Database Systems

SQL: Data Manipulation (DML)
SQL Joins

SELECT Statement - Syntax

```
SELECT [DISTINCT | ALL]
    {* | [columnExpression [AS newName]]
        [,...] }
FROM TableName [alias] [, ...]
[WHERE  condition]
[GROUP BY    columnList]
[HAVING condition]
[ORDER BY    columnList]
```

- Syntax order of the clauses cannot be changed
- Only SELECT and FROM are mandatory

SELECT Statement - Coding

Logical / Interpretation Order:

1.	FROM	Specifies table(s) to be used
2.	WHERE	Filters rows by some conditions
3.	GROUP BY	Forms groups of rows with same column value
4.	HAVING	Filters groups subject to some conditions
5.	SELECT	Specifies which columns are to appear in output
6.	ORDER BY	Specifies the order of the output

Books Database Schema

authors (au_id, au_fname, au_lname, phone, address, city, state, zip)

title_authors (title_id, au_id, au_order, royalty_share)

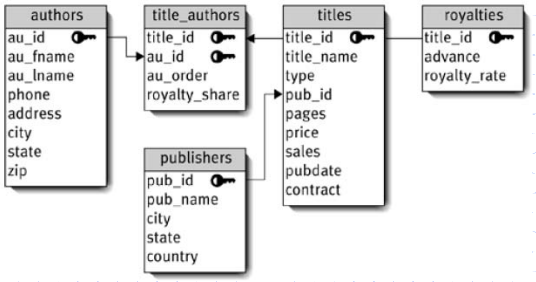
publishers (pub_id, pub_name, city, state, country)

royalties (title_id,advance,royalty_rate)

titles (title_id, title_name, type, pub_id, pages, price, sales, pubdate, contract)

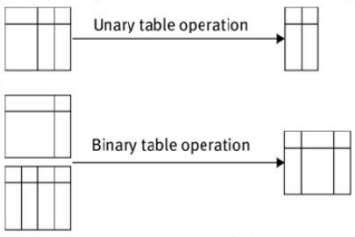
http://www.fehily.com/books/SQL-Visual-QuickStart-Guide-3rd.html

Books Database Schema



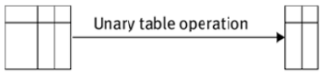
Relational Algebra

- Relational algebraic operations:
  - Projection
  - Restriction
  - Product
  - Join
  - Division
  - Union
  - Intersection
  - Difference



Relational Algebra

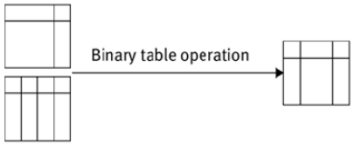
- Unary operators:



  - Projection – pick out columns
  - SQL: SELECT <columns>
  - Restriction – pick out rows meeting certain conditions
  - SQL: WHERE <conditions>

Relational Algebra

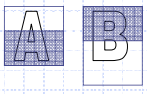
- Binary operators



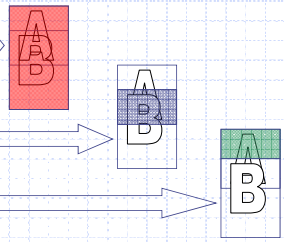
  - Product – SQL: CROSS JOIN
  - Join – SQL: INNER JOIN, OUTER JOIN
    SQL: FROM clause
  - Division – no single SQL counterpart (later)

Relational Algebra

- Binary SET operators
  Union Compatible



  - A Union B :
    All rows from both tables
  - A Intersect B :
    Rows existed in both tables
  - A Difference B :
    Rows in one but not in other table



