Database Systems (ISYS1001/ISYS5008)

Lecture 2

SQL for defining tables with keys, manipulating single table with SQL quesris, some functions from MySQL

Updated: 04th August,2021

Discipline of Computing
School of Electrical Engineering, Computing and Mathematical Sciences (EECMS)

CRICOS Provide Code: 00301J

Copyright Warning

COMMONWEALTH OF AUSTRALIA
Copyright Regulation 1969

WARNING

This material has been copied and communicated to you by or on behalf of **Curtin University of Technology** pursuant to Part VB of the *Copyright Act 1968* (**the Act**)

The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice

Objectives

- Describe main constructs of Relational model
- Describe the how SQL is useful in database systems
- Create a relation schema using SQL
- Use SQL to modify table schema, create table with a key constraint
- Create SQL queries with SELECT, FROM, WHERE clauses on a single table with additional clauses such as IN, NOT IN, BETWEEN, logical operators, mathematical operators , LIKE and DISTINCT
- Create SQL statements to manage NULL values properly
- Create queries with MySQL data and time functions and ROUND, TRUNCATE, CONCAT functions

Revisit : Basics of relational model

- Attributes : column name; describes the meaning of the values in the columns
- Schemas : name of the relation and the set of attributes of the relation.

  Example

  Books(title,author) , Hardcovers(isbn, jacket, price)

  - Attributes of a relation schema are a set, not a list therefore order does not matter
  - However, the order which is used to define the schema is used as the 'standard order' when displaying data
- Tuples : row of a relation; has a value for each attribute of the relation.
  - Following notation is used when a tuple is displayed not as part of a relation

  Example : (The Two Towers, J.R.R. Tolkein)
- Domains : each attribute of a relation has a domain from which the values can be taken (data type) ; this is an elementary type of data. Data type can be included in the schema as follows:

  Books(title : string, author: string)

  Books(title: VARCHAR(30), author: VARCHAR(15))

Revisit : basics of relational model

- Relations are **sets** of tuples , not lists of tuples
  - Therefore the order of tuples is not important
  - Attributes of a relation also can be reordered, however need to do so carefully as column order changes when doing so.

title	author
The Two Towers	J.R.R. Tolkein
The Silicon Mage	B. Hambly

title	author
The Silicon Mage	B. Hambly
The Two Towers	J.R.R. Tolkein

author	title
J.R.R. Tolkein	The Two Towers
B. Hambly	The Silicon Mage

Keys of a relation

- Relational model allows to define different constraints on the relations of the database schema.
- Most common constraint is the key constraint
- Key Constraint:
  - Key constant : A set of attributes forms a key for a relation if two tuples in a relation instance is not allowed to have the same values in all the attributes of the key.
  - Key is denoted by underlining the required attributes in the schema

    Books(isbn,title)

    - The key is an attribute that completely determines the tuple.
    - In this case, it is possible to have two books with the same title, but we can't have two books with the same isbn.

SQL

Defining tables with some constraints,

Queries using a single table

The SQL Query Language

- *"S.Q.L." or "sequel"*
  - Developed by IBM (system R) in the 1970s
  - Supported by most major commercial database systems
- Standardized – many new features over time
  - SQL-86
  - SQL-89 (minor revision)
  - SQL-92 (major revision)
  - SQL-99 or SQL3 (major extensions)
  - SQL:2003 (minor extensions)
  - SQL:2008 (minor extensions)
  - SQL:2011 (minor extensions)
  - SQL:2016 (current standard) ISO/IEC 9075-1:2016
- We will use MySQL version 5.7 in Curtin labs, which is an open-source implementation of SQL. Current version is 8.0

Revisit: Why SQL?

