# Databases

John Wordsworth
University of Reading

## How the course works

- Nine lectures, three each morning: Copies of the slides are provided.
- Three practicals, one each afternoon
- A text book for supplementary reading: Connolly and Begg, *Database Systems: A practical approach to design, implementation, and management*. Third Edition, Addison-Wesley, 2000

## What the course covers

- The relational model of data
- Using entity-relationship modelling to construct a relational database
- Using SQL to create, modify, and interrogate a database
- Database file organisations and indexing
- Transaction properties

What is your interest in this material?

## Lecture 1: Introduction

- What a database is, and how we managed before they were invented
- What data is, and how business uses it
- A three-level architecture for data
- Three models of data: hierarchical, network, and relational
- What a Database Management System (DBMS) is

## What is a database?

- "A shared collection of logically related data, and a description of that data, designed to meet the information needs of an organisation"
- A collection of files managed by a DBMS so as to maintain and make available the information needed by an organisation

## Application development: old-style

- Each application has its own files (Payroll, Stock control, Personnel, Sales, Invoicing).
- Each file has its own layout, and program structures are closely bound to file structures.
- Data is often duplicated, but cannot be easily shared.
- There is no uniform view of data and its importance to the business.

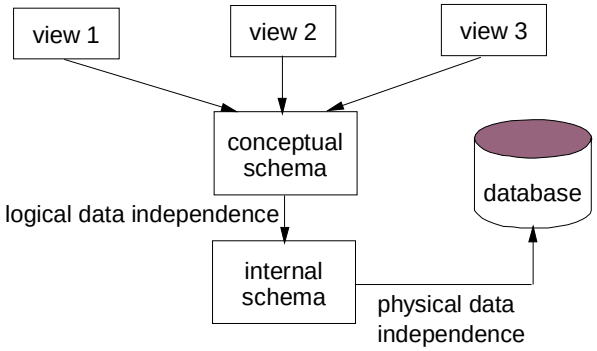
## What is data?

- Information about things and their relationships
- Things (entities, objects) have attributes.
  - A student has a registration number and an address.
  - A book has a title and a shelf number.
  - A course has a title and a maximum class size.
- Relationships relate things to one another.
  - A student enrols on a course.
  - A student borrows a book from the library.

## What is business?

- Creating new instances of existing things
  - Enrolling a new student
- Deleting existing instances of things
  - Withdrawing a book from the library
- Modifying existing instances of things
  - Changing a student's address
- Modifying existing relationships
  - Enrolling a student on a course
- Creating new kinds of thing and new kinds of relationship - less frequent

## ANSI-SPARC three-level architecture




## Three models of data

- Hierarchical
  - uses hierarchies of parent-child relationships
  - IMS, DL/I
- Network
  - uses sets defined by logical links
  - CODASYL
- Relational
  - uses tables based on mathematical relations
  - DB2, Oracle, Ingres, etc.

## What is a DBMS?

- "A DBMS is a software system that enables users to define, create, maintain, and control access to the database."
- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Provides security, integrity, concurrency, recovery, views.

## Summary

- Data is vital to business, and its management has moved from file-based processing to databases.
- Data is things and their relationships.
- The ANSI-SPARC architecture guides the analysis and implementation of databases.
- There are different models of data; we concentrate on the relational model.
- A DBMS is a software system that makes it all work.

## Lecture 2: Relational model and normalisation

- What the relational model is
- Keys and integrity
- Functional dependencies
- Anomalies of table structures
- Normalisation to remove anomalies

## The relational model

- The relational model models data as tables with rows and coumns.
- Each row in a table represent a thing.
- Each column represents an attribute.
- The cells contain values from an appropriate domain.
- Duplicate rows are not allowed.

## Two relations

Student

| SReg | SName | SHall | LName |
|---|---|---|---|
| 72978 | Rawsthorne | Cavendish | ABC |
| 25477 | Bliss | Cavendish | DEF |
| 40935 | Howells | Fitzroy | ABC |
| 78883 | Rutter | Russell | DEF |

Lecturer

| LName | LPhone |
|---|---|
| ABC | 3456 |
| DEF | 4567 |

## Keys and integrity

- Each table has one or more (simple or composite) candidate keys, one of which is chosen as primary key.
- Nulls can appear in a cell if a value is not known, or not applicable.
- Nulls can not appear in a primary key.
- A primary key in one table can be an attribute in another. This is a foreign key.
- Every value in a foreign key must be present in the other table.