Cross Join / Product Example

```
SQL> SELECT * FROM mypubs;

PUB PUB_NAME
--- --------------------
P01 Pearson
P02 McGraw-Hill
P03 Morgan-Kaufmann

SQL> SELECT * FROM mytitles;

TIT TITLE_NAME   TYPE       PUB PUBDATE
--- ------------ ---------- --- ----------
T01 1977!        history    P01 2000-08-01
T02 World War I  history    P03 1998-04-01
T03 Database     computer   P03 2000-09-01
T05 Your Ideas   psychology P01 2001-01-01
T10 Mr President biography  P02
T13 Civil War    history    P02 1999-05-31
```



```
SQL> SELECT *
  2  FROM mypubs CROSS JOIN mytitles;

PUB PUB_NAME           TIT TITLE_NAME   TYPE       PUB PUBDATE
--- ------------------ --- ------------ ---------- --- ----------
P01 Pearson            T01 1977!        history    P01 2000-08-01
P01 Pearson            T02 World War I  history    P03 1998-04-01
P01 Pearson            T03 Database     computer   P03 2000-09-01
P01 Pearson            T05 Your Ideas   psychology P01 2001-01-01
P01 Pearson            T10 Mr President biography  P02
P01 Pearson            T13 Civil War    history    P02 1999-05-31
P02 McGraw-Hill        T01 1977!        history    P01 2000-08-01
P02 McGraw-Hill        T02 World War I  history    P03 1998-04-01
P02 McGraw-Hill        T03 Database     computer   P03 2000-09-01
P02 McGraw-Hill        T05 Your Ideas   psychology P01 2001-01-01
P02 McGraw-Hill        T10 Mr President biography  P02
P02 McGraw-Hill        T13 Civil War    history    P02 1999-05-31
P03 Morgan-Kaufmann    T01 1977!        history    P01 2000-08-01
P03 Morgan-Kaufmann    T02 World War I  history    P03 1998-04-01
P03 Morgan-Kaufmann    T03 Database     computer   P03 2000-09-01
P03 Morgan-Kaufmann    T05 Your Ideas   psychology P01 2001-01-01
P03 Morgan-Kaufmann    T10 Mr President biography  P02
P03 Morgan-Kaufmann    T13 Civil War    history    P02 1999-05-31

18 rows selected.
```

```
SQL> SELECT *
  2  FROM mypubs, mytitles;
```


Cross Join / Product

- SQL1:
  - **SELECT [DISTINCT | ALL]**
    {* | columnList}
    FROM Table1, Table2;
- SQL2:
  - **SELECT [DISTINCT | ALL]**
    {* | columnList}
    FROM Table1 CROSS JOIN Table2;

Inner Join

- List the names of all publishers which have published a book:
-
```
SELECT *
FROM mypubs
   INNER JOIN mytitles
      ON mypubs.pub_id = mytitles.pub_id;
```

```
PUB PUB_NAME            TIT TITLE_NAME   TYPE       PUB PUBDATE
--- ------------------- --- ------------ ---------- --- ----------
P01 Pearson             T01 1977!        history    P01 2000-08-01
P03 Morgan-Kaufmann     T02 World War I  history    P03 1998-04-01
P03 Morgan-Kaufmann     T03 Database     computer   P03 2000-09-01
P01 Pearson             T05 Your Ideas   psychology P01 2001-01-01
P02 McGraw-Hill         T10 Mr President biography  P02
P02 McGraw-Hill         T13 Civil War    history    P02 1999-05-31

6 rows selected.
```



```
SQL> SELECT *
  2  FROM mypubs CROSS JOIN mytitles
     ;

PUB PUB_NAME            TIT TITLE_NAME   TYPE       PUB PUBDATE
--- ------------------- --- ------------ ---------- --- ----------
P01 Pearson             T01 1977!        history    P01 2000-08-01
P01 Pearson             T02 World War I  history    P03 1998-04-01
P01 Pearson             T03 Database     computer   P03 2000-09-01
P01 Pearson             T05 Your Ideas   psychology P01 2001-01-01
P01 Pearson             T10 Mr President biography  P02
P01 Pearson             T13 Civil War    history    P02 1999-05-31
P02 McGraw-Hill         T01 1977!        history    P01 2000-08-01
P02 McGraw-Hill         T02 World War I  history    P03 1998-04-01
P02 McGraw-Hill         T03 Database     computer   P03 2000-09-01
P02 McGraw-Hill         T05 Your Ideas   psychology P01 2001-01-01
P02 McGraw-Hill         T10 Mr President biography  P02
P02 McGraw-Hill         T13 Civil War    history    P02 1999-05-31
P03 Morgan-Kaufmann     T01 1977!        history    P01 2000-08-01
P03 Morgan-Kaufmann     T02 World War I  history    P03 1998-04-01
P03 Morgan-Kaufmann     T03 Database     computer   P03 2000-09-01
P03 Morgan-Kaufmann     T05 Your Ideas   psychology P01 2001-01-01
P03 Morgan-Kaufmann     T10 Mr President biography  P02
P03 Morgan-Kaufmann     T13 Civil War    history    P02 1999-05-31
```

```
SQL> SELECT *
  2  FROM mypubs, mytitles
     ;
```



```
SQL> SELECT *
  2  FROM mypubs CROSS JOIN mytitles;

PUB PUB_NAME            TIT TITLE_NAME   TYPE       PUB PUBDATE
--- ------------------- --- ------------ ---------- --- ----------
P01 Pearson             T01 1977!        history    P01 2000-08-01

P01 Pearson             T05 Your Ideas   psychology P01 2001-01-01

P02 McGraw-Hill         T10 Mr President biography  P02
P02 McGraw-Hill         T13 Civil War    history    P02 1999-05-31

P03 Morgan-Kaufmann     T02 World War I  history    P03 1998-04-01
P03 Morgan-Kaufmann     T03 Database     computer   P03 2000-09-01

6 rows selected.
```

```
SQL> SELECT *
  2  FROM mypubs, mytitles;
```

