# Databases

## The Relational Data Model

Dr Konrad Dabrowski
konrad.dabrowski@durham.ac.uk

Online Office Hour:
Mondays 13:30–14:30
See Duo for the Zoom link

# Relational Data Model

- Data Definition Language (DDL) is too low-level
  - not easily understandable by most of users

⟹ We need a Data Model:

  - a collection of intuitive concepts describing *data*, their *relationships* and *constraints*

- Relational Data Model:
  – relations between data ⟶ stored in tables
  – based on the concept of mathematical relations
  – the most widely used Data Model (for structured data)

- Intuitively:
  – in our data, every entity (e.g. customer) combines (or "relates") various "attributes" together (e.g. *name*, *id*, *address*, etc.)

# Relational Data Model

- The schema of a relation:
  - the description of a particular collection of data in the model
- Let $A_1, A_2, \ldots, A_n$ be a set of attributes that can be "related"
  - i.e. there exists an entity with some values for these attributes
- Then $R(A_1, A_2, \ldots, A_n)$ is the schema of the relation $R$

  e.g. if $R$ denotes customers, the schema of $R$ could be:

  *R(name, id, address, town, date-of-last-purchase)*

  - *this means: every customer has values for exactly these attributes*
- In a relation schema:
  - the ordering of the attributes does not matter!
    (in contrast to mathematical relations)

# Relational Data Model

- A database has many entities
  - each with its own attributes
- This information is decomposed into smaller pieces
  - every relation stores only one piece of the information
- Example:
  - the relation *account* stores info for the accounts
  - *depositor* stores info about which customer has this account
  - *customer* stores info about all customers
  - the whole info of the DB could be stored in one relation, e.g.

    *R(account, balance, cust-name, …)*

  but then:
  - data duplication (e.g. two customers have the same account)
  - we need many null values (e.g. insert customer with no account)

# Relational Data Model

Customer

| Cust-id | Cust-Name | Cust-address | Cust-city |
|---|---|---|---|
| 192-83-7465 | Johnson | Bakerstr. 1 | Durham |
| 019-28-3746 | Dunn | Halkinstr. 23 | Newcastle |
| 677-89-9011 | Richardson | Pensburystr. 12 | London |
| 182-73-6091 | Edwards | Gardenstr. 25 | Manchester |

customer with no accounts

Account

| Account number | Balance |
|---|---|
| A-101 | 500 |
| A-215 | 700 |
| A-102 | 400 |
| A-305 | 350 |
| A-222 | 700 |

customer with 3 accounts

| Cust-id | Account number |
|---|---|
| 192-83-7465 | A-101 |
| 019-28-3746 | A-215 |
| 677-89-9011 | A-102 |
| 019-28-3746 | A-305 |
| 019-28-3746 | A-222 |

Depositor
(i.e. customer with an account)

# Relational Model Terminology

- A relation is a table (with rows and columns)
  - this only refers to the logical structure of the DB, not the physical structure
- An attribute is a named column of a relation
  - every attribute has a unique name
- The domain of an attribute is the set of allowable values
- A tuple is a row of a relation
  - every tuple has a concrete value for every attribute!
- A cell of a relation is the intersection of a row and a column
- The degree of a relation is the number of attributes
  - i.e. every row stores as many values as the degree of the relation
- The cardinality of a relation is the number of tuples

# Relational Model Terminology

- A relation is normalized if it is “appropriately structured”, e.g.
  - every cell has exactly one value (not more / less!)
  - no repetitions of two identical rows
- A Relational Database is a collection of normalized relations
  - each of them with distinct relation names
- An alternative terminology for the Relational Model:

| Formal terms | Alternative 1 | Alternative 2 |
|---|---|---|
| Relation | Table | File |
| Tuple | Row | Record |
| Attribute | Column | Field |

# Instances of Branch and Staff Relations

Attributes

Cell

**Branch**

| branchNo | street | city | postcode |
|---|---|---|---|
| B005 | 22 Deer Rd | London | SW1 4EH |
| B007 | 16 Argyll St | Aberdeen | AB2 3SU |
| B003 | 163 Main St | Glasgow | G11 9QX |
| B004 | 32 Manse Rd | Bristol | BS99 1NZ |
| B002 | 56 Clover Dr | London | NW10 6EU |

