Relational Query Languages

Languages of DBMS

- Data Definition Language DDL
  - define the schema and storage stored in a Data Dictionary
- Data Manipulation Language DML
  - *Manipulative* populate schema, update database
  - *Retrieval* querying content of a database
- Data Control Language DCL
  - permissions, access control etc...

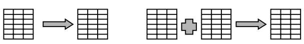
Data Manipulation Language

- Theory behind operations is formally defined and equivalent to a first-order logic (FOL)
- Relational Calculus ($\forall$, $\exists$) $\equiv$ Relational Algebra
- Relational algebra is a retrieval query language based on set operators and relational operators

- Relational operators transform either a single relation or a pair of relations into a result that is a relation that can be used as an operand on later operations
- For every operator operand and result, relations are free of duplicates
- Operators are *tuple oriented* or *set oriented*
- Structured Query Language (SQL)
- an ANSI standard for relational databases, based on relational algebra/calculus
  - SQL2 1992
  - SQL3 1998

Operations in the Relational Model

- Theory behind operations is formally defined and equivalent to a first order logic (FOL)
- Relational operators transform either a simple relation or a pair of relations into a result that is a relation



- The result can be used as an operand on later activities
- For every operand and result, relations are free of duplicates
- Operators are tuple oriented or set oriented

Query Operators

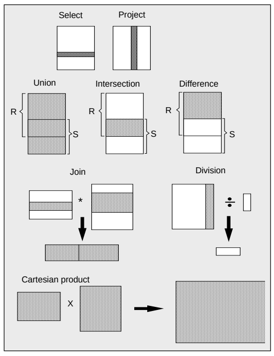
- *Relational Algebra*
  - tuple (unary) Selection, Projection
  - set (binary) Union, Intersection, Difference
  - tuple (binary) Join, Division
- *Additional Operators*
  - Outer Join, Outer Union

A Retrieval DML Must Express

- Attributes required in a result
  - *target list*
- Criteria for selecting tuples for that result
  - *qualifier*
- The relations that take part in the query
  - set generators

- Independent of the instances in the database
- Expressions are in terms of the database schema

Relational Algebra



SQL Retrieval Statement

```
SELECT[all|distinct]
     {*|{table.*|expr[alias]|view.*}
     [,{table.*|expr[alias]}]...}
FROM table [alias][,table[alias]] ...
[WHERE condition]
[CONNECT BY condition
     [START WITH condition]]
[GROUP BY expr [,expr] ...]
[HAVING condition]
[{UNION|UNION ALL|INTERSECT|MINUS}
     SELECT ...]
[ORDER BY {expr|position}
[ASC|DESC][,expr|position}[ASC|DESC].
[FOR UPDATE OF column [,column] ... [NOWAIT]]
```

π Project Operator

selects a subset of the attributes of a relation

attribute list are drawn from the specified relation;
if the key attribute is in the list then card(result) = card(relation)

Result = $\pi_{\text{(attribute list)}}$(relation name)

resulting relation has only the attributes in the list, in same order as they appear in the list

the degree(result) = number of attributes in the attribute list

no duplicates in the result

π Project Operator

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

tutor
bush
kahn
goble
zobel

π_{tutor}(STUDENT)

π Project Operator SELECT

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

```
select *
from student;
```

tutor
bush
kahn
goble
goble
zobel
kahn

```
select tutor
from student;
```

σ Select Operator

selects a subset of the tuples in a relation that satisfy a selection condition

a boolean expression specified on the attributes of a specified relation

Result = $\sigma_{\text{(selection condition)}}$(relation name)

a relation that has the same attributes as the source relation;

- stands for the usual comparison operators '<', '<>', '<=', '>', '>=', etc
- clauses can be arbitrarily connected with boolean operators AND, NOT, OR

degree(result) = degree(relation);
card(result) <= card(relation)