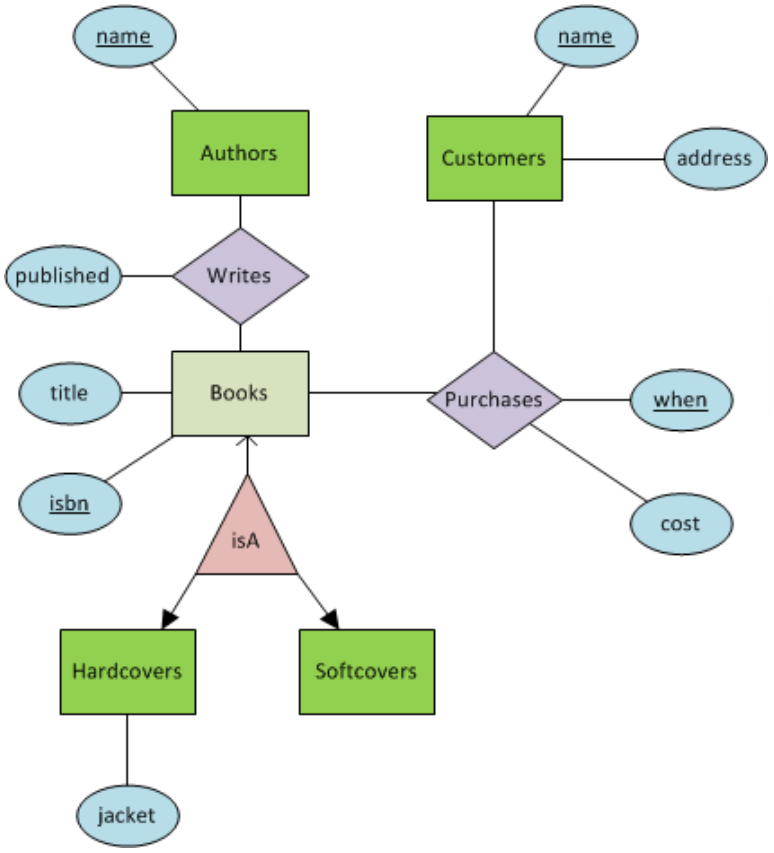
- SQL is a very-high-level language.
  - Declarative, based on relational algebra
  - Say “what to do” rather than “how to do it.”
  - Avoid details needed in procedural languages like C++ or Java.
- Database management system figures out “best” way to execute query.
  - Called “query optimization.”
- Interactive via GUI or prompt, or embedded in programs

SQL commands

- Data Definition Language (DDL)
  - Create table, **alter table, drop table**, …
- Data Manipulation Language (DML)
  - **Select, insert, delete, update**
- Other Commands
  - indexes, constraints, views, triggers, transactions, authorization, …

Our Running Example

- Most of our SQL queries in this week will be based on the Bookseller database schema from the last week .
  - Underline indicates key attributes.



Authors(name)

Writes(authorname, bookid, published)

Books(isbn,title)

Hardcovers(isbn, jacket, price)

Softcovers(isbn, price)

Purchases(name, bookid,when, cost)

Customers(name, address)

Modifying relation Schema

- Adding a new attribute to a relation

```
ALTER TABLE <name> (
        ADD <new attribute name and it's type>
    );
```

  - All values of the new attribute will have NULL as the value of all existing tuples

- Delete an attribute of a relation

```
ALTER TABLE <name> (
        DROP <attribute name>
    );
```

Declaring keys when creating tables

- In order to have a tuple, the key must have a unique value – hence a key (as a whole) is always both `NOT NULL` and `UNIQUE`.
- Keys can be declared as part of creating the key attribute of the table:

```
CREATE TABLE Books(
        isbn INT(13) PRIMARY KEY,
        title VARCHAR(50)
    );
```

- They can also be declared on a separate line:

```
CREATE TABLE Books(
        isbn INT(13),
        title VARCHAR(50),
        PRIMARY KEY(isbn)
    );
```

Declaring keys

- Sometimes there is no single attribute that completely identifies the tuples in a table.
- In this case you can choose a PRIMARY KEY that includes multiple attributes.
- This must be done using the second declaration format.

```
CREATE TABLE Writes(
      authorname VARCHAR(50),
      bookid INT(13),
      published DATE,
      PRIMARY KEY(authorname,bookid)
   );
```