## Candidate keys

University

| LName | LPhone | SReg | SName | SHall |
|---|---|---|---|---|
| ABC | 3456 | 72978 | Rawsthorne | Cavendish |
| DEF | 4567 | 25477 | Bliss | Null |
| ABC | 3456 | 40935 | Howells | Fitzroy |
| DEF | 4567 | 78883 | Rutter | Russell |

## Functional dependencies

- An attribute group B is functionally dependent on an attribute group A if distinct values of B are always associated with distinct values of A.
- Every attribute in a table is functionally dependent on the primary key.
- An attribute dependent on part of a composite primary key is partially dependent on the primary key.
- Transitive dependencies can occur.

## Keys and dependencies

University

| LName | LPhone | SReg | SName | SHall |
|---|---|---|---|---|
| ABC | 3456 | 72978 | Rawsthorne | Cavendish |
| DEF | 4567 | 25477 | Bliss | Cavendish |
| ABC | 3456 | 40935 | Howells | Fitzroy |
| DEF | 4567 | 78883 | Rutter | Russell |

## Partial dependencies

Enrolled

| SReg | SName | UCode | UTitle |
|---|---|---|---|
| 72978 | Rawsthorne | AB12 | Operating systems |
| 72978 | Rawsthorne | BC23 | Database |
| 25477 | Bliss | AB12 | Operating systems |
| 40935 | Howells | BC23 | Database |
| 78883 | Rutter | CD34 | Formal methods |

## Anomalies

- Deleting a row might remove information that should be kept.
- Insertion of a new row might require more information than is presently available.
- Updating one piece of information needs changes to several rows.

## Table with anomalies

University

| LName | LPhone | SReg | SName | SHall |
|---|---|---|---|---|
| ABC | 3456 | 72978 | Rawsthorne | Cavendish |
| DEF | 4567 | 25477 | Bliss | Cavendish |
| ABC | 3456 | 40935 | Howells | Fitzroy |
| DEF | 4567 | 78883 | Rutter | Russell |

## First normal form

- Every cell contains one value (or null)

University

| LName | LPhone | SReg | SName | SHall |
|---|---|---|---|---|
| ABC | 3456 | 72978<br>40935 | Rawsthorne<br>Howells | Cavendish<br>Fitzroy |
| DEF | 4567 | 25477<br>78883 | Bliss<br>Rutter | Cavendish<br>Russell |

## Second normal form

- Eliminate partial dependencies

Student

| SReg | SName |
|---|---|
| 72978 | Rawsthorne |
| 25477 | Bliss |
| 40935 | Howells |
| 78883 | Rutter |

Unit

| UCode | UTitle |
|---|---|
| AB12 | Operating systems |
| BC23 | Database |
| CD34 | Formal methods |

Enrolment

| SReg | UCode |
|---|---|
| 72978 | AB12 |
| 72978 | BC23 |
| 25477 | AB12 |
| 40935 | BC23 |
| 78883 | CD34 |

## Third normal form

- Eliminate transitive dependencies

Student

| SReg | SName | SHall | LName |
|---|---|---|---|
| 72978 | Rawsthorne | Cavendish | ABC |
| 25477 | Bliss | Cavendish | DEF |
| 40935 | Howells | Fitzroy | ABC |
| 78883 | Rutter | Russell | DEF |

Lecturer

| LName | LPhone |
|---|---|
| ABC | 3456 |
| DEF | 4567 |

## Other normal forms

- Boyce-Codd: Eliminate partial dependencies on candidate keys that are not the primary key.
- Fourth: Eliminate multivalued dependencies.
- Fifth: Eliminate join dependencies.

## Summary

- The relational model views data as a set of tables.
- Tables have keys, and there are integrity constraints between values.
- Functional dependencies of various kinds can exist between attributes in a table.
- Tables can be subject to various anomalies in processing.
- Normalisation is a systematic process for removing anomalies.

## Lecture 3: Entity-relationship modelling

- Entities and entity types
- Relationships
- The Unified Modelling Language (UML)
- Cardinality and participation in relationships
- Recursive relationships and roles
- Creating an E-R model
- Creating a relational model from an E-R model

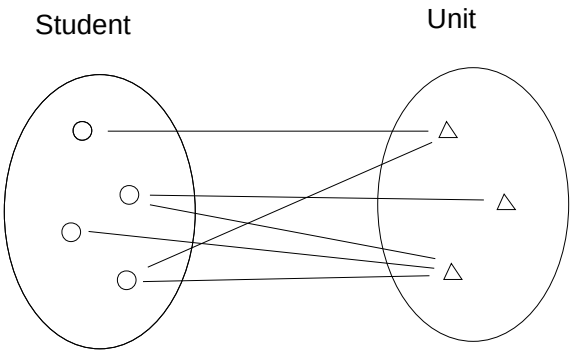
## Entities and entity types

- Entities are objects of interest to an organisation.
- Entities of the same kind constitute an entity type: Book is the entity type of books.
- Entities have attributes; each attribute has a domain; one or more attributes can be used as a key.
- Entity types whose existence depends on other entity types are weak; others are strong.

