Cross Platform SQL

Strategies and Methodologies for Write Once, Run Anywhere
Today’s Speaker

Andrew Clark

CTO
RT Analytics
aclark@rtanalytics.com
Today’s Agenda

Why Cross-Platform SQL Matters
ANSI SQL Overview
Data Types
Functions
Cross-Platform SQL
References

Comparison of different SQL implementations
http://troels.arvin.dk/db/rdbms/

ANSI SQL Scalar Functions
http://users.atw.hu/sqlnut/sqlnut2-chp-4-sect-4.html

MySQL Function and Operator Reference

Postgres Functions and Operators
https://www.postgresql.org/docs/9.1/static/functions.html

DB2 for i Scalar Functions

SQL Server Built-in Functions

ODBC Scalar Functions
Database Versions Tested

- Access 2003
- SQL Server 2012
- DB2 for i 7.2
- Postgres 9.5.1
- MySQL 5.6.11
- Oracle 11g
The "Big 5" (and Jet)

95% of World's Relational Data

MySQL, ORACLE, IBM DB2, SQL Server, PostgreSQL

...don't forget Jet i.e. "Big 6"...
# Relational Database Popularity

<table>
<thead>
<tr>
<th>Rank</th>
<th>DBMS</th>
<th>Score Jan 2017</th>
<th>Score Dec 2016</th>
<th>Score Jan 2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Oracle</td>
<td>1416.72</td>
<td>+12.32</td>
<td>-79.36</td>
</tr>
<tr>
<td>2.</td>
<td>MySQL</td>
<td>1366.29</td>
<td>-8.12</td>
<td>+67.03</td>
</tr>
<tr>
<td>3.</td>
<td>Microsoft SQL Server</td>
<td>1220.95</td>
<td>-5.70</td>
<td>+76.89</td>
</tr>
<tr>
<td>4.</td>
<td>PostgreSQL</td>
<td>330.37</td>
<td>+0.35</td>
<td>+47.97</td>
</tr>
<tr>
<td>5.</td>
<td>DB2</td>
<td>182.49</td>
<td>-1.85</td>
<td>-13.88</td>
</tr>
<tr>
<td>6.</td>
<td>Microsoft Access</td>
<td>127.45</td>
<td>+2.75</td>
<td>-6.59</td>
</tr>
</tbody>
</table>

ANSI SQL

Works with sets, based in Mathematics

SELECT CUSNO, CNAME, SUM(AMTDU)
FROM TABLE
WHERE cusno<100200
GROUP BY CNAME
HAVING SUM(AMTDU)>10000
ORDER BY 2

Not Freely Available... (SQL2011 latest)
ANSI SQL

Inherent Limits

- No calculations in SELECT clause
  ```sql
  SELECT CUSNO+10 AS CUSNO10
  FROM TABLE
  WHERE cusno10<100200
  ```

- Poor handling of Null values
  ```sql
  SELECT *
  FROM CUSTMAST
  WHERE CUSNO=:CUSNO
  ```

- Limited Scalar Functions, biased toward math functions
  - No Repeat(), no Round(), yes Tan()

- Overly complex/wordy
  - Char_Length(), Substring(), Position(), Extract(), Interval