Data manipulation with SQL

- Basis is relational algebra
- Limited set of operations on relations (tables)
- Main operations :
  - Usual set operations applied to relations
    - Union, intersection, and difference
  - Operations that remove part of the relation
    - ‘selection’ eliminates some rows
    - ‘projection’ eliminates some columns
  - Operations that combine tuples from two relations
    - ‘Cartesian product’ - combine all tuples of two relations in all possible ways
    - ‘join’ operations – selectively combine tuples of two relations
  - ‘renaming’ operation- does not affects the tuples but affects the relation schema

SQL Query

- Specifies a result table
- When entering a query start each clause in a new line for readability and ease of debugging

```
SELECT name, book_id
FROM Purchases
WHERE cost>25;
```

- SQL is case **insensitive** except for **quoted strings**
  - Keywords like SELECT or WHERE can be written in upper/lower case.
  - Quoted strings that are values of character string data types.
- MySQL is case sensitive on table and database names

Select statement

- To tell the DBMS what information to retrieve
- Basic form:

```
SELECT A1, A2, ..., An
    [FROM R1, R2, ..., Rm
      [WHERE condition ]
        ]
```

  - **SELECT** desired columns
  - **FROM** one or more tables
  - **WHERE** condition about tuples of the tables

Meaning of Single-Relation Query

```
SELECT name, book_id
FROM Purchases
WHERE cost>25;
```

Results of the Query:

