CPE 365 Study Guide

Motivation

Speed of access to disk is very slow. Easy to get lots of data with a database.

Basics

BasicRDM

Normalization is the process of factoring data into tables in such a way as to avoid redundancies or repetitions. Allows us to avoid innacurate edits as well as save space. We only need to edit data in one place to have it effect information throughout the system.

Keys And Joins

Key Citeria

1. Every row has a value for the key.
2. No two rows have the same key.
3. Key is minimal. No superfluous data.

Primary Key

Primary keys are more efficient than normal keys and have the fastest lookup time. Foreign keys target primary keys for this reason.

Cartesian Product

Product of two tables combines every combination of rows from each table.

Query optimizer figures out the best way to carry out a join without generating excess data.
Secondary sorting key is the second value in an order by statement that breaks the tie when the first sorting key match.
ER diagrams or Entity-Relationship Diagram, shows relationships in a database.

n-m relationships need an additional table.

Use a **table alias** to reference a table multiple times in a statement.

DB Design

Creating DB's - DDL

DDL is a subset of commands from SQL.

Why are there so many column type choices?

1. Types allow for flexible data size.
2. Allows for the prevention of human error.
3. Disk storage is made easier with predictable sized data.

Create Command

Make new databases':

```
create database name;
```

collation order is the order in which characters are sorted. Can be set when creating a new database.

Make new tables:

```
create table <name>
```

followed by a comma separated list of column declarations wrapped in parentheses.

To create a table with identical columns as that of another table use the *like* syntax:

```
create table newTable like oldTable;
```

Each table needs to have a default value or be initalized to null.

Any comparisons with null are automatically false. Need to check using special comparisons:

```
is null or is not null
```

Data Types

Integer Types

Type	Bytes	Range
tinyint	1	-128, 127
smallint	2	+/- 32768
mediumint	3	+/- 16 million
int, integer	4	+/- 2 billion
bigint	8	+/- 9 quintillion

All of the types above are signed but can also be declared as unsigned. Parenthesized numbers after the integer are formatting cues for displaying text.

Decimal Types

Use decimal for exact values where you want no rounding.

Type	Bytes
decimal(digits, dec points)	2 * # of digits

Floating Point Types

Type	Bytes	Precision

float	4	7 decimal points
double	8	16 decimal points

The parenthasized value after the type adjusts the value to be either float or double. For example:

```
float(10)
```

will choose whichever data type supports 10 digits of accuracy.

String Types

Type	Bytes	Notes
char(n)	n	stores exactly n chars. empty spots are blank padded
varchar(n)	n+length	only uses as much space as needed for each string. stores the string length

varchar can cause disk fragmentation. A varchar(255) will only use one byte to store the length but is bad becuase it doesn't catch human errors.

Large Data Types

Databases can also store large binary data like images in something called a blob (Binary Large OBject). Very similar to a string in that both are sequences of bytes.

Type	Bytes	Notes
binary(n)	n	similar to char
varbinary(n)	n+length	similar to varchar

For cases where size checking isn't needed text or blob makes a good fit. text is is meant to store string data with character encoding and blob is for raw binary data. Compared to binary forms, text and blob do not allow for scrubbing of the data.

Type	Bytes	Size
tinytext	1	255 bytes
text	2	65 KiB
mediumtext	3	16 MiB
longtext	4	4 GiB
tinyblob	1	255 bytes
blob	2	65 KiB
mediumblob	3	16 MiB
longblob	4	4 GiB

Date and Time Types

Type	Format
datetime	YYYY-MM-DD HH:MM:SS
date	YYYY-MM-DD
mediumtext	HH:MM:SS
timestamp	just like datetime

time can represent values up to 840 hours. timestamp and datetime both record time in microsecond accuracy since epoch, Jan 1, 1970. timestamp will show the time whatever the current timezone is translating from UTC. All datetime values are in UTC time.

Enum Types

Enumeration variables take on values from a fixed list of values. Design goals for enumerations require:

1. Way to ensure that values into an enum column are from the allowed set of values.
2. Order for the values of an enum.
3. Way to translate enum values to readable strings, preferably internationally.

Enums as Integers

The simplest approach is to store each enum as an integer. Doesn't meet criteria 1 or 3. This can be ok as many interactions do not involve any humans therefore a human readable format isn't dire. Chaning the ordering requires for all of the instances of the enum to be modified accordingly.

Enums as Lookup Tables

Create a separate table of allowed enum values. This method meets the criteria except that it doesn't allow for internationalization. This problem can be fixed by adding an additional language column. Downside to this method is that it requires an additional database join.

Built-In Enum Types

A built-in enum type lists requires enum values listed directly in the declaration as shown:

```
builtinColor enum('red', 'green', 'blue')
```

You address these columns as if there were strings (e.g `where color = 'blue'`) but the values are stored as integers with values 1 , 2 and 3. The 0 value is reserved for an error indication. If you try and assign an invalid value into the enum column you get the error value assigned. This method does not support internationalization. Error handling is also weak as it is silent.

Primary Key Contraints

Create table statements can also include **constraint** declerations. Declaring a column as a primary key will setup an index on that column and also the column will be non-nullable.

Indexes

An index is usually a tree that allows for the quick access of data in a table. A B-tree is a fast version of this which only applies to the primary key's index. One can add an index to any column by doing the following:

```
INDEX index_name (column_name, column_name, ...)
```

An index does not require that a column be unique. Indexes are spece-expensive so use them wisely. A **clustered index** allows for the rows in memory to be stored sequentially making it easy to access a range of values. In MySQL the primary key is a clustered index.

Unique Key Contraints

To ensure that a column has all unique values one can put a unique constraint on the column.

```
UNIQUE KEY `UKkind_flavor` (`kind`, `flavor`)
```

Unique key constraint does allow for null values.

Foreign Key Constraints

Foreign key constraints connect parent tables to child tables. The foreign key lives in the child table and references the primary key in the parent table. This creates a 1-n relationship where the n is the child and the 1 is the parent.

```
CONSTRAINT `FKReceipt_customerId` FOREIGN KEY (`customerId`) REFERENCES
`Customer` (`id`)
```

Adding this constraint ensures that the foreign key to correspond to an actual row in the target table. This constraint can add an index on the key.

Good Database Design

DRY: Don't Repeat Yourself. Very applicable in database design.

Have no **obligatory nulls**, temporary nulls to mean something.

Modifying DB's

Updating Rows

The `update` command changes volumn values in existing rows:

```
update Person set unitsCompleted = 18 where id = 3;
```

without a where clause the entire table is modified. Can set multiple columns by separating the assignments with commas.

Inserting Rows

To add a new row to a table, use the `insert` statement:

```
insert into Person values (11, 'Staley', 'Clint', 'M', NULL, NULL, 'Computer
Science');
```

Inserts must comply with constraints.

To conduct a column specific insert use the following syntax:

```
insert into Person(lastName, firstName, gender, dept) values ('Smith', 'Jane',
'F', 'English');
```

All other columns are left unaffected.

Deleting Rows

Use the SQL `delete` command to remove rows from a table:

```
delete from Person where id = 13;
```

Moving a row with a foreign key constraint has different effects depending on tthe foreign key constraint.

Altering Columns

DB Analysis

Simple Functions

Aggregation

Adv Queries

Subqueries

Outer Joins

DB Coding

JDBC Intro

Transactions

DB Theory

Basics

Relational Algebra