(R \bowtie S)$

Age
31
23

Note: This is called an SPJ query, because it uses Select, Project & Join, which is a common combination.

Extended Example

Given 3 Tables:

- Freq (Drinker, Bar)
  "Drinker Frequents Bar"

- Serves (Bar, Beer)
  "Bar Serves Beer"

- Likes (Drinker, Beer)
  "Drinker Likes Beer"

FREQ	
DRINKER	BAR
Joe	Bull
Joe	Bear
Bob	Tap
Jack	Bear

SERVES	
BAR	BEER
Bull	Bud
Bull	Mich
Bear	Bud
Tap	Mich

LIKES	
DRINKER	BEER
Joe	Mich
Bob	Bud
Bob	Coors
Jack	Bud

eg

SERVES ⋈ LIKES

BAR	BEER	DRINKER
Bull	Bud	Bob
Bull	Bud	Jack
Bull	Mich	Joe
Bear	Bud	Bob
Bear	Bud	Jack
Tap	Mich	~~Jack~~ Joe

meaning:

BAR serves BEER, which DRINKER likes.

$\Pi_{BAR, DRINKER}$ (SERVES $\bowtie$ LIKES)

BAR,	DRINKER
Bull	Bob
Bull	Jack
Bull	Joe
Bear	Bob
Bear	Jack
Tap	~~Jack~~ Joe

Meaning:

BAR serves some beer that DRINKER likes.

$FREQ \cap \Pi_{BAR, DRINKER} (SERVES \bowtie LIKES)$

BAR	DRINKER
Bull	Joe
Bear	Jack

Meaning: DRINKER Frequents BAR, and BAR serves some beer he likes.

$\Pi_{DRINKER} [FREQ \cap \Pi_{BAR, DRINKER} (SERVES \bowtie LIKES)]$

DRINKER
Joe
Jack

Meaning: DRINKER Frequents some bar that serves some beer that he likes.

Algebraic Laws

$$R \cup (S \cup T) = (R \cup S) \cup T$$

$$R \cup S = S \cup R$$

$$R \bowtie S = S \bowtie R$$

$$R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$$

$$\sigma_{F \wedge G} R = (\sigma_F R) \cap (\sigma_G R) = \sigma_F \sigma_G R$$

$$\sigma_{F \vee G} R = (\sigma_F R) \cup (\sigma_G R)$$

Such laws are the basis of algebraic query optimization.

SQL

(A commercial query language)

Syntax:

```
Select  Attributes
From    Tables
where   Conditions
```

+ some frills

- Queries over a single table
- Queries over multiple tables
- Queries that use the same table multiple times

① Queries over a single table

eg. Given a table

PERSON (NAME, AGE, SALARY)

- retrieve the name & age of each person.

```
Select  NAME, AGE
From    PERSON
```

} $\Pi_{NAME, AGE}$ PERSON

- retrieve those persons over 21.

i)
```
Select  NAME, AGE, SALARY
From    PERSON
where   AGE >21
```

} $\sigma_{AGE>21}$ PERSON

ii) Select * From PERSON where AGE >21.

"Retrieve the name of each person over 21."

```
Select  NAME
From    PERSON
where   AGE >21
```

$\Pi_{NAME}\ \sigma_{AGE>21}\ PERSON$

or alternatively,

```
Select  PERSON.NAME
From    PERSON
where   PERSON.AGE >21
```

This "dot" notation is useful when a query involves several tables.

(2) Queries over Multiple Tables: Examples.

Given two relations $R(A, B)$, $S(C, D)$,

```
Select  A, B, C, D
From    R, S
```

$\Big\}$ $R \times S$

i.e., return (a, b, c, d) iff $(a, b) \in R$ and $(c, d) \in S$

```
Select  A, B, C, D
From    R, S
where   B = C
```

$\Big\}$ $\sigma_{B=C}(R \times S)$

Queries over Multiple Tables with Shared Attributes

Given relations $R(A, B)$, $S(B, C)$

```
Select  A, R.B, S.B, C
From    R, S
```

$R \times S$

or equivalently

```
Select  R.A, R.B, S.B, S.C
From    R, S
```

$R \times S$

Joins.

Given $R(A,B)$, $S(B,C)$:

```
Select  R.A, R.B, S.C
From    R, S
where   R.B = S.B
```

} $R \bowtie S$

Given $R(A,B,C)$, $S(B,C,D)$

```
Select  R.A, R.B, R.C, S.D
from    R, S
where   R.B = S.B
  and   R.C = S.C
```

} $R \bowtie S$

Project-Select-Join (PSJ) Queries (Very Common)

Join several tables together,

Select tuples of interest

Display (Project onto) attributes of interest.

eg. EMP(NAME, DEPT)

DEPT(NAME, MGR)

"retrieve all employees managed by Bonner."

```
Select  EMP.NAME
From    EMP, DEPT
where   EMP.DEPT = DEPT.NAME
 and    DEPT.MGR = "Bonner"
```

(c) Using the Same Table more than Once

g.

PERSON

NAME	PARENT
tony	gord
gord	sue
sue	joe
joe	rose
rose	alice

...

"retrieve the grandparents of each person."

ake two copies of the table, and join them.

PERSON P1

NAME	PARENT
tony	gord
gord	sue
sue	joe
joe	rose
...	...

PERSON P2

NAME	PARENT
tony	gord
gord	sue
sue	joe
joe	rose
...	...

```
Select  P1.NAME, P2.PARENT
From    PERSON P1, PERSON P2
where   P1.PARENT = P2.NAME
```

gives:

tony	sue
gord	joe
sue	rose
...	

Extended Example

SUPPLIERS

NAME	ITEM	PRICE
Acme	Brie	3.49
Acme	Perrier	1.19
Acme	Snails	.25
Ajax	Brie	3.98
Ajax	Perrier	1.09

ORDERS

NUM	DATE	CUST
1024	Jan 3	Zack
1025	Jan 3	Ruth
1026	Jan 4	Zack

INCLUDES

ORD	ITEM	QUANTITY
1024	BRIE	3
1024	Perrier	6
1025	Brie	5

SUPPLIERS (NAME, ITEM, PRICE)

ORDERS (NUM, DATE, CUST)

INCLUDES (ORD, ITEM, QUANTITY)

"retrieve the items in order 1024"

```
Select ITEM
FROM INCLUDES
where ORD = 1024.
```

"retrieve the items ordered by Jack"

```
Select ITEM
From ORDERS, INCLUDES
where CUST = 'Jack'
  and NUM = ORD.
```

"Retrieve those suppliers that supply an item ordered by Jack."

'Jack'
=
ORDERS (NUM, DATE, CUST)
=
INCLUDES(ORD, ITEM, QUANTITY)
=
SUPPLIERS(NAME, ITEM, PRICE)

```
Select  NAME
From    ORDERS, INCLUDES, SUPPLIERS
where   CUST = 'Jack'
  and   NUM = ORD
  and   INCLUDES.ITEM = SUPPLIERS.ITEM
```

Summary

- In relational databases, data is stored in tables (or relations).
- A relation is a <u>set</u> of records.
  - Each record appears only once.
  - The order of records is unimportant.
- Relational Algebra is a small set of operations for deriving new relations from old relations.
- SQL is a (very popular) commercial version of Relational Algebra.