INTRODUCTION TO STRUCTURED QUERY LANGUAGE

by
Justin Burruss

Presented at
General Atomics
San Diego, California

June 2000

109-00/df

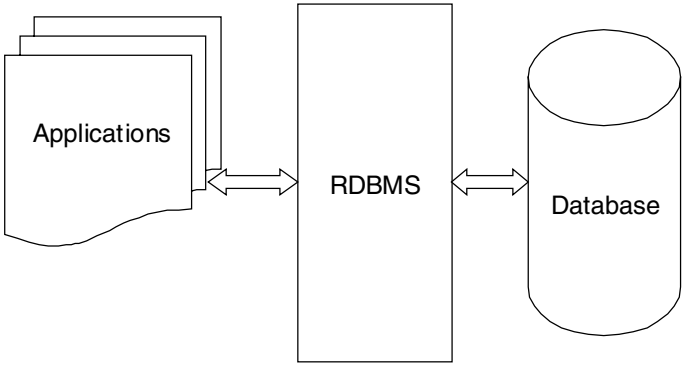
DIII-D
NATIONAL FUSION FACILITY
SAN DIEGO

OUTLINE

1. Relational Databases
2. Structured Query Language
3. References

A RELATIONAL DATABASE MANAGES DATA

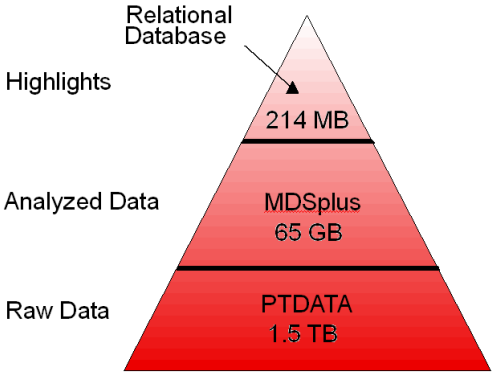
- Applications do not interact with the data directly but instead access the database through the Relational Database Management System (RDBMS).



- A relational database works in concert with MDSplus/PTDATA.
  - MDSplus is not optimized for queries across multiple shots.
  - MDSplus is hierarchical, but not relational.
  - MDSplus/PTDATA stores all the data.

- A relational database stores highlights of the data.
  - Optimized for queries (e.g. what shots had plasma in 1999?)

- You can survey, then drill down for more detailed information.



RELATIONAL DATABASES STRUCTURE DATA INTO RELATIONS

- A relational database is a set of relations...
- A relation is a set of tuples...
- A tuple is a set of attributes.

- In more familiar terms:
  - A relational database is a set of tables...
  - A table is a set of rows...
  - A row is a set of fields.

- Tables in a database should be related. These example tables are related through the owner and username columns.

Computers

Computer Id	Type	Owner
101	DEC Alpha	keithk
102	NT Server	parker
103	NT Server	meyer
104	SPARC-2	keithk

Users

Username	First Name	Last Name
keithk	Kristi	Keith
parker	Carl	Parker
meyer	William	Meyer

A RELATIONAL DATABASE IS STRUCTURED

- Each table is unique—no two tables may have the same name.
- Each row in a table is unique—no two rows in a table may be the same. One or more columns in the row should uniquely identify that row. This unique identifier is called a **primary key**.
- Each field is complete value (no pointers or derived values).
- An empty field has a well defined value: **null**. Null is not the same as an empty string or zero—null is a distinct value.
- Each table should be related to other tables in the database (if its unrelated you should put it in a different database).
- You can access any value using the table name, column name, and the value of the primary key that defines the row in which it is stored.

SQL IS THE LANGUAGE USED BY ALL LEADING RELATIONAL DATABASE MANAGEMENT SYSTEMS

- SQL (Structured Query Language) was developed in the 1970s.
- It was standardized by ANSI and ISO in the 1980s.
- It is supported by all major database vendors.
- It is a declarative language (and thus easier to use than procedural languages).
- It is used for:
  - Building databases
  - Storing data
  - Retrieving data
  - Managing databases
- We will only discuss retrieving data...

