Database Systems

Relational Algebra

H. Turgut Uyar   Şule Öğüdücü

2002-2012

License

Topics

Closure

Definition
closure: the input and output of all operations are relations

- the output of one operation can be the input of another
- operations can be nested

Example Relations

Example (MOVIE)

MOVIE#	TITLE	YEAR	SCORE	VOTES	DIRECTOR#
6	Usual Suspects	1995	8.7	35027	639
70	Being John Malkovich	1999	8.3	13809	1485
107	Batman & Robin	1997	3.5	10577	105
110	Sleepy Hollow	1999	7.5	10514	148
112	Three Kings	1999	7.7	10319	1070
151	Gattaca	1997	7.4	8388	2020
213	Blade	1998	6.7	6885	2861
228	Ed Wood	1994	7.8	6587	148
251	End of Days	1999	5.5	6095	103
281	Dangerous Liaisons	1988	7.7	5651	292
373	Fear and Loathing in Las Vegas	1998	6.5	4658	59
432	Stigmata	1999	6.1	4141	2557
433	eXistenZ	1999	6.9	4130	97
573	Dead Man	1995	7.4	3333	175
1468	Europa	1991	7.6	1042	615
1512	Suspiria	1977	7.1	1004	2259
1539	Cry-Baby	1990	5.9	972	364

Example Relations

Example (PERSON)

PERSON#	NAME
9	Arnold Schwarzenegger
26	Johnny Depp
59	Terry Gilliam
97	David Cronenberg
103	Peter Hyams
105	Joel Schumacher
138	George Clooney
148	Tim Burton
175	Jim Jarmusch
187	Christina Ricci
243	Uma Thurman
282	Cameron Diaz
292	Stephen Frears
302	Benicio Del Toro
308	Gabriel Byrne
350	Jennifer Jason Leigh
364	John Waters
406	Patricia Arquette
503	John Malkovich
615	Lars von Trier
639	Bryan Singer
745	Udo Kier
793	Jude Law
1070	David O. Russell
1485	Spike Jonze
1641	Iggy Pop
2020	Andrew Niccol
2259	Dario Argento
2557	Rupert Wainwright
2861	Stephen Norrington
3578	Traci Lords

Example Relations

Example (CASTING)

MOVIE#	ACTOR#	ORD
6	308	2
6	302	3
70	282	2
70	503	14
107	9	1
107	138	2
107	243	4
110	26	1
110	187	2
112	138	1
112	1485	4
151	243	2
151	793	3
213	745	6
213	3578	8
228	26	1
228	406	4
251	9	1
251	308	2
251	745	10
281	243	7
281	503	2
373	26	1
373	187	6
373	282	8
373	302	2
432	308	2
432	406	1
433	350	1
433	793	2
573	26	1
573	308	12
573	1641	6
1468	745	3
1512	745	9
1539	26	1
1539	1641	5
1539	3578	7

Selection

Definition
selection: selecting tuples that satisfy a condition

Statement

```
relation WHERE condition
```

- output header = input header

Selection Examples - 1

Example
- the movies with more than 10000 votes (S1)

```
MOVIE WHERE (VOTES > 10000)
```

S1

MOVIE#	TITLE	YEAR	SCORE	VOTES	DIRECTOR#
6	Usual Suspects	1995	8.7	35027	639
70	Being John Malkovich	1999	8.3	13809	1485
107	Batman & Robin	1997	3.5	10577	105
110	Sleepy Hollow	1999	7.5	10514	148
112	Three Kings	1999	7.7	10319	1070

Selection Examples - 2

Example
- the movies older than 1992, with scores higher than 7.5 (S2)

```
MOVIE WHERE ((YEAR < YEAR(1992))
        AND (SCORE > SCORE(7.5)))
```

S2

MOVIE#	TITLE	YEAR	SCORE	VOTES	DIRECTOR#
281	Dangerous Liaisons	1988	7.7	5651	292
1468	Europa	1991	7.6	1042	615

