Database models. RDM.

DEFINITIONS

Database, DBMS

- Data are characteristics or information, are collected through observation. In a more technical sense, data is a set of values of qualitative or quantitative variables about one or more objects. Data and information have different mining.
- A database is an organized collection of data, generally stored and accessed electronically from a computer system.
- The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data.
- A database model is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized and manipulated.
    - Hierarchical, Network, Flat, Relational, Object-oriented, Graph, Document-oriented…
- Data modeling is a process used to define and analyze data requirements needed to support the business processes within the scope of corresponding information systems in organizations.

wiki

Relational model

- The relational model (RM) for database management is an approach to managing data using a structure and language consistent with first-order predicate logic, first described in 1969 by English computer scientist Edgar F. Codd,[1][2] where all data is represented in terms of tuples, grouped into relations.
- A database organized in terms of the relational model is a relational database.
- Most relational databases use the SQL data definition and query language

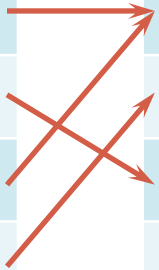
Relational model concepts

- **Tables** – In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.
- **Tuple** – A single row of a table, which contains a single record for that relation is called a tuple.
- **Relation instance** – A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.
- **Relation schema** – A relation schema describes the relation name (table name), attributes, and their names.
- **Relation key** – Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.
- **Attribute domain** – Every attribute has some pre-defined value scope, known as attribute domain.

Relational model concepts

id	emp_name	job	dept_id
7839	KING	PRESIDENT	1
7698	BLAKE	MANAGER	3
7782	CLARK	MANAGER	1
7566	JONES	MANAGER	2

id	dept_name	loc
1	ACCOUNTING	NEW YORK
2	RESEARCH	DALLAS
3	SALES	CHICAGO
4	OPERATIONS	BOSTON



Normalization

- Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization rules divides larger tables into smaller tables and links them using relationships.
- The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

Database Normal Forms

- 1NF (First Normal Form)
- 2NF (Second Normal Form)
- 3NF (Third Normal Form)
- BCNF (Boyce-Codd Normal Form)
- 4NF (Fourth Normal Form)
- 5NF (Fifth Normal Form)
- 6NF (Sixth Normal Form)

- **Note**. However, in most practical applications, normalization achieves its best in 3rd Normal Form.

1NF (First Normal Form)

- Each table cell should contain a single value.
- Each record needs to be unique.

…	BLAKE, MANAGER, RESEARCH	…

emp_name	job	dept_name
BLAKE	MANAGER	RESEARCH

Keys

- Candidate Key – **CK**
  - Primary key – **PK**
  - Alternate key – **AK** | Secondary Key – **SK** | Unique Key – **UK**
- Foreign key – **FK**

Primary key

- A primary is a single column value used to identify a database record uniquely.
  - A primary key cannot be NULL;
  - A primary key value must be unique;
  - The primary key values should rarely be changed;
  - The primary key must be given a value when a new record is inserted.

- Composite Key
  - A composite key is a primary key composed of multiple columns used to identify a record uniquely

2NF (Second Normal Form)

- Be in 1NF
- Single Column Primary Key

emp_name	job	dpet_name	loc
King	President	Accounting	New york
Blake	Manager	Research	Dallas
Clark	Manager	Sales	Chicago
Clark	Manager	Operations	Boston

Composite key

id	emp_name	job	dept_id
7839	King	President	**1**
7698	Blake	Manager	**3**
7782	Clark	Manager	**1**
7566	Clark	Manager	**2**

PK FK

id	dept_name	loc
1	Accounting	New york
2	Research	Dallas
3	Sales	Chicago
4	Operations	Boston

PK

Foreign key

- Foreign Key references the primary key of another Table
  - A foreign key can have a different name from its primary key.
  - It ensures rows in one table have corresponding rows in another.
  - Unlike the Primary key, they do not have to be unique. Most often they aren’t.
  - Foreign keys can be null even though primary keys can not.

3NF (Third Normal Form)

- Be in 2NF
- Has no transitive functional dependencies
    - A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change

id	emp_name	job	salutation	dept_id
7839	KING	PRESIDENT	Mr.	1
7698	BLAKE	MANAGER	Mr.	3
7782	CLARK	MANAGER	Mr.	1
7566	CLARK	MANAGER	Mr.	2

BCNF (Boyce-Codd Normal Form)

- Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one Candidate Key.

Relational Database Integrity

- In a database application, **maintaining data integrity** means ensuring that the data in the tables that the application manipulates conform to the appropriate business rules.
- Database can check that all the data in a table obeys a constraint faster than application code can do the equivalent checking.

  - Data integrity
  - Referential integrity

Data integrity

- Primary key constraint
- Unique key constraint
- Values constraints
- NOT NULL constraints

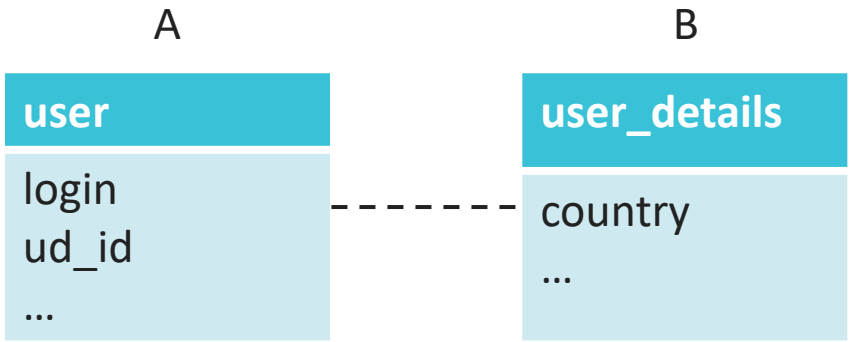
- Default Column Values

Relationships

- There are 3 types of relationships in relational database design. They are:
  - One-to-One (1:1)
  - One-to-Many (1:N)
  - Many-to-Many (N:M)

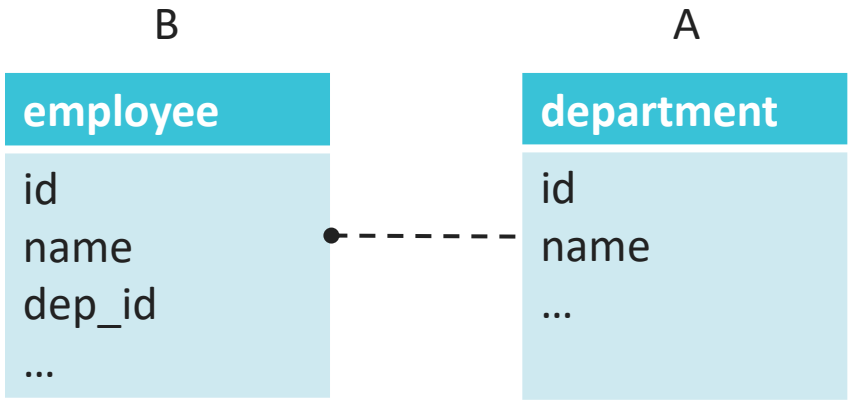
One-to-One (1:1) Relationship

- This is not a common relationship type, as the data stored in table B could just have easily been stored in table A.
- However, there are some valid reasons for using this relationship type.
- A one-to-one relationship can be used for security purposes, to divide a large table, and various other specific purposes.



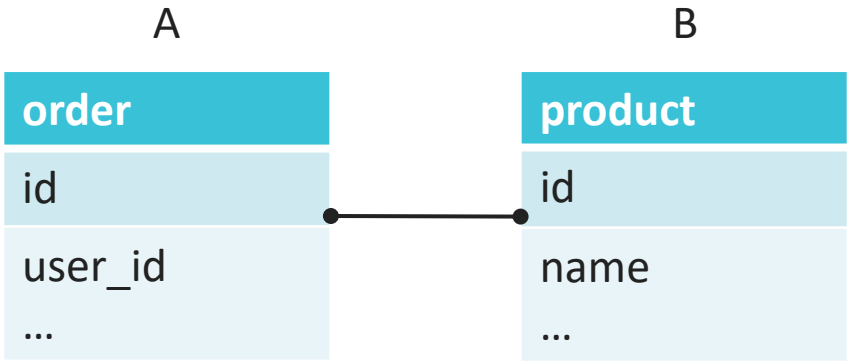
One-to-Many (1:N) Relationship

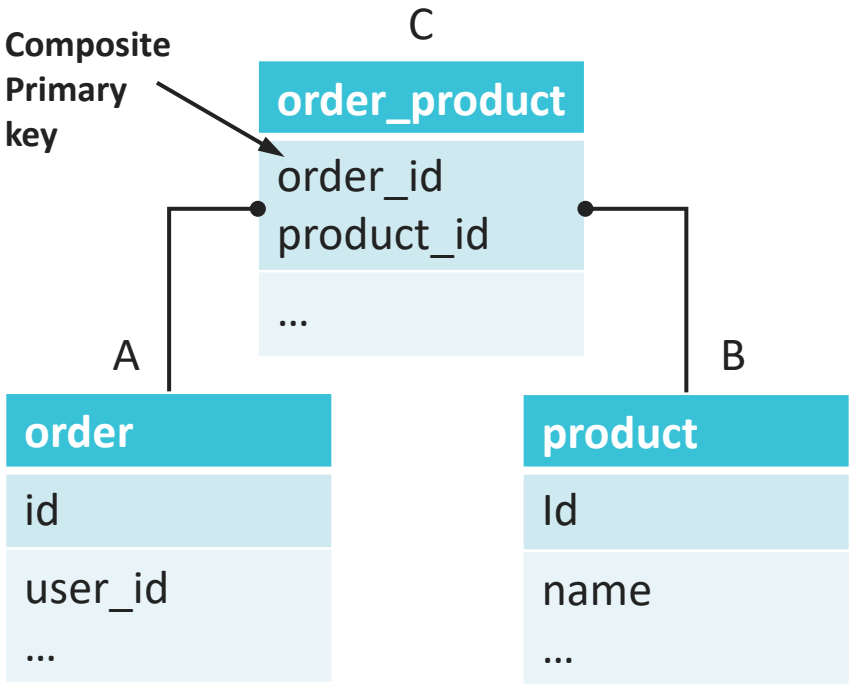
- This is the most common relationship type.
- In this type of relationship, a row in table A can have many matching rows in table B, but a row in table B, can have only one matching row in table A.



Many-to-Many (N:M) Relationship

- In a many-to-many relationship, a row in table A can have many matching rows in table B, and vice versa.





References

- Identifying
  - the primary key of the independent entity is included in the primary key of the dependent entity
- Non-identifying
  - the primary key of the independent entity is included in the child entity but not as part of the dependent entity’s primary key.

Referential integrity

- Referential integrity is a property of data stating that all its references are valid.
- Referential integrity means that two specific database rules must exist between tables:
    - Do not delete data required by another table.
        - You must check that there are no records in related tables that require the record which you are deleting.
    - Do not add incomplete data to a table.
        - You must check that the related information exists in another table before adding information to a table.

- Foreign key constraint

Examples

NAMING CONVENTION

Naming Conventions

- Lowercase.
  - Identifiers should be written entirely in lower case. This includes tables, views, column, and everything else too.
- Data types are not names.
  - Database object names, particularly column names, should be a noun describing the field or object. Avoid using words that are just data types such as text or timestamp. The latter is particularly bad as it provides zero context.
- Underscores separate words.
  - Object name that are comprised of multiple words should be separated by underscores (ie. snake case).
  - Use ***word_count***, ***team_member_id***, ***create_time***, ***is_deleted***

Naming Conventions

- Full words, not abbreviations.
  - Object names should be full English words. In general avoid abbreviations, especially if they're just the type that removes vowels.
  - *Use **middle_name**, not **mid_nm***
- Use common abbreviations.
  - For a few long words the abbreviation is both more common than the word itself. "Internationalization" and "localization" are the two that come up most often as **i18n** and **l10n** respectively. In these cases use the abbreviation.
  - If you're in doubt, use the full English word. It should be obvious where the abbreviation makes sense.

Naming Conventions

- Avoid reserved words.
  - Avoid using any word that is considered a reserved word in the database that you are using. There aren't that many of them so it's not too much effort to pick a different word. Depending on the context, reserved words may require quoting. This means sometimes you'll write **"user"** or **`user`** and sometimes just **user**.
  - Avoid using words like ***user***, ***lock***, or ***table***

- Tables, views, and other relations that hold data should have singular names, not plural.
  - This means our tables and views would be named ***team***, not ***teams***.

Naming Conventions

- Single column primary key fields should be named **id**.
  - It's short, simple, and unambiguous. This means that when you're writing SQL you don't have to remember the names of the fields to join on.
  - Some guides suggest prefixing the table name in the primary key field name, ie. **person_id** vs **id**.
  - The extra prefix is redundant.
  - All field names in non-trivial SQL statements (i.e. those with more than one table) should be explicitly qualified and prefixing as a form of namespacing field names is a bad idea.

Naming Conventions

- Foreign key fields should be a combination of the name of the referenced table and the name of the referenced fields.
- For single column foreign keys (by far the most common case) this will be something like foo_id.

Naming Conventions

- Constraints:
  - **PK_TableName** for **p**rimary **k**ey constraints
  - **FK_TableName_ReferencedTableName[_n]** for **f**oreign **k**ey constraints
  - **UQ_TableName_ColumnName[_ColumnName2...]** for **u**nique **c**onstraints
  - **CK_TableName_ColumnName** (or **CK_TableName_n**) for **c**heck **c**onstraints
  - **IX_TableName_ColumnName** for **i**nde**x**
  - **Table_Name_BIS_TRG** for trigger