USE SELECT TO RETRIEVE DATA

```
SELECT  columns you want
  FROM  table
```

- Examples:

```
SELECT  shot, time_of_shot, pulse_length
  FROM  summaries

SELECT  *
  FROM  shots
```

- The * is a shortcut for selecting all columns in a table.
- You can use the DISTINCT keyword to remove duplicate values.
- Example:

```
SELECT DISTINCT  shot
          FROM  entries
```

USE THE WHERE CLAUSE TO SPECIFY WHICH ROWS YOU WANT

```
SELECT  columns you want
  FROM  table
 WHERE  condition
```

- **You can use the following operands in your condition:**

=	equal to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
<>	not equal to
IS NULL	equal to null
BETWEEN *x* AND *y*	between *x* and *y* inclusive
IN(s_1, s_2, ..., s_n)	in the set *s*

- **Examples:**

```
SELECT  first_name, last_name
  FROM  personnel
 WHERE  uid > 500

SELECT  first_name, last_name
  FROM  personnel
 WHERE  uid IN( 315, 316, 708 )
```

YOU CAN USE LIKE WHEN DEALING WITH STRINGS

- LIKE lets you match strings. You can use the % wildcard to match 0 or more characters. The _ wildcard will match exactly one character.

- Examples:
  - ‘fusion’ matches ‘FUSION’ and ‘fusion’
  - ‘%ion’ matches ‘fusion’, ‘cold fusion’, and ‘a red lion’
  - ‘_ion’ matches ‘lion’ but not ‘fusion’

- This example finds people with the name ‘Peterson’ and ‘Petersen’:

```
SELECT  first_name, last_name, job
  FROM  people
 WHERE  last_name LIKE('peters_n')
```

- Results:

first_name	last_name	job
Chris	Petersen	2nd Baseman, Chicago Cubs
Peter	Petersen	Assistant Program Director, DIII-D Program
Cassandra	Peterson	Actress
Peter	Peterson	Chairman, Council on Foreign Relations

USE AND, OR & NOT TO SPECIFY MULTIPLE CONDITIONS

- **Examples:**

```
SELECT  shot, time_of_shot, pulse_length
  FROM  summaries
 WHERE  ip > 1000000
    OR  btor > 2

SELECT  *
  FROM  entries
 WHERE  topic = 'BEAMS'
   AND  username = 'phillips'

SELECT  *
  FROM  summaries
 WHERE  patch_panel = '1.5DNBUP5'
   AND  ip >= 1500000
   AND  ( btorsign = -1 OR  btormax > 2.05 )
   AND  kappa BETWEEN 1.5 AND 1.8
AND NOT pulse_length < 3.5
```

USE AGGREGATE FUNCTIONS TO DO SOME SIMPLE MATH

- **Aggregate Functions:**

COUNT(*x*)	Count non-null occurrences of *x*
SUM(*x*)	sum of *x*
AVG(*x*)	average of *x* (ignoring null values)
MIN(*x*)	minimum *x*
MAX(*x*)	maximum *x*

- **Examples:**

```
SELECT  COUNT(shot)
  FROM  shots

SELECT  MAX(ip)
  FROM  summaries

SELECT  MAX(ip) / 1000000
  FROM  summaries
```

- **Note: the / 1000000 just divides the result by 1000000.**

USE ORDER BY TO SORT YOUR RESULTS

- You may choose to sort your query results using ORDER BY.

```
SELECT  columns
  FROM  table
ORDER BY  criteria
```

- You may use the ASC and DESC keywords to specify ascending or descending order.