## Relationships

- A relationship is an association between entity types.
- A borrower (an entity from entity type Borrower) and a book (from entity type Book) become associated when the borrower borrows the book.
- A student and a unit become associated when a student enrols on a unit.
- Relationships can involve more than two entity types: binary, ternary, quaternary, etc.

## Relationships continued



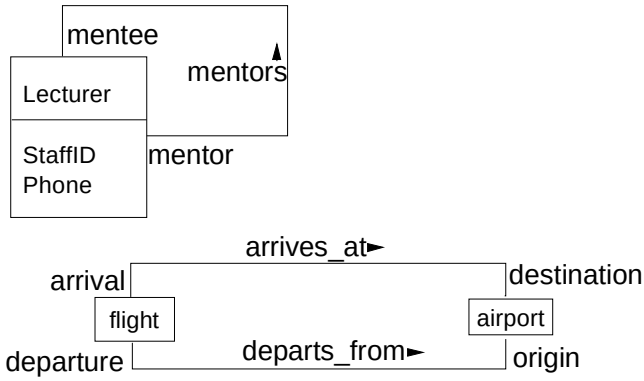

## Unified Modelling Language

- Devised for recording object-oriented models.
- Entity types are the classes.
- Each entity type has a name, and attributes.
- Relationships are noted by (directed) links between entity types.
- Relationships can have additional attributes.

## Cardinality and participation

- One book has at most one borrower. The Book-to-Borrower relationship is many-to-one, and Book and borrower both have partial participation in the relationship.
- Each lecturer has exactly one phone number: A one-to-one relationship with total participation by Lecturer.
- Each student can enrol on many units, and each unit can accommodate many students: A many-to-many relationship.

## Recursive relationships and roles



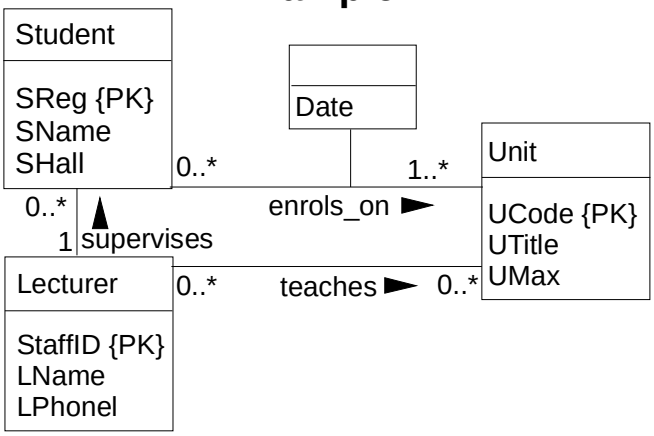

## Constructing and E-R model

- Identify the entity types.
- Choose the primary key attribute for each entity type.
- Identify the relationships between entity types, their cardinality and participation.
- Identify and record other attributes of the entity types.
- Validate the model against the proposed uses of it.

## Constructing a relational model

- Create a table for each entity type.
- Make the attributes of the entity type the coumns of the table.
- For a one-to-many relationship, add a foreign key to the "many" end to identify the "one" entity.
- For a many-to-many relationship, create a new entity type and table with two attributes, each atrribute being the primay key of the participating entity.

## Example




## Tables for the example

- Student with attributes SReg, SName, SHall, StaffID
- Lecturer with attributes StaffID, LName, LPhone
- Unit with attributes UCode, UTitle, UMax
- Enrolment with attributes SReg, UCode, Date
- Teaches with attributes StaffID, UCode

## Summary

- Entity-relationship models record an organisation's view of its data.
- The Unified Modelling Language can be used to record E-R models.
- Relational models can be derived from E-R models.
- E-R models are the key to good relational models of an organisation's data.

## Additional material for day 1

http://www.rdg.ac.uk/~sis00jbw/NLOD.html

- Supplementary reading to support the lectures
- Instructions for the lab work

## Lectures 4 and 5: Relational algebra and relational calculus

- Relational algebra as a collection of operations on tuples
- Unary operations of relational algebra
- Binary operations of relational algebra
  - Set operations
  - Joins
- Types of relational calculus
- Tuple relational calculus

## What is relational algebra?

- Relational algebra is a collection of operations on sets of tuples (ordered sets of values).
- A table in a relational model is regarded as a set of tuples (decorated with attribute names).
- Relational algebra operations can be used to express solutions to information problems.
- Relational algebra is the (formal) basis of most (user-friendly) DMLs.

## Projection

- A unary operation (Table -> Table) that produces a table containing specified columns from a table.
- $\pi_{\text{column\_list}}$ (T1) denotes the table formed from T1 by retaining only the columns named in column_list.
- Duplicate rows are absorbed.