σ Select Operator

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

studno	name	hons	tutor	year
s4	bloggs	ca	goble	1

$\sigma_{name=\text{'bloggs'}}$(STUDENT)

retrieve tutor who tutors Bloggs

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

$\pi_{tutor}(\sigma_{name=\text{'bloggs'}}(\text{STUDENT}))$

```
select tutor from student
    where name = 'bloggs';
```

SQL retrieval expressions

- `select studentno, name from student`
  `where hons != 'ca' and`
  `(tutor = 'goble' or tutor = 'kahn');`
- `select * from enrol`
  `where labmark > 50;`
- `select * from enrol`
  `where labmark between 30 and 50;`

- `select * from enrol`
  `where labmark in (0, 100);`
- `select * from enrol`
  `where labmark is null;`
- `select * from student`
  `where name is like 'b%';`
- `select studno, courseno,`
  `exammark+labmark total from enrol`
  `where labmark is not NULL;`

Cartesian Product Operator

Definition:

- The cartesian product of two relations R1(A_1,A_2,...,A_n) with cardinality i and R2(B_1,B_2,...,B_m) with cardinality j is a relation R3 with degree k=n+m, cardinality i*j and attributes (A_1,A_2,...,A_n,B_1,B_2,...,B_m)
- The result, denoted by R1XR2, is a relation that includes all the possible combinations of tuples from R1 and R2
- Used in conjunction with other operations

Cartesian Product Example

STUDENT	
studno	name
s1	jones
s2	brown
s6	peters

COURSE		
courseno	subject	equip
cs250	prog	sun
cs150	prog	sun
cs390	specs	sun

STUDENT X COURSE				
studno	name	courseno	subject	equip
s1	jones	cs250	prog	sun
s1	jones	cs150	prog	sun
s1	jones	cs390	specs	sun
s2	brown	cs250	prog	sun
s2	brown	cs150	prog	sun
s2	brown	cs390	specs	sun
s6	peters	cs250	prog	sun
s6	peters	cs150	prog	sun
s6	peters	cs390	specs	sun

X Cartesian Product

STUDENT				
studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

STAFF	
lecturer	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

studno	name	hons	tutor	year	lecturer	roomno
s1	jones	ca	bush	2	kahn	IT206
s1	jones	ca	bush	2	bush	2.26
s1	jones	ca	bush	2	goble	2.82
s1	jones	ca	bush	2	zobel	2.34
s1	jones	ca	bush	2	watson	IT212
s1	jones	ca	bush	2	woods	IT204
s1	jones	ca	bush	2	capon	A14
s1	jones	ca	bush	2	lindsey	2.10
s1	jones	ca	bush	2	barringer	2.125
s2	brown	cis	kahn	2	kahn	IT206
s2	brown	cis	kahn	2	bush	2.26
s2	brown	cis	kahn	2	goble	2.82
s2	brown	cis	kahn	2	zobel	2.34
s2	brown	cis	kahn	2	watson	IT212
s2	brown	cis	kahn	2	woods	IT204
s2	brown	cis	kahn	2	capon	A14
s2	brown	cis	kahn	2	lindsey	2.10
s2	brown	cis	kahn	2	barringer	2.125
s3	smith	cs	goble	2	kahn	IT206
s3	smith	cs	goble	2	bush	2.26
s3	smith	cs	goble	2	goble	2.82
s3	smith	cs	goble	2	zobel	2.34
s3	smith	cs	goble	2	watson	IT212
s3	smith	cs	goble	2	woods	IT204
s3	smith	cs	goble	2	capon	A14
s3	smith	cs	goble	2	lindsey	2.10
s3	smith	cs	goble	2	barringer	2.125
s4	bloggs	ca	goble	1	kahn	IT206
s4	bloggs	ca	goble	1	bush	2.26

θ Join Operator

Definition:

The join of two relations $R1(A_1,A_2,...,A_n)$ and $R2(B_1,B_2,...,B_m)$ is a relation R3 with degree k=n+m and attributes $(A_1,A_2,...,A_n, B_1,B_2,...,B_m)$ that satisfy the join condition

$$\text{Result} = R1 \bowtie_{(\theta \text{ join condition})} R2$$

The result is a concatenated set but only for those tuples where the condition is true. It does not require union compatibility of R1 and R2

•stands for the usual comparison operators '<', '<>', '<=', '>', '>=', etc

•comparing terms in the Θ clauses can be arbitrarily connected with boolean operators AND, NOT, OR

θ Join Operator

STUDENT				
studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

STAFF	
lecturer	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

stud no	name	hons	tutor	year	lecturer	roomno
s1	jones	ca	bush	2	bush	2.26
s2	brown	cis	kahn	2	kahn	IT206
s3	smith	cs	goble	2	goble	2.82
s4	bloggs	ca	goble	1	goble	2.82
s5	jones	cs	zobel	1	zobel	2.34
s6	peters	ca	kahn	3	kahn	IT206

More joins

ENROL			
stud no	course no	lab mark	exam mark
s1	cs250	65	52
s1	cs260	80	75
s1	cs270	47	34
s2	cs250	67	55
s2	cs270	65	71
s3	cs270	49	50
s4	cs280	50	51
s5	cs250	0	3
s6	cs250	2	7

STUDENT				
studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

stud no	name	hons	tutor	year	stud no	course no	lab mark	exam mark
s1	jones	ca	bush	2	s1	cs250	65	52
s1	jones	ca	bush	2	s1	cs260	80	75
s1	jones	ca	bush	2	s1	cs270	47	34
s2	brown	cis	kahn	2	s2	cs250	67	55
s2	brown	cis	kahn	2	s2	cs270	65	71
s3	smith	cs	goble	2	s3	cs270	49	50
s4	bloggs	ca	goble	1	s4	cs280	50	51
s5	jones	cs	zobel	1	s5	cs250	0	3
s6	peters	ca	kahn	3	s6	cs250	2	7

Natural Join Operator

- Of all the types of θ-join, the equi-join is the only one that yields a result in which the compared columns are redundant to each other—possibly different names but same values
- The natural join is an equi-join but one of the redundant columns (simple or composite) is omitted from the result
- Relational join is the principle algebraic counterpart of queries that involve the existential quantifier ∃

Self Join: Joins on the same relation

$\pi_{(lecturer,\ roomno,appraiser,\ approom)}$ (staff $_{(appraiser\ =\ lecturer)}$ staff)

STAFF

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

lecturer	roomno	appraiser	approom
kahn	IT206	watson	IT212
bush	2.26	capon	A14
goble	2.82	capon	A14
zobel	2.34	watson	IT212
watson	IT212	barringer	2.125
woods	IT204	barringer	2.125
capon	A14	watson	IT212
lindsey	2.10	woods	IT204

```
select e.lecturer, e.roomno, m.lecturer appraiser, m.roomno approom
from staff e, staff m
where  e.appraiser = m.lecturer
```

Exercise

Get student's name, all their courses, subject of course, labmark for course, lecturer of course and lecturer's roomno for 'ca' students

University Schema

- STUDENT(<u>studno</u>,name,hons,tutor,year)
- ENROL(<u>studno,courseno</u>,labmark,exammark)
- COURSE(<u>courseno</u>,subject,equip)
- STAFF(<u>lecturer</u>,*roomno*,appraiser)
- TEACH(<u>courseno,lecturer</u>)
- YEAR(<u>yearno</u>,*yeartutor*)

Set Theoretic Operators

- *Union*, *Intersection* and *Difference*
- Operands need to be union compatible for the result to be a valid relation

Definition:

Two relations

$R1(A_1,A_2,...,A_n)$ and $R2(B_1,B_2,...,B_m)$

are union compatible iff:

$n = m$ and, $dom(A_i)= dom(B_i)$ for $1 \leq i \leq n$

∪ Union Operator

Definition:

The union of two relations

$R1(A_1,A_2,...,A_n)$ and $R2(B_1,B_2,...,B_m)$

is a relation $R3(C_1,C_2,...,C_n)$ such that $dom(C_i)= dom(A_i) =dom(B_i)$ for $1 \leq i \leq n$

- The result $R1 \cup R2$ is a relation that includes all tuples that are either in R1 or R2 or in both without duplicate tuples
- The resulting relation might have the same attribute names as the first or the second relation

Retrieve all staff that lecture or tutor

STUDENT

<u>studno</u>	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

TEACH

<u>courseno</u>	<u>lecturer</u>
cs250	lindsey
cs250	capon
cs260	kahn
cs260	bush
cs270	zobel
cs270	woods
cs280	capon

Lecturers $\pi_{(lecturer)}$TEACH
Tutors $\pi_{(tutor)}$STUDENT
Lecturers ∪ Tutors

lecturer
lindsey
capon
kahn
bush
zobel
woods
capon
goble

∩ Intersection Operator

Definition:

The intersection of two relations $R1(A_1,A_2,...,A_n)$ and $R2(B_1,B_2,...,B_m)$ is a relation $R3(C_1,C_2,...,C_n)$ such that

$dom(C_i)= dom(A_i) \cap dom(B_i)$ for $1 \leq i \leq n$

- The result $R1 \cap R2$ is a relation that includes only those tuples in R1 that also appear in R2
- The resulting relation might have the same attribute names as the first or the second relation

Retrieve all staff that lecture and tutor

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

TEACH

courseno	lecturer
cs250	lindsey
cs250	capon
cs260	kahn
cs260	bush
cs270	zobel
cs270	woods
cs280	capon

Lecturers $\pi_{(lecturer)}$TEACH
Tutors $\pi_{(tutor)}$STUDENT
Lecturers $\cap$ Tutors

lecturer
kahn
bush
zobel

– Difference Operator

Definition:
The difference of two relations $R1(A_1,A_2,\ldots,A_n)$ and $R2(B_1,B_2,\ldots,B_m)$ is a relation $R3(C_1,C_2,\ldots,C_n)$ such that

$$dom(C_i)= dom(A_i) - dom(B_i) \quad \text{for } 1 \le i \le n$$

- The result R1 – R2 is a relation that includes all tuples that are in R1 and not in R2
- The resulting relation might have the same attribute names as the first or the second relation

Retrieve all staff that lecture but don't tutor

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

TEACH

courseno	lecturer
cs250	lindsey
cs250	capon
cs260	kahn
cs260	bush
cs270	zobel
cs270	woods
cs280	capon

Lecturers $\pi_{(lecturer)}$TEACH
Tutors $\pi_{(tutor)}$STUDENT
Lecturers -Tutors

lecturer
lindsey
capon
woods

Outer Join Operation

- In an equi-join, tuples without a 'match' are eliminated
- Outer join keeps all tuples in R1 or R2 or both in the result, padding with nulls
  - Left outer join R1 R2
    - keeps every tuple in R1
    - select * from R1, R2 where R1.a = R2.a (+)
  - Right outer join R1 R2
    - keeps every tuple in R2
    - select * from R1, R2 where R1.a (+) = R2.a
  - Double outer join R1 R2
    - keeps every tuple in R1 and R2
    - select * from R1, R2 where R1.a (+) = R2.a (+)

Outer Join Operator

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	*null*	1
s5	jones	cs	zobel	1
s6	peters	ca	*null*	3

STAFF

lecturer	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

`select * from` student, staff
`where` tutor = lecturer

studno	name	hons	tutor	year	lecturer	roomno
s1	jones	ca	bush	2	bush	2.26
s2	brown	cis	kahn	2	kahn	IT206
s3	smith	cs	goble	2	goble	2.82
s5	jones	cs	zobel	1	zobel	2.34

