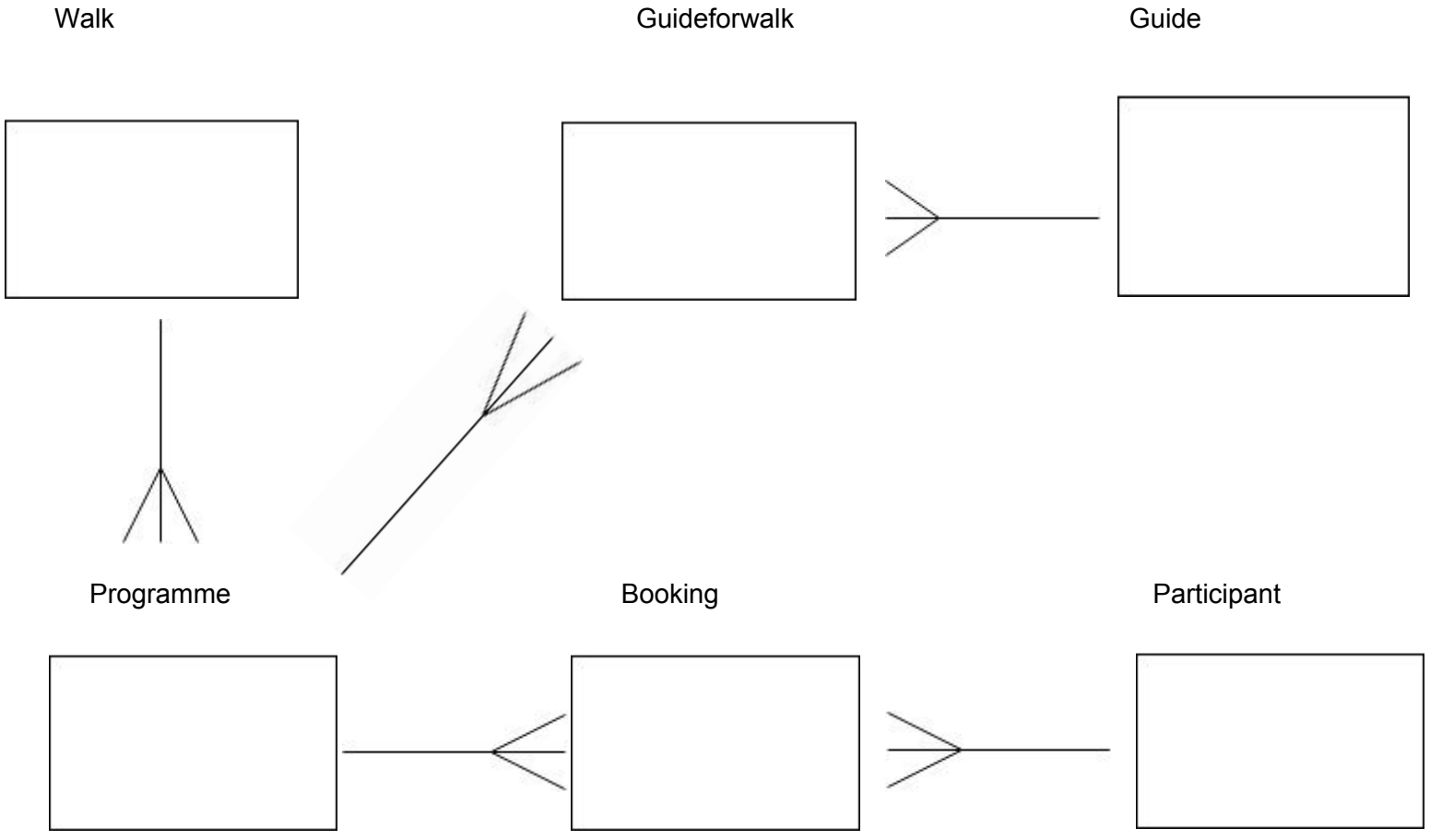
Introduction to Database Technology TMA

Nuchem Katz
nkatz01
Ms Ping Brennan

14/06/2017

Capital Promenades Database

Entity-relationship diagram



Details for each table

Table: Guide			
Column	Datatype	Attribute	Default
guide_id	INT(11)	PK, NN, AI	
guide_title	CHAR(4)		NN[1]
guide_fname	VARCHAR(20)	NN	
guide_surname	VARCHAR(20)	NN	
guide_address1	VARCHAR(20)	NN	
guide_address2	VARCHAR(20)		NULL
guide_town	VARCHAR(30)	NN	
guide_postcode	CHAR(10)	NN	
guide_mobile	VARCHAR(15)	NN	
guide_email	VARCHAR(25)	NN	
guide_description	VARCHAR(255)	NN	

Table: Walk			
Column	**Datatype**	**Attribute**	**Default**
walk_id	INT(11)	PK, NN, AI	
walk_title	VARCHAR(30)	NN	
Walk_durationinminutes	INT(11)	NN	
walk_startpoint	VARCHAR(50)	NN	
walk_finishpoint	VARCHAR(50)	NN	
walk_description	TEXT	NN	

[1] The requirements did not ask for a title.

Table: Participant			
Column	**Datatype**	**Attribute**	**Default**
participant_id	INT(11)	PK, NN, AI	
participant_title	CHAR(4)	NN	
participant_fname	VARCHAR(20)	NN	
participant_surename	VARCHAR(20)	NN	
participant_phone	VARCHAR(15)		NULL
participant_email	VARCHAR(25)		NULL
participant_membershiplevel	ENUM(‘REG’, ‘CON’)	NN	‘REG’

Table: Guideforwalk			
Column	**Datatype**	**Attribute**	**Default**
guideforwalk_id	INT(11)	PK, NN, AI	
guideforwalk_programme_id	INT(11)	PK, NN FK reference Programme_programme_id	
guideforwalk_guide_id	INT(11)	PK, NN FK reference Guide_guide_id	

Table: Programme			
Column	**Datatype**	**Attribute**	**Default**
programme_id	INT(11)	PK, NN, AI	
programme_guideforwalk_id	INT(11)	PK, NN FK reference Guideforwalk_guideforwalk_id	
programme_walk_id	INT(11)	PK, NN FK reference Walk_walk_id	
programme_date	DATE	NN	
programme_starttime	CHAR(7)	NN	
programme_maxparticipants	INT(11)	NN	

Table: Booking			
Column	**Datatype**	**Attribute**	**Default**
booking_id	INT(11)	UQ, NN, AI	
booking_programme_id	INT(11)	PK, NN FK reference Programme_programme_id	
booking_participant_id	INT(11)	PK, NN FK reference Participant_id	

[2]

[2] booking _programme_id together with booking_participant_id form a composite primary key for this table so I reckoned since the booking_id isn't being used anywhere else, I would give it a UQ attribute.

Describe three different data types

CHAR(n)

CHAR is a data type used for values that are predictable always to hold up a known amount of places and so this data type stores up to n places (255) and in a case where less is entered, would store space characters to fill up the rest. (as opposed to VARCHAR which stores only however much you enter and up to the maximum you've declared it to be but not more than 255.)

In table **Programme,** I've used CHAR to as a datatype for '**starttime**' column, as CHAR(7) accommodates digits and so using the 12 hour clock (as per the examples in the Walk Programme table within the the Capital Promenades Walk Programme pdf sheet), 7 characters to account for 4 digits, a colon and am or pm will always suffice for this column.

(Although I could use the datatype INT or the TIME data type which as I quote from the MySQL manual "retrieves and displays TIME values in 'HH:MM:SS' format", I did choose not to use them. A. Because as far as I recall, datatype Time wasn't covered in the lecture slides and b) because in the example in the Walk Programme table within the the Capital Promenades Walk Programme pdf sheet, 'pm' or 'am' were attached to the time.

ENUM

In the **Participant** table, column **participant_membershiplevel** I've used ENUM which allows me to specify an option of two values one or the other, in our case, whether the participant is a 'regular' member or a 'concessionary' member. As a default I've put 'regular', because I assumed that this is the default. ENUMs are often used when the designer wants to restrict the values that can be put in a field or to assure that no other values are by mistake inserted.

INT

INT stores a minimum of 4 bytes which can represent a number long up to 11 places, so really a specification of (11) only provides for how many of those 11 places to be printed. Furthermore, in most places SMALLINT and perhaps at 1 instance (in Programme table, column: programme_maxparticipants) even TINYINT would be sufficient. (See https://dev.mysql.com/doc/refman/5.7/en/integer-types.html where TINYINT is the smallest and BIGINT is the largest. INT provides for 11 places including the sign.)

Still, because during class, practice was always to choose the default of INT(11), I chose to put them this way, which in any case allows for the maximum number of digits to be displayed if required.

Describe the relationships

a) The WALK table lists a set of walks that are all unique. However, the problem is that a participant can choose the same walk title but which occur at different days or times. Also, more than one participant are to opt for the same walk title.

b) Furthermore, there is a table listing a list of Guides that run/lead the walks. All unique. The problems however is that one particular guide can run different walk titles as well as the same walk title but at different days or times. In addition, there can be more than one guide to a given variant of a walk title and they can vary each time the walk occurs.

To fix those problems, I've first introduced the booking_programme_id, booking_participant_id columns in the Booking table. As foreign keys, they links participants' ids (from the Participant table) together with their respective orderitem from the Programme table. In other words, instead of booking a Walk, they book a programme item, which refers to an occurrence of a walk.

I've then introduced the Guideforwalk table which through the foreign key-columns guideforwalk_programme_id, guideforwalk_guide_id, each guide is linked with an occurrence of a programme. And so multiple guides can be allocated to the same occurrence or one particular guide can be allocated to different walks as well as different occurrences of the same walk..

So there was a many to many relationship 1. between Programmes and Guide and 2. Between Walks and Programmes and Participants and by introducing the linking table Guideforwalk, as well as the foreign keys in table Programmes and Bookings, I've managed to break those many to many relationships and convert them to a one-to-many.