Lecture Notes

Concept of Keys

Sample Table:

customer_id	name	phone	age
1	Akon	9876723452	17
2	Akon	9991165674	19
3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	17

Super Key

- An attribute or set of attributes which can uniquely identify a tuple
- It may be redundant or reducible which means that a proper subset of it might also uniquely identify a record in the table
- Example
    - Assuming customer_id can uniquely identify a record, it can be the super key. (customer_id, name) can also be the super key though the attribute “name” is redundant.
    - Similarly, the phone number for every customer will be unique, hence again, the phone can also be the super key.

Candidate Key

- The minimal set of fields (non-reducible) that can uniquely identify each record in a table
- Every table must have at least a single candidate key (obviously can have multiple candidate keys) which may have single or multiple attributes

- Attributes which are part of candidate keys must not contain NULL values
- In our example, customer_id and phone both are candidate keys for table customer.
- **Prime Attributes** - An attribute that is present in any of the candidate keys is a prime attribute

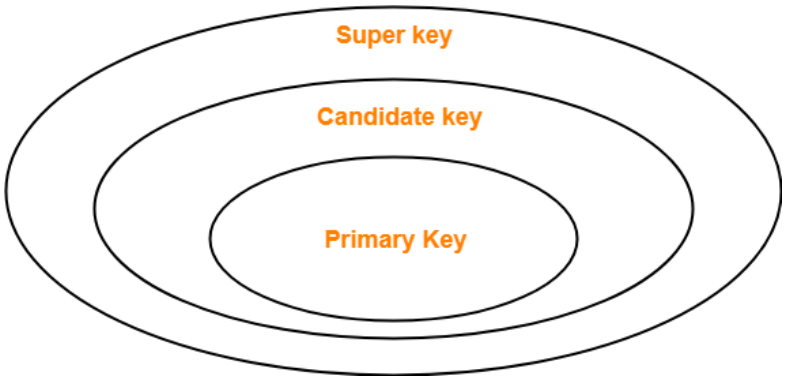
Primary Key

- A primary key is a candidate key that is most appropriate to become the main key for a table in the view of a database designer
- Like candidate keys, the value of the primary key must always be unique and can never be NULL.
- The value of the primary key must be assigned when inserting a record and can never be updated.
- Unlike candidate keys, a relation is allowed to have only one primary key.

Question - In our example of customer relation, among the 2 candidate keys customer_id and phone, what should be the primary key and why?

Ans - The most appropriate candidate key that should be the primary key is **customer_id**

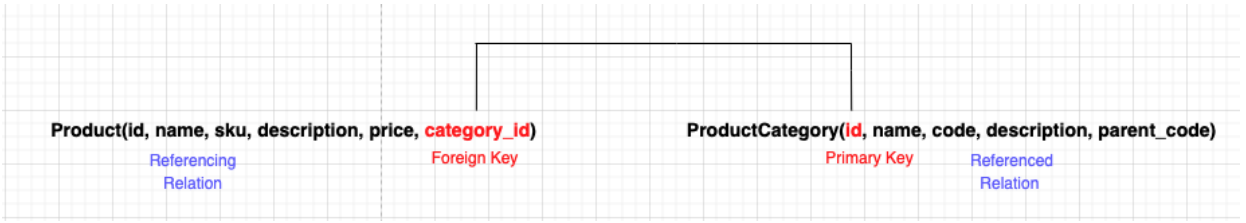
But, why not phone, even if it satisfies all the criteria?
The answer is it can be. But, what if a customer wants to update his/her phone number!



Foreign Key

- In the order table, we have customer_id attribute which is a primary key in the Customer table.
- Customer_id in the order table is called a foreign key as it will help us establish the relationship with the Customer table.
- So, if I need customer details from an order, I can access the customer_id and then get all the details about that customer_id from the customer table.