Projection

Definition
projection: selecting a set of attributes

Statement

```
relation { attribute_name [, ...] }
```

- output header = attribute list

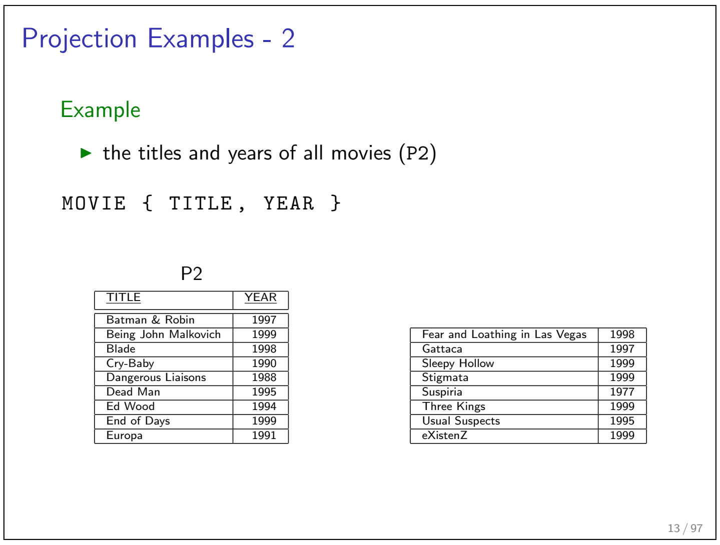
Projection Examples - 1

Example
- the titles of all movies (P1)

```
MOVIE { TITLE }
```

P1

TITLE
Usual Suspects
Being John Malkovich
Batman & Robin
Sleepy Hollow
Three Kings
Gattaca
Blade
Ed Wood
End of Days
Dangerous Liaisons
Fear and Loathing in Las Vegas
Stigmata
eXistenZ
Dead Man
Europa
Suspiria
Cry-Baby



Projection Examples - 2

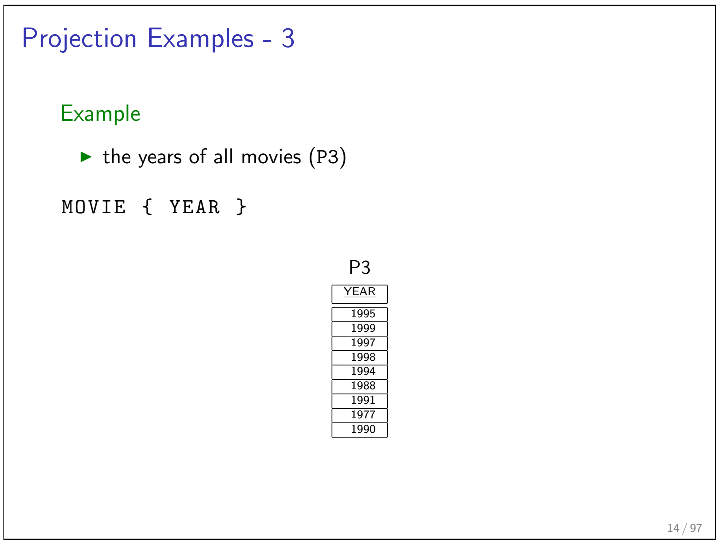
Example

- the titles and years of all movies (P2)

```
MOVIE { TITLE, YEAR }
```

P2

TITLE	YEAR
Batman & Robin	1997
Being John Malkovich	1999
Blade	1998
Cry-Baby	1990
Dangerous Liaisons	1988
Dead Man	1995
Ed Wood	1994
End of Days	1999
Europa	1991
Fear and Loathing in Las Vegas	1998
Gattaca	1997
Sleepy Hollow	1999
Stigmata	1999
Suspiria	1977
Three Kings	1999
Usual Suspects	1995
eXistenZ	1999



