Conceptual and Logical Database Design Recap

Watch video: https://youtu.be/nTWEyZEqPY4?t=10m0s

Overview Database Design Methodology

- Step 1 Build conceptual data model.
- Step 2 Build and validate logical data model.
- Step 3 Translate logical data model for target DBMS.
- Step 4 Design file organisations and indexes.
- Step 5 Design user views.
- Step 6 Design security mechanisms.
- Step 7 Consider the introduction of controlled redundancy.
- Step 8 Monitor and tune the operational system.

Overview Database Design Methodology

Build conceptual data model

- Step 1.1 Identify entity types.
- Step 1.2 Identify relationship types.
  - 1.2.1 Cardinality.
  - 1.2.2 Participation.
- Step 1.3 Identify and associate attributes with entity or relationship types.
- Step 1.4 Determine attribute domains.
- Step 1.5 Determine candidate, primary, and alternate key attributes.
- Step 1.6 Consider use of enhanced modelling concepts (optional step).
- Step 1.7 Check model for redundancy.
- Step 1.8 Validate conceptual model against user transactions.
- Step 1.9 Review conceptual data model with user.

Overview Database Design Methodology

Build and validate Logical Data Model

- Step 2.1 Derive relations for logical data model.
- Step 2.2 Validate relations using normalisation.
- Step 2.3 Validate relations against user transactions.
- Step 2.4 Define integrity constraints.
- Step 2.5 Review logical data model with user.
- Step 2.6 Merge logical data models into global model (optional step).
- Step 2.7 Check for future growth.

Worked Example

- A company wishes to create a database about the running of training courses for its staff training system.
- Each staff member may attend a number of training courses and a training course must involve at least 5 staff members. A training course is run by either one, two or three trainers and each trainer will run one or more training courses. Each training course is held at just one venue and most (but not all) venues hold several training courses. Every training course involves just one course theme and course themes are involved in just one training course (with some themes recorded for some future allocation to an actual training course).

Worked Example

- Staff member details to be recorded include staff number (unique), staff name, department, phone number, and a number of qualifications. Trainer details include trainer code (unique), trainer name and phone number. Training course details include course code (unique), course date and length. Course Theme includes theme number (unique), theme name and theme area. Venue details include venue name (unique), capacity and distance. The outcome for each staff member who attends a particular course (i.e. pass/fail) is also recorded. The role of the trainer on each particular training course (lead, support, administration) is also recorded.

Worked Example

- **Step 1.1 Identify entities**

Worked Example

- **Step 1.1 Identify entities**
  - Staff
  - Training Course
  - Trainer
  - Venue
  - Theme

Worked Example

- **Step 1.2 Identify relationships**

Worked Example

- **Step 1.2 Identify relationships**

  - Each staff member may attend a number of training courses and a training course must involve at least 5 staff members.
    - Staff 5..* **attends** 0..* Training Course

  - A training course is run by either one, two or three trainers and each trainer will run one or more training courses.
    - Trainer 1..3 **runs** 0..* Training Course

Worked Example

- **Step 1.2 Identify relationships**
  - Each training course is held at just one venue and most (but not all) venues hold several training courses.
    - Venue 1..1 **holds** 0..* Training Course
  - Every training course involves just one course theme and course themes are involved in just one training course (with some themes recorded for some future allocation to an actual training course).
    - Training Course 0..1 **involves** 1..1 Theme

Worked Example

- **Step 1.3 Identify and associate attributes with entities or relationships**

Worked Example

- **Step 1.3 Identify and associate attributes with entities or relationships**

- **Entity Type attributes:**
  - **Staff**:staffNo, name (fName, lName), department, contactNumber, qualifications
  - **Training Course**: courseCode, courseDate, length
  - **Trainer**: trainerCode, name (fName, lName), phoneNumber
  - **Venue**: venueName, capacity, distance
  - **Theme**: themeNumber, themeName, themeArea

Worked Example

- **Step 1.3 Identify and associate attributes with entities or relationships**
  - **Relationship Type attributes:**
    - The *outcome* for each staff member who attends a particular course (i.e. pass/fail) is also recorded.
    - The *role* of the trainer on each particular training course (lead, support, administration) is also recorded.

Worked Example

- **Step 1.4 Determine attribute domains**

- In the data dictionary record the allowable set of values for the attribute; and the size and format of the attribute.

Worked Example

- **Step 1.5 Determine candidate, primary, and alternate key attributes**

Worked Example

- **Step 1.5 Determine candidate, primary, and alternate key attributes**

  - **Customer**: Candidate keys: staffNo, contactNumber
    Primary key: staffNo

  - **Training Course**: Candidate key: courseCode
    Primary key: courseCode

  - **Trainer**: Candidate key: trainerCode
    Primary key: trainerCode

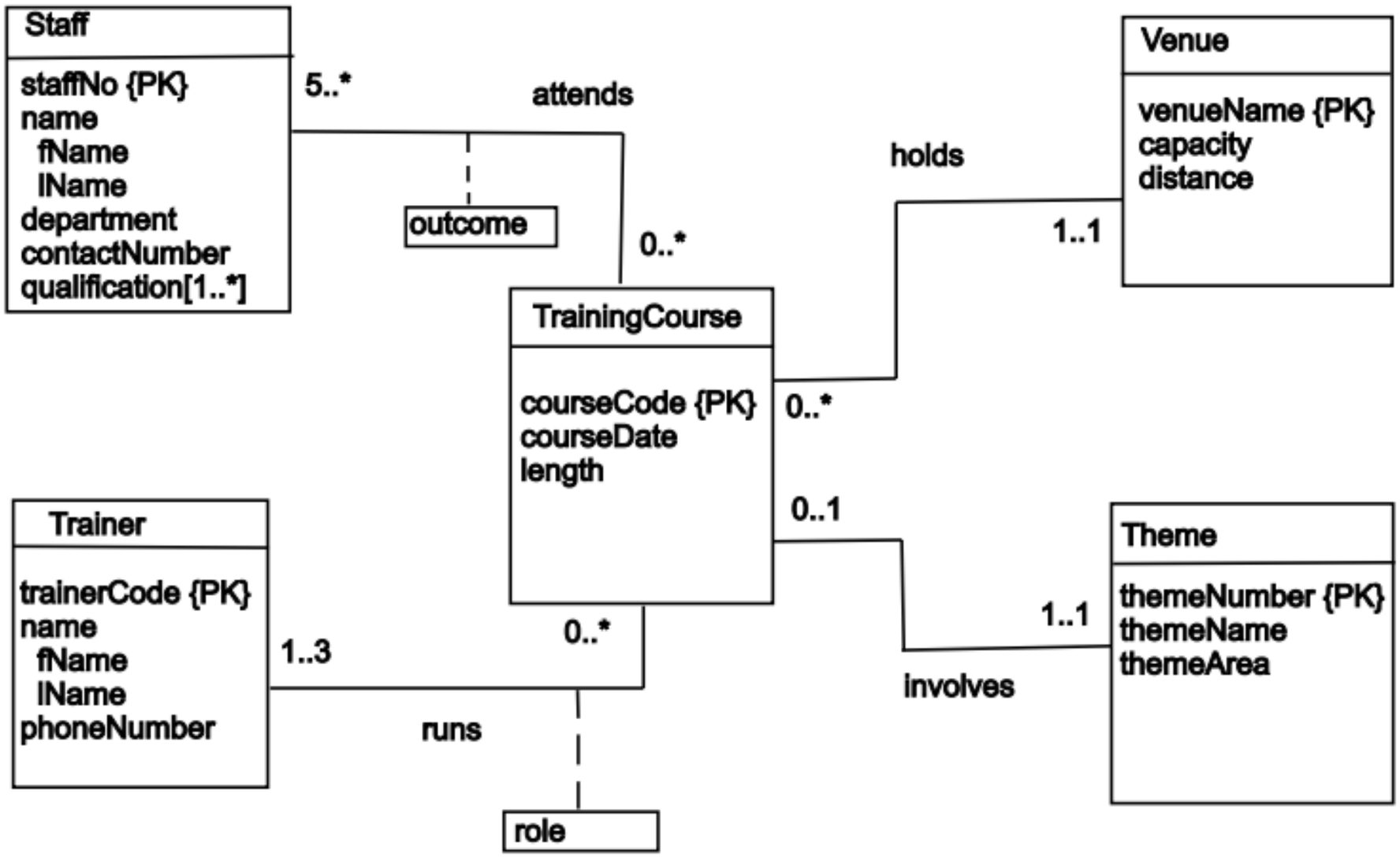
Worked Example

- **Step 1.5 Determine candidate, primary, and alternate key attributes**

  - **Venue**: Candidate keys: venueName
    Primary key: venueName

  - **Theme**: Candidate keys: themeNumber, themeName
    Primary key: themeNumber

Worked Example



Worked Example

- Transform the Conceptual Data Model into a set of relations.

Worked Example

- Firstly, we will map all the entity types to a set of relations:

Worked Example

- Firstly, we will map all the entity types to a set of relations:

  Staff(staffNo, fName, lName, department, contactNumber
  Primary key staffNo

  - We will deal with the multivalued attribute *qualification* by creating a new relation staffQualifications and adding it and the associated Primary key attribute as relation attributes.

  staffQualifications(staffNo, qualification)
  Primary key staffNo, qualification
  Foreign key staffNo references Staff(staffNo)

Worked Example

TrainingCourse(courseCode, courseDate, length
Primary key courseCode

Trainer(trainerCode, fName, lName, phoneNumber
Primary key trainerCode

Venue(venueName, capacity, distance
Primary key venueName

Theme(themeNumber, themeName, themeArea
Primary key themeNumber

Worked Example

- Secondly, we will map the relationships:

Worked Example

- Secondly, we will map the relationships:
  - *attends* relationship
    - The ‘attends’ relationship between Staff and Training Course is *:* and optional on the Staff side. Since this is a *:* relationship we create a new relation.

Worked Example

- Secondly, we will map the relationships:
  - *attends* relationship
    - The ‘attends’ relationship between Staff and Training Course is *:*. Since this is a *:* relationship we create a new relation.

attends(staffNo, courseCode, outcome)
Primary key staffNo, courseCode
Foreign key staffNo references Staff(staffNo)
Foreign key courseCode references TrainingCourse(courseCode)

Worked Example

- *runs* relationship
  - The ‘runs’ relationship between Trainer and Training Course is *:*. Since this is a *:* relationship we create a new relation.

Worked Example

- *runs* relationship
  - The ‘runs’ relationship between Trainer and Training Course is *:*. Since this is a *:* relationship we create a new relation.

runs(trainerCode, courseCode, role)
Primary Key trainerCode, courseCode
Foreign Key trainerCode references Trainer(trainerCode)
Foreign Key courseCode references TrainingCourse(courseCode)

Worked Example

- *holds* relationship
  - The ‘holds’ relationship between Venue and Training Course is 1:* and optional on the Venue side. Since this is a 1:* relationship we post the primary key from the ‘one’ side into the table on the ‘many’ side as a foreign key.

Worked Example

- *holds* relationship
  - The ‘holds’ relationship between Venue and Training Course is 1:* and optional on the Venue side. Since this is a 1:* relationship we post the primary key from the ‘one’ side into the table on the ‘many’ side as a foreign key.

TrainingCourse(courseCode, courseDate, length, venueName
Primary key courseCode
Foreign Key venueName references Venue(venueName)

Worked Example

- *involves* relationship
  - The ‘involves ’ relationship between Training Course and Theme is 1:1 and optional on the Theme side. Since this is a 1:1 relationship and optional on one side, post the primary key from the ‘optional’ side into the table on the ‘mandatory’ side as a foreign key.

Worked Example

- *involves* relationship
  - The ‘involves ’ relationship between Training Course and Theme is 1:1 and optional on the Theme side. Since this is a 1:1 relationship and optional on one side, post the primary key from the ‘optional’ side into the table on the ‘mandatory’ side as a foreign key.

TrainingCourse(courseCode, courseDate, length, venueName, themeNumber
Primary key courseCode
Foreign Key venueName references Venue(venueName)
Foreign key themeNumber references Theme(themeNumber)

Worked Example

- ***Full set of tables:***

  Staff(staffNo, fName, lName, department, contactNumber)
  Primary key staffNo

  staffQualifications(staffNo, qualification)
  Primary key staffNo, qualification
  Foreign key staffNo references Staff(staffNo)

  TrainingCourse(courseCode, courseDate, length, venueName, themeNumber)
  Primary key courseCode
  Foreign Key venueName references Venue(venueName)
  Foreign key themeNumber references Theme(themeNumber)

  Trainer(trainerCode, fName, lName, phoneNumber)
  Primary key trainerCode

Worked Example

Venue(venueName, capacity, distance)
Primary key venueName

Theme(themeNumber, themeName, themeArea)
Primary key themeNumber

attends(staffNo, courseCode, outcome)
Primary key staffNo, courseCode
Foreign key staffNo references Staff(staffNo)
Foreign key courseCode references TrainingCourse(courseCode)

runs(trainerCode, courseCode, role)
Primary Key trainerCode, courseCode
Foreign Key trainerCode references Trainer(trainerCode)
Foreign Key courseCode references TrainingCourse(courseCode)

Worked Example

- All of the tables are in 3NF (and indeed BCNF).

Worked Example

Referential Integrity Constraints

Staff(staffNo, fName, lName, department, contactNumber)
Primary key staffNo

staffQualifications(staffNo, qualification)
Primary key staffNo, qualification
Foreign key staffNo references Staff(staffNo) on update cascade
on delete cascade

TrainingCourse(courseCode, courseDate, length, venueName, themeNumber)
Primary key courseCode
Foreign Key venueName references Venue(venueName)
on update cascade on delete set null
Foreign key themeNumber references Theme(themeNumber)
on update cascade on delete set null

Worked Example

Referential Integrity Constraints

attends(staffNo, courseCode, outcome)
Primary key staffNo, courseCode
Foreign key staffNo references Staff(staffNo) on update cascade
on delete cascade
Foreign key courseCode references TrainingCourse(courseCode)
on update cascade on delete no action

runs(trainerCode, courseCode, role)
Primary Key trainerCode, courseCode
Foreign Key trainerCode references Trainer(trainerCode)
on update cascade on delete no action
Foreign Key courseCode references TrainingCourse(courseCode)
on update cascade on delete no action