Inner Join

- List the names of all publishers which have published a history book:
-
```
SELECT mypubs.pub_id, pub_name
FROM mypubs
  INNER JOIN mytitles
     ON mypubs.pub_id = mytitles.pub_id
WHERE type = 'history';
```

```
PUB PUB_NAME
--- ----------------
P01 Pearson
P03 Morgan-Kaufmann
P02 McGraw-Hill
```


Qualifying Column Names

- Need to qualify column names when there is ambiguity
- Can use table names, except in self-join
- Can use an alias for a table named in the FROM clause, separated from table name with a space; AS keyword is optional
- Cannot mix – once aliases are defined, must use aliases to qualify column names

Inner Join Using Table Aliases

- List the names of all publishers which have published a history book:
-
```
SELECT p.pub_id, pub_name
FROM mypubs p
  INNER JOIN mytitles t
     ON p.pub_id = t.pub_id
WHERE type = 'history';
```

```
PUB PUB_NAME
--- ----------------
P01 Pearson
P03 Morgan-Kaufmann
P02 McGraw-Hill
```

Types of Join

- CROSS JOIN (no conditions)
- [INNER] JOIN
  - EQUI-JOIN (ON conditions are equalities)
  - THETA-JOIN (any comparison ON conditions)
- NATURAL JOIN (equality of common columns)
- SELF-JOIN (mostly to other rows of same table)
- [OUTER] JOIN
  - LEFT [OUTER] JOIN
  - RIGHT [OUTER] JOIN
  - FULL [OUTER] JOIN

INNER JOIN vs NATURAL JOIN

- SQL2 provides alternative ways for inner joins:

```
1. FROM mypubs p JOIN mytitles t
     ON p.pub_id = t.pub_id
2. FROM mypubs p JOIN mytitles t
     USING (pub_id)
3. FROM mypubs NATURAL JOIN mytitles
```

- Note1: uses FROM clause only, not WHERE clause
- Note2: 1st format produces result with 2 pub_id columns
- Note3: 2nd and 3rd formats gives result with only 1 pub_id column (eliminates duplicates)

Which JOIN Syntax?

- If result columns come from more than 1 table, must have a way to combine them.
- Older SQL-89
  - FROM : tables separated by commas
  - WHERE : the join column(s) comparisons
- Newer SQL-92
  - FROM : tables and join column(s)
  - There are multiple syntax alternatives

Which JOIN Syntax?

- Older SQL-89

```
SELECT au_fname, au_lname, a.city
  FROM authors a, publishers p
  WHERE a.city = p.city;
```

- Newer SQL-92

```
SELECT au_fname, au_lname, a.city
  FROM authors a
  INNER JOIN publishers p
    ON a.city = p.city;
```


Sorting a Join Result

- List the names of all publishers which have published a history book:
-
```
SELECT mypubs.pub_id, pub_name
FROM mypubs
  INNER JOIN mytitles
     ON mypubs.pub_id = mytitles.pub_id
WHERE type = 'history'
ORDER BY pub_name;
```

```
PUB PUB_NAME
--- ----------------
P02 McGraw-Hill
P03 Morgan-Kaufmann
P01 Pearson
```