- Examples:

```
SELECT  shot, a, r, kappa
  FROM  summaries
ORDER BY  shot DESC

SELECT  first_name, last_name
  FROM  personnel
ORDER BY  last_name ASC
```

USE GROUP BY TO GROUP YOUR QUERY RESULTS

- The GROUP BY clause lets you group your results based on the criteria you supply.

```
   SELECT  columns
     FROM  table
 GROUP BY  criteria
```

- This example finds the number of males and females in the people table:

```
   SELECT  sex, count(last_name)
     FROM  people
 GROUP BY  sex
```

- Results:

sex	
m	220
f	216

USE THE HAVING CLAUSE TO APPLY A SEARCH CONDITION TO GROUPS

- The HAVING clause is used to apply search conditions to groups.

```
SELECT   columns
  FROM   table
GROUP BY criteria
HAVING   condition
```

- Example:

```
SELECT   shot, COUNT(shot)
  FROM   entries
GROUP BY shot
HAVING   COUNT(shot) > 15
```

- Results:

shot	
98303	22
98777	16

USE JOINS WHEN YOU NEED DATA FROM TWO OR MORE TABLES

- It is often necessary to look in multiple tables for the data you need. To get data from more than one table, use joins.
- A join combines two or more tables into a single (larger) table.
- Example:

Users

Username	First Name	Last Name
keithk	Kristi	Keith
parker	Carl	Parker
meyer	William	Meyer

Computers

Computer Id	Type	Owner
101	DEC Alpha	keithk
102	NT Server	parker
103	NT Server	meyer
104	SPARC-2	keithk

Users joined with Computers where owner = username

Username	First Name	Last Name	Computer Id	Type
keithk	Kristi	Keith	101	DEC Alpha
parker	Carl	Parker	102	NT Server
meyer	William	Meyer	103	NT Server
keithk	Kristi	Keith	104	SPARC-2

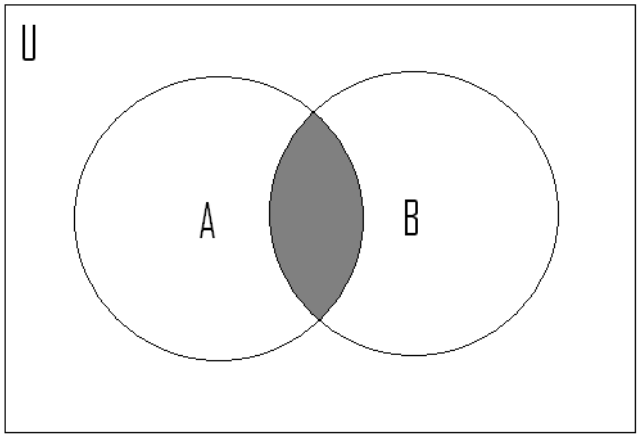
AN INNER JOIN IS LIKE AN INTERSECT

Users

Username	First Name	Last Name
keithk	Kristi	Keith
parker	Carl	Parker
meyer	William	Meyer
schacht	Jeff	Schachter

Computers

Computer Id	Type	Owner
101	DEC Alpha	keithk
102	NT Server	parker
103	NT Server	meyer
104	iMac	nobody
105	SPARC-2	keithk



A INNER JOIN B

Users inner joined with Computers

Username	First Name	Last Name	Computer Id	Type
keithk	Kristi	Keith	101	DEC Alpha
parker	Carl	Parker	102	NT Server
meyer	William	Meyer	103	NT Server
keithk	Kristi	Keith	104	SPARC-2

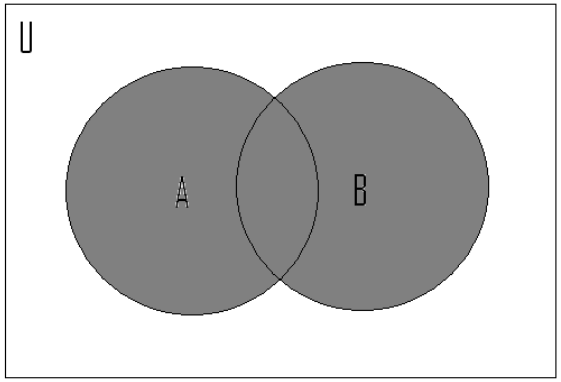
OUTER JOINS ARE LIKE UNIONS

Users

Username	First Name	Last Name
keithk	Kristi	Keith
parker	Carl	Parker
meyer	William	Meyer
schacht	Jeff	Schachter

Computers

Computer Id	Type	Owner
101	DEC Alpha	keithk
102	NT Server	parker
103	NT Server	meyer
104	iMac	null
105	SPARC-2	keithk