- Evolving standard, poorly implemented by vendor DBMS
## Data Type Support

<table>
<thead>
<tr>
<th>SQL Type</th>
<th>SQLServer</th>
<th>Jet</th>
<th>MySQL</th>
<th>Oracle</th>
<th>DB2</th>
<th>Postgres</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Binary</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>BIT</td>
<td>BIT/YESNO</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BINARY</td>
<td>X 8000</td>
<td>x 255</td>
<td>X 255</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARBINARY</td>
<td>X 8000</td>
<td>X 255</td>
<td>X 64K</td>
<td>RAW 2000</td>
<td>X 32K</td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td>VARBINARY(MAX) 2GB</td>
<td>LONGBINARY 1GB</td>
<td>LONGBLOB 4GB</td>
<td>X 4GB</td>
<td>X 2GB</td>
<td>BYTEA 1GB</td>
</tr>
<tr>
<td><strong>String</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>X 8000</td>
<td>x²</td>
<td>X 255</td>
<td>X 2000</td>
<td>X 255</td>
<td>X 1GB</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>X 8000</td>
<td>X 255</td>
<td>X 64K</td>
<td>VARCHAR2 4000</td>
<td>X 32K</td>
<td>X 1GB</td>
</tr>
<tr>
<td>CLOB</td>
<td>VARCHAR(MAX) 2GB</td>
<td>LONGBINARY/ MEMO 64K</td>
<td>LONGTEXT 4GB</td>
<td>X 4GB</td>
<td>X 2GB</td>
<td>TEXT No limit</td>
</tr>
<tr>
<td>NCHAR</td>
<td>X 4000</td>
<td>x²</td>
<td>X 255</td>
<td>X 2000</td>
<td>X 16K</td>
<td></td>
</tr>
<tr>
<td>NVARCHAR</td>
<td>X 4000</td>
<td>x²</td>
<td>X 64K</td>
<td>NVARCHAR2 4000</td>
<td>X 16K</td>
<td></td>
</tr>
<tr>
<td>NCLOB</td>
<td>NVARCHAR(MAX) 1GB</td>
<td></td>
<td>X 4GB</td>
<td></td>
<td>X 1GB</td>
<td></td>
</tr>
<tr>
<td><strong>Numeric</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>INTEGER</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DECIMAL</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FLOAT(n)</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>REAL</td>
<td></td>
<td></td>
<td>SINGLE</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MONEY</td>
<td></td>
<td></td>
<td>CURRENCY</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Date/Time</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>DATE</td>
<td>X⁸</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>DATETIME</td>
<td>DATETIME</td>
<td>X</td>
<td></td>
<td>X⁸</td>
<td>X</td>
</tr>
<tr>
<td>INTERVAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

*Note: The table represents the compatibility and support levels of various SQL data types across different databases. The 'X' indicates support, while 'null' indicates non-support.*
SQL Functions

Least-Common-Denominator Functions
User-Defined Functions
ODBC/JDBC Escape Syntax
ANSI SQL Trivia

How many ANSI SQL scalar functions are there?
31

How many are supported by all databases?
1: ABS
...technically...
5: ABS, COS, EXP, SIN, TAN
LCD Functions

LTRIM
Removes blanks or hexadecimal zeros from the beginning of a string expression

REPLACE
Returns the string `<str>` with all occurrences of the string `<from_str>` replaced by the string `<to_str>`

ROUND
Returns a number rounded to the specified number of places to the right or left of the decimal place

RTRIM
Removes blanks or hexadecimal zeros from the end of a string expression

SPACE
Returns a string consisting of `<N>` space characters
LCD\(^5\) Functions

**LOWER**
Converting a string to lowercase. Access=LCASE

**REPEAT**
Returns a string consisting of the string <str> repeated <count> times. Access 2007=STRING

**SIGN**
Returns the sign of the argument as -1, 0, or 1, depending on whether the value is negative, zero, or positive. Access=SGN.