Projection Examples - 3

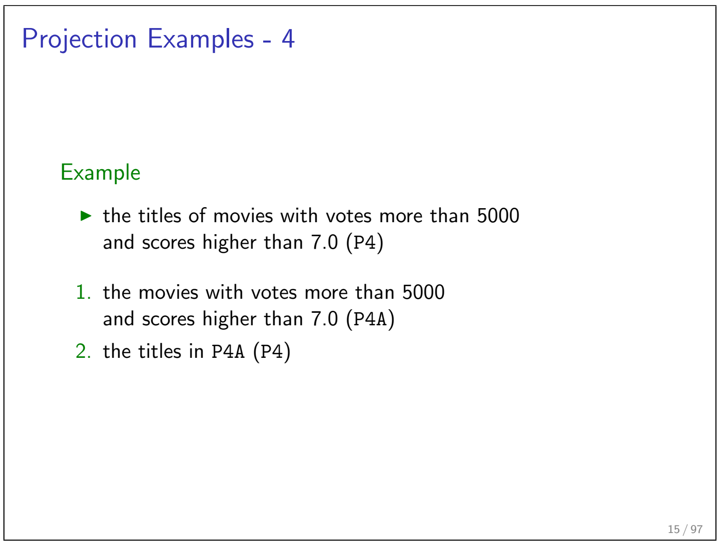
Example

- the years of all movies (P3)

```
MOVIE { YEAR }
```

P3

YEAR
1995
1999
1997
1998
1994
1988
1991
1977
1990

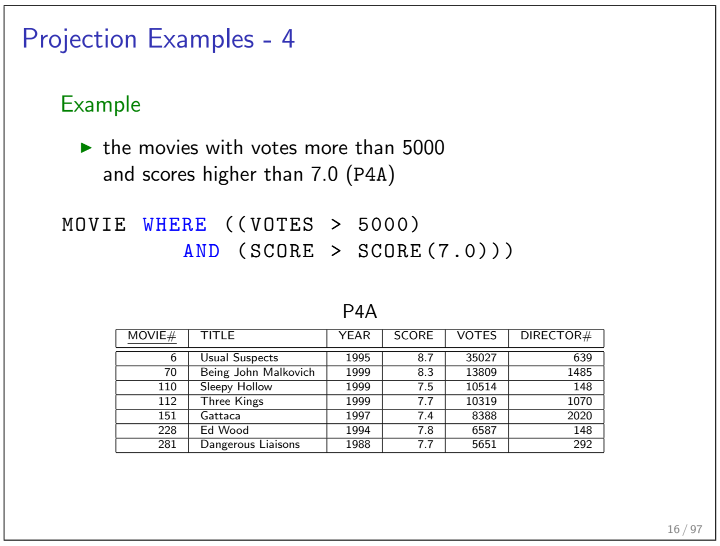


Projection Examples - 4

Example

- the titles of movies with votes more than 5000 and scores higher than 7.0 (P4)

1. the movies with votes more than 5000 and scores higher than 7.0 (P4A)
2. the titles in P4A (P4)



Projection Examples - 4

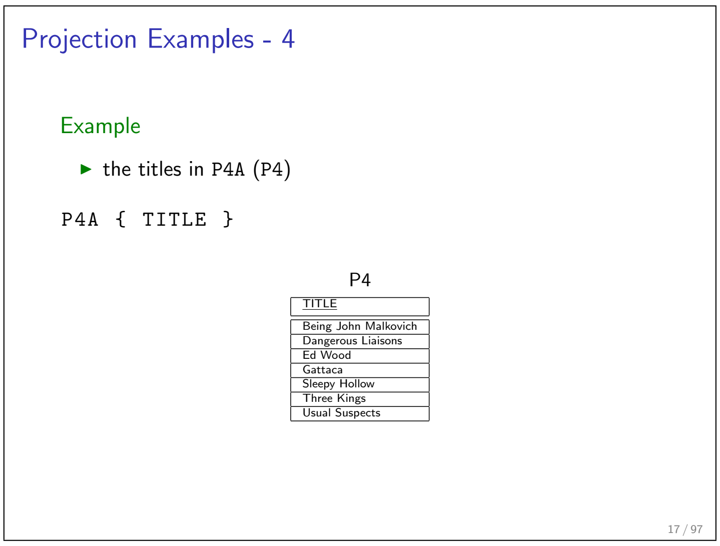
Example

- the movies with votes more than 5000 and scores higher than 7.0 (P4A)

```
MOVIE WHERE ((VOTES > 5000)
         AND (SCORE > SCORE(7.0)))
```

P4A

MOVIE#	TITLE	YEAR	SCORE	VOTES	DIRECTOR#
6	Usual Suspects	1995	8.7	35027	639
70	Being John Malkovich	1999	8.3	13809	1485
110	Sleepy Hollow	1999	7.5	10514	148
112	Three Kings	1999	7.7	10319	1070
151	Gattaca	1997	7.4	8388	2020
228	Ed Wood	1994	7.8	6587	148
281	Dangerous Liaisons	1988	7.7	5651	292



Projection Examples - 4

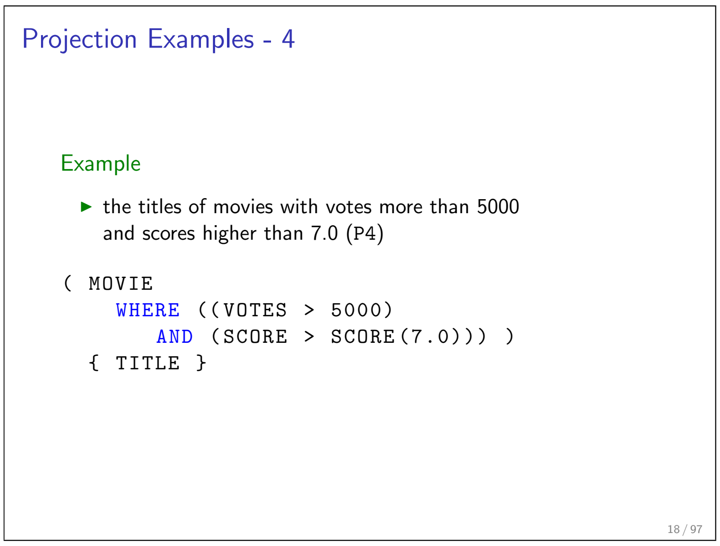
Example

- the titles in P4A (P4)

```
P4A { TITLE }
```

P4

TITLE
Being John Malkovich
Dangerous Liaisons
Ed Wood
Gattaca
Sleepy Hollow
Three Kings
Usual Suspects