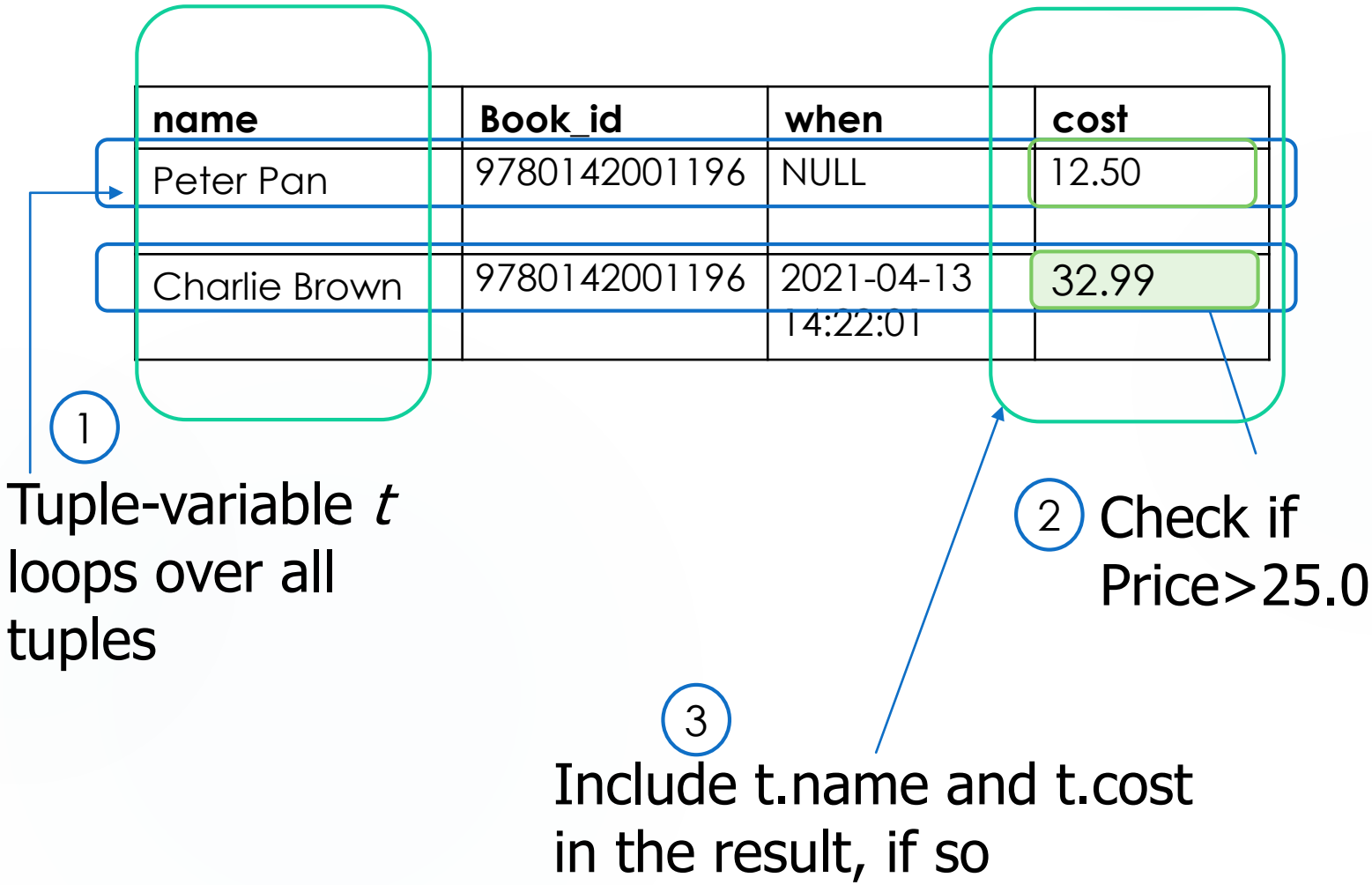
name	book
Charlie Brown	9780142001196

Purchases Table

name	Book_id	when	cost
Peter Pan	….	…	12.50
Charlie Brown	…	..	32.99

- Conceptual evaluation strategy:
  - Begin with the table in the `FROM` clause.
  - Discard tuples that fail conditions in the `WHERE` clause.
  - Retrieve columns in the `SELECT` clause.
- Note that we need to match case on `Purchases` but not book or cost.

Operational Semantics



Operational Semantics

- Think of a *tuple variable* visiting each tuple of the relation mentioned in `FROM`.
- Check if the "current" tuple satisfies the `WHERE` clause.
- If so, compute the attributes or expressions of the `SELECT` clause using the components of this tuple.

* in SELECT clauses

- When there is one relation in the FROM clause, * in the SELECT clause stands for “all attributes of this relation.”
- Example: Using Purchases (name, bookid, when, cost) :

```
SELECT *
FROM Purchases
WHERE name = 'Charlie Brown';
```

Results of the Query:

name	bookid	when	cost
Peter Pan	9780142001196	NULL	12.50
Charlie Brown	9780142001196	2021-04-13 14:22:01	32.99

Note the result has each of the attributes of Purchases.

Complex Conditions in WHERE Clause

- Comparisons **=, >, <, <=, >=, <>, !=**
  - And other operators that produce Boolean-valued results
- Boolean operators AND, OR, NOT.
  - Multiple criteria can be combined using AND, OR
- BETWEEN , IN, NOT IN
- LIKE
- Arithmetic operators +,-,*,/
- ..
- Can be combined to get desired results

Complex Conditions in WHERE Clause : comparisons and multiple criteria

- Example: Using Hardcover(isbn, jacket, cost), find all hard cover books that have a dust jacket (not 'none') and cost less than $20:

```
SELECT isbn
FROM Hardcover
WHERE jacket != 'None' AND
        cost < 20;
```

- Example: Searching for hardcover books that have a dust jacket or are priced below $20, using the schema Hardcover(isbn, jacket, cost) :

```
SELECT isbn
FROM Hardcover
WHERE jacket != 'None' OR
        cost < 20;
```

Complex Conditions in WHERE Clause : range

- Between A and B
- Example: Using Hardcover(isbn, jacket, cost), find all hard cover books that have cost between $10 and $30 , including both $10 and $30.

```
SELECT isbn, cost
FROM Hardcover
WHERE cost BETWEEN 10 AND 30;
```

- Same as :

```
    … cost>=10 AND cost<=30;
```

- Can be used with other data types, e.g., date.

```
SELECT *
FROM Purchases
WHERE date BETWEEN '2014-07-01' AND '2015-06-30';
```