**UPPER**
Converting a string to all uppercase characters. Access=UCASE
# Database Function Compatibility

<table>
<thead>
<tr>
<th>Function Name</th>
<th>DB2</th>
<th>SQL Server</th>
<th>Oracle</th>
<th>MySQL</th>
<th>Access</th>
<th>ODBC</th>
<th>Postgres</th>
<th>SQL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Only common function</td>
</tr>
<tr>
<td>SIGN</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>SGN</td>
<td>X</td>
<td>X</td>
<td></td>
<td>+1, 0 or -1</td>
</tr>
<tr>
<td>ROUND</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Functions common to all databases</td>
</tr>
<tr>
<td>LTRIM/RTRIM REPLACE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>(but not SQL!)</td>
</tr>
<tr>
<td>TRIM(&lt;s1&gt; FROM &lt;s2&gt;)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>i</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOWER UPPER</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>LCASE</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REPEAT</td>
<td>X</td>
<td>REPLICATE</td>
<td>X</td>
<td>STRING</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Repeats char n times</td>
</tr>
<tr>
<td>LEFT/RIGHT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBSTR</td>
<td>X</td>
<td>SUBSTRING</td>
<td>X</td>
<td>X</td>
<td>MID</td>
<td>X</td>
<td>SUBSTRING X</td>
<td></td>
<td>Index of s1 in s2</td>
</tr>
<tr>
<td>LOCATE</td>
<td>X</td>
<td>CHARINDEX</td>
<td>X</td>
<td>X</td>
<td>INSTR</td>
<td>X</td>
<td>X</td>
<td>POSITION( &lt;s1&gt; in &lt;s2&gt;)</td>
<td></td>
</tr>
<tr>
<td>CONCAT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>CONCATenate 2 strings</td>
<td></td>
</tr>
<tr>
<td>SOUNDEX</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>'Sound' of value</td>
<td></td>
</tr>
<tr>
<td>DIFFERENCE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>0-4 'sounds alike'</td>
<td></td>
</tr>
<tr>
<td>CHAR_LENGTH</td>
<td>X</td>
<td>LEN</td>
<td>LENGTH</td>
<td>X</td>
<td>LEN</td>
<td>X</td>
<td>X</td>
<td>Char count in string</td>
<td></td>
</tr>
<tr>
<td>GREATEST LEAST</td>
<td>MAX</td>
<td>MIN</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>1st non-NULL</td>
</tr>
<tr>
<td>COALESCE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>IFNULL</td>
<td>X</td>
<td>X</td>
<td>1st non-NULL</td>
</tr>
<tr>
<td>CURRENT_DATE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>DATE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CURRENT_TIME</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>TIME</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CURRENT_TIMESTAMP</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>NOW</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YEAR MONTH</td>
<td>X</td>
<td>DATEPART</td>
<td>EXTRACT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>EXTRACT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAY</td>
<td>X</td>
<td>DATEPART</td>
<td>EXTRACT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>DAYOFMONTH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOUR MINUTE SECOND</td>
<td>X</td>
<td>DATEPART</td>
<td>EXTRACT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>EXTRACT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAYOFWEEK</td>
<td>X</td>
<td>DATEPART</td>
<td>TO_CHAR</td>
<td>X</td>
<td>WEEKDAY</td>
<td>X</td>
<td>EXTRACT( dow, &lt;d&gt;)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAYOFYEAR</td>
<td>X</td>
<td>DATEPART</td>
<td>TO_CHAR</td>
<td>X</td>
<td>DATEPART( 'y', &lt;d&gt;)</td>
<td>X</td>
<td>EXTRACT( doy, &lt;d&gt;)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMPDIFF</td>
<td>X</td>
<td>DATEDIFF</td>
<td>X</td>
<td>DATEDIFF</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMESTAMPADD</td>
<td>DATEADD</td>
<td>DATEADD</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO_CHAR(&lt;date&gt;)</td>
<td>CONVERT</td>
<td>X</td>
<td>DATE_FORMAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO_CHAR(&lt;num&gt;)</td>
<td>CONVERT</td>
<td>X</td>
<td>FORMAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Filling Function Gaps with UDFs

-- SQL Server Create Function
CREATE FUNCTION dbo.TRIM(@string VARCHAR(MAX))
RETURNS VARCHAR(MAX)
BEGIN
RETURN LTRIM(RTRIM(@string))
END
ODBC

• ODBC Microsoft extension of CLI (Call Level Interface)
• 73 Explicitly defined functions
• ODBC 1.0 almost universally supported
• ODBC 3.0 current standard
ODBC* Escape Syntax

