Objectives

After completing this chapter, you should be able to:

- write advanced queries in SQL including several tuple variables over the same relation.
- enumerate the main constructs of SQL, at least of the SQL-86 standard (e.g., all kinds of conditions).
- evaluate the portability of certain constructs.
- evaluate the syntactical correctness of a given query.

SQL

Today, SQL is the only database language for relational database management systems (industry standard).

Today, SQL is the only database language for relational databases over the same relation.

Overview

1. General Remarks about SQL
2. Lexical Syntax
3. SELECT-FROM-WHERE, Tuple Variables
4. Terms and Conditions
5. NULL Values

SQL Tod ay, SQL is the only database language for relational database management systems (industry standard). RDBMS today must offer an SQL interface. SQL is used for:

- Interactive "ad-hoc" commands and application program development (embedded into other languages like C, Java, HTML, PL/SQL).
- All constructs from the SQL-86 standard are treated in the SQL-86 standard (e.g., all kinds of conditions).
- NOT EXISTS / NOT IN are treated in Part 10.
- Aggregate functions / HAVING are well supported. Relevant papers include O’Gorman's "SQL/DW in Perspective."
History
SEQUEL, an earlier version of SQL, was designed by Chamberlin, Boyce, et al. at IBM Research in 1974.

SEQUEL stands for Structured English Query Language.

Some people pronounce SQL this way. Others use "ess-cue-ell." The name was changed for legal reasons.

SQL was the language of System/R (IBM, 1976/77).

System/R was a very influential research product.

First commercial systems supporting SQL were Oracle (1979) and IBM SQL/DS (1981).

SQL was the Influence of System/R (IBM, 1976/77).

Codd was also in San Jose when he invented the relational model in 1970.

"SELECT, FROM, WHERE, GROUP BY." These are the basic commands for relational databases.

Some people pronounce SQL the way Codd used to pronounce English Query language.

SEQUEL stands for Structured English Query Language.


SEQUEL, in earlier version of SQL, was designed by...
### Identifiers (1)

- **Whitespace and Comments**
  - White space is allowed between words (tokens):
    - Spaces
    - Tabulator characters
    - Line breaks
  - Comments:
    - `/*` to `*/`
    - `--` to `--`

- **Identifiers**
  - Identifiers are used as table names and column names.
  - Identifiers are not case sensitive.
  - SQL/*Plus converts all letters outside quotes to uppercase.

#### SQL Standard, Oracle, SQL Server, IBM DB2

<table>
<thead>
<tr>
<th>Length</th>
<th>First Character</th>
<th>Other Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-8</td>
<td>A-Z, a-z</td>
<td>Any character except <code>&quot;</code></td>
</tr>
<tr>
<td>9-127</td>
<td>A-Z, a-z, 0-9</td>
<td>Any character except <code>&quot;</code></td>
</tr>
<tr>
<td>128</td>
<td>A-Z, a-z, 0-9, <code>.</code></td>
<td>Any character except <code>&quot;</code></td>
</tr>
<tr>
<td>129</td>
<td>A-Z, a-z, 0-9, <code>@</code></td>
<td>Any character except <code>&quot;</code></td>
</tr>
<tr>
<td>130</td>
<td>A-Z, a-z, 0-9, <code>#</code></td>
<td>Any character except <code>&quot;</code></td>
</tr>
<tr>
<td>131</td>
<td>A-Z, a-z, 0-9, <code>$</code></td>
<td>Any character except <code>&quot;</code></td>
</tr>
</tbody>
</table>

- Identifiers can have up to 18 characters (at least).

#### Delimited Identifiers (1)

- It is possible to use any sequence of characters in double quotes.
  - Such identifiers are case-sensitive.
  - There are no conflicts with reserved words.
  - SQL/*Plus does not contain them.

- Names must be different from all reserved words.

### Identifiers (2)

- There are a lot of reserved words, e.g., `ALTER`, `CREATE`, `DROP`, `GRANT`, `REVOKE`, `USE`.

- Identifiers (and keywords) are not case sensitive.

- SQL/*Plus converts all letters outside quotes to uppercase.

- SQL/*Plus converts all letters outside quotes to uppercase.

- `"id/",/"` is a valid identifier.

### Delimited Identifiers (2)

- Any character except `"`.