Complex Conditions in WHERE Clause : IN and NOT IN

- IN: match with any value in parentheses
  - Example: Using Writes(authorname, bookid, published), get all books written by Arthur Conan Doyle, Agatha Christie or Ian Fleming.

```
SELECT bookid
FROM Writes
WHERE author IN ('A.C.Doyle','A.Christie','I.Fleming');
```

  - Same as:

```
… author = 'A.C.Doyle'OR … OR author = 'I.Fleming';
```

- NOT IN
  - not equal to any value in parentheses
  - Example:Using Writes(authorname, bookid, published), get all books not written by Author R. Greene or R. Barnes.

```
SELECT bookid
FROM Writes
WHERE author NOT IN ('R.Greene', 'R.Barnes');
```

Complex Conditions in WHERE Clause : **LIKE** operator

- `LIKE` to match string patterns in the where clause.
  - '_' stands for any single character
  - '%' stands for any 0 or more characters.
- Example: Using Books(isbn, title), find books that have titles starting with the letter D.

```
SELECT *
FROM Books
WHERE title LIKE 'D%';
```

'D' as the first letter

- Example: Find books that have titles with 't' as the second letter.

```
SELECT *
FROM Books
WHERE title LIKE '_t%';
```

't' as the second letter, first letter and other letters after second letter can be anything

- `NOT LIKE` for negation

```
. . .
WHERE title NOT LIKE '%Guide%';
```

Renaming Attributes when selecting data

- If you want the result to have different attribute names, use “AS <new name>” to rename an attribute
  - **AS** keyword is optional.
- Example: Using Books(isbn, title):

```
SELECT title AS bookTitle
FROM Books;
```

Results of the Query:

bookTitle
The Complete Sherlock Holmes
Grimm’s Fairy Tales

```
SELECT title
FROM Books;
```

Results of the Query:

title
The Complete Sherlock Holmes
Grimm’s Fairy Tales

Select with Arithmetic expressions

- + add , - subtract, * multiply, / divide
- Arithmetic expressions can appear
  - in the SELECT clause or
  - in a predicate (*e.g.*, WHERE clause)
- Example: Using Purchases(name, bookid, when, cost):

```
SELECT name, book_id, cost,
       cost*100 AS costInYen
FROM Purchases;
```

Results of the Query:

name	bookid	cost	costInYen
Peter Pan	9780142001196	12.50	1250
Charlie Brown	9780142001196	32.99	3299

Constants as Expressions

- To get a particular constant in each row, use it as an expression in the Select clause.
- Using Purchases(name, bookid, date, cost):

```
SELECT name,bookid,
       '2021Purchases' AS purchased
FROM Purchases
WHERE date >='2021-01-01'
```

Results of the Query:

name	bookid	purchased
Charlie Brown	9780142001196	2021Purchases

Purchases Table:

name	bookid	when	cost
Peter Pan	9780142001196	NULL	12.50
Charlie Brown	9780142001196	2021-04-13 14:22:01	32.99

Removing Duplicates

- **Duplicates** retained unless keyword DISTINCT is used

```
SELECT jacket
FROM Hardcover;
```

Results of the Query:

jacket
fairy
None
None

```
SELECT DISTINCT jacket
FROM Hardcover;
```

Results of the Query:

jacket
fairy
None

Hardcovers(isbn, jacket, price)

isbn	jacket	cost
9780330320559	fairy	16.28
9780786965625	None	49.95
9781435114944	None	18.50

NULL Values

- Tuples in SQL relations can have `NULL` as a value for one or more components
- Every type implicitly includes `NULL`
- Meaning depends on context. Two common cases:
  - *Missing value* : *e.g.*, we know that a hardcover book has some dust jacket, but not what it displays
  - *Inapplicable* : *e.g.*, the value of attribute spouse for an unmarried person

- The logic of conditions in SQL is really 3-valued logic: TRUE, FALSE, UNKNOWN.
- Comparing any value (including NULL itself) with NULL yields UNKNOWN.
- A tuple is in a query answer iff when the WHERE clause is TRUE (not FALSE or UNKNOWN).

