SQL
A Query Language for Relational Databases

History
- Structured Query Language, invented in 1970s at IBM Research Labs, San Jose
- Has evolved, acquiring more and more features
- SQL 92 is widely supported at various levels.
- SQL 1999 (ANSI and ISO) is current standard.

A Sample Database

<table>
<thead>
<tr>
<th>student</th>
<th>sid</th>
<th>sname</th>
<th>yog</th>
<th>gpa</th>
<th>sex</th>
<th>major</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>Larry</td>
<td>grad</td>
<td>3.5</td>
<td>M</td>
<td>cs</td>
<td>cs</td>
</tr>
<tr>
<td>s2</td>
<td>Moo</td>
<td>2005</td>
<td>3.2</td>
<td>M</td>
<td>math</td>
<td>math</td>
</tr>
<tr>
<td>s3</td>
<td>Curly</td>
<td>2004</td>
<td>4</td>
<td>M</td>
<td>cs</td>
<td>cs</td>
</tr>
<tr>
<td>s4</td>
<td>Shemp</td>
<td>2000</td>
<td>M</td>
<td>cs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A Sample Database, cont.

<table>
<thead>
<tr>
<th>course</th>
<th>cid</th>
<th>sec_no</th>
<th>semest</th>
<th>instruct room</th>
<th>time_slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>201</td>
<td>I2005</td>
<td>f1</td>
<td>Olsen 420</td>
<td>T 5-7:30</td>
</tr>
<tr>
<td>c2</td>
<td>202</td>
<td>I2005</td>
<td>f1</td>
<td>Olsen 399</td>
<td>T 5-7:30</td>
</tr>
<tr>
<td>c3</td>
<td>201</td>
<td>I2005</td>
<td>f3</td>
<td>Olsen 115</td>
<td>M 3-4:30</td>
</tr>
<tr>
<td>c4</td>
<td>201</td>
<td>I2005</td>
<td>f4</td>
<td>Olsen 333</td>
<td>Th 8-10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enr</th>
<th>sid</th>
<th>cid</th>
<th>sec_no</th>
<th>semest</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>c1</td>
<td>201</td>
<td>I2004</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>s2</td>
<td>c2</td>
<td>202</td>
<td>I2005</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>s3</td>
<td>c1</td>
<td>201</td>
<td>I2004</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>s4</td>
<td>c2</td>
<td>202</td>
<td>I2005</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

CREATE TABLE

```sql
CREATE TABLE student
    (sid CHAR(5),
     sname CHAR(16),
     yog CHAR(4),
     gpa DECIMAL(3,2),
     sex CHAR(1),
     major CHAR(16),
     PRIMARY KEY (sid))
```

CREATE TABLE, cont’d

```sql
CREATE TABLE enr
    (sid CHAR(5),
     cid CHAR(5),
     sec_no CHAR(5),
     semester CHAR(4),
     grade CHAR(2),
     PRIMARY KEY (sid, cid, sec_no, semester),
     FOREIGN KEY (sid) REFERENCES student,
     FOREIGN KEY (cid, sec_no, semester) REFERENCES course_off)
```
Domains

CREATE DOMAIN SID_DOMAIN CHAR(5);
CREATE DOMAIN SNAME_DOMAIN CHAR(16);
CREATE TABLE student
(sid  SID_DOMAIN,
sname SNAME_DOMAIN, ...
);

Domain Constraints

CREATE DOMAIN SEX_DOMAIN CHAR(1)
CONSTRAINT CHECK_SEX
CHECK ( VALUE IN ('m', 'f'))

DROP TABLE

DROP TABLE student

removes the student table, including schema

Indices (not part of SQL standards)

CREATE CLUSTERED INDEX StudentInd
ON student (sid)
builds an index on student.sid

DROP StudentInd
deletes the index

Updates

INSERT INTO student
(sid, sname, yog, gpa, sex, major)
VALUES
('s1', 'Larry', 'grad', 4.0, 'm', 'cs');

UPDATE student
SET major = 'cs'
WHERE gpa = 4.0;

DELETE FROM student
WHERE yog = '2005';

Syntax - The SELECT Statement

SELECT [DISTINCT] column_list
FROM tablename_list
[ WHERE search_condition ]
[ GROUP BY column_list ]
[ HAVING group_condition ]
[ ORDER BY column_list ]
Select Statement

Give the names of Computer Science (CS) majors.

```sql
SELECT sname
FROM student
WHERE major = 'cs'
```

Select Statement, cont.

Give the names of students who have received at least one grade of 'A'.

```sql
SELECT DISTINCT student.sname
FROM student, enr
WHERE enr.grade = 'a'
AND student.sid = enr.sid
```

Select Statement, cont.

Give the names of students who have received at least one grade of 'A'.

```sql
SELECT student.sname
FROM student, enr
WHERE enr.grade = 'a'
AND student.sid = enr.sid
```

UNIONs

Give the cid's of courses that either have prerequisites or are prerequisites for other courses or both.

```sql
SELECT cid FROM prereq
UNION
SELECT prereq_cid FROM prereq
```

Relation Aliases

Give names of courses that have prerequisites and the names of the prerequisites.

```sql
SELECT c1.cname AS course_name,
c2.cname AS prereq_name
FROM course c1, prereq, course c2
WHERE c1.cid = prereq.cid
AND prereq.prereq_cid = c2.cid
```

SQL Views

Create a view that joins together students with their enrollments.

```sql
CREATE VIEW student_enr
AS
SELECT sid, sname, major, yog,
cid, sec_no, semester, grade
FROM student, enr
WHERE student.sid = enr.sid
```
Using Views

Give the names of students who have received at least one grade of ‘A’.

```sql
SELECT sname
FROM student_enr
WHERE grade = 'a'
```

Pattern Matching

Give IDs of courses named "Database something-or-other".

```sql
SELECT cid
FROM course
WHERE cname LIKE 'Database%'
```

Pattern Matching, cont.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%</code></td>
<td>0 or more characters</td>
<td>'John%' (a string beginning with 'John')</td>
</tr>
<tr>
<td><code>_</code></td>
<td>any one character</td>
<td>'a_' ('a0' or 'a1' or _ 'aa' or _ )</td>
</tr>
<tr>
<td><code>[ ]</code></td>
<td>one character in the brackets</td>
<td>'a[bcdf]' ('ab' or 'ac' or 'ad')</td>
</tr>
</tbody>
</table>

GROUP BY

Give each (enrolled) student's sid and number of courses he has been enrolled in.

```sql
SELECT sid,
       COUNT(DISTINCT cid) AS num
FROM enr
GROUP BY sid
```

HAVING

Give the sid's of students who have been enrolled in more than two different courses.

```sql
SELECT sid
FROM enr
GROUP BY sid
HAVING COUNT(DISTINCT cid) > 2
```

JOIN

Give the names of students who have received at least one grade of ‘A’.

```sql
SELECT sname
FROM student JOIN enr
ON student.sid = enr.sid
HAVING enr.grade = 'a'
```
OUTER JOIN

Tell me everything about students and their enrollments. Include students who don't have any enrollments.

```
SELECT *
FROM student LEFT OUTER JOIN enr
ON student.sid = enr.sid
```

A Left Outer Join

```
+---+-------+---+-----+-------+---+-------+---------+---+
| st.sid | name   | year | gpa | sex major | enr.sid | cid    | sec sem | grade |
|-------+--------+------+-----+----------+--------+--------+---------+-------|
| s1    | Larry  | grad | 3.5 | M cs      | s1     | c1     | 201     | 2004  |
| s1    | Larry  | grad | 3.5 | M cs      | x1     | c2     | 202     | 2005  |
| s2    | Moe    | 2003 | 3.2 | M math    | s2     | c1     | 201     | 2004  |
| s2    | Moe    | 2005 | 3.2 | M math    | x2     | c2     | 202     | 2005  |
| s2    | Moe    | 2005 | 3.2 | M math    | x2     | c3     | 201     | 2005  |
| s2    | Moe    | 2005 | 3.2 | M math    | x4     | c4     | 201     | 2005  |
| s3    | Curly  | 2004 | 4   | M cs      | s3     | c1     | 201     | 2005  |
| s3    | Curly  | 2004 | 4   | M cs      | x3     | c2     | 202     | 2000  |
| s4    | Shemp  | 2000 | M   | physics   |
+---+-------+---+-----+-------+---+-------+---------+---+
```

Nesting SELECT Blocks

Give the names of students who have enrolled in course ‘c1’.

```
SELECT name
FROM student
WHERE sid IN
    (SELECT sid
     FROM enr
     WHERE cid = 'c1')
```

SQL:1999

- Previously known as SQL3
- Much new functionality added 😊
- Nothing taken away 😊
- Most important extensions deal with user-defined types, complex types, inheritance, and other “object-oriented” features

Competing Standard APIs: ODBC and JDBC

- Function calls are standardized, unlike embedded SQL.
- Access to different data sources is mediated by ODBC/JDBC drivers.
- Unlike with embedded SQL, binding to database system is done at run-time.

Conclusions

- SQL is an extremely ugly language. Alternatives such as Quel are cleaner.
- SQL:1999 adds many new features. It’s clearly designed by committee. SQL will continue to grow.
- SQL is here to stay. Current work includes extensions to query XML data.