**不保证考试会出现下文没有的词！！**

**高分党请路过**

红色的个人觉得概率大点

绿色的个人觉得不太会考

填空题40分，40个空不太可能，20个空，算算几个最可能的，也就那么几个了。

有些词不在填空考也会出现在简答里。比方说事物的几个性质。

A **database** is a collection of interrelated data items

**A DBMS (DataBase Management System) provide a way to store and retrieve information about a particular enterprise**

**Physical level:** describes***how*** a record (e.g., customer) is stored.

**Logical level:** describes ***what*** data are stored in database, and what relationships exist among those data

**View level:** application programs hide details of data types. Views can also hide information (such as an employee’s salary) for security purposes.

**Schema** – the logical structure of the database

**Physical schema**(Internal schema): database design at the physical level

**Logical schema**(Conceptual schema): database design at the logical level

**Subschema**(external schema): describe different views of the database

**Instance** – the actual content of the database at a particular point in time

**Physical Data Independence** – the ability to modify the physical schema without changing the logical schema

**Logical Data Independence** – the ability to modify the logical schema without updating application

**Data Model**: A collection of conceptual tools for describing

Two classes of languages

**Procedural** – user specifies what data is required and how to get those data

**Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data

**SQL**: widely used non-procedural language

**Storage manager** is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

A **transaction** is a collection of operations that performs a single logical function in a database application

**Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

**Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

**Users** are differentiated by the way they expect to interact with

the system

**Application programmers** – interact with system through DML calls

**Sophisticated users** – form requests in a database query language

**Specialized users** – write specialized database applications that do not fit into the traditional data processing framework

**Naïve users** – invoke one of the permanent application programs that have been written previously

Formally, given sets *D*1, *D*2, …. *Dn* a **relation** *r* is a subset of   
 *D*1 x *D*2  x … x *Dn*

The set of allowed values for each attribute is called the **domain** of the attribute

**Domain** – the set of permitted values for each attribute

Attribute values are (normally) required to be **atomic**; that is, indivisible

An element *t* of relation *r* is a ***tuple***, represented by a *row* in a table

*K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*

*K* is a **candidate key** if *K* is minimal  
Example: {*customer\_name*} is a candidate key for *Customer*, since it is a superkey (assuming no two customers can possibly have the same name), and no subset of it is a superkey.

**Primary Key**

**Foreign Key**

**Referencing relation**

**Referenced relation**

A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.

A **candidate key** of an entity set is a minimal super key

Although several candidate keys may exist, one of the candidate keys is selected to be the **primary key**. Need to consider semantics of relationship set in selecting the ***primary key*** in case of more than one **candidate key**

Notation: *σ* *p*(*r*)

*p* is called the **selection predicate**

Defined as:  
  
 *σp*(***r***) = {*t* | *t* ∈ *r* **and** *p(t)*}

Where *p* is a formula in propositional calculus consisting of **terms** connected by : ∧ (**and**), ∨ (**or**), ¬ (**not**)  
Each **term** is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: =, ≠, >, ≥. <. ≤

For *r* ∪ *s* to be valid.

*r,* *s* must have the *same* **arity** (same number of attributes)

The attribute domains must be **compatible** (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

**Aggregation function** takes a collection of values and returns a single value as a result.

**Aggregate operation** in relational algebra

To eliminate the duplications, append ‘***distinct***’ to the end of the function name.

**Conceptual-design** --- Chooses a data model, translates the requirements into a **conceptual schema** of database

**Redundancy**

**Incompleteness**

Implementation

**Logical-design**

**Physical-design**

An **entity** is an object that exists and is distinguishable from other objects

Entities have ***attributes***

An **entity set** is a set of entities of the same type that share the same properties.

A **relationship** is an association among several entities

A **relationship set** is a mathematical relation among *n* ≥ 2 entities, each taken from entity sets

The association between entity sets is referred to as **participation**

**Total participation** (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set

**Partial participation**: some entities may not participate in any relationship in the relationship set

***Relationship instance***represents an association between the named entities in the real-world enterprise that is being modeled

**Role**---The function that an entity plays in a relationship

**Recursive relationship set**---the same entity set participate in a relationship set more than once

**Descriptive attribute** --An **attribute** can also be property of a relationship set.

**Degree**-Refers to number of entity sets that participate in a relationship set. Relationship sets that involve two entity sets are **binary** (or degree two).