Outer Join Operator

STUDENT

studno	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	*null*	1
s5	jones	cs	zobel	1
s6	peters	ca	*null*	3

STAFF

lecturer	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

`select * from` student, staff
`where` tutor = lecturer `(+)`

studno	name	hons	tutor	year	lecturer	roomno
s1	jones	ca	bush	2	bush	2.26
s2	brown	cis	kahn	2	kahn	IT206
s3	smith	cs	goble	2	goble	2.82
s5	jones	cs	zobel	1	zobel	2.34
s4	bloggs	ca	null	1	*null*	*null*
s6	peters	ca	*null*	3	*null*	*null*

Outer Self Join

π (lecturer, roomno,appraiser, approom) (staff (appraiser = lecturer) staff)

STAFF

lecturer	roomno	appraiser
kahn	IT206	watson
bush	2.26	capon
goble	2.82	capon
zobel	2.34	watson
watson	IT212	barringer
woods	IT204	barringer
capon	A14	watson
lindsey	2.10	woods
barringer	2.125	*null*

lecturer	roomno	appraiser	approom
kahn	IT206	watson	IT212
bush	2.26	capon	A14
goble	2.82	capon	A14
zobel	2.34	watson	IT212
watson	IT212	barringer	2.125
woods	IT204	barringer	2.125
capon	A14	watson	IT212
lindsey	2.10	woods	IT204
barringer	2.125	*null*	*null*

```
select e.lecturer, e.roomno, m.lecturer appraiser, m.roomno approom
from staff e, staff m
where  e.appraiser = m.lecturer (+)
```

Outer Union

- Takes the union of tuples from two relations that are *not* union compatible
- The two relations, R1 and R2, are partially compatible—only some of their attributes are union compatible
- The attributes that are not union compatible from either relation are kept in the result and tuples without values for these attributes are padded with nulls

Ordering results

```
select *
from enrol, student
where labmark is not null and
student.studno = enrol.studno
order by
  hons, courseno, name
```

default is ascending

Completeness of Relational Algebra

- Five fundamental operations
- σ π X $\cup$ —
- Additional operators are defined as combination of two or more of the basic operations,
- e.g.
  $R1 \cap R2 = R1 \cup R2 — ((R1 — R2) \cup (R2—R1)$
  $R1 \;_{<condition>} R1 = \sigma_{<condition>}(R1 \text{ X } R2)$

÷ Division Operation

Definition:

The division of two relations $R1(A_1,A_2,...,A_n)$ with cardinality i and $R2(B_1,B_2,...,B_m)$ with cardinality j is a relation R3 with degree k=n-m, cardinality i*j and attributes

$(A_1,A_2,...,A_n,B_1,B_2,...,B_m)$

that satisfy the division condition

- The principle algebraic counterpart of queries that involve the universal quantifier $\forall$
- Relational languages do not express relational division

÷ Division Operation Example

Retrieve the studnos of students who are enrolled on all the courses that Capon lectures on

- Small_ENROL ÷ Capon_TEACH

Small_ENROL

studno	courseno
s1	cs250
s1	cs260
s1	cs280
s2	cs250
s2	cs270
s3	cs270
s4	cs280
s4	cs250
s6	cs250

Capon_TEACH

courseno
cs250
cs280

result
s1
s4

Aggregation Functions

- Aggregation functions on collections of data values: average, minimum, maximum, sum, count
- Group tuples by value of an attribute and apply aggregate function independently to each group of tuples

ENROL

studno	courseno	lab mark	exam mark
s1	cs250	65	52
s1	cs260	80	75
s1	cs270	47	34
s2	cs250	67	55
s2	cs270	65	71
s3	cs270	49	50
s4	cs280	50	51
s5	cs250	0	3
s6	cs250	2	7