Projection Examples - 4

Example

- the titles of movies with votes more than 5000 and scores higher than 7.0 (P4)

```
( MOVIE
    WHERE ((VOTES > 5000)
       AND (SCORE > SCORE(7.0))) )
  { TITLE }
```

Join

Definition

join: matching tuples of two relations
over common values of one or more attributes

- from the Cartesian product of the two relations,
  selecting the tuples which have the same values
  for the given attributes
- matching attributes are not repeated in the output
- natural join: over common values of all the attributes
  with the same name

Join

Statement

```
relation1 JOIN relation2
```

- output header = relation1 header ∪ relation2 header

Join Examples - 1

Example

- the titles of all movies and the names of their directors (J1)

1. all movies and their directors (`J1A`)
2. the movie titles and director names in `J1A` (J1)

Join Examples - 1

Example

- all movies and their directors (`J1A`)

```
MOVIE JOIN
  (PERSON RENAME (PERSON# AS DIRECTOR#))
```

J1A

MOVIE#	TITLE	...	DIRECTOR#	NAME
6	Usual Suspects	...	639	Bryan Singer
70	Being John Malkovich	...	1485	Spike Jonze
107	Batman & Robin	...	105	Joel Schumacher
...	...	...	...	...
1468	Europa	...	615	Lars von Trier
1512	Suspiria	...	2259	Dario Argento
1539	Cry-Baby	...	364	John Waters

Join Examples - 1

Example

- the movie titles and director names in `J1A` (J1)

```
J1A { TITLE, NAME }
```

J1

TITLE	NAME
Batman & Robin	Joel Schumacher
Being John Malkovich	Spike Jonze
Blade	Stephen Norrington
...	...
Three Kings	Spike Jonze
Usual Suspects	Bryan Singer
eXistenZ	David Cronenberg

Join Examples - 2

Example

- the titles of all movies and the names of their actors,
  along with their ordinal numbers (J2)

1. all movies and their casting data (`J2A`)
2. all data in `J2A` matched with persons (`J2B`)
3. the movie titles, actor names and ordinal numbers in `J2B` (J2)

Join Examples - 2

Example

- all movies and their casting data (J2A)

```
MOVIE JOIN CASTING
```

J2A

MOVIE#	TITLE	...	ACTOR#	ORD
6	Usual Suspects	...	302	3
6	Usual Suspects	...	308	2
70	Being John Malkovich	...	282	2
70	Being John Malkovich	...	503	14
...	...	...	...	...
1539	Cry-Baby	...	26	1
1539	Cry-Baby	...	1641	5
1539	Cry-Baby	...	3578	7

Join Examples - 2

Example

- all data in J2A matched with persons (J2B)

```
J2A JOIN (PERSON RENAME (PERSON# AS ACTOR#))
```

J2B

MOVIE#	TITLE	...	ACTOR#	ORD	NAME
6	Usual Suspects	...	302	3	Benicio Del Toro
6	Usual Suspects	...	308	2	Gabriel Byrne
70	Being John Malkovich	...	282	2	Cameron Diaz
70	Being John Malkovich	...	503	14	John Malkovich
...	...	...	...	...	...
1539	Cry-Baby	...	26	1	Johnny Depp
1539	Cry-Baby	...	1641	5	Iggy Pop
1539	Cry-Baby	...	3578	7	Traci Lords

Join Examples - 2

Example

- the movie titles, actor names and ordinal numbers in J2B (J2)

```
J2B { TITLE, NAME, ORD }
```

J2

TITLE	NAME	ORD
Usual Suspects	Benicio Del Toro	3
Usual Suspects	Gabriel Byrne	2
Being John Malkovich	Cameron Diaz	2
Being John Malkovich	John Malkovich	14
...	...	...
Cry-Baby	Johnny Depp	1
Cry-Baby	Iggy Pop	5
Cry-Baby	Traci Lords	7

Join Examples - 3

Example

- the names of the actors in Johnny Depp's movies (J3)

1. the identifiers of Johnny Depp's movies (J3A)
2. the identifiers of the actors in the movies in J3A (J3B)
3. the names of the actors in J3B (J3)

Join Examples - 3

Example

- the identifiers of Johnny Depp's movies (J3A)

```
(((PERSON RENAME (PERSON# AS ACTOR#))
    JOIN CASTING)
  WHERE (NAME = 'Johnny Depp')) { MOVIE# }
```

J3A

MOVIE#
110
228
373
573
1539

Join Examples - 3

Example

- the identifiers of the actors in the movies in J3A (J3B)

```
(J3A JOIN CASTING) { ACTOR# }
```