An entity set that does not have a primary key is referred to as a **weak entity set**

The existence of a weak entity set depends on the existence of a **identifying entity set**

**Identifying relationship**

The **discriminator**(*or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.

**Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked

The ISA relationship also referred to as **superclass - subclass** relationship

Constraint on which entities can be members of a given lower-level entity set.

**condition-defined**

Example: all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.---**attribute-defined**

**user-defined**

Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.

**Disjoint**

an entity can belong to only one lower-level entity set

Noted in E-R diagram by writing ***disjoint*** next to the ISA triangle

**Overlapping**

an entity can belong to more than one lower-level entity set

**Completeness constraint** -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.

we must create foreign key constraints on the lower-level entity sets

**decompose**

Denote as a **functional dependency**:

*loan\_number* → *amount*

A **functional dependency** is a generalization of the notion of a *key.*

**holds on**

Domain is **atomic** if its elements are considered to be indivisible units

**first normal form**

***third normal form***

If a relation *r* is legal under a set *F* of functional dependencies, we say that *r* **satisfies** *F.*

*A* functional dependency is **trivial** if it is satisfied by all instances of a relation

***closure***

***dependency preserving***

**restriction**

Lossless-join decide whether the relation is able todecompose or not

***canonical cover***

**volatile storage:** loses contents when power is switched off

**non-volatile storage**

**Optical storage**

**Tape storage**-- **sequential-access**

**Cache** – fastest and most costly form of storage; volatile; managed by the computer system hardware.

**Main memory/Second memory/Flash memory**

**Magnetic-disk**

**direct-access** – possible to read data on disk in any order, unlike magnetic tape

**primary storage:** Fastest media but volatile (cache, main memory).

**secondary storage:** next level in hierarchy, non-volatile, moderately fast access time

also called **on-line storage**

**tertiary storage:** lowest level in hierarchy, non-volatile, slow access time

also called **off-line storage**

Surface of platter divided into circular **tracks**

Each track is divided into **sectors.**

**Cylinder** *i*consists of *i*th track of all the platters

**Block** – a contiguous sequence of sectors from a single track

**Disk controller** – interfaces between the computer system and the disk drive hardware.

Computes and attaches **checksums** to each sector to verify that data is read back correctly

**Access time** – the time it takes from when a read or write request is issued to when data transfer begins

**Seek time** – time it takes to reposition the arm over the correct track

**Rotational latency** – time it takes for the sector to be accessed to appear under the head.

**Data-transfer rate** – the rate at which data can be retrieved from or stored to the disk

**Mean time to failure (MTTF)** – the average time the disk is expected to run continuously without any failure.

**Disk-arm-scheduling** algorithms order pending accesses to tracks so that disk arm movement is minimized

**elevator algorithm** : move disk arm in one direction (from outer to inner tracks or vice versa), processing next request in that direction, till no more requests in that direction, then reverse direction and repeat

Files may get **fragmented** over time

**Nonvolatile write buffers** speed up disk writes by writing blocks to a non-volatile RAM buffer immediately

**RAID: Redundant Arrays of Independent Disks**

high **capacity** and high **speed** by using multiple disks in parallel, and

high **reliability** by storing data redundantly, so that data can be recovered even if a disk fails

**Redundancy** – store extra information that can be used to rebuild information lost in a disk failure

**Mirroring** (or **shadowing**)

**Buffer** – portion of main memory available to store copies of disk blocks.

**Buffer manager** – subsystem responsible for allocating buffer space in main memory

**least recently used** (LRU strategy)

**Most recently used (MRU) strategy**

**Slotted page**

**Heap** – a record can be placed anywhere in the file where there is space

**Sequential** – store records in sequential order, based on the value of the search key of each record

**Hashing** – a hash function computed on some attribute of each record; the result specifies in which block of the file the record should be placed

**multitable clustering**

**Search Key** - attribute to set of attributes used to look up records in a