## Selection

- A unary operation (Table -> Table) that produces a table containing specified rows of a table.
- $\sigma_{\text{predicate}}$ (T1) denotes the table formed from T1 by retaining only the rows that satisfy predicate.

## Union

- Union is a binary operation (Table x Table -> Table) that combines the contents of two tables that have the same columns.
- Duplicate rows are absorbed.

L1

| LName | LPhone |
|---|---|
| ABC | 3456 |
| DEF | 4567 |

L2

| LName | LPhone |
|---|---|
| ABC | 3456 |
| GHI | 5678 |

L1 ∪ L2=

| LName | LPhone |
|---|---|
| ABC | 3456 |
| DEF | 4567 |
| GHI | 5678 |

## Set difference

- Set difference is a binary operation (Table x Table -> Table) that produces a table that contains the rows of the first table that are not rows of the second, the two tables having the same attributes.

L1

| LName | LPhone |
|---|---|
| ABC | 3456 |
| DEF | 4567 |

L2

| LName | LPhone |
|---|---|
| ABC | 3456 |
| GHI | 5678 |

L1 - L2=

| LName | LPhone |
|---|---|
| DEF | 4567 |

## Intersection

- Intersection is a binary operation that produces a table that contains the rows common to two tables that have the same attributes.

L1

| LName | LPhone |
|---|---|
| ABC | 3456 |
| DEF | 4567 |

L2

| LName | LPhone |
|---|---|
| ABC | 3456 |
| GHI | 5678 |

L1 ∩ L2=

| LName | LPhone |
|---|---|
| ABC | 3456 |

## Cartesian product

- Cartesian product is a binary operation that produces the table formed by concatenating every tuple of the first with every tuple of the second.
- If the first table has *h* rows and *j* columns, and the second has *m* rows and *n* columns, the Cartesian product has *h*x*m* rows and *j*+*n* columns.

## Cartesian product example (1)

S1

| SReg | SName | LName |
|---|---|---|
| 72978 | Rawsthorne | ABC |
| 25477 | Bliss | DEF |
| 40935 | Howells | ABC |
| 78883 | Rutter | DEF |

L1

| LName | LPhone |
|---|---|
| ABC | 3456 |
| DEF | 4567 |

## Cartesian product example (2)

S1 x L1 =

| SReg | SName | S1. LName | L1. LName | LPhone |
|---|---|---|---|---|
| 72978 | Rawsthorne | ABC | ABC | 3456 |
| 25477 | Bliss | DEF | ABC | 3456 |
| 40935 | Howells | ABC | ABC | 3456 |
| 78883 | Rutter | DEF | ABC | 3456 |
| 72978 | Rawsthorne | ABC | DEF | 4567 |
| 25477 | Bliss | DEF | DEF | 4567 |
| 40935 | Howells | ABC | DEF | 4567 |
| 78883 | Rutter | DEF | DEF | 4567 |

## Theta join and equijoin

- Applies a selection predicate to a Cartesian product expression.
- The predicate must be a comparison of the values of two attributes, one from each table.
- Comparison means >, ≥, <, ≤, =, ≠
- An equijoin is a theta join with an equality.

$R \bowtie_{predicate} S = \sigma_{predicate} (R \times S)$

## Natural join

- The two tables have at least one attribute in common.
- The equijoin predicate requires each common attribute to have the same value in both tables.
- Only one copy of each of the common attributes is retained.

## Natural join example (1)

S2

| SReg | SName |
|---|---|
| 72978 | Rawsthorne |
| 25477 | Bliss |

E2

| SReg | UCode |
|---|---|
| 72978 | AB12 |
| 72978 | BC23 |
| 40935 | BC23 |

S2 ⋈ E2 =

| SReg | SName | UCode |
|---|---|---|
| 72978 | Rawsthorne | AB12 |
| 72978 | Rawsthorne | BC23 |

## Natural join example (2)

Book

| BNum | BTitle | WNum |
|---|---|---|
| 1 | War and Peace | 44 |
| 2 | War and Peace | |
| 4 | Recursive Analysis | 55 |
| 8 | The Crimson Field | |

Borrower

| WNum | WName |
|---|---|
| 33 | Gainsborough |
| 44 | Reynolds |
| 55 | Bacon |
| 66 | Sutherland |
| 77 | Winterhalter |

Book ⋈ Borrower =

| BNum | BTitle | WNum | WName |
|---|---|---|---|
| 1 | War and Peace | 44 | Reynolds |
| 4 | Recursive Analysis | 55 | Bacon |

## Semijoin

- A semijoin is a projection of the attributes of the first table from one of the foregoing kinds of join.