J3B

ACTOR#
26
187
282
302
308
406
1641
3578

Join Examples - 3

Example

- the names of the actors in J3B (J3)

```
((J3B RENAME (ACTOR# AS PERSON#))
    JOIN PERSON) { NAME }
```

J3

NAME
Johnny Depp
Christina Ricci
Cameron Diaz
Benicio Del Toro
Gabriel Byrne
Patricia Arquette
Iggy Pop
Traci Lords

Division

Definition
division: from the tuples of the first relation,
selecting the ones that match all the tuples
of the second relation in a mediating relation

Statement

```
relation1 DIVIDEBY relation2
  PER (relation3)
```

Division Example

Example

- the titles of the movies in which both Johnny Depp and Christina Ricci played (V1)

1. the identities of Johnny Depp and Christina Ricci (V1A)
2. the identities of the movies with all the actors in V1A (V1B)
3. the titles of the movies in V1B (V1)

Division Example

Example

- the identities of Johnny Depp and Christina Ricci (V1A)

```
(PERSON
    WHERE ((NAME = "Johnny Depp")
        OR (NAME = "Christina Ricci")))
  { PERSON# }
```

V1A

PERSON#
26
187

Division Example

Example

- the identities of movies with all the actors in V1A (V1B)

```
(MOVIE { MOVIE# })
    DIVIDEBY (V1A RENAME (PERSON# AS ACTOR#))
      PER (CASTING { MOVIE#, ACTOR# })
```

V1B

MOVIE#
110
373

Division Example

Example

- the titles of the movies in V1B (V1)

```
(V1B JOIN MOVIE) { TITLE }
```

V1

TITLE
Fear and Loathing in Las Vegas
Sleepy Hollow

Intersection

Definition

intersection: selecting the tuples found in both relations

Statement

```
relation1 INTERSECT relation2
```

- output header = relation1 header = relation2 header

Intersection Example

Example

- the names of all directors who also acted (`I1`)

1. the identifiers of all directors who also acted (`I1A`)
2. the names of all the persons in `I1A` (`I1`)

Intersection Example

Example

- the identifiers of all directors who also acted (`I1`)

```
(MOVIE { DIRECTOR# }
      RENAME (DIRECTOR# AS PERSON#))
  INTERSECT
(CASTING { ACTOR# }
      RENAME (ACTOR# AS PERSON#))
```

I1A

PERSON#
1485

Intersection Example

Example

- the names of all the persons in `I1A` (`I1`)

```
(I1A JOIN PERSON) { NAME }
```

I1

NAME
Spike Jonze

Union

Definition

union: selecting the tuples found in at least one of two relations

Statement

```
relation1 UNION relation2
```

- output header = relation1 header = relation2 header

Union Example

Example

- the names of all the directors and actors of all movies newer than 1997 (`U1`)

1. the identifiers and director identifiers of all movies newer than 1997 (`U1A`)
2. the identifiers of all the actors in the movies in `U1A` (`U1B`)
3. the identifiers of the directors and actors in at least one of `U1A` and `U1B` (`U1C`)
4. the names of all the persons in `U1C` (`U1`)

Union Example

Example

- the identifiers and director identifiers of all movies newer than 1997 (U1A)

```
(MOVIE WHERE (YEAR > YEAR(1997)))
  { MOVIE#, DIRECTOR# }
```

U1A

MOVIE#	DIRECTOR#
70	1485
110	148
112	1070
213	2861
251	103
373	59
432	2557
433	97

Union Example

Example

- the identifiers of all the actors in the movies in U1A (U1B)

```
(U1A JOIN CASTING) { ACTOR# }
```

U1B

ACTOR#
9
26
138
187
282
302
308
350
406
503
745
793
1485
3578

Union Example

Example

- the identifiers of the directors and actors in at least one of U1A and U1B (U1C)

```
(U1A { DIRECTOR# }
      RENAME (DIRECTOR# AS PERSON#))
UNION (U1B RENAME (ACTOR# AS PERSON#))
```

U1C

PERSON#
9
26
59
97
103
138
148
187
282
302
308
350
406
503
745
793
1070
1485
2557
2861
3578

Union Example

Example

- the names of all the persons in U1C (U1)

```
(U1C JOIN PERSON) { NAME }
```

U1

NAME
Arnold Schwarzenegger
Benicio Del Toro
Cameron Diaz
Christina Ricci
David Cronenberg
David O. Russell
Gabriel Byrne
George Clooney
Jennifer Jason Leigh
John Malkovich
Johnny Depp
Jude Law
Patricia Arquette
Peter Hyams
Rupert Wainwright
Spike Jonze
Stephen Norrington
Terry Gilliam
Tim Burton
Traci Lords
Udo Kier