Truth tables for 3-valued logic

AND	**T**	**F**	**U**
T	T	F	U
F	F	F	F
U	U	F	U

OR	**T**	**F**	**U**
T	T	T	T
F	T	F	U
U	T	U	U

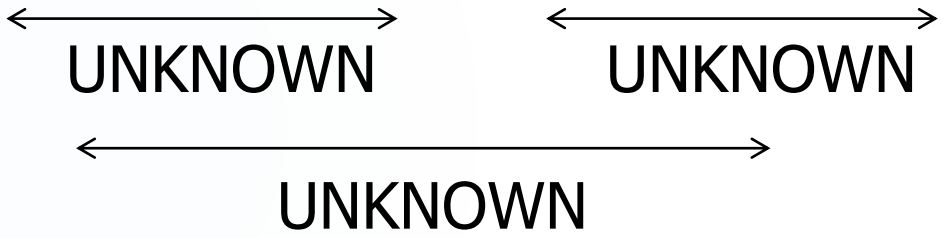
NOT	
T	F
F	T
U	U

NULL Values

- From the following `Purchases` relation:

name	bookid	when	cost
Peter Pan	9780142001196	NULL	12.50
Charlie Brown	9780142001196	2021-04-13 14:22:01	32.99

```
SELECT *
FROM Purchases
WHERE cost < 20.00 OR cost >= 20.00;
```



name	bookid	when	cost
Charlie Brown	9780142001196	2021-04-13 14:22:01	32.99

Comparing NULL's to Values

- To retrieve based on the presence of NULL, use `x IS NULL` or `x IS NOT NULL` to compare an attribute x with null.
  - CANNOT use ' = NULL' or '<> NULL' for this purpose.
  - Example:

```
SELECT jacket
FROM Hardcover
WHERE jacket IS NULL;
```

```
SELECT jacket
FROM Hardcover
WHERE jacket IS NOT NULL;
```

Comments in SQL

- SQL script is a file of SQL statements. Comments can be included to improve readability.
  - Text in a line after two hyphens treated as comment

```
-- select from books table
```

(at least one space is required after second hyphen)

  - Comments over several lines is given as follows

```
/* List all the books written
    by A.C.Doyle */
```

  - Additionally, text in a line after # is also treated as comment in MySQL

```
# creating tables
```

Example:

```
SELECT book_id
FROM Purchases -- sales table
WHERE Cost <20.0;
```

Some functions in MySQL

Number, Character and Date Functions

MySQL built-in functions

- MySQL offers many built-in functions in addition to SQL standard constructs
- Any DBMS software (such as Oracle, PostgreSQL) also have own built-in functions

- Refer the MySQL reference manual to know about different functions

Emp table schema used in examples

Name	Null?	Type
EMPNO	NOT NULL	CHAR(6)
FIRSTNAME	NOT NULL	VARCHAR2(12)
MIDINIT	NOT NULL	CHAR(1)
LASTNAME	NOT NULL	VARCHAR2(15)
WORKDEPT		CHAR(3)
PHONENO		CHAR(4)
HIREDATE		DATE
JOB		CHAR(8)
EDLEVEL		DECIMAL(2)
SEX		CHAR(1)
BIRTHDATE		DATE
SALARY		DECIMAL(8,2)
BONUS		DECIMAL(8,2)
COMM		DECIMAL(8,2)

Functions on numerical data

- ROUND(n,[m])
  - n is the number to be rounded. Result rounded to m places to the right of the decimal point.

```
SELECT lastname, ROUND(salary/26.2,2) pay
FROM Emp
WHERE workdept='E21';
```

lastname	pay
George	2807.70
James	2500.00
Kent	2269.23

  - ROUND(n) rounded to the nearest integer

- TRUNCATE(n,m)
  - n is the number to be truncated. Result truncated to m places to the right of the decimal point.

```
SELECT TRUNCATE(23.567,1);
```