$R \triangleright_{predicate} S = \pi_A ( R \bowtie_{predicate} S )$

where A is the set of attributes of R.

## Semijoin example

Book

| BNum | BTitle | WNum |
|---|---|---|
| 1 | War and Peace | 44 |
| 2 | War and Peace | |
| 4 | Recursive Analysis | 55 |
| 8 | The Crimson Field | |

Borrower

| WNum | WName |
|---|---|
| 33 | Gainsborough |
| 44 | Reynolds |
| 55 | Bacon |
| 66 | Sutherland |
| 77 | Winterhalter |

Book $\triangleright_{Book.WNum = Borrower.WNum}$ Borrower =

| BNum | BTitle | WNum |
|---|---|---|
| 1 | War and Peace | 44 |
| 4 | Recursive Analysis | 55 |

## Outer join

- A left (right) outer join contains all the rows in the first (second) table extended with the attributes of the second (first) table. If the row of the first (second) table matched a row of the second (first) table, that row is extended with the values of the attributes in the corresponding row of the second (first) table. Otherwise it is extended with null values.
- There is a full outer join.

## Left outer join example

Book $\bowtie$ Borrower =

| BNum | BTitle | WNum | WName |
|---|---|---|---|
| 1 | War and Peace | 44 | Reynolds |
| 2 | War and Peace | | |
| 4 | Recursive Analysis | 55 | Bacon |
| 8 | The Crimson Field | | |

## Right outer join example

Book $\bowtie$ Borrower =

| WNum | WName | BNum | BTitle |
|---|---|---|---|
| 33 | Gainsborough | | |
| 44 | Reynolds | 1 | War and Peace |
| 55 | Bacon | 4 | Recursive analysis |
| 66 | Sutherland | | |
| 77 | Winterhalter | | |

## Division

- Let A be the attributes of table R.
- Let B be the attributes of table S.
- Suppose that B is a subset of A.
- Let C = A - B.

$R \div S = \pi_C ( R ) - \pi_C ( ( S \times \pi_C ( R ) ) - R )$

## Relational calculi

- Tuple relational calculus: A predicate calculus whose variables range over tuples in one or more relations
- Domain relational calculus: A predicate calculus whose variables range over values in attribute domains.

## The tuple relational calculus (1)

- Mathematical expressions are used to denote sets of tuples of interest.
- { tuples descriptor | predicate } denotes the set of tuples satisfying the predicate.
- { b | Book ( b ) } denotes the set of tuples in the relation Book.
- { b.BNum, b.BTitle | Book (b) } denotes the set of tuples of book number and book title in the relation Book.

## The tuple relational calculus (2)

- { s.SName,s.SHall | Student (s) /\ s.LName = "ABC" } denotes the set of tuples of student names and halls for students whose tutor is ABC.
- { b | Book ( b ) /\ ( ∃w )( Borrower ( w ) /\ b.WNum = w.WNum ) } denotes the set of books borrowed by someone in the Borrower table.

## Summary

- Relational algebra is a set of operations on tuples that allow us to write explicit specifications of the information we want to get from relations (tables).
- Relational calculus is a set of mathematical expressions that allow us to write implicit specifications of the information we want to get from relations.

## Lectures 6 and 7: SQL

- Origins of SQL
- SELECT statement for projection
- SELECT statement for selection
- SELECT statement: more elaborate cases
- INSERT statement for adding rows
- UPDATE statement for changing values of attributes in rows
- DELETE statement for deleting rows

## Overview

- Language invented in IBM in the 1970s to support relational databases, and now an international standard.
- Three aspects:
  - A Data Definition Language
  - A Data Manipulation Language
    - SELECT
    - UPDATE
    - INSERT
    - DELETE
  - A Data Control Language

## SELECT syntax

```
SELECT [ DISTINCT | ALL ]
  { * | column_expression [ AS newname ] [ , ... ] }
  FROM table_name [ alias ] [ , ... ]
  [ WHERE condition ]
  [ GROUP BY column_list ]
  [ HAVING condition ]
  [ ORDER BY column_list ]
  ;
```

## The Book table

| BNum | BTitle | BDewey | WNum | BAuthor |
|---|---|---|---|---|
| 1 | War and Peace | | 44 | Tolstoy |
| 2 | War and Peace | | | Tolstoy |
| 3 | Introduction to Metamathematics | 510.1 | | Kleene |
| 4 | Recursive Analysis | 511.35 | 55 | Goodstein |
| 5 | The Buildings of England: Dorset | 720.9423 | | Pevsner |
| 6 | Oxford Companion to Music | 780.3 | 66 | Scholes |
| 7 | The Bach Reader | 780.924 | 66 | David |
| 8 | The Crimson Field | | | Sutcliffe |

## The Borrower table

| WNum | WName |
|---|---|
| 33 | Gainsborough |
| 44 | Reynolds |
| 55 | Bacon |
| 66 | Sutherland |
| 77 | Winterhalter |

## SELECT * and SELECT DISTINCT

SELECT * from Book ;

denotes the whole table Book.