Difference

Definition
difference: selecting the tuples which are found in the first but not in the second relation

Statement

```
relation1 MINUS relation2
```

- output header = relation1 header = relation2 header

Difference Example

Example

- the names of all the actors who have not played in Johnny Depp's movies (D1)

1. the identities of all the actors who played in Johnny Depp's movies (J3B)
2. the names of all the actors who are not in J3B (D1)

Difference Example

Example

- the names of all the actors who are not in J3B (D1)

```
(((CASTING { ACTOR# } MINUS J3B)
      RENAME (ACTOR# AS PERSON#))
  JOIN PERSON) {NAME}
```

D1

NAME
Arnold Schwarzenegger
George Clooney
Jennifer Jason Leigh
John Malkovich
Jude Law
Spike Jonze
Udo Kier
Uma Thurman

References

Required Reading: Date

- Chapter 7: Relational Algebra
  7.1. Introduction
  7.2. Closure Revisited
  7.4. The Original Algebra: Semantics

Simple Queries

Statement

```
SELECT [ ALL | DISTINCT ] column_name [, ...]
  FROM table_name
```

- duplicate rows are allowed
  - ALL: preserve duplicate rows (default)
  - DISTINCT: take only one of duplicate rows
- *: all columns

Query Examples

Example (all data of all movies)

```
SELECT * FROM MOVIE
```

Example (titles and years of all movies)

```
SELECT TITLE, YR FROM MOVIE
```

Example (years when movies were filmed)

```
SELECT DISTINCT YR FROM MOVIE
```

Sorting Results

Statement

```
SELECT [ ALL | DISTINCT ] column_name [, ...]
  FROM table_name
  [ ORDER BY { column_name [ ASC | DESC ] }
      [, ...] ]
```

- sort order:
  - ASC: ascending (default)
  - DESC: descending

Query Examples

Example (years when movies were filmed, in ascending order)

```
SELECT DISTINCT YR FROM MOVIE
  ORDER BY YR
```

Example (years when movies were filmed, in descending order)

```
SELECT DISTINCT YR FROM MOVIE
  ORDER BY YR DESC
```

Expressions

Statement

```
SELECT [ ALL | DISTINCT ]
  { expression [ AS column_name ] } [, ...]
  FROM table_name
  [ ORDER BY { column_name [ ASC | DESC ] }
      [, ...] ]
```

- the new column can be named
- the name or order of the column can be used for sorting

Query Examples

Example (titles and total scores of all movies)

```
SELECT TITLE, SCORE * VOTES
  FROM MOVIE
```

Query Examples

Example (titles and total scores of all movies,
in descending order of total scores)

```
SELECT TITLE, SCORE * VOTES AS POINTS
  FROM MOVIE
  ORDER BY POINTS DESC

SELECT TITLE, SCORE * VOTES
  FROM MOVIE
  ORDER BY 2 DESC
```

Selecting Rows

Statement

```
SELECT [ ALL | DISTINCT ]
  { expression [ AS column_name ] } [, ...]
  FROM table_name
  [ WHERE condition ]
  [ ORDER BY { column_name [ ASC | DESC ] }
      [, ...] ]
```

- comparison operators:
  `= < > <= >= <>`
- connectives:
  `NOT AND OR`

Condition Expressions

- whether a column is empty or not:

  ```
  column_name IS { NULL | NOT NULL }
  ```

- set membership:

  ```
  column_name IN (value_set)
  ```

- string comparison

  ```
  column_name LIKE pattern
  ```

  - in the pattern, % can substitute any symbol group

Query Examples

Example (years of movies with the title "Citizen Kane")

```
SELECT YR FROM MOVIE
  WHERE (TITLE = 'Citizen Kane')
```

Example (titles of movies with scores less than 3
and votes more than 10)

```
SELECT TITLE FROM MOVIE
  WHERE ((SCORE < 3) AND (VOTES > 10))
```

Query Examples

Example (titles of movies with unknown year)

```
SELECT TITLE FROM MOVIE
  WHERE (YR IS NULL)
```

Example (titles of movies in years 1967, 1954 and 1988)

```
SELECT TITLE, YR FROM MOVIE
  WHERE (YR IN (1967, 1954, 1988))
```

Query Examples

Example (titles and scores of "Police Academy" movies)

```
SELECT TITLE, SCORE FROM MOVIE
  WHERE (TITLE LIKE 'Police Academy%')
```

Grouping

Statement

```
SELECT [ ALL | DISTINCT ]
  { expression [ AS column_name ] } [, ...]
  FROM table_name
  [ WHERE condition ]
  [ GROUP BY column_name [, ...] ]
  [ HAVING condition ]
  [ ORDER BY { column_name [ ASC | DESC ] }
      [, ...] ]
```