A OUTER JOIN B

Users outer joined with Computers

Username	First Name	Last Name	Computer Id	Type
keithk	Kristi	Keith	101	DEC Alpha
parker	Carl	Parker	102	NT Server
schacht	Jeff	Schachter	null	null
null	null	null	104	iMac
meyer	William	Meyer	103	NT Server
keithk	Kristi	Keith	104	SPARC-2

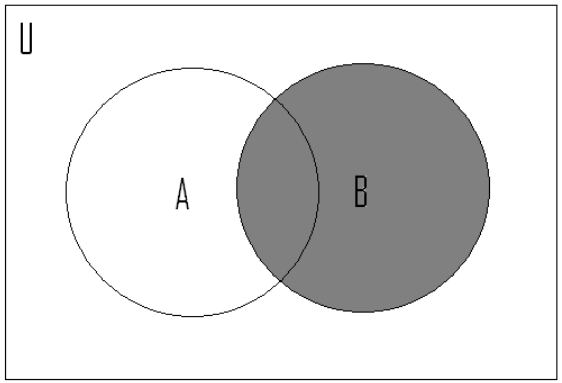
"A RIGHT OUTER JOIN B" GRABS ALL OF B, BUT ONLY THE PART OF A THAT MATCHES B

Users

Username	First Name	Last Name
keithk	Kristi	Keith
parker	Carl	Parker
meyer	William	Meyer
schacht	Jeff	Schachter

Computers

Computer Id	Type	Owner
101	DEC Alpha	keithk
102	NT Server	parker
103	NT Server	meyer
104	iMac	null
105	SPARC-2	keithk



A RIGHT OUTER JOIN B

Users outer joined with Computers

Username	First Name	Last Name	Computer Id	Type
keithk	Kristi	Keith	101	DEC Alpha
parker	Carl	Parker	102	NT Server
null	null	null	104	iMac
meyer	William	Meyer	103	NT Server
keithk	Kristi	Keith	105	SPARC-2

THE SQL2 SYNTAX FOR JOINS USES KEYWORDS

- SQL2 syntax for inner join:

```
SELECT  columns
  FROM  table1 INNER JOIN table2
    ON  table1.keycolumn = table2.keycolumn
```

- Example:

```
SELECT  shots.shot, shot_type, time_of_shot
  FROM  shots INNER JOIN summaries
    ON  shots.shot = summaries.shot
```

- Notice that we use *tablename.columnname* to indicate which column we are referring to.

- Join keywords are: INNER, LEFT OUTER, RIGHT OUTER, and FULL OUTER.

- Example:

```
SELECT  shots.shot, shot_type, ip
  FROM  shots LEFT OUTER JOIN summaries
    ON  shots.shot = summaries.shot
```

- There is an older syntax (SQL1) for joins that we'll save for another discussion.

SUBQUERIES CAN BE NESTED INSIDE OF QUERIES

- A subquery is any query embedded inside another query.

- Examples:

```
SELECT  *
  FROM  shots
 WHERE  shot = ( SELECT MAX(shot) FROM summaries )

SELECT  run, shot
  FROM  shots
 WHERE  shot IN ( SELECT shot FROM summaries )
```

WE'VE DISCUSSED MOST OF THE QUERYING FEATURES OF SQL

- **We've discussed:**
  - SELECT
  - WHERE
  - LIKE
  - AND, OR, and NOT
  - GROUP BY
  - HAVING
  - ORDER BY
  - The most common types of Joins

- **We will leave these SQL features for another discussion:**
  - Cross and Union Joins
  - SQL1 Syntax Joins
  - Aliases
  - EXISTS, ANY, and ALL (subquery tests)
  - Unions
  - Indices & Views
  - Updates & Deletions
  - Creating Tables
  - SQL Security

REFERENCES

Groff, James R., Weinber, Paul N., LAN Times Guide to SQL, (Osborne MacGraw-Hill, Berkeley, California, 1994)

Codd, E.F., "A Relational Model of Data for Large Shared Data Banks", reprinted from Communications of the ACM, Vol. 13, No. 6 (1970) 377.

http://d3dnff.gat.com/D3DRDB/resources.html