Dr Greg Wadley

THE UNIVERSITY OF MELBOURNE

INFO90002 Database Systems & Information Modelling

Week 03

Data Modelling (1)

Today's session

- Data modelling
  - ER modelling conventions
  - Identifying entities and business rules
  - one-to-many (1-M) relationships
- SQL
  - Overview and history
  - Create tables
  - Insert data into tables
  - Read data from tables

© University of Melbourne 2018

Databases and ER models

- A database can be thought of as a representation of
  - a collection of entity sets, and
  - relationships between the entities
- An entity is an object or abstract concept or event which can be distinguished from other entities
  - example: product, order, sale, person, movie, tweet
- Entities have attributes that describe the entity and distinguish it from other entities in the same entity set
  - example attributes: EmployeeName, Address

- (reminder – what are “sets”?
  - union, intersection, Cartesian product)

© University of Melbourne 2018

Valid and invalid entities

- An Entity
  - Will have many instances in the database
  - Has several attributes
  - Is necessary for the system to work
- Examples
  - Person: EMPLOYEE, STUDENT, PATIENT
  - Place: STORE, WAREHOUSE, STATE, CITY
  - Object: PRODUCT, MACHINE, BUILDING, VEHICLE
  - Event: SALE, REGISTRATION, BROADCAST
  - Abstract: ACCOUNT, UNI SUBJECT, ROLE
- Entities do not usually include:
  - An output of the system (i.e. a report)
  - The system itself
  - The company that owns the system

© University of Melbourne 2018

Naming Conventions

- Entities
  - singular nouns
    - Employee, Customer, Sale (without or without capital letter)
- Attributes
  - usually a noun
    - itemColour, quantitySold, id
- Relationships
  - verbs or verb phrases
    - has, wants, manages, performs work for
- Use names meaningful to the domain
  - try not to abbreviate names
    - except num or nbr for number, ID for identifier
  - conventions on UPPER and lower case

© University of Melbourne 2018

Modelling a Single Entity

- By searching for nouns in the case we can identify entities

  (for example – Customer)

- What things would we need to record about the Customer
  - these become the customer's Attributes

- How can we identify individual Customers?
  - by name?
  - by address?
- Now we can draw it as an entity in the ER diagram

© University of Melbourne 2018

Mapping ER diagram to database tables

Employee	
PK	**EmployeeId**
	EmployeeName
	EmployeeSalary
FK1	**DepartmentId**
FK2	BossId

EmployeeID	EmployeeName	EmployeeSalary	DepartmentID	BossID
1	Alice	75000.00	1	0
2	Ned	45000.00	11	1
3	Andrew	25000.00	11	2
4	Clare	22000.00	11	2
5	Todd	38000.00	2	1
6	Nancy	22000.00	2	5
7	Brier	43000.00	9	1
8	Sarah	56000.00	9	7
9	Sophie	35000.00	10	1
10	Sanjay	15000.00	6	3

- Entity set
  - Often corresponds to a table in the database
- Entity instance
  - Often corresponds to a row in a table
- Attribute
  - Often corresponds to a column in a table
- Relationship set (link between entity sets)
  - Often corresponds to a Foreign Key in a table
- Relationship instance (link between entity instances)
  - Foreign Key value = Primary Key value

© University of Melbourne 2018

How to draw ER diagrams?

- You can use paper or the software of your choice
- *Visio* is a good (commercial) tool, available in labs
- https://www.lucidchart.com/ and other web apps
- Assignment 1 model should be made in *MySQL Workbench*
- In the exam you'll need to model *on paper*
- Diagrams in lecture slides are made in Workbench and Visio
- My suggestion is:
  - Use pen-and-paper or whiteboard for early Conceptual modelling
  - Use paper, Visio or Workbench for subsequent Logical modelling
  - Use Workbench for the final Physical model

© University of Melbourne 2018

Variation in ER diagram standards

© University of Melbourne 2018

Variation in ER diagram standards

Variation in ER diagram standards

Representing entities and attributes

- Entity

Entity1	
PK	Identifier
	Ent1Attribute1 Ent1Attribute2

- Attributes

- Key (or Identifier)
  - Fully identifies an instance
- Partial Key
  - Partially identifies an instance
- Attributes
  - Mandatory
  - Optional
  - Derived
    - [YearsEmployed]
  - Multivalued
    - {Skill}
  - Composite
    - Name (First, Middle, Last)

© University of Melbourne 2018

Conceptual design for single entity

(drawn using Visio software)

Customer1	
PK	**CustomerID**
	CustFirstName CustMiddleName **CustLastName** BusinessName **CustType** **CustAddress(Line1, Line 2, Suburb, Postcode, Country)**

- underline = primary key
- bold = not null
- () = composite attribute

© University of Melbourne 2018

Convert to *logical* design

Customer1	
PK	**CustomerID**
	CustFirstName CustMiddleName **CustLastName** BusinessName **CustType** **CustAddress(Line1, Line 2, Suburb, Postcode, Country)**

Customer1	
PK	**CustomerID**
	CustFirstName CustMiddleName **CustLastName** BusinessName **CustType** **CustAddLine1** **CustAddLine2** **CustSuburb** **CustPostcode** **CustCountry**

- Composite attributes become individual attributes
- Multi-valued attributes become a new table
- Resolve many-many relationships via a new table
- Add foreign keys at crows foot end of relationships

© University of Melbourne 2018

Convert to *physical* design

- Determine data types for each attribute
- Key, nullable

Customer1		
PK	CustomerID	SMALLINT
	CustFirstName	VARCHAR(100)
	CustMiddleName	VARCHAR(100)
	CustLastName	**VARCHAR(100)**
	BusinessName	VARCHAR(100)
	CustType	**CHAR(1)**
	CustAddLine1	**VARCHAR(100)**
	CustAddLine2	**VARCHAR(100)**
	CustSuburb	**VARCHAR(60)**
	CustPostcode	**CHAR(6)**
	CustCountry	**VARCHAR(60)**

Physical Name	Data Type	Req'd	PK	Notes
CustomerID	SMALLINT	☑	☑	CustomerID identifies Customer1
CustFirstName	VARCHAR(100)	☐	☐	CustFirstName is of Customer1
CustMiddleName	VARCHAR(100)	☐	☐	CustMiddleName is of Customer1
CustLastName	VARCHAR(100)	☑	☐	CustLastName is of Customer1
BusinessName	VARCHAR(100)	☐	☐	BusinessName is of Customer1
CustType	CHAR(1)	☑	☐	NOTE: This will be implemented as an ENUM type in MySQL with t
CustAddLine1	VARCHAR(100)	☑	☐	CustAddLine1 is of Customer1
CustAddLine2	VARCHAR(100)	☑	☐	CustAddLine2 is of Customer1
CustSuburb	VARCHAR(60)	☑	☐	CustSuburb is of Customer1
CustPostcode	CHAR(6)	☑	☐	CustPostcode is of Customer1
CustCountry	VARCHAR(60)	☑	☐	CustCountry is of Customer1

© University of Melbourne 2018

Dealing with multi-valued attributes

Conceptual Design

Employee	
PK	EmpID
	EmpFirstName EmpLastName {Role}

Logical Design

StaffRole is an example of a *weak entity*

© University of Melbourne 2018

Business Rules and Relationships

- Business rules are assertions that constrain entities
- Can impact structure and behaviour of the database
- Business rules can be assertions about attributes
  - "Quantity bought must be between 1 and 200." (assertion)
    - quantity bought (attribute)
- Or business rules can be assertions about entities
  - "A customer sets up at least one account." (assertion)
    - customer (entity)
    - account (entity)
- The latter kind of business rules are represented in our data models as *relationships between entities*.

© University of Melbourne 2018

Keys

- Keys or Identifiers are used to identify individual entity instances
  - Primary Key
    - (set of) columns, the values in which uniquely identify each instance
    - no column can be removed from the key without losing uniqueness
  - Candidate Key
    - the set of possible primary keys (choose one to be the PK)
  - Surrogate Key
    - system-assigned serial number (used if natural PK is unavailable or unsuitable)
  - Composite Key
    - a key which is made up of more than one attribute
      - e.g. for the entity “airline flight” we might use the composite key
        - » FlightNumber + FlightDate
  - Foreign Key
    - the key used to link to a primary key in another table
      - helps us to join tables in a Select statement
- Primary Keys are
  - unique
  - never null
  - do not change their value

© University of Melbourne 2018

Two entities with 1-M relationship

- Example: "A customer can set up several accounts."
  - The tables get linked through a foreign key

CustID	Customer FirstName	CustMiddle Name	CustLast Name	Business Name	CustType
1	Peter		Smith		Personal
2	James		Jones	JJ Enterprises	Company

AccountID	AccountName	Outstanding Balance	CustID
01	Peter Smith	245.25	1
05	JJ Ent.	552.39	2
06	JJ Ent. Mgr	10.25	2

© University of Melbourne 2018

Conceptual to logical design – with FK

- Conceptual Design

- Logical Design

– Add foreign keys at crows feet end of relationships
  - FK1 – CustomerID
    – This is the link to the customer table
    – Every CustomerID in Account must be present in Customer
      » Referential integrity

© University of Melbourne 2018

Physical design - with FK

- Attribute data types

FK must have the same data type as the PK it refers to.

© University of Melbourne 2018

Now we can create the tables

```
CREATE TABLE Customer (
  CustomerID        smallint                    auto_increment,
  CustFirstName     varchar(100),
  CustMiddleName    varchar(100),
  CustLastName      varchar(100)                NOT NULL,
  BusinessName      varchar(200),
  CustType          enum('Personal','Company')  NOT NULL,
  PRIMARY KEY  (CustomerID)
) ENGINE=InnoDB;
```

```
CREATE TABLE Account (
  AccountID             smallint        auto_increment,
  AccountName           varchar(100)    NOT NULL,
  OutstandingBalance    DECIMAL(10,2)   NOT NULL,
  CustomerID            smallint        NOT NULL,
  PRIMARY KEY  (AccountID),
  FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Referential Actions how foreign keys guarantee referential integrity.

© University of Melbourne 2018

Insert - with FK

Current Database

CustID	CustomerFirstName	CustMiddle Name	CustLastName	BusinessName	CustType
1	Peter		Smith		Personal
2	James		Jones	JJ Enterprises	Company

AccountID	AccountName	OutstandingBalance	CustID

Insert a row…

```
INSERT INTO ACCOUNT VALUES (DEFAULT, 'My New Account', 0, 5);
```

What happens?

INSERT INTO ACCOUNT VALUES (DEFAULT, ...	Error Code: 1452. Cannot add or update a child row: a f

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`db_seanbm/account`, CONSTRAINT `account_ibfk_1` FOREIGN KEY (`CustomerID`) REFERENCES `customer` (`CustomerID`))

© University of Melbourne 2018

Insert - with FK

- Run the Inserts…

```
INSERT INTO ACCOUNT VALUES (DEFAULT, 'Peter Smith', 245.25, 1);
INSERT INTO ACCOUNT VALUES (DEFAULT, 'JJ ENt.', 552.39, 2);
INSERT INTO ACCOUNT VALUES (DEFAULT, 'JJ ENt. Mgr', 10.25, 2);
```

CustID	CustomerFirstName	CustMiddle Name	CustLastName	BusinessName	CustType
1	Peter		Smith		Personal
2	James		Jones	JJ Enterprises	Company
3	Akin		Smithies	Bay Wart	Company

AccountID	AccountName	OutstandingBalance	CustID
01	Peter Smith	245.25	1
02	JJ Ent.	552.39	2
03	JJ Ent. Mgr	10.25	2

© University of Melbourne 2018

Relationship Degree

How many entities take part in the relationship?

Unary (1)

Binary (2)

Ternary (3)

© University of Melbourne 2018

Relationship Cardinality

- One to One
  - Each entity in one set is related to 0 or 1 in the other.
- One to Many
  - Each entity in one set is related to many in the other.
- Many to Many
  - Each entity in either set can be related to many in the other set
  - These require an extra step to implement in a relational database.

Optional Many

Mandatory Many

Optional One

Mandatory One

© University of Melbourne 2018

Strong and Weak entities

- Strong Entity
  - entity 2's PK is independent of the PKs of other entities

Weak Entity

- entity 2's PK depends on (includes) the PK of entity 1

© University of Melbourne 2018