- selected rows can be grouped
- groups can be filtered

Processing Order

- the rows that satisfy the WHERE condition are selected
- selected rows are grouped using the columns specified in the GROUP BY clause
  - if no group, the entire result is one group
- the groups that satisfy the HAVING condition are selected
- the expressions given in the column list are calculated
- the result is sorted on the column list specified in the ORDER BY clause

Group Values

- one value for each group
  - the value of the grouping column
  - the result of an aggregate function
- aggregate functions:
  COUNT SUM AVG MAX MIN
  - column name as parameter
  - null values are ignored

Query Examples

Example (for every year, the number of movies with score greater than 8.5 in that year)

```
SELECT YR, COUNT(*) FROM MOVIE
  WHERE (SCORE > 8.5)
  GROUP BY YR
```

Query Examples

Example (score of the favorite movie of every year,
in ascending order of years)

```
SELECT YR, MAX(SCORE) FROM MOVIE
  GROUP BY YR
  ORDER BY YR
```

Query Examples

Example (total number of votes)

```
SELECT SUM(VOTES) FROM MOVIE
```

Query Examples

Example (averages of movie scores which were filmed in the
years
where there are at least 25 movies
for which more than 40 people have voted,
in ascending order of years)

```
SELECT YR, AVG(SCORE)
  FROM MOVIE
  WHERE (VOTES > 40)
  GROUP BY YR
  HAVING (COUNT(ID) >= 25)
  ORDER BY YR
```

Join

- joining can be achieved using WHERE conditions
  - list the tables to join in the table list
  - use the dotted notation for columns with identical names
- processing order:
  - the Cartesian product of the tables is taken
  - the rows that satisfy the WHERE condition are selected
  - ...

Query Examples

Example (names of the directors of the movies
with the title "Star Wars")

```
SELECT NAME
  FROM MOVIE, PERSON
  WHERE ((DIRECTORID = PERSON.ID)
     AND (TITLE = 'Star Wars'))
```

Query Examples

Example (names of the actors of the movies with the title
"Alien")

```
SELECT NAME
  FROM MOVIE, PERSON, CASTING
  WHERE ((TITLE = 'Alien')
     AND (MOVIEID = MOVIE.ID)
     AND (ACTORID = PERSON.ID))
```

Query Examples

Example (titles of movies where actors
with the name "Harrison Ford" played)

```
SELECT TITLE
  FROM MOVIE, PERSON, CASTING
  WHERE ((NAME = 'Harrison Ford')
     AND (MOVIEID = MOVIE.ID)
     AND (ACTORID = PERSON.ID))
```

Query Examples

Example (titles of movies where actors
with the name "Harrison Ford" played, but not the lead role)

```
SELECT TITLE
  FROM MOVIE, PERSON, CASTING
  WHERE ((NAME = 'Harrison Ford')
     AND (MOVIEID = MOVIE.ID)
     AND (ACTORID = PERSON.ID)
     AND (ORD > 1))
```

Query Examples

Example (titles and names of the lead actors of the movies in 1962)

```
SELECT TITLE, NAME
  FROM MOVIE, PERSON, CASTING
  WHERE ((YR = 1962)
     AND (MOVIEID = MOVIE.ID)
     AND (ACTORID = PERSON.ID)
     AND (ORD = 1))
```

Table Expressions

- the join operation can be expressed as a table expression:
  - product
  - using conditions
  - over columns with the same name
  - natural join
  - outer join

Statement

```
SELECT ...
  FROM table_expression [ AS table_name ]
  WHERE selection_condition
  ...
```

Join Expressions

product

```
table1_name CROSS JOIN table2_name
```

using conditions

```
table1_name JOIN table2_name
  ON condition
```

Query Examples

Example (names of the directors of the movies
with the title "Star Wars")

```
SELECT NAME
  FROM MOVIE JOIN PERSON
          ON (DIRECTORID = PERSON.ID)
  WHERE (TITLE = 'Star Wars')
```

Join Expressions

over columns with the same name

```
table1_name JOIN table2_name
  USING (column_name [, ...])
```

- repeated columns are taken once

natural join

```
table1_name NATURAL JOIN table2_name
```

Outer Join

- in "inner" join, if a row does not match any row of the other table, it is not included in the result
- in outer join, every unmatched row is included in the result where the columns from the other table are set to null

Statement

```
table1_name [ LEFT | RIGHT | FULL ]
       [ OUTER ] JOIN table2_name
```

Outer Join Examples

Example (left outer join)

T1

NUM	NAME
1	a
2	b
3	c

T2

NUM	VALUE
1	xxx
3	yyy
5	zzz

SELECT * FROM T1 LEFT JOIN T2

NUM	NAME	NUM	VALUE
1	a	1	xxx
2	b		
3	c	3	yyy

Outer Join Examples

Example (right outer join)