Relation

Cardinality

Degree

Tuple

**Staff**

| staffNo | fName | lName | position | sex | DOB | salary | branchNo |
|---|---|---|---|---|---|---|---|
| SL21 | John | White | Manager | M | 1-Oct-45 | 30000 | B005 |
| SG37 | Ann | Beech | Assistant | F | 10-Nov-60 | 12000 | B003 |
| SG14 | David | Ford | Supervisor | M | 24-Mar-58 | 18000 | B003 |
| SA9 | Mary | Howe | Assistant | F | 19-Feb-70 | 9000 | B007 |
| SG5 | Susan | Brand | Manager | F | 3-Jun-40 | 24000 | B003 |
| SL41 | Julie | Lee | Assistant | F | 13-Jun-65 | 9000 | B005 |

Relation

# Instances of Branch and Staff Relations

- An illustration of the domains of the attributes:

| Attribute | Domain Name | Meaning | Domain Definition |
|---|---|---|---|
| branchNo | BranchNumbers | The set of all possible branch numbers | character: size 4, range B001–B999 |
| street | StreetNames | The set of all street names in Britain | character: size 25 |
| city | CityNames | The set of all city names in Britain | character: size 15 |
| postcode | Postcodes | The set of all postcodes in Britain | character: size 8 |
| sex | Sex | The sex of a person | character: size 1, value M or F |
| DOB | DatesOfBirth | Possible values of staff birth dates | date, range from 1-Jan-20, format dd-mmm-yy |
| salary | Salaries | Possible values of staff salaries | monetary: 7 digits, range 6000.00–40000.00 |

- NULL value:
  - a special case of a cell entry
  - it represents an attribute value that is:
    - either currently unknown
    - or not applicable
  - not the same as the value “0”!
  - may / may not belong to the domain of the attribute

# Instances of Branch and Staff Relations

Another example for the domains:

- the schema of the relation $R$ of the customers is:

  *R(name, address, town)*

- where:
  - the domain of “name” is {Johnson, Dunn, Edwards, Richardson}
  - the domain of “address” is {Bakerstr, Halkinstr, Gardenstr}
  - the domain of “town” is {London, Durham, Glasgow}

  which one is a correct relation?

| name | address | town |
|---|---|---|
| Johnson | Bakerstr | London |
| Edwards | Halkinstr | Durham |
| Richardson | Gardenstr | London |

| name | address | town |
|---|---|---|
| Johnson | Bakerstr | London |
| Edwards | Halkinstr | Durham |
| Richardson | Gardenstr | Leeds |

# Properties of Relations

- The relation name is distinct from all other relation names in the relational schema
- Each attribute (within a relation) has a distinct name
  - possibly two attributes of different relations may have the same name
  - e.g. “name”, “id” etc.
- Values of an attribute are all from the same domain
- Each cell of relation contains exactly one atomic (single) value
- Each tuple is distinct among the tuples of the relation
  - there are no duplicate tuples
- The ordering of attributes has no significance
  - unlike mathematical relations
  - e.g. Cartesian product of two sets: ordered pairs of elements
- The ordering of tuples has no significance
  - just unordered rows

# Structuring concept: Keys

- How do we uniquely identify a tuple in a (normalized) table?
  - attribute names are unique within a table (‘id’, ‘sex’, …)
  - but two tuples may share attribute values (both have sex ‘M’)

- Every table must have some attributes, such that:
  - their value uniquely determines a tuple of the table
  - these attributes are the primary key of the table

**CD table**

| *CD #* | *Song* | *Artist* | *Position* | *Month* | *Year* |
|---|---|---|---|---|---|
| 0001 | Mandolin Wind | Rod Stewart | 1 | 02 | 1974 |
| 0002 | Gallows Pole | Led Zeppelin | 2 | 04 | 1985 |
| 0003 | Comfortably Numb | Pink Floyd | 5 | 04 | 1989 |
| 0004 | Paint it Black | Rolling Stones | 6 | 09 | 1967 |

# Structuring concept: Keys

- Candidate key: (of a relation)
  - a minimal (not minimum!) set of attributes ("keys") whose values uniquely identify the tuples
- Primary key:
  - The candidate key selected to identify rows uniquely within the table
- Alternate key:
  - Those candidate key(s) not selected as primary key
- Simple key:
  - The key consists of only one attribute
- Composite key:
  - The key consists of several attributes

# Primary keys

- An example:

**Employee**

| Initial | Surname | Job Title | Car No. | Department |
|---|---|---|---|---|
| N | Cook | Salesman | 5 | Sales |
| A | Randell | Programmer | null | Computer |
| M | Cook | Consultant | 12 | Insurance |

- Is it possible to use *Surname* as the primary key?

Maybe sometimes, but not in this case ...

- What about using *Initial*? Even worse!
  - In this example, initials are all different, so *Initial* can be a primary key
  - However: if *Mark Washington* gets a job at this company, the DBMS will not allow his name to be added

# Primary keys

- An example:

**Employee**

| Initial | Surname | Job Title | Car No. | Department |
|---|---|---|---|---|
| N | Cook | Salesman | 5 | Sales |
| A | Randell | Programmer | null | Computer |
| M | Cook | Consultant | 12 | Insurance |

- Could we use *Initial* and *Surname* as a (composite) primary key?
  - Much better, but still…
  - *What happens if another Nick Cook appears?*

- Which primary key should be used?
  - The best choice is the *Employee No.* which is unique for every person!
  - Or alternatively: to add another new “identification” attribute

# Composite keys

- We can create composite keys:
  - by combining two (or more) attributes
- In the table below *Song* and *Artist* are a composite key
  - this key uniquely identifies a row

| **<u>Song</u>** | **<u>Artist</u>** | **Position** | **Month** | **Year** |
|---|---|---|---|---|
| Tubular Bells | Mike Oldfield | 1 | 02 | 1974 |
| Gallows Pole | Led Zeppelin | 2 | 04 | 1985 |
| Comfortably Numb | Pink Floyd | 5 | 04 | 1989 |
| Paint it Black | Rolling Stones | 6 | 09 | 1967 |

- Why use composite keys?
  - Providing that an artist never reissues the same song title, *Song* and *Artist* are a valid primary key
  - *Song* on its own would run into difficulties (other artists may use this title)

# Foreign keys

- To keep track of who borrowed a CD, we need to store:
  - the details of our customers
  - the details of all the transactions

**Customer table**

| ***Cust #*** | ***Name*** | ***Address*** |
|---|---|---|
| 001 | Konrad Dabrowski | Stockton Road, Durham |
| 002 | Janet Lavery | Stockton Road, Durham |
| 003 | Stephan Jamieson | Old Elvet, Durham |

**Transaction table**

| ***Trans#*** | ***CD#*** | ***Cust #*** | ***Hire date*** | ***Duration*** |
|---|---|---|---|---|
| 00001 | 0001 | 001 | 12/9/06 | 1 |
| 00002 | 0001 | 002 | 14/9/06 | 3 |
| 00003 | 0004 | 002 | 15/9/06 | 1 |

- How do we “match” the customers in the two tables?
  - using foreign keys!

# Foreign keys

- Foreign key:
  - An attribute in one table $A$ whose values must:
    - either match the primary key of another table $B$ (then $A$ references $B$)
    - or be NULL (e.g. staff has not been yet assigned to a branch)

table $B$:

Attributes

**Branch**

Relation

| branchNo | street | city | postcode |
|---|---|---|---|
| B005 | 22 Deer Rd | London | SW1 4EH |
| B007 | 16 Argyll St | Aberdeen | AB2 3SU |
| B003 | 163 Main St | Glasgow | G11 9QX |
| B004 | 32 Manse Rd | Bristol | BS99 1NZ |
| B002 | 56 Clover Dr | London | NW10 6EU |

Cardinality

Degree

table $A$:

**Staff**

Relation

| staffNo | fName | lName | position | sex | DOB | salary | branchNo |
|---|---|---|---|---|---|---|---|
| SL21 | John | White | Manager | M | 1-Oct-45 | 30000 | B005 |
| SG37 | Ann | Beech | Assistant | F | 10-Nov-60 | 12000 | B003 |
| SG14 | David | Ford | Supervisor | M | 24-Mar-58 | 18000 | B003 |
| SA9 | Mary | Howe | Assistant | F | 19-Feb-70 | 9000 | B007 |
| SG5 | Susan | Brand | Manager | F | 3-Jun-40 | 24000 | B003 |
| SL41 | Julie | Lee | Assistant | F | 13-Jun-65 | 9000 | B005 |

# Integrity Constraints

- So far we have seen:
  - domain constraints for the attributes
- Entity integrity:
  - every attribute of a primary key cannot be null

    (otherwise we do not need all attributes of the primary key to identify the tuples)
- Purpose of entity integrity:
  - guarantees that each entity has unique identifier
  - ensures that foreign key values can reference primary key values
- Example:
  - no invoice can have a duplicate number, nor can it be null
  - all invoices uniquely identified by the invoice number

# Integrity Constraints

- Referential integrity:
  - a foreign key either matches the primary key in the table it refers to
  - or it is null
- Purpose of referential integrity:
  - any reference between tables is valid (or it has not been set yet)
  - prevents deleting a row in a table $B$, if the primary key of $B$ has a matching foreign key in another table $A$
- Example:
  - a customer will be always assigned to a valid sales representative
  - unless (s)he is not yet assigned to any representative