SELECT DISTINCT BTitle FROM Book ;

denotes a table with one column that lists the titles, without repetitions, of all the books in the library.

## Result of SELECT DISTINCT

| BTitle |
|---|
| War and Peace |
| Introduction to Metamathematics |
| Recursive Analysis |
| The Buildings of England: Dorset |
| Oxford Companion to Music |
| The Bach Reader |
| The Crimson Field |

## COUNT, GROUP and ORDER BY (1)

SELECT BTitle, COUNT(*) AS BookCount
FROM Book GROUP BY BTitle ;

denotes a table with two columns, BTitle and BookCount, the first containing the titles, and the second the number of books with that title.

SELECT * FROM Book ORDER BY BDewey ;

denotes the Book table with the rows in ascending order of DDN, nulls first.

## COUNT, GROUP and ORDER BY (2)

| BTitle | BookCount |
|---|---|
| War and Peace | 2 |
| Introduction to Metamathematics | 1 |
| Recursive Analysis | 1 |
| The Buildings of England: Dorset | 1 |
| Oxford Companion to Music | 1 |
| The Bach Reader | 1 |
| The Crimson Field | 1 |

## COUNT, GROUP and ORDER BY (3)

| BNum | BTitle | BDewey | WNum | BAuthor |
|---|---|---|---|---|
| 1 | War and Peace | | 44 | Tolstoy |
| 2 | War and Peace | | | Tolstoy |
| 8 | The Crimson Field | | | Sutcliffe |
| 3 | Introduction to Metamathematics | 510.1 | | Kleene |
| 4 | Recursive Analysis | 511.35 | 55 | Goodstein |
| 5 | The Buildings of England: Dorset | 720.9423 | | Pevsner |
| 6 | Oxford Companion to Music | 780.3 | 66 | Scholes |
| 7 | The Bach Reader | 780.924 | 66 | David |

## WHERE (1)

```
SELECT * FROM Book
WHERE BDewey>"5" AND BDewey<"6" ;
```

denotes a table of books with DDNs in the 500s.

```
SELECT * FROM Book
WHERE BDewey IS NULL ;
```

denotes a table of books with null DDNs.

## WHERE (2)

```
SELECT * FROM Book
WHERE BDewey LIKE "5*" ;
```

denotes a table of books with DDNs in the 500s. (The standard wild-card is "%".)

| BNum | BTitle | BDewey | WNum | BAuthor |
|---|---|---|---|---|
| 3 | Introduction to Metamathematics | 510.1 | | Kleene |
| 4 | Recursive Analysis | 511.35 | 55 | Goodstein |

## Cartesian product

```
SELECT Book.*, Borrower.*
FROM Book, Borrower ;
```

denotes the Cartesian product of Book and Borrower.

## Equijoin (1)

```
SELECT BTitle , WName
FROM Book, Borrower
WHERE  Book.WNum = Borrower.WNum ;
```

denotes a projection of BTitle and WName from the equijoin of Book and Borrower over WNum.

## Equijoin (2)

| BTitle | WName |
|---|---|
| War and Peace | Reynolds |
| Recursive Analysis | Bacon |
| Oxford Companion to Music | Sutherland |
| The Bach Reader | Sutherland |

## Equijoin with COUNT, etc. (1)

```
SELECT WName ,
COUNT (BNum) AS Borrowed
FROM Book, Borrower
WHERE Book.WNum = Borrower.WNum
GROUP BY WName ;
```

denotes a table that lists the borrowers and the number of books that they have borrowed.

## Equijoin with COUNT, etc. (2)

| WName | Borrowed |
|---|---|
| Bacon | 1 |
| Reynolds | 1 |
| Sutherland | 2 |

## SELECT with nested query (1)

```
SELECT BTitle, BAuthor
FROM Book
WHERE WNum IN
  ( SELECT WNum
    FROM Borrower
    WHERE WNum > "40" AND WNum < "60")
```

denotes a table that lists the titles and authors of books whose borowers are in the stated range.

## SELECT with nested query (2)

| **BTitle** | **BAuthor** |
|---|---|
| War and Peace | Tolstoy |
| Recursive Analysis | Goodstein |

## SELECT for an outer join (1)

```
SELECT BTitle, WName
FROM Book LEFT JOIN Borrower ON
Book.WNum=Borrower.WNum;
```

denotes a table that contains the titles of all the books, and the names of the borrowers of the books that are on loan.

