4. Relational Databases

Fundamentals of Data Management

Dr Charaka Palansuriya

Outline

- What is a database
- Keys and constraints
- Design considerations
- Querying a relational database
- Accessing relational databases via applications
- Transactions
- Performance optimisation
- Backup and restore

What is a database

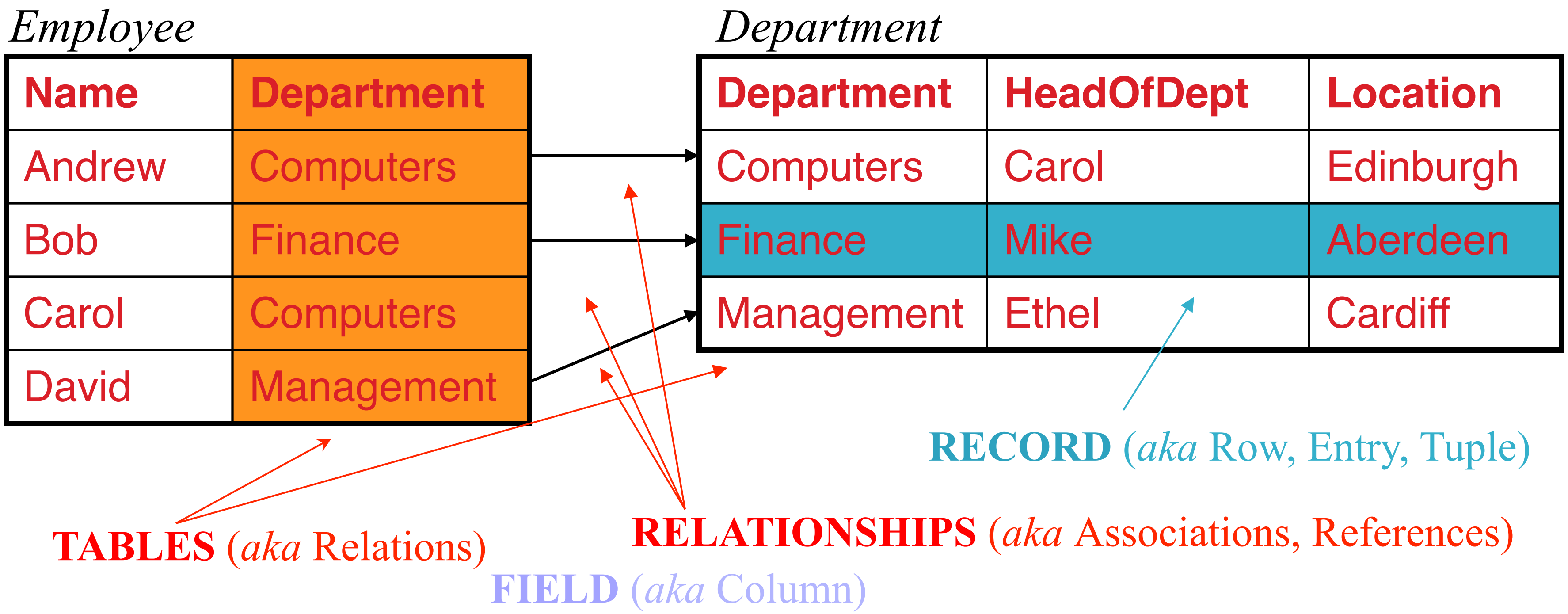
- Means of storing and accessing data efficiently
- Usually contains a Database Management System (DBMS)
- DBMS provides:
  - Mechanisms to create data structure (e.g., Tables) and content
  - A means of querying and modifying content
  - Ways to optimise performance
  - A tool to backup or archive
  - Ways to allow applications to access data

Types of databases

- There are various types of databases
  - Relational Databases
  - NoSQL Databases
  - Other databases
    - Hybrids,
      - PostgresSQL (object-relational)
- This course will look at two of the most popular and widely used types:
  - Relational
  - NoSQL
- This lecture focus on Relational Databases
  - By far the most common
  - Examples: Oracle, Microsoft SQL Server, MySQL

What is a Relational Database?

