Status:	New	**Priority:**	Should have
Author:	Adrian Föder	**Category:**	
Created:	2012-01-19	**Assigned To:**	
Updated:	2012-03-06	**Due date:**	
Subject:	General date and time handling rules		

Description

Date and time handling discussion

This issue is considered as invitation for discussion, comment as you like.

Introduction

When handling trivially said "date and time", running in technical difficulties is waiting to happen. The purpose of this document is to carry-together this difficulties and lastly, possibly, provide a best practice both in the Framwork API and in documentation, how to handle date and times in user's applications.

Professional considerations

Different meanings of "point in time"

Regarding a "time", "event" or generally saying "point in time" could result in two meanings. *(The author likes to have this terms ubiquitous, so feel free to improve them)*

Physical Point In Time

The Physical Point In Time means a time stamp that describes a "physical" time of an occurrence without doubt. That might be the creation or modification time of a record, the moment of a happening like the exact time of the birth of a person or the time where a football goal was achieved.
It describes an event that occurs on a specific point of the global, physical timeline no matter where or under what conditions this was (like DST or such).

Logical Point In Time

The Logical Point In Time is a time the user is familiar or comfortable with without caring about localization and such. If a user invites to a party saying it begins at 19 o'clock he means the time all people know and are used to around the event. 19 o'clock than means 19 o'clock at the location where the party happens under the circumstances around that date, e.g. active DST etc. It's the time being seen when you look at your watch.

Date Only

In some use cases, the date only may matter; mostly usual when describing a birth date. In such cases timezone and DST doesn't matter, only the day-date is important. It even would be contra-productive to even attempt to store a time besides that date, because it's intrinsically unimportant.

Technical considerations

Basically the author came up to the following idea: MySQL's DATETIME format stores a date and time as-is, so without any automatic timezone conversion (**unlike**TIMESTAMP does). *TODO: find source of this statement*

So it could be a general best practice to decide that every Point In Time is to be stored and therefore assumed to be in UTC. That's what UTC is made for: to describe an event happening at a Physical Point In Time.

If a Date Only is required, it could be an idea to force UTC 0:00 as corresponging time (unfortunately DateTimes are always bound together in most implementations we know, therefore no **real**Date Only is provided)

Before storing and after retrieving is has to be decided what the intention of the date is: Physical or Logical Point In Time or Date Only . Dependant of this, the adjustments could be made, i.e. deciding what Timezone the user intended to provide resp. what timezone he expects.

If the user, for example, provides an invitation to a party in Stuttgart, taking place May 30, 19 o'clock, this is a Logical Point In Time. Disregarding locational aspects, we assume DST and therefore store May 30, 18 o'clock in the Database (which is implicitly UTC because per design).

History

#1 - 2012-03-06 16:05 - Adrian Föder

amendment: Source for the DATETIME and TIMESTAMP statement: MySQL's DATETIME reference (~4th paragraph)