Result : 23.5

```
SELECT TRUNCATE(122,-2);
```

Result : 100

Character data type

- Columns defined as CHAR(n) or VARCHAR(n)
- Concatenation function CONCAT
  - List of names with a space between first and last name.

```
SELECT CONCAT(firstname,' ',lastname) AS fullname,
   salary
FROM Emp;
```

fullname	salary
Peter George	73000
James Keiman	65000
John Hayman	65000
Mike Kent	59000

data types for date, time values

- DATE is the third most common type of data in databases
- DATE, DATETIME types are related functions to manage data and time values.
- MySQL `DATE` type contains only the date
  - critical to remember when comparing two dates
- Default format: YYYY-MM-DD
  - *e.g.* 2015-08-15
- MySQL `DATETIME` type is used for values that contain both data and time
  - critical to remember when comparing two dates
- Default date – time format: YYYY-MM-DD HH:MM:SS
  - *e.g.* 2015-08-15 10:03:58

Time and Date functions

- `SYSDATE(), CURDATE(), CURTIME()`
  - Current date and/or time
- `DATEDIFF,PERIOD_DIFF,SUBTIME,TIMEDIFF,`
  - Difference between dates or times

`DATEDIFF(p1,p2)`	returns expr1 − expr2 expressed as a value in days from one date to the other. expr1 and expr2 are date or date-and-time expressions. Only the date parts of the values are used in the calculation.
`PERIOD_DIFF(exp1,exp2)`	Returns the number of months between periods ***P1*** and ***P2***. ***P1*** and ***P2*** should be in the format ***YYMM*** or ***YYYYMM***.
`SUBTIME(exp1,exp2)`	returns expr1 − expr2 expressed as a value in the same format as expr1. expr1 is a time or datetime expression, and expr2 is a time expression.
`TIMEDIFF(exp1,exp2)`	returns expr1 − expr2 expressed as a time value. expr1 and expr2 are time or date-and-time expressions, but both must be of the same type.

Time and Date functions

- Example:

```
SELECT lastname,
MOD(DATEDIFF(SYSDATE(), birthdate), 365) AS 'days to birthday'
FROM Emp;
```

lastname	days to birthday
George	23
James	2
Kent	22

MOD(n, m) : Modulo operation. Returns the remainder of n divided by m.

- In this lecture, we will revise the relational model again and look at SQL command to define and modify tables, retrieve data from a table using different clauses . MySQL functions for date- time, number and string manipulation also will be covered.

Date and time display formats

- To override the default, use DATE_FORMAT in the form

  DATE_FORMAT(<date>, '<format>')
- 33 options for format
- Some common options:

day and month

%e	Day of the month, numeric (0..31)
%d	Day of month (e.g., 04)
%a	Abbreviated name of day (e.g., Fri)
%W	week day fully spelled out (e.g., Friday)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%b	Abbreviated month name (e.g., Jul)

time

%p	Meridian indicator, AM or PM
%l	hour of day (1-12)
%k	hour of day (0-23)
%i	minute (0-59)
%s	second (0-59)

year

%Y	4-digit year (e.g., 2011)
%y	last 2 digits of the year (e.g., 11)

Date and time display formats

Example:

```
SELECT DATE_FORMAT(SYSDATE, '%b') AS 'Month Short'
FROM Emp;
```

Month Short
Aug

```
SELECT DATE_FORMAT(birthdate, '%M %e') AS bdate
FROM Emp;
```

bdate
March 6
June 30
February 20

```
SELECT DATE_FORMAT(SYSDATE,'%W, %e %M') AS today;
```

today
Thursday,5 August

```
SELECT DATE_FORMAT(SYSDATE, '%l:%i:%s');
```

Summary

- Relational model supports SQL
- SQL query manipulation is based on relational algebra
- Key identify a tuple uniquely; a Key can be defined with PRIMARY KEY in schema definition
- Basic SQL use commands such as
  - SELECT, FROM, WHERE clauses
  - Predicates: comparisons, BETWEEN, IN, LIKE
  - Arithmetic operators
  - DISTINCT
  - Comparing nulls, UNKNOWN
- In MySQL, there are large number of additional functions. ROUND, TRANCATE on numbers , CONCAT on strings and Date and time functions are commonly used.

Happy Database systems

Next week : ER model

Practical : Practical test -1 will be held during the lab time.

Lab worksheet - 3 will be also covered in the next lab.