  Example -



- Similarly, category_id can take only those values which are present in id in the ProductCategory table since only those products actually exist.
- Foreign key references the primary key of the other table.
- A foreign key may have a name other than that of a primary key. It can take only those values which are present in the primary key of the referenced relation.
- There is no restriction on a foreign key to be unique. It can have NULL values as well.

Composite Key

- A candidate key consisting of multiple attributes and not just a single attribute is called a composite key. It holds all the properties of candidate keys.
- Example - In the customer relation (customer_id, phone) could be a composite key

Unique Key

- It is unique for all the records of the table.
- Once assigned, its value can not be changed i.e. it is non-updatable
- It may have a NULL value.

Example - Aadhaar Card Numbers!

How?

- The Adhaar Card Number is unique for all the citizens (tuples) of India (table).
- If it gets lost and another duplicate copy is issued, then the duplicate copy always has the same number as before. Thus, it is non-updateable.
- Few citizens may not have got their Aadhaar cards, so for them, its value is NULL.

ACID Property

Case

- In our ecommerce example, let's say there are 2 buyers (A,B). They made some transactions.

- Do you think the sum of account balances of the 3 accounts (A, B and you) of your accounts will change before and after the payment?

- If there are other customers who are also paying to your account, how will it affect your transaction?

In the context of databases, a transaction is not just related to a financial transaction.

What is a transaction?

→ A transaction is a single logical unit of work that accesses and possibly modifies the contents of a database. Transactions access data using read and write operations.

In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called ACID properties.

ACID stands for Atomicity, Consistency, Isolation and Durability.
Let's go over each of these properties

1. **Atomicity**
    a. All changes to data are performed as if they are a single operation. That is, all the changes are performed, or none of them is.

    b. For example, the atomicity property ensures that, if a debit is made successfully from one account, the corresponding credit is made to the other account.

    c. If anything breaks in between the database needs to be rolled back to the previous consistent state. (What is this consistent state?)

2. **Consistency**
    a. Data is in a consistent state when a transaction starts and when it ends.
    b. For example, the consistency property ensures that the total value of funds in both the accounts is the same at the start and end of each transaction.

3. **Isolation**
    a. The intermediate state of a transaction **is invisible to other transactions**. As a result, transactions that run concurrently appear to be serialized.

b. One of the goals of **isolation is to allow multiple transactions to occur at the same time** without adversely affecting the execution of each.

c. If **A** issues a database transaction at the same time that **B** issues a different transaction, **both transactions should operate on the database in an isolated manner. The database should either perform A's entire transaction before executing B's or vice-versa.**

4. **Durability**
    a. After a transaction is successfully completed, changes to a database should persist and should not get lost, even in the event of a system failure.

    b. For example, the durability property ensures that the changes made to each account will not be reversed.

    c. Durability is ensured by using **database backups and transaction logs** that facilitate the **restoration** of committed transactions despite any subsequent software or hardware failures.

Cheat Sheet for MySQL Functions:

a. Aggregate function

Aggregate function	**Description**
AVG()	Return the average of non-NULL values.
BIT_AND()	Return bitwise AND.
BIT_OR()	Return bitwise OR.
BIT_XOR()	Return bitwise XOR.
COUNT()	Return the number of rows in a group, including rows with NULL values.
GROUP_CONCAT()	Return a concatenated string.
JSON_ARRAYAGG()	Return result set as a single JSON array.

JSON_OBJECTAGG()	Return result set as a single JSON object.
MAX()	Return the highest value (maximum) in a set of non-NULL values.
MIN()	Return the lowest value (minimum) in a set of non-NULL values.
STDEV()	Return the population standard deviation.
STDDEV_POP()	Return the population standard deviation.
STDDEV_SAMP()	Return the sample standard deviation.
SUM()	Return the summation of all non-NULL values a set.
VAR_POP()	Return the population standard variance.
VARP_SAM()	Return the sample variance.
VARIANCE()	Return the population standard variance.

b. Date Functions

Function	Description
CURDATE	Returns the current date.
DATEDIFF	Calculates the number of days between two DATE values.
DAY	Gets the day of the month of a specified date.
DATE_ADD	Adds a time value to date value.
DATE_SUB	Subtracts a time value from a date value.
DATE_FORMAT	Formats a date value based on a specified date format.
DAYNAME	Gets the name of a weekday for a specified date.