Generically allows ODBC function calls regardless of DB implementation

\{fn\ \text{FUNCTION}(PARAMETERS)\}\}

**SQL Server Ok:**
Select \{fn\ locate('A', Customers.CompanyName)\}
From Customers

**SQL Server Fails:**
Select locate('A', Customers.CompanyName)
From Customers

Cross-Platform SQL

Naming
Literals
Casts
String Concatenation
Date/Time Handling
Leveraging CTEs and Derived Tables
Cross-Platform Naming

Adhere to SQL standards:

- **Schema.Table naming**
  
  ```sql
  SELECT * from SEQUELEX.CUSTMAST
  ```

  ...not SEQUELEX/CUSTMAST

  Use double quotes for "non-standard" names ($@# are non-standard)

  ```sql
  SELECT * FROM "$NON@NAM#"
  ```

  ...not [$NON@NAM#] not `$NON@NAM#`

- **Create Views where appropriate**
  
  ```sql
  CREATE VIEW NON_STANDARD AS
  SELECT * FROM [Cr@zily non'standard \Name]
  ```
Strings and Concatenation

- **SQL**
  
  ```sql
  Select 'abc' || 'def', 'ghi'
  Concat('abc', 'def')
  ```

- **SQL Server/Access**
  
  ```sql
  Select 'abc' + 'def'
  ```

- **ODBC (only cross-platform way)**
  
  ```sql
  {fn Concat('abc', 'def')}
  ```
Numeric Literals

- Watch out for decimal literals without 0 e.g.
  Select `amtdu+.32` from customers
  Select `amtdu+0.32` from customers

- Hex literals are non-standard (and EBCDIC):
  DB2 and SQL (not `x'F1'`):
    Select `X'F1'` from customers
  Most Others (and ODBC):
    Select `0x31` from customers
Date/Time Literals

ISO Date Literals are universal across Big 6

'2011-03-19'

CAST('2011-03-19' as DATE)

ODBC

{d '2011-03-19'}
{t '13:01:02'}
{ts '2011-03-19 13:01:02'}
Casts

Instead of DB2-specific casts:

```
INTEGER, DATE, DECIMAL
```

Select `Integer(char_field)`

Can use SQL CAST instead

```
CAST(EXPRESSION as DATA_TYPE)
```

Select `Cast(char_field as integer)`

ODBC

```
{fn convert(char_field, SQL_INTEGER)}
```
Extracts

Instead of DB2-specific extracts:
  DAY, MONTH, HOUR
  Select day(date_field)

Can use SQL EXTRACT instead
  EXTRACT(TYPE from EXPRESSION)
  Select Extract(DAY from date_field)

ODBC 3.0
  {fn extract(DAY from date_field)}
Fun with Date Arithmetic

Select Current_Date - Cast('2016-12-17' As date)

- DB2
  100
- Postgres, Access
  31
- Oracle
  31.713
- MySQL
  89 {???}
- SQL Server
  2/1/1900
- SQL
  INTERVAL '1' MONTH
The Strangely Bizarre Interval Type

- Two types of intervals
  INTERVAL YEAR [TO MONTH] (YYYY-MM)
  Day-Time (DD HH:MM:SS.mmm)

- For year-month literals the syntax is:
  INTERVAL 'YY[-MM]' YEAR [TO MONTH]'
  {year-month}

- For day-time literals the syntax is:
  INTERVAL 'DD [HH[:MM[:SS[:mmm]]]] [DAY|HOUR|MINUTE|SECOND] TO
  [HOUR|MINUTE|SECOND]'
  {day hour:minute:second.microsecond}

- Each Set(YEAR, MONTH, etc.) has an optional (leading precision) which is
  2 by default?? Except microsecond which appears to be 9...
Interval Examples

- 1 year = INTERVAL '1' YEAR
- 1 year 3 months = INTERVAL '1-3' YEAR TO MONTH
- 999 YEARS = INTERVAL '999' YEAR(3)

```sql
  Select CURRENT_DATE + INTERVAL '999' YEAR(3)
```