### White Space and Comments

- White space is allowed between words (tokens):
Other Constants

Character String Constants

• A sequence of characters enclosed in single quotes, e.g. 'abc'.
  • Single quotes in a string must be doubled, e.g. 'John''s'.
  • Any character except ' can be inside string constants.
  • In Oracle, SQL Server, and DB2 allow line breaks inside strings.
  • Microsoft SQL Server accepts also strings that are enclosed in
    double quotes. This does not conform to the standard.
  • Oracle does not allow line breaks in strings.
  • The SQL-92 standard allows splitting strings between these
    double quotes.

Character String Constants (2)

Delimited Identifiers

• Do not confuse delimited identifiers with string constants.
  • See above for "set quoted-identifiers on".
  • Double quotes. This does not conform to the standard.
  • Microsoft SQL Server accepts also strings that are enclosed in
    double quotes.
  • Oracle, SQL Server, and DB2 allow line breaks inside strings.
  • Strings can be combined with the concatenation operator.
  • Neither Oracle nor SQL Server support this, however.
  • Each segment must be enclosed in single quotes.
  • The SQL-92 standard allows splitting strings between these
    double quotes.

Delimited Identifiers (2)

Number Constants/Literals:

• [ ]
  • [ ]
  • [ ]
  • [ ]
  • [ ]
  • [ ]
• [ ]
• [ ]
  • If the decimal point is present, there must be at least one
  • E/.g./: 1.2, -3.4, .67e-8
  • The base can be any single character.
  • Microsoft SQL Server accepts also string literals enclosed in
    double quotes.
  • This does not conform to the standard.

Course on Database Systems
Stefan Brass, 2000
Overview

General Remarks about SQL

Lexical Syntax

SELECT FROM WHERE, Tuple Variables

Terms and Conditions

Null Values

Example Database (1)

Example Database (2)

Basic Query Syntax (1)

The basic SQL query (extensions follow) has the form:

\[
\text{SELECT } A_1, \ldots, A_n \text{ FROM } R_1, \ldots, R_m \text{ WHERE } \theta\]

It corresponds to the following algebra expression:

\[
\pi_{A_1, \ldots, A_n} (\sigma_{\theta} (R_1 \times \ldots \times R_m))
\]

Example:

\[
\text{SELECT INSTRUCTOR FROM COURSES WHERE TOPIC = 2710}
\]

SELECT

A_1, \ldots, A_n

FROM

R_1, \ldots, R_m

WHERE

\theta

The Example Database (1) contains:

<table>
<thead>
<tr>
<th>SSN</th>
<th>FIRST</th>
<th>LAST</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Smith</td>
<td></td>
</tr>
<tr>
<td>Ann</td>
<td>Miller</td>
<td></td>
</tr>
<tr>
<td>David</td>
<td>Meyer</td>
<td></td>
</tr>
<tr>
<td>Mary</td>
<td>Jones</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CRN</th>
<th>SSN</th>
<th>CN</th>
<th>FIRST</th>
<th>LAST</th>
</tr>
</thead>
<tbody>
<tr>
<td>22332</td>
<td>3893</td>
<td>2711</td>
<td>John</td>
<td>Smith</td>
</tr>
<tr>
<td>37339</td>
<td>41590</td>
<td>2711</td>
<td>David</td>
<td>Meyer</td>
</tr>
<tr>
<td>36439</td>
<td>23560</td>
<td>2710</td>
<td>Smith</td>
<td>Jones</td>
</tr>
<tr>
<td>36894</td>
<td>26100</td>
<td>2710</td>
<td>Mary</td>
<td>Jones</td>
</tr>
<tr>
<td>22334</td>
<td>41590</td>
<td>2711</td>
<td>David</td>
<td>Meyer</td>
</tr>
</tbody>
</table>

Example:

\[
\text{SELECT ENROLLMENTS.CRN, STUDENTS.SSN FROM ENROLLMENTS, STUDENTS WHERE STUDENTS.SSN = ENROLLMENTS.SSN}
\]

Example Database (2)

<table>
<thead>
<tr>
<th>CRN</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2688</td>
<td>DBM</td>
</tr>
<tr>
<td>3739</td>
<td>OOP</td>
</tr>
<tr>
<td>4159</td>
<td>DS</td>
</tr>
<tr>
<td>4196</td>
<td>DBM</td>
</tr>
<tr>
<td>2321</td>
<td>DSE</td>
</tr>
<tr>
<td>2710</td>
<td>DBM</td>
</tr>
</tbody>
</table>

Course on Database Systems

Stefan Brass

Edition 2000

5. Null Values

4. Terms and Conditions

3. SELECT FROM WHERE, Tuple Variables

2. Lexical Syntax

1. General Remarks about SQL

Overview
Basic Query Syntax

A join must be done by including the join condition in the FROM clause.

Forgetting the join condition is a common error.

If no projection is needed, instead of `/*` or `;;`:

```sql
SELECT /* FROM COURSES
```

If no selection is needed, the WHERE-clause can be omitted.

E.g., to list the complete table `COURSES`:

```sql
SELECT * FROM COURSES;
```

Oracle provides a relation `DUAL` which has only one row:

```sql
SELECT TO_CHAR(SQRT(2)) FROM DUAL
```

In SQL Server, `FROM` can be omitted:

```sql
SELECT C.INSTRUCTOR FROM COURSES WHERE C.TOPIC = 210;
```

SELECT-CorT-Expression (Simplified):

```sql
SELECT C.INSTRUCTOR FROM COURSES WHERE C.TOPIC = 210
```

Tuple Variables

The FROM clause can be understood as declaring variables ranging over all tuples of a relation:

```sql
SELECT C.INSTRUCTOR FROM COURSES WHERE C.TOPIC = 210
```

A tuple variable is always created if not given a name explicitly. It will have the name of the relation explicitly. It will have the name of the relation.

Oracle provides a relation `DUAL` which has only one row:

```sql
SELECT TO_CHAR(SQRT(2)) FROM DUAL
```

In SQL Server, `FROM` can be omitted:

```sql
SELECT C.INSTRUCTOR FROM COURSES WHERE C.TOPIC = 210
```

SELECT-CorT-Expression (Simplified):

```sql
SELECT C.INSTRUCTOR FROM COURSES WHERE C.TOPIC = 210
```

Tuple Variables

The FROM clause can be understood as declaring variables ranging over all tuples of a relation:

```sql
SELECT C.INSTRUCTOR FROM COURSES WHERE C.TOPIC = 210
```

A tuple variable is always created if not given a name explicitly. It will have the name of the relation.

Oracle provides a relation `DUAL` which has only one row:

```sql
SELECT TO_CHAR(SQRT(2)) FROM DUAL
```

In SQL Server, `FROM` can be omitted:

```sql
SELECT C.INSTRUCTOR FROM COURSES WHERE C.TOPIC = 210
```

SELECT-CorT-Expression (Simplified):

```sql
SELECT C.INSTRUCTOR FROM COURSES WHERE C.TOPIC = 210
```
TupleVariables

If at tuple variable name is explicitly declared, e.g.,

`COURSES`.

Another way to understand `FROM COURSES DISTINCT` is:

`COURSES`.

Tables of other users can be referenced in the FROM-list with

TableNames

Tablesofotheruserscanbereferencedinthe

FROM

Listwith

thefollowingsyntax/(ifreadpermissionwasgranted)/:

```
SELECT/*
FROM sbrass.DEPT
```

The username is really a name of a database schema

(onedatabaseinstancecanmanageseveralschemas)/.

InOracle/,schemaanduseraremoreorlessthesame/:

Everyuserhashis//herownschema/,everyschema/has

a user, and user is more or less the same:

(One database instance can manage several schemas).

InOracle/, the database is really a name of a database schema.

InSQLServer/, a fully qualified name has the form

```
\server/.database/.owner/.table/
```

There are various abbreviations including

```
server|database|owner|table
```

Forjoinsinthe FROM-list (not in Oracle) see Part 10.

Variables ranging over the rows of the same table.

Variables which match be less natural than thinking about two
tables, which may be less natural than thinking about two

In SOL-92, SOL-server, and VAPID in SOL-92 and DB2

Variables ranging over the rows of the same table.

If at tuple variable name is explicitly declared, e.g.,

`COURSES`.
Joins

Joins (4)

Joins

Joins (3)

Joins (2)

Joins (1)
Attributes can be accessed in the form Variable.Attribute. If only one variable has this attribute, the variable name can be left out, e.g., LAST. So the following query is legal:

```
SELECT CRN FROM STUDENTS, ENROLLMENTS, COURSES
WHERE S.SSN = E.SSN AND E.CRN = C.CRN
AND FIRST = 'Ann' AND LAST = 'Miller'
```

But:

```
SELECT C.CRN, TITLE FROM STUDENTS, ENROLLMENTS, COURSES
WHERE S.SSN = E.SSN AND E.CRN = C.CRN
AND FIRST = 'Ann', LAST = 'Miller'
```

Wrong! SQL requires that the user specifies whether he/she wants C.CRN or E.CRN in the SELECT-clause, although both are equal, so it actually does not matter which one. The rule is purely syntactic: if more than one tuple variable in the FROM clause has the attribute, the tuple variable cannot be left out, or Oracle will print the error ORA-00918: column ambiguously defined.
Query Formulation (1)

Query Formulation (2)

Unnecessary Joins (1)

Unnecessary Joins (2)
Self Joins

1. Suppose that S and E are assigned these tuples:

<table>
<thead>
<tr>
<th>SSN</th>
<th>FIRST</th>
<th>LAST</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-66-7890</td>
<td>John</td>
<td>Smith</td>
</tr>
<tr>
<td>41590</td>
<td>98765</td>
<td>Smith</td>
</tr>
</tbody>
</table>

2. The WHERE clause in this query can never be true:

SELECT S.FIRST, S.LAST
FROM STUDENTS S, ENROLLMENTS E1, ENROLLMENTS E2
WHERE S.SSN = E1.SSN AND S.SSN = E2.SSN
   AND E1.CRN = 41590 AND E2.CRN = 41690
   AND E1.TOPIC = 25 AND E2.TOPIC = 7111
   AND E1.CRN = 41590 AND E2.CRN = 41690
   WHERE S.SSN = E1.SSN
FROM STUDENTS S, ENROLLMENTS E1, ENROLLMENTS E2
WHERE S.SSN = E1.SSN AND S.SSN = E2.SSN
   AND E1.CRN = 41590 AND E2.CRN = 41690
   AND E1.TOPIC = 25 AND E2.TOPIC = 7111
   AND E1.CRN = 41590 AND E2.CRN = 41690
   WHERE S.SSN = E1.SSN

3. The condition in this query can never be true:

With CRN 31864 and the course with CRN 41950.

Task: Print a list of students enrolled in both the course with CRN 31864 and the course with CRN 41950.

4. To find students registered for both courses, two tuple variables over the relation ENROLLMENTS are required.

To find students registered for both courses, two tuples must be considered together (at the same time). For one specific assignment of a tuple to E.

So E.CRN = 41590 and E.CRN = 31864 can never be

Self Joins (2)
SelfJoins: Different tuple variables can refer to the same tuple. 

**Example:**

```
SELECT S.FIRST, S.LAST
FROM STUDENTS S, ENROLLMENTS E1, ENROLLMENTS E2
WHERE S.SSN = E1.SSN AND S.SSN = E2.SSN
```

To emphasize that there are duplicates and that they are not all the same:

```
SELECT DISTINCT S.FIRST, S.LAST
```

If you suspect that the query may contain duplicates, you can use the `DISTINCT` keyword:

```
SELECT DISTINCT S.FIRST, S.LAST
```

Duplicate Elimination: Onedifference of SQL to relational algebra is that duplicates have to be eliminated.

**Exercise:**

Which courses have two sessions?

```
SELECT INSTRUCTOR
FROM COURSES
```

In SQL-92, Oracle, and SQL Server, `DISTINCT` is also needed, see below.

### Tautology

**Exercise:**

Which courses have the same topic number but different CRNs?

```
SELECT INSTRUCTOR
FROM COURSES
```

### Self Join

Different tuple variables can refer to the same tuple.

```
SELECT S.FIRST, S.LAST
FROM STUDENTS S, ENROLLMENTS E1, ENROLLMENTS E2
WHERE S.SSN = E1.SSN AND S.SSN = E2.SSN
```
SELECTSYNTAX/(/1/)/

Goal/-List/(after
SELECT

DISTINCT

ALL

/*

?*/

Goal/-Element

/*

Variable

AS

Term

Coefficient:

Terms (1) (2)

Overview

5. Null Values

4. Terms and Conditions

3. SELECT-FROM-WHERE, Tuple Variables

2. Lexical Syntax

1. General Remarks about SQL

Terms (1)

Terms (2)

SELECT SYNTAX (1)
Conditions (1):

- Conditions consist of atomic formulas, e.g., \( \text{SEATS} > 50 \), connected by the logical connectives \( \text{AND} \), \( \text{OR} \), \( \text{NOT} \).
- Conditions are based on the logical connectives "and", "or", "not".
- Conditions consist of atomic formulas, e.g., \( \text{SEATS} > 50 \).

Conditions (2):

- Exception: ... \( \text{AND} \) ...
- SQL (not supported in Oracle 8.0, SQL Server, DB2):
- SQL-92 allows "IS NOT TRUE", "IS FALSE", etc., after formulas.

Terms (1):

- Terms are used in conditions, e.g., \( \text{SEATS} > 50 \).
- Terms are frequently used instead of terms, e.g., \( \text{SEATS} > 50 \).
- Terms are used in conditions, e.g., \( \text{SEATS} > 50 \).
Conditions

Atomic Formula (Form 1):

- Term
- Comparison-op
- Term

Comparison operators:
- =
- ,
- <>
- <
- >
- </=
- >/=

Comparison operators can be used for numbers as well as for strings.

Note equals/"is written in standard SQL as/".]

Oracle, SQL Server, and DB2 understand also/!

=/.

^=/

"works in Oracle and DB2, but not in SQL Server."

Comparison operations can be used for numbers as well as for strings.

E.g.:

SEATSBETWEEN 20 AND 40

Table: Seats

Conditions (4)

Atomic Formula (Form 3):

- Character
- ESCAPE
- Term
- NOT
- LIKE
- Term

E.g.: INSTRUCTOR LIKE '/%Brass/%/' is true for all instructors containing the substring "Brass".

Tousethe characters % and _ without their special meaning in the second argument, an escape character is

required. To use the characters % and _ without their special meaning in the second argument, an escape character is used. It removes the special meaning of the following character. It must be explicitly declared. e.g.:

Programme LIKE '\% ESCAPE '\'

Conditions (5)

Atomic Formula (Form 2):

- Term
- AND
- Term

E.g.: SEATS BETWEEN 20 AND 40

Table: Seats

Conditions (7)

Atomic Formula (Form 5):

- Term
- BETWEEN
- Term

Conditions (6)

Atomic Formula (Form 3):
WHERE NOT (INSTRUCTOR = 'Brass', 'Press', 'S', 'B')
FROM COURSES
SELECT CRN, TITLE

5. NULL Values
4. Terms and Conditions
3. SELECT-FROM-WHERE, Tuple Variables
2. Lexical Syntax
1. General Remarks about SQL

Overview

Conditions (1)

Conditions (9)

Conditions (9)

Conditions (9)
null values in SQL I

1. If SQL Server, "INSTRUCTOR = NULL" works after both operands are null, but inserted "unknown".
   - Both operands are null, but inserted "unknown".
   - SQL Server returns either true or false when one of
     the operands is null.
   - The result of the "or" or "and" operator in SQL 9-2 and
     Oracle, and SQL Server is a special format in SQL 9-2 and
     Oracle.
   - The test for an null value can only be done in this way.

2. E.g., INSTRUCTOR IS NULL

   ![Diagram](image)

   Atomic Formula (Form 5):

   ![Table](image)

   Test for NULL

   ![Table](image)

   Null Values (4)

   ![Table](image)

   Null Values (3)

   ![Table](image)

   Null Values (2)
ProblemsofNullValues

Forthoseaccustomedtoworkingwithatwovaledlogics/nullvaluescansometimesteleadtosurprises:

SomestandardlogicalequivalencesdonotholdinSQL:

E.g.,courses taughtbyBrassandcoursesnottaughtbyBrassarecounted,one normalassumetogetall
courses. ButthisisnottrueinSQL; thosewithanull
valueintheINSTRUCTOR
columnarenotcounted.

E.g.,X=X
evaluateto"unknown",notto"true"if
Xisnull.

Sincethenullvalueisusedwithdif-
ferentmeanings,theremight
be no satisfyingsemanticsfor a
querylanguage:

E.g.,themeaning"valueexists,
butunknown"(X)
canbeinconflictwith"X"(X).

ExpressionswithNullValues

Datatypfunctionswillnormallyreturnnullifoneoftheir
argumentsisnull:

E.g.,ifAisnull,
A+Bwillreturnnull.

InOracle,
A||B(returnstheconcatenationofstringsAandB)
returnsBifAisnull(violatestheSQL-2standard).

NULLbyitselfisnotaterm(expression),althoughitcanbe
usedinmanycontextswhereotherwiseanexpression
isneeded.

NULLhasnotype,soatleastweneedaclearcontextinwhich
we'reworking:

NULLmustbeconvertedtoanelementofthespecified
type:

E.g.,CAST(NULLAStype)
givesanullvalueofthespecified

Forthoseaccustomedtoworkingwithatwovaledlogics...