# Summary: Characteristics of a relational table

- A relation is represented by a two-dimensional table
- Each row (tuple) signifies an entity occurrence
- No two rows can be identical (each row of the table is unique)
- Each column represents an attribute and has a distinct name
- The intersection of a row and a column has a single value (atomic)
- All values in a column must be of the same type (e.g. integers)
  - they have the same domain
- One (or more) attributes uniquely identify each row (primary key)
  - primary keys are not allowed a NULL value
- Two tables can be dependent
  - the primary key is the foreign key of another table
- The ordering of rows and columns does NOT matter

## Views

- So far all relations (tables) we have seen:
  - base relations
  - its tuples are physically stored in the database
- A different type of a relation: a view
  - a virtual relation
  - it does not exist physically in the database
- The content of a view:
  - is derived from one (or more) base relations
  - is computed upon request by a user, at the time of request
  - changes when the underlying base relations change
- Main use:
  - show customized information to every user (e.g. show "loan number" but not "amount borrowed")
  - compute dynamic quantities (e.g. "age" from "date-of-birth")

# Alternatives to the Relational Data Model

Other models before the development of the Relational Data Model:

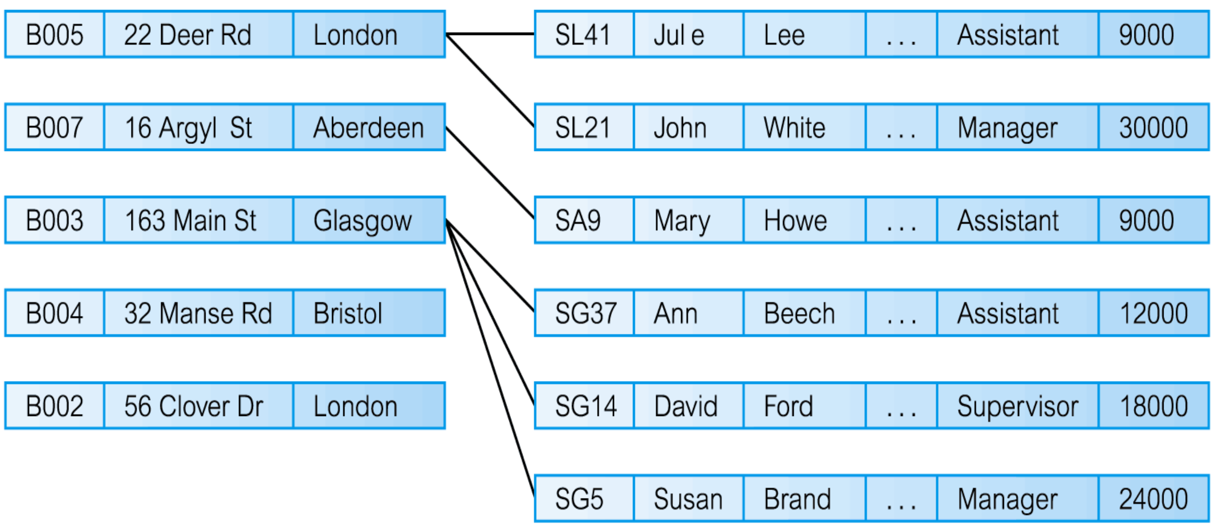
1. Network Data Model
   - records (tuples) appear as nodes
   - relationships (foreign keys) appear as edges

**Branch**

| branchNo | street | city | postCode |
|---|---|---|---|
| B005 | 22 Deer Rd | London | SW1 4EH |
| B007 | 16 Argyll St | Aberdeen | AB2 3SU |
| B003 | 163 Main St | Glasgow | G11 9QX |
| B004 | 32 Manse Rd | Bristol | BS99 1NZ |
| B002 | 56 Clover Dr | London | NW10 6EU |

**Staff**

| staffNo | fName | lName | position | sex | DOB | salary | branchNo |
|---|---|---|---|---|---|---|---|
| SL21 | John | White | Manager | M | 1-Oct-45 | 30000 | B005 |
| SG37 | Ann | Beech | Assistant | F | 10-Nov-60 | 12000 | B003 |
| SG14 | David | Ford | Supervisor | M | 24-Mar-58 | 18000 | B003 |
| SA9 | Mary | Howe | Assistant | F | 19-Feb-70 | 9000 | B007 |
| SG5 | Susan | Brand | Manager | F | 3-Jun-40 | 24000 | B003 |
| SL41 | Julie | Lee | Assistant | F | 13-Jun-65 | 9000 | B005 |



# Alternatives to the Relational Data Model

Other models before the development of the Relational Data Model:

## 2. Hierarchical Data Model

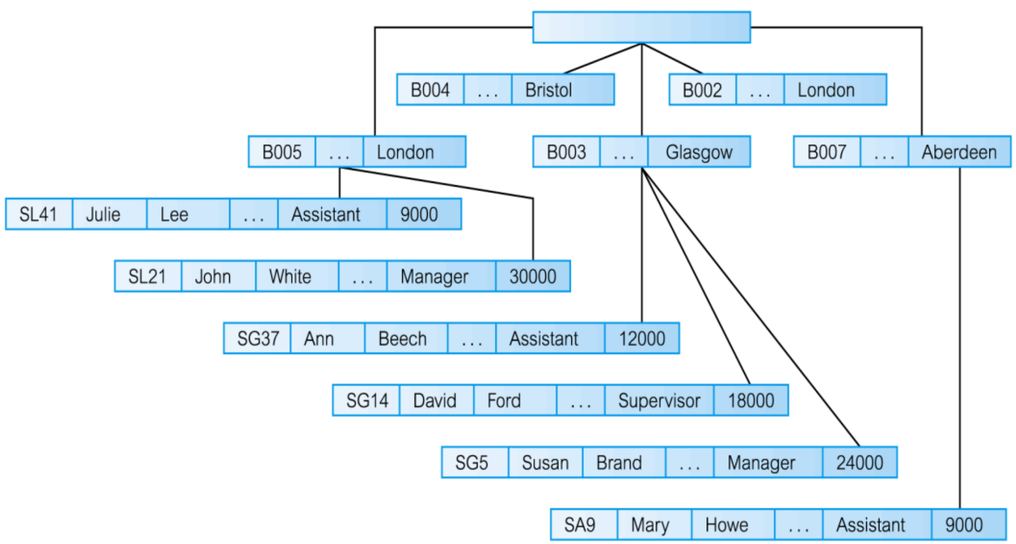
- special case of the Network Data Model, where the graph is a tree graph
- its structure mirrors parent-child relationship (one parent, many children)
- limitations of this model, e.g.
  - deleting a parent,
  - adding a record without a parent

Branch

| branchNo | street | city | postCode |
|---|---|---|---|
| B005 | 22 Deer Rd | London | SW1 4EH |
| B007 | 16 Argyll St | Aberdeen | AB2 3SU |
| B003 | 163 Main St | Glasgow | G11 9QX |
| B004 | 32 Manse Rd | Bristol | BS99 1NZ |
| B002 | 56 Clover Dr | London | NW10 6EU |

Staff

| staffNo | fName | lName | position | sex | DOB | salary | branchNo |
|---|---|---|---|---|---|---|---|
| SL21 | John | White | Manager | M | 1-Oct-45 | 30000 | B005 |
| SG37 | Ann | Beech | Assistant | F | 10-Nov-60 | 12000 | B003 |
| SG14 | David | Ford | Supervisor | M | 24-Mar-58 | 18000 | B003 |
| SA9 | Mary | Howe | Assistant | F | 19-Feb-70 | 9000 | B007 |
| SG5 | Susan | Brand | Manager | F | 3-Jun-40 | 24000 | B003 |
| SL41 | Julie | Lee | Assistant | F | 13-Jun-65 | 9000 | B005 |



# Additional Slides

## Mathematical Relations

In mathematics (background):

- for $n$ sets $D_1, D_2, \ldots, D_n$ , their *Cartesian product* is:
  $D_1 \times D_2 \times \ldots \times D_n = \{(d_1, d_2, \ldots, d_n) \mid d_1 \in D_1, \ldots, d_n \in D_n\}$,
  i.e. all the *ordered* $n$-tuples of these sets
- example: $D_1 = \{1, 2\},\ D_2 = \{a, b\},\ D_3 = \{x, y\}$, thus:

$$
\begin{aligned}
D_1 \times D_2 \times D_3 = \{&(1, a, x), (1, a, y),\\
&(1, b, x), (1, b, y),\\
&(2, a, x), (2, a, y),\\
&(2, b, x), (2, b, y)\}
\end{aligned}
$$

- in mathematics, a relation among the sets $D_1, D_2, \ldots, D_n$
  is any (ordered) subset $R \subseteq D_1 \times D_2 \times \ldots \times D_n$

# Mathematical Relations

**However,**

in the Databases terminology,

the order of attributes $(D_1, D_2, \ldots, D_n)$,

i.e. column headings, in a relation does not matter!

- in mathematics, a relation among the sets $D_1, D_2, \ldots, D_n$

  is any (ordered) subset $R \subseteq D_1 \times D_2 \times \ldots \times D_n$