- A Collection of tables (i.e., Relations) with associated relationships



Employee

Name	Department
Andrew	Computers
Bob	Finance
Carol	Computers
David	Management

Department

Department	HeadOfDept	Location
Computers	Carol	Edinburgh
Finance	Mike	Aberdeen
Management	Ethel	Cardiff

RECORD (*aka* Row, Entry, Tuple)

TABLES (*aka* Relations)

RELATIONSHIPS (*aka* Associations, References)

FIELD (*aka* Column)

Primary Key and Indexes

- A Primary Key is a field which is guaranteed to hold a unique value in every entry in a table
  - A unique key may be generated automatically by the database management system
- An index is a means of accessing the entries in a table efficiently
  - i.e., indexing is used to improve the performance of querying a database

Constraints

- Domain Constraints
  - A value for a field must be picked from a particular set (or range) of pre-defined values
- Uniqueness Constraints
  - If a field *F* is designated as a key, it is not possible to add a record which has a value of *F* equal to that of any existing record
- Null Constraint
  - Specifies whether or not a field may be Null
- Referential integrity
  - When a table refers to a primary key of another table, i.e., foreign key, a new record can only be added if the record refers to a valid primary key of the other table. For example, if a foreign key is deleted then all records referring to this foreign key must be deleted.

Constraints - continued

- Sematic Constraints
  - *e.g.,* The salary of an employee must be less than that of his boss
- Dynamic Constraints
  - *e.g.,* The salary of an employee can only increase

Design Considerations

- Minimise data redundancy

Name	Company	Address	Postcode
A. Smith	Smith & Son Ltd	51 High Street	PC7 4LT
J. Smith	Smith & Son Ltd	51 High Street	PC7 4LT
T. Jones	Smith & Son Ltd	51 High St	PC7 4LT
N. Dupont	Flash Lighting Co	14 Howe Crescent	RN4 8PU

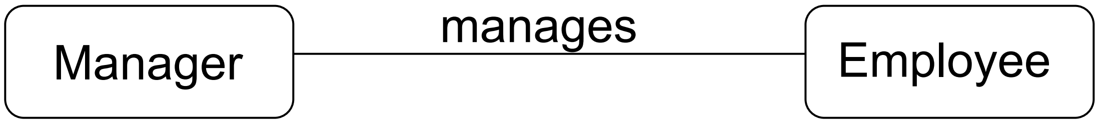
Redundancy

Design Considerations: Data Normalisation

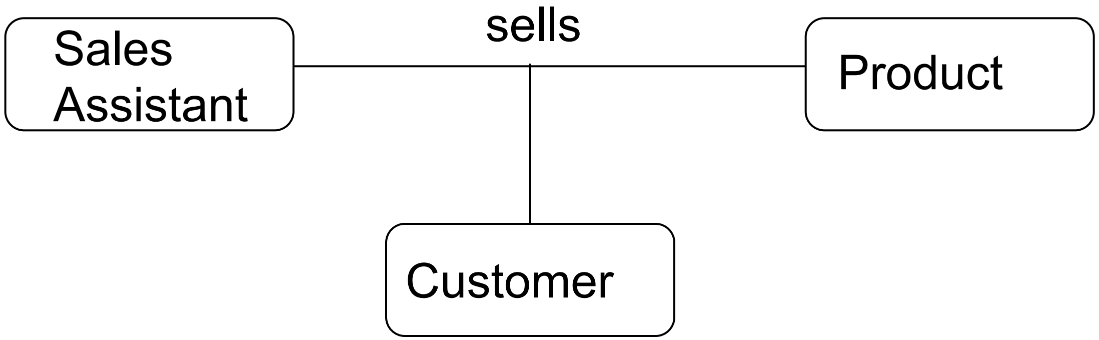
- First Normal Form
  - Eliminate repeating sets of related data in an individual table
  - Create a separate table for each set of related data
  - Identify each set of related data with a primary key
- Second Normal Form
  - Non-primary key columns must depend on the entire primary key, not just part of the primary key
    - Applies when a primary key is based on more than one column
- Third Normal Form
  - Non-primary key columns must depend only on primary key
- Higher Normal Forms
  - Fourth (Boyce Codd) Normal Form and Fifth Normal Form do exist in theory but rarely used in practice