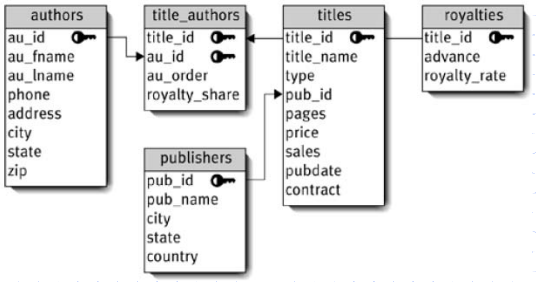

Three Table Join – SQL-89

```
SELECT t.title_id, t.pub_id,
       p.pub_name, r.advance
FROM publishers p, titles t,
     royalties r
WHERE p.pub_id = t.pub_id
  AND t.title_id = r.title_id
  AND r.advance < 20000;
```

Three Table Join – SQL-92

```
SELECT t.title_id, t.pub_id,
       p.pub_name, r.advance
FROM publishers p
  INNER JOIN titles t
     ON p.pub_id = t.pub_id
  INNER JOIN royalties r
     ON t.title_id = r.title_id
WHERE r.advance < 20000;
```


Books Database Schema



Computing an Inner Join

- **Procedure for generating the result of a join CONCEPTUALLY:**

1. Form CROSS JOIN of the tables named in FROM clause
2. If WHERE clause, apply the conditions to each row, retaining rows meeting conditions
3. Apply SELECT list to retain those columns
4. If DISTINCT, eliminate any duplicate rows
5. If ORDER BY clause, sort result table as required

A JOIN Example

- Give me a list of the authors and publishers involved in publishing computer books.

authors (au_id, au_fname, au_lname, phone, address, city, state, zip)

title_authors (title_id, au_id, au_order, royalty_share)

publishers (pub_id, pub_name, city, state, country)

royalties (title_id,advance,royalty_rate)

titles (title_id, title_name, type, pub_id, pages, price, sales, pubdate, contract)

A JOIN Example

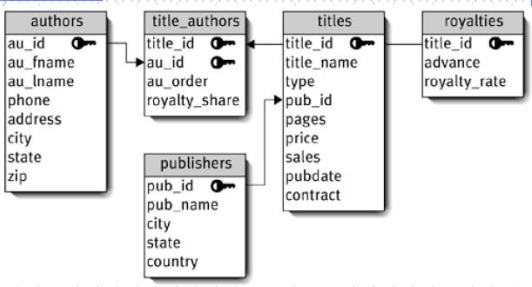
- Give me a list of the authors and publishers involved in publishing computer books.

A JOIN Example

- Give me a list of the authors and publishers involved in publishing computer books.

1. FROM ??? -- 3 questions

Give me a list of the authors and publishers involved in publishing computer books.



Outer Joins

- If one row of a joined table is unmatched, that row is omitted from result table
- Outer join operations retain rows that do not satisfy the join conditions
- Consider the following query:
  - Find the publishers who have not yet published a book.
  - Find the books without a publisher lined up as yet

Find the publishers who have not yet published a book

```
SQL> SELECT * FROM mypubs;

PUB PUB_NAME
--- --------------------
P01 Pearson
P02 McGraw-Hill
P03 Morgan-Kaufmann
P04 New World Book

SQL> SELECT * FROM mytitles;

TIT TITLE_NAME   TYPE       PUB PUBDATE
--- ------------ ---------- --- ----------
T01 1977!        history    P01 2000-08-01
T02 World War I  history    P03 1998-04-01
T03 Database     computer   P03 2000-09-01
T05 Your Ideas   psychology P01 2001-01-01
T10 Mr President biography
T13 Civil War    history    P02 1999-05-31
```

Find the publishers who have not yet published a book

- An INNER JOIN can never help to answer such queries:

```
SELECT *
FROM mypubs p
  JOIN mytitles t        -- INNER JOIN
    ON p.pub_id = t.pub_id;
```

```
PUB PUB_NAME            TIT TITLE_NAME   TYPE       PUB PUBDATE
--- ------------------- --- ------------ ---------- --- ----------
P01 Pearson             T01 1977!        history    P01 2000-08-01
P03 Morgan-Kaufmann     T02 World War I  history    P03 1998-04-01
P03 Morgan-Kaufmann     T03 Database     computer   P03 2000-09-01
P01 Pearson             T05 Your Ideas   psychology P01 2001-01-01
P02 McGraw-Hill         T13 Civil War    history    P02 1999-05-31
```


LEFT [OUTER] JOIN Example

- Find the publishers who have not yet published a book (& what is the percentage of publishers)
-
```
SELECT *
FROM mypubs p
  LEFT JOIN mytitles t   -- LEFT JOIN
    ON p.pub_id = t.pub_id
WHERE t.pub_id IS NULL;
```

```
PUB PUB_NAME            TIT TITLE_NAME   TYPE       PUB PUBDATE
--- ------------------- --- ------------ ---------- --- ----------
P01 Pearson             T01 1977!        history    P01 2000-08-01
P03 Morgan-Kaufmann     T02 World War I  history    P03 1998-04-01
P03 Morgan-Kaufmann     T03 Database     computer   P03 2000-09-01
P01 Pearson             T05 Your Ideas   psychology P01 2001-01-01
P02 McGraw-Hill         T13 Civil War    history    P02 1999-05-31
P04 New World Book
```


Outer Join Result Includes:

- INNER JOIN result rows (can have multiple matched rows)
- Additional unmatched rows with NULL-fills (only 1 row per unmatched row)
  - LEFT JOIN – extra rows from LEFT table
  - RIGHT JOIN – extra rows from RIGHT table
  - FULL JOIN – extra rows from both tables

RIGHT [OUTER] JOIN Example

- Find the books without a publisher lined up as yet
-
```
SELECT *
FROM mypubs p
  RIGHT JOIN mytitles t   -- RIGHT JOIN
    ON p.pub_id = t.pub_id
WHERE p.pub_id IS NULL;
```

```
PUB PUB_NAME            TIT TITLE_NAME    TYPE       PUB PUBDATE
--- ------------------- --- ------------- ---------- --- ----------
P01 Pearson             T01 1977!         history    P01 2000-08-01
P03 Morgan-Kaufmann     T02 World War I   history    P03 1998-04-01
P03 Morgan-Kaufmann     T03 Database      computer   P03 2000-09-01
P01 Pearson             T05 Your Ideas    psychology P01 2001-01-01
P02 McGraw-Hill         T13 Civil War     history    P02 1999-05-31
                        T10 Mr President  biography
```


FULL [OUTER] JOIN Example

- I can match the publishers who have not yet published a book with books without a publisher
-
```
SELECT *
FROM mypubs p
  FULL JOIN mytitles t     -- FULL JOIN
    ON p.pub_id = t.pub_id
WHERE      ;               -- what ???
```

```
PUB PUB_NAME            TIT TITLE_NAME    TYPE       PUB PUBDATE
--- ------------------- --- ------------- ---------- --- ----------
P01 Pearson             T01 1977!         history    P01 2000-08-01
P03 Morgan-Kaufmann     T02 World War I   history    P03 1998-04-01
P03 Morgan-Kaufmann     T03 Database      computer   P03 2000-09-01
P01 Pearson             T05 Your Ideas    psychology P01 2001-01-01
P02 McGraw-Hill         T13 Civil War     history    P02 1999-05-31
P04 New World Book
                        T10 Mr President  biography
```


Self Join

```
emp_id emp_name           boss_id
------ ----------------   -------
E01    Lord Copper        NULL
E02    Jocelyn Hitchcock  E01
E03    Mr. Salter         E01
E04    William Boot       E03
E05    Mr. Corker         E03
```

- List each employee with his/her boss
-
```
SELECT e1.emp_name AS "Employee",
       e2.emp_name AS "Boss"
   stf.staffNo, stf.fName, stf.lName
FROM employee e1
  JOIN employee e2
    ON e1.boss_id = e2.emp_id;
```

Self Join

- List each employee with his/her boss
-
```
SELECT e1.emp_name AS "Employee",
       e2.emp_name AS "Boss"
    stf.staffNo, stf.fName, stf.lName
FROM employee e1
  JOIN employee e2
    ON e1.boss_id = e2.emp_id;
```

```
Employee            Boss
----------------    ------------------
Jocelyn Hitchcock   Lord Copper
Mr. Salter          Lord Copper
William Boot        Mr. Salter
Mr. Corker          Mr. Salter
```


Self Join

- List each employee with his/her boss
-
```
SELECT e1.emp_name AS "Employee",
       e2.emp_name AS "Boss"
    stf.staffNo, stf.fName, stf.lName
FROM employee e1
  LEFT JOIN employee e2
    ON e1.boss_id = e2.emp_id;
```

```
Employee            Boss
----------------    ------------------
Lord Copper
Jocelyn Hitchcock   Lord Copper
Mr. Salter          Lord Copper
William Boot        Mr. Salter
Mr. Corker          Mr. Salter
```