- 10 hours = INTERVAL '10' HOUR

- 4 days, 5 hours, 12 minutes, 10 seconds = INTERVAL '4 5:12:10.222' DAY TO SECOND

- 11 hours, 12 minutes, and 10.2222222 seconds = INTERVAL '11:12:10.2222222' HOUR TO SECOND(7)

- ODBC Escape literals are formatted exactly like this with brackets {} around them:
  {INTERVAL '1' YEAR}
  {INTERVAL '4 5:12:10.222' DAY TO SECOND}

- SQL
  Select CURRENT_DATE + INTERVAL '1-3' YEAR TO MONTH

- DB2
  Select CURRENT_DATE + 1 YEAR + 3 MONTHS)

- ODBC {poorly implemented}
  Select {fn timestampadd(SQL_TSI_YEAR, 1, {fn curdate()})

- Access (UDF)
  Select DateAdd('yyyy', 1, '2017-01-17')

- SQL Server
  Select DateAdd('yyyy, 1, '2017-01-17')
TIMESTAMP with TIMEZONE

Comparing TIMESTAMPS as coordinated events
• Think Rocket Launch, Stock price change
• Not employee clock in, log file

'2017-01-17 11:37:58' - '2017-01-16 11:37:58'

INTERVAL '1' DAY

Chicago UTC -6
Paris UTC +1

'2017-01-17 11:37:58 +01:00' - '2017-01-16 11:37:58 -06:00'

INTERVAL '31' HOUR

INTERVAL '1 7' DAY TO HOUR
DB2 TIMESTAMP UTC

Chicago UTC -6 (Current Timezone)
Paris UTC +1

Select
(tsl + 1 hour) - (ts2 + CURRENT TIMEZONE)
CURRENT TIMESTAMP

SQL standard defines CURRENT_TIMESTAMP as TIMESTAMP with TIMEZONE

Postgres/Oracle:
   LOCALTIMESTAMP = {DB2/SQL Server/MySQL} CURRENT_TIMESTAMP

UTC CURRENT_TIMESTAMP without TIMEZONE support
MySQL:
   UTC_TIMESTAMP()
SQL Server:
   GETUTCDATE()
DB2:
   CURRENT_TIMESTAMP + CURRENT TIMEZONE
CREATE FUNCTION ANDREW.DATEDIFF(datepart char(4),startdate date, enddate date) returns integer
language sql
BEGIN
RETURN(case upper(datepart)
when 'YYYY' then int((enddate-startdate)/10000)
when 'MM' then months_between(startdate,enddate)
when 'DD' then days(enddate)-days(startdate)
end);
END
CTEs and SQL Server Constants

• SQL Server

Select DateAdd(YYYY, 1, '2017-01-17')

With

Constants As (Select 'YYYY' As YYYY
From RECORD1)

Select DateAdd(YYYY, 1, '2017-01-17')
From

Customers, Constants
Derived Tables and SQL Server Constants

- SQL Server

  Select DateAdd(YYYY, 1, '2017-01-17')

  Select
  DateAdd(YYYY, 1, '2017-01-17')
  From
  Customers,
  (Select
   'YYYY' As YYYY
  From
   Record1) as Constants
Summary

Why Cross-Platform SQL Matters
ANSI SQL Overview
Data Types
Functions
Cross-Platform SQL
Questions?
Thank you for attending!

Andrew Clark
CTO
RT Analytics
aclark@rtanalytics.com

References:
Comparison of different SQL implementations
http://troels.arvin.dk/db/rdbms/

ANSI SQL Scalar Functions
http://users.atw.hu/sqlnut/sqlnut2-chp-4-sect-4.html

MySQL Function and Operator Reference

Postgres Functions and Operators
https://www.postgresql.org/docs/9.1/static/functions.html

DB2 for i Scalar Functions

SQL Server Built-in Functions

ODBC Scalar Functions