<grouping attributes> ƒ <function list> (relation name)

studno ƒ COUNT courseno (ENROL)

studno	count
s1	3
s2	2
s3	1
s4	1
s5	1
s6	1

Aggregation Functions in SQL

ENROL

studno	courseno	lab mark	exam mark
s1	cs250	65	52
s1	cs260	80	75
s1	cs270	47	34
s2	cs250	67	55
s2	cs270	65	71
s3	cs270	49	50
s4	cs280	50	51
s5	cs250	0	3
s6	cs250	2	7

```
select studno, count(*),
  avg(labmark)
from enrol
group by studno
```

studno	count	avg
s1	3	64
s2	2	66
s3	1	49
s4	1	50
s5	1	0
s6	1	2

Aggregation Functions in SQL

ENROL

studno	courseno	lab mark	exam mark
s1	cs250	65	52
s1	cs260	80	75
s1	cs270	47	34
s2	cs250	67	55
s2	cs270	65	71
s3	cs270	49	50
s4	cs280	50	51
s5	cs250	0	3
s6	cs250	2	7

```
select studno, count(*),
  avg(labmark)
from enrol
group by studno
having count(*) >= 2
```

studno	count	avg
s1	3	64
s2	2	66

Nested Subqueries

- Complete select queries within a where clause of another outer query
- Creates an intermediate result
- No limit to the number of levels of nesting

List all students with the same tutor as bloggs

```
select studno, name, tutor
from student
where tutor =(select tutor
              from student
              where name = 'bloggs')
```

studno	name	tutor
s3	smith	goble
s4	bloggs	goble

Nested Subqueries

```
select distinct name
from student
where studno in
  (select studno
  from enrol, teach, year
  where
  year.yeartutor = teach.lecturer and
  teach.courseno = enrol.courseno)
```

Union compatibility in nested subqueries

```
select distinct studno
from enrol
where (courseno, exammark) in
     (select courseno, exammark
     from student s, enrol e
     where s.name = 'bloggs' and
     e.studno = s.studentno);
```

Nested subqueries set comparison operators

- Outer query qualifier includes a value v compared with a bag of values V generated from a subquery
- Comparison v with V evaluates TRUE

v in V	if v is one of the elements V
v = any V	if v equal to some value in V
v > any V	if v > some value in V (same for <)
v > all V (same for <)	if v greater than all the values in V

Subqueries

May be used in these situations:

- to define the set of rows to be inserted in the target table of an `insert, create table` or `copy` command
- to define one or more values to be assigned to existing rows in an `update` statement
- to provide values for comparison in `where, having` and `start with` clauses in `select, update` and `delete` commands

Correlated subqueries

- A condition in the where clause of a nested query references some attribute of a relation declared in the outer (parent) query
- The nested query is evaluated once for each tuple in the outer (parent) query

```
select name
from student
where 3 >  (select count (*)
        from enrol
        where student.studno=enrol.studno)
```

Exists and correlated sub queries

- Exists is usually used to check whether the result of a correlated nested query is empty
- Exists (Q) returns TRUE if there is at least one tuple in the results query Q and FALSE otherwise

```
select name
from student
where exists (select *
        from enrol, teach
        where student.studno=enrol.studno and
            enrol.courseno = teach.courseno and
            teach.lecturer = 'Capon')
```

Retrieve the names of students who have registered for at least one course taught by Capon

Exists and correlated sub queries

- Not Exists (Q) returns FALSE if there is at least one tuple in the results query Q and TRUE otherwise

```
select name
from student
where not exists (select *
        from enrol
        where student.studno=enrol.studno)
```

Retrieve the names of students who have no enrolments on courses

Conclusions

- The only logical structure is that of a relation
- Constraints are formally defined on the concept of domain and key
- Operations deal with entire relations rather than single record at a time
- Operations are formally defined and can be combined in a declarative language to implement user queries on the database

Conclusions on SQL

- Retrieval: many ways to achieve the same result—though different performance costs
- Comprehensive and powerful facilities
- Non-procedural
- Can’t have recursive queries
- Limitations are overcome by use of a high-level procedural language that permits embedded SQL statements