DAYOFWEEK	Returns the weekday index for a date.
EXTRACT	Extracts a part of a date.
LAST_DAY	Returns the last day of the month of a specified date
NOW	Returns the current date and time at which the statement is executed.
MONTH	Returns an integer that represents a month of a specified date.
STR_TO_DATE	Converts a string into a date and time value based on a specified format.
SYSDATE	Returns the current date.
TIMEDIFF	Calculates the difference between two TIME or DATETIME values.

TIMESTAMPDIFF	Calculates the difference between two DATE or DATETIME values.
WEEK	Returns a week number of dates.
WEEKDAY	Returns a weekday index for a date.
YEAR	Return the year for a specified date

C. String Functions

Name	**Description**
CONCAT	Concatenate two or more strings into a single string
INSTR	Return the position of the first occurrence of a substring in a string
LENGTH	Get the length of a string in bytes and in characters
LEFT	Get a specified number of leftmost characters from a string

LOWER	Convert a string to lowercase
LTRIM	Remove all leading spaces from a string
REPLACE	Search and replace a substring in a string
RIGHT	Get a specified number of rightmost characters from a string
RTRIM	Remove all trailing spaces from a string
SUBSTRING	Extract a substring starting from a position with a specific length.
SUBSTRING_INDEX	Return a substring from a string before a specified number of occurrences of a delimiter
TRIM	Remove unwanted characters from a string.
FIND_IN_SET	Find a string within a comma-separated list of strings
FORMAT	Format a number with a specific locale, rounded to the number of decimals
UPPER	Convert a string to uppercase

E.Math Functions

Name	Description
ABS()	Returns the absolute value of a number
CEIL()	Returns the smallest integer value greater than or equal to the input number (n).
FLOOR()	Returns the largest integer value not greater than the argument
MOD()	Returns the remainder of a number divided by another
ROUND()	Rounds a number to a specified number of decimal places.
TRUNCATE()	Truncates a number to a specified number of decimal places
ACOS(n)	Returns the arc cosine of n or null if n is not in the range -1 and 1.
ASIN(n)	Returns the arcsine of n which is the value whose sine is n. It returns null if n is not in the range -1 to 1.
ATAN()	Returns the arctangent of n.
ATAN2(n,m), ATAN(m,n)	Returns the arctangent of the two variables n and m

CONV(n,from_base,to_base)	Converts a number between different number bases
COS(n)	Returns the cosine of n, where n is in radians
COT(n)	Returns the cotangent of n.
CRC32()	Computes a cyclic redundancy check value and returns a 32-bit unsigned value
DEGREES(n)	Converts radians to degrees of the argument n
EXP(n)	Raises to the power of e raised to the power of n
LN(n)	Returns the natural logarithm of n
LOG(n)	Returns the natural logarithm of the first argument
LOG10()	Returns the base-10 logarithm of the argument
LOG2()	Returns the base-2 logarithm of the argument
PI()	Returns the value of PI
POW()	Returns the argument raised to the specified power
POWER()	Returns the argument raised to the specified power

RADIANS()	Returns argument converted to radians
RAND()	Returns a random floating-point value
SIGN(n)	Returns the sign of n that can be -1, 0, or 1 depending on whether n is negative, zero, or positive.
SIN(n)	Returns the sine of n
SQRT(n)	Returns the square root of n
TAN(n)	Returns the tangent of n

F. Comparison Functions

COALESCE	Return the first non-NULL arguments, which is very handy for substitution ofNULL.
GREATEST & LEAST	take n arguments and return the greatest and least values of the n arguments respectively.
ISNULL	return 1 if the argument is NULL, otherwise, return zero.