## SELECT for an outer join (2)

| BTitle | WName |
|---|---|
| War and Peace | Reynolds |
| War and Peace | |
| The Crimson Field | |
| Introduction to Metamathematics | |
| Recursive Analysis | Bacon |
| The Buildings of England: Dorset | |
| Oxford Companion to Music | Sutherland |
| The Bach Reader | Sutherland |

## SELECT with UNION

( SELECT BTitle, BAuthor
FROM Book WHERE BDewey IS NULL )
UNION (SELECT BTitle, BAuthor
FROM Book WHERE WNum IS NULL);

denotes the titles and authors of the books that are fiction (no DDN) or not on loan.

## SELECT with INTERSECTION

( SELECT BTitle, BAuthor
FROM Book WHERE BDewey IS NULL )
INTERSECTION
(SELECT BTitle, BAuthor
FROM Book WHERE WNum IS NULL);

denotes the titles and authors of the fiction books (no DDN) that are not on loan.

## SELECT with EXCEPT

( SELECT BTitle, BAuthor
FROM Book WHERE BDewey IS NULL )
EXCEPT
(SELECT BTitle, BAuthor
FROM Book WHERE WNum IS NULL);

denotes the titles and authors of the fiction books (no DDN) that are on loan.

## INSERT syntax

INSERT INTO table_name [ ( column_list ) ]
{ VALUES ( data_value_list ) | select_expression
} ;

## Adding a new row to a table

INSERT INTO Book
VALUES ('23', 'The Sorrows of Satan', NULL, NULL, 'Corelli') ;

Adds a new row to the Book table with the stated values for the attributes.

Beware of key violations and data conversion errors.

## UPDATE syntax

UPDATE table_name
SET column_name = data_value , ...
[ WHERE search_condition ] ;

## Updating a row

UPDATE Book SET WNum = '33'
WHERE BNum='23';

Records the fact that borrower 33 has borrowed book 23.

## DELETE syntax

DELETE FROM table_name
WHERE search_condition ;

## Deleting rows

DELETE FROM Book WHERE BNum='23' ;

Deletes all the rows in which BNum is 23. (In our sample there is only one.)

DELETE FROM Book ;

Deletes all the books.

## Summary

- The SELECT statement can be used for making queries, simple or complex, on one or more tables of a database.
- The INSERT, UPDATE, and DELETE statements are used for managing the contents of tables in a database.

## Lecture 8: Physical representations

- Physical design activity
- Tables and files
- Access methods and file organisations
- Role of the DBMS
- Indexing

## Physical design activity

- Translates the global logical data model into a physical model using the DBMS's facilities.
  - Analyse the transactions.
  - Choose the file oganisations.
  - Choose secondary indexes.
  - Consider introducing data redundancy.
  - Calculate disk space requirements.
  - Design security mechanisms.
  - Monitor and tune the database.

## Tables and files

Table: Lecturer

| LName | LPhone | LDept |
|---|---|---|
| ABC | 3456 | CS |

File: Lecturer

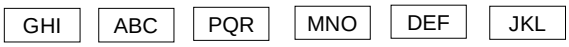
| | | | |
|---|---|---|---|
| 0 | ABC | 3456 | CS |
| | DEF | 4567 | AG |
| | GHI | 5678 | CS |
| 1 | JKL | 7890 | MU |
| | MNO | 8901 | CS |
| | PQR | 9012 | AG |

## Role of the access method

- Software provided with the operating system
- Mediates between the programmer's logical view of data and the operating systems view of the I/O hardware (close to the disk's architecture).
- The implementer of the DBMS works with file organisations provided by the operating system's access methods.

## Heap files

- Records are stored in the order in which they are added.
- New records are added at the end.
- Deletions (if any) by marking the records.
- Retrieval needs a linear search.




## Ordered files

- Records stored in order of (primary) key.
- Access method might leave gaps for additions.
- New records added in place.
- Deletion by marking; reuse possible.
- Retrieval might use binary search.


## Hash files (1)

- The file space is organised into buckets.
- To add a record the key is hashed into a bucket number.
- To retrieve a record by key, hash the key, then search the bucket.
- To delete, mark the record; space can be reused.
- Two keys with the same hash value are synonyms.
- Overflow areas needed.

## Hash files (2)

| Bucket 0 | Bucket 1 | Bucket 2 | Bucket 3 | Overflow |
|---|---|---|---|---|
| | JKL | ABC | DEF | |
| | PQR | GHI | | |
| | MNO | | | |

## Role of the DBMS

- Allows the database designer to exploit access method facilities, or does not.
- There are some standard SQL DDL commands for setting up indexes, but they do not go far.
- Allows the database designer to make some configuration decisions that might improve performance.