Design Considerations: ER Diagrams

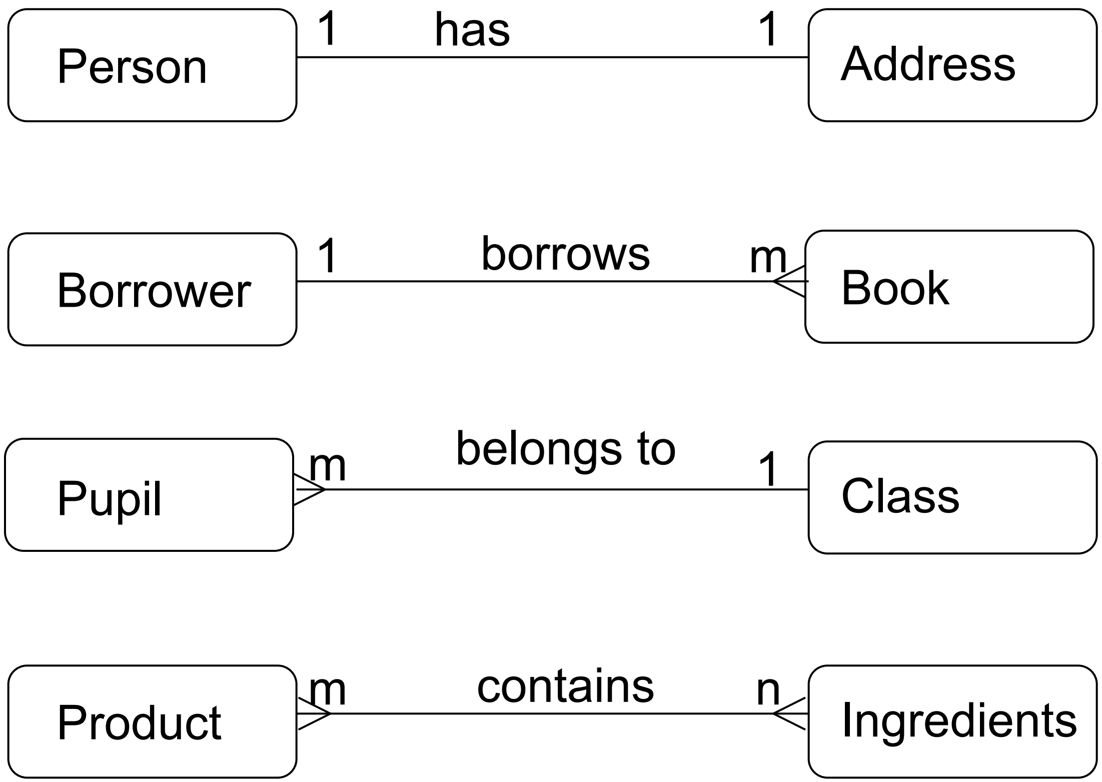
- Binary Relationship



- Tertiary Relationship



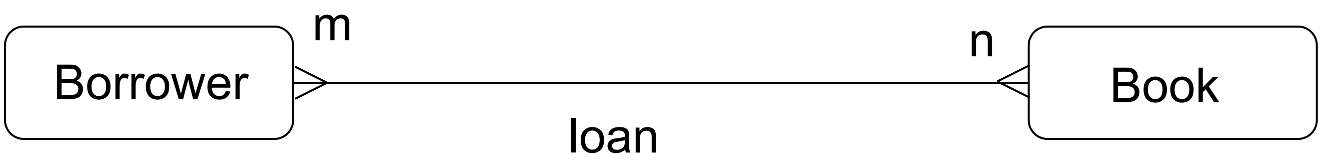
Cardinality

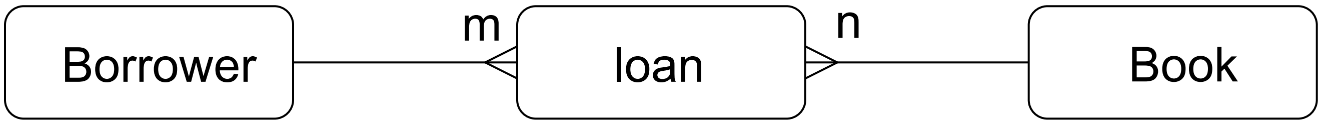


How ER Diagrams Can Be Useful

- Produces a data model representing a real-world situation
  - Identifies important entities (tables)
  - Relationship between the entities.
- Allows simplification of the data model
  - E.g., Remove many-to-many relationships

Removing Many-to-Many Relationships





Querying a relational database

- Relational databases are queried using SQL (Structured Query Language)
- Used by all major DBMSs
- Standards exist
  - SQL:2003
  - Dialects exists too
- The result of an SQL query is another table

SQL SELECT

- SELECT Name, Department FROM Employee WHERE Department='Computers'

Name	Department
Andrew	Computers
Carol	Computers

- Print out all fields using the * notation:

  SELECT * FROM Employee WHERE Department='Computers'

Joins

- Joins enable extraction of information from more than one table
- SELECT Name, Employee.Department, Department.Location AS Name, Dept, Location FROM Employee, Department WHERE Employee.Department=Department.Department