T1

NUM	NAME
1	a
2	b
3	c

T2

NUM	VALUE
1	xxx
3	yyy
5	zzz

SELECT * FROM T1 RIGHT JOIN T2

NUM	NAME	NUM	VALUE
1	a	1	xxx
3	c	3	yyy
		5	zzz

Outer Join Examples

Example (full outer join)

T1

NUM	NAME
1	a
2	b
3	c

T2

NUM	VALUE
1	xxx
3	yyy
5	zzz

SELECT * FROM T1 FULL JOIN T2

NUM	NAME	NUM	VALUE
1	a	1	xxx
2	b		
3	c	3	yyy
		5	zzz

Query Examples

Example (titles of movies with no known actors)

```
SELECT TITLE
  FROM MOVIE LEFT JOIN CASTING
         ON (MOVIEID = MOVIE.ID)
  WHERE (ACTORID IS NULL)
```

Self Join

- if the columns to join are in the same table
- give a new name to the table in the expression

Query Examples

Example (titles of all movies with the same number of votes)

```
SELECT M1.TITLE, M2.TITLE
  FROM MOVIE AS M1, MOVIE AS M2
  WHERE (M1.VOTES = M2.VOTES)
    AND (M1.ID < M2.ID)
```

Subqueries

Statement

```
SELECT ...
  WHERE expression operator
       [ ALL | ANY ] (subquery)
  ...
```

- using subquery results in condition expression
  - row and column counts of subquery must be suitable
  - ALL: for all values from subquery
  - ANY: for at least one value from subquery

Query Examples

Example (titles and scores of all movies with scores higher than that of "Star Wars", in descending order of scores)

```
SELECT TITLE, SCORE FROM MOVIE
   WHERE (  SCORE >
      ( SELECT SCORE FROM MOVIE
          WHERE (TITLE = 'Star Wars') )
) ORDER BY SCORE DESC
```

Query Examples

Example (titles of movies with more votes than the sum of all "Police Academy" movies)

```
SELECT TITLE FROM MOVIE
   WHERE (  VOTES >
      ( SELECT SUM(VOTES) FROM MOVIE
          WHERE (TITLE LIKE 'Police Academy%') )
)
```

Query Examples

Example (titles of movies with scores less than the scores of all "Police Academy" movies)

```
SELECT TITLE FROM MOVIE
   WHERE (  SCORE < ALL
      ( SELECT SCORE FROM MOVIE
          WHERE (TITLE LIKE 'Police Academy%') )
)
```

Query Examples

Example (titles of movies with less votes than any movie made before 1930)

```
SELECT TITLE FROM MOVIE
   WHERE ((YR >= 1930) AND (  VOTES < ANY (
      ( SELECT VOTES FROM MOVIE
          WHERE (YR < 1930) )
))
```

Set Operations

- an operation on two subquery results
- basic set operations in the relational model:
  - intersection: INTERSECT
  - union: UNION
  - difference: EXCEPT
- duplicate rows are not allowed in the result

Query Examples

Example (number of people who both directed and acted)

```
SELECT COUNT(*) FROM (
   ( SELECT DISTINCT DIRECTORID FROM MOVIE )
   INTERSECT
   ( SELECT DISTINCT ACTORID FROM CASTING )
) AS DIRECTOR_ACTOR
```

Query Examples

Example (number of directors who have not acted)

```
SELECT COUNT(*) FROM (
   ( SELECT DISTINCT DIRECTORID FROM MOVIE )
   EXCEPT
   ( SELECT DISTINCT ACTORID FROM CASTING )
) AS DIRECTOR_ONLY
```

Query Examples

Example (number of people who worked in movies before 1930)

```
SELECT COUNT(*) FROM (
   ( SELECT DISTINCT DIRECTORID FROM MOVIE
       WHERE (YR < 1930) )
   UNION
   ( SELECT DISTINCT ACTORID FROM CASTING
       WHERE (MOVIEID IN
                ( SELECT ID FROM MOVIE
                    WHERE (YR < 1930) )) )
) AS OLD_MOVIE_PERSON_IDS
```

Extra Examples

- number of movies John Travolta acted in every year
- titles and number of actors of the movies in 1978, in ascending order of actor counts
- names of actors who played with Johnny Depp
- titles and names of lead actors of the movies Uma Thurman was in
- names of actors with at least 10 lead roles

References

Required Reading: Date

- Chapter 8: Relational Calculus
  - 8.6. SQL Facilities
- Appendix B: SQL Expressions
- Chapter 19: Missing Information

Supplementary Reference

- A Gentle Introduction to SQL:
  `http://sqlzoo.net/`