## Sparse indexes

| Key | Track | Key | Track | Key | Track |
|---|---|---|---|---|---|
| GHI | 0 | PQR | 1 | | |

| | | | | |
|---|---|---|---|---|
| Track 0 | ABC | DEF | GHI | |
| Track 1 | JKL | MNO | PQR | |
| Track 2 | | | | |

## Secondary indexes

| Key | T | R |
|---|---|---|
| AG | 0 | 1 |

| Key | T | R |
|---|---|---|
| AG | 1 | 2 |

| Key | T | R |
|---|---|---|
| CS | 0 | 0 |

| Key | T | R |
|---|---|---|
| CS | 0 | 2 |

| Key | T | R |
|---|---|---|
| CS | 1 | 2 |

| Key | T | R |
|---|---|---|
| MU | 1 | 0 |

| | | | |
|---|---|---|---|
| T0 | ABC CS | DEF AG | GHI CS |
| T1 | JKL MU | MNO CS | PQR AG |

## Multilevel indexes

- An index can be regarded as a file of records.
- A sparse index of an index is a second-level index.
- The creation of higher-level indexes can be continued as far as is practicable.

## Summary

- Physical design maps relational model to files.
- Each table corresponds to a file of records; the attributes of the table are the fields of the records.
- Access methods support various file organisations.
- The DBMS manages the data using the access methods.
- Indexing

## Lecture 9: Transactions

- What is a transaction?
- Transaction program
- Resource managers and locking
- Transaction manager
- Transaction processing monitor
- Two-phase commit processing

## What is a transaction?

- A small piece of work done in the database to satisfy a user request
  - Cash withdrawal from an ATM
  - Add details of a new student
- Types of transaction
  - Inquiry: changes nothing
  - Simple update: changes tables in a single database, one DBMS
  - Complex update: changes tables in several databases, several DBMSs

## Transaction specification

- Inputs required: domains, preconditions
- Outputs produced
- Effect on contents of the tables in the relational models.

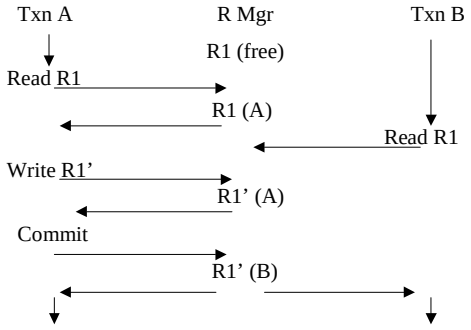
## Transaction program

- A repository of business logic
  - Analyse the transaction request.
  - Create the DML (SQL) to effect the transaction, and get the DBMS (or DBMSs) to do it.
  - Send the response.

## Resource manger (DBMS)

- Know who is asking for changes.
- Advise transaction manager about interest in transaction.
- Manage locks on resources (tables, rows).
- Make provisional changes to resources.
- Be prepared to:
  - Commit changes
  - Rollback changes

## Locking



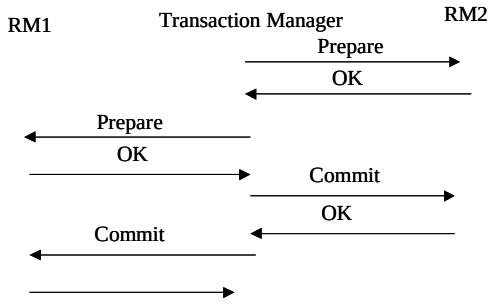
## Transaction manager

- Keep account of transactions that are active.
- Keep account of which resource managers have interests in which transactions.
- Manage the committing (at normal end of transaction) or rolling back (at abnormal end of transaction) of the provisional changes made by a transaction in order to maintain data integrity.
- For multiple resource managers, use the two-phase commit protocol.

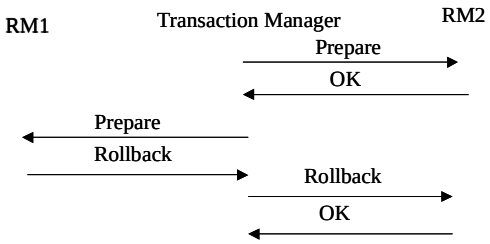
## Transaction processing monitor

- Provides an environment for multiple transaction programs:
  - Access to resource mangers (files, databases, devices, …)
  - Access to private resources (storage, temporary storage queues, …)
- Provides transaction management facilities
- Provides operations management facilities (distributed transaction processing, multiple platforms, …)

## Two-phase commit (1)




## Two-phase commit (2)




## Summary

- A transaction is a small piece of work that might affect many resources (databases).
- A resource manager keep control of who is doing what to its resources, and manages resource locks.
- A transaction manager keeps control of the relationship between transactions and resource managers.
- A transaction processing monitor manages the transaction program environment and two-phase commit processing.