Name	Dept	Location
Andrew	Computers	Edinburgh
Bob	Finance	Aberdeen
Carol	Computers	Edinburgh
David	Management	Cardiff

AND, OR, Numerical Comparison

- Boolean operators “AND” and “OR” can be used:
  - **SELECT name,department FROM Employee WHERE name='Andrew' OR name='Bob’**
- Brackets can be used to group conditions in the usual way.
- For numerical fields, all of the usual operators exist: <,>,=,>=,<=,<> (some implementations also accept !=)
- Negation can also be performed with NOT.
  - **SELECT student FROM class WHERE NOT(mark1>50)** is equivalent to
  - **SELECT student FROM class WHERE mark1<=50**

Pattern Matching SELECT

- SQL includes simple pattern matching/wildcards
- Wildcards % (zero or more characters) and _ (exactly one character)
  - Don’t work for equality expressions; need to use LIKE keyword.
  - SELECT name FROM addressbook WHERE name LIKE 'Car%’
    - Would pick out Caroline, Carl, Carol, cAroLinE
  - SELECT name FROM addressbook WHERE name LIKE 'A_a_'
    - Would pick out Adam, Alan (but not Armstrong)

Aggregation

- SQL includes aggregation functions
- SUM, AVG, COUNT, MIN, MAX
- SELECT count(*) FROM Customers
  - Will count the number of customers
- Some implementations provide many other functions

Ordering

- Results (*not* the original tables) can be sorted by one or more fields
  - SELECT name, salary FROM employees ORDER BY salary, name
    - Lists all employees ordered by (increasing) salary, and then those with the same salary are listed alphabetically
- Reverse order sorting can be performed with the DESC keyword (for DESCending)
  - SELECT name, salary FROM employees ORDER BY salary DESC, name

Arithmetic Operations

- Arithmetic can be performed on numerical fields with the usual operators: +,-,*,/.
- The BETWEEN keyword can be used
  - SELECT name,salary FROM employee WHERE salary BETWEEN 10000 AND 20000
  - Above is equivalent to:
    - SELECT name,salary FROM employee WHERE salary >=10000 AND salary <=20000
  - Note that limits *are* included

Accessing Relational Databases via applications

- Databases could be accessed using the Command Line Tool (CLT) provided by a RDBMS
- Typically access to the relational databases are restricted and access provided via applications
- Standards exists for accessing relational databases via applications
  - JDBC: for Java applications
  - ODBC: for example used by Microsoft databases and applications
- Each RDBMS provides drivers which an application should use to access the database
  - For example: MySQL JDBC driver and ODBC driver

Performance Optimization

- Use Index
  - Index in a book provide a quick way to locate information.
  - Similarly, indexes in a table give a quick way to find relevant rows with specific column values in a query.
    - i.e., instead of scanning entire table row at a time, the query executioner can jump straight to the relevant rows using the index.
  - Typically single column is used for indexing; e.g.
    - Primary key
    - Important field that is most likely to be use in WHERE clause
      - E.g., name in an Employee table

Transactions

- A sequence of SQL statements executed as a single unit or they are all undone
  - Commonly referred to as a transaction is **committed** or **rolled back**
  - E.g., moving money between bank accounts

- Used for
  - Data consistency
    - When two or more users updating the same data at the same time
  - Data concurrency
    - Allowing two or more users to update data simultaneously

Transactions:ACID

- Transactions are ACID:
  - Atomicity
    - All commands of a transaction is performed or none of them
  - Consistency
    - A transaction takes a database from one consistent state to another consistent state.
  - Isolation
    - When transactions are executed in parallel, the effects of one transaction must not visible to another transaction until the transaction is committed.
  - Durability
    - After a transaction is committed, the changes made by that transaction is permanent. This is important, for example, if a system failure occurs.

Backup and restore

- Relational databases support backup and restoring databases
  - Useful for
    - recovering from system failures
    - Archiving data
- For example
  - A MySQL databases can be backed up by using “mysqldump” command
    - Backs up a database to a single file
  - Restore the database using the “source” command

Summary

- Good database design requires thought!
- SQL is used to query relational databases
- Database access is generally done via application
- Large databases would require performance optimisation

References

- Database design basic, http://office.microsoft.com/en-us/access-help/database-design-basics-HA001224247.aspx
- Oracle database documentation, http://docs.oracle.com/cd/E11882_01/
- A relational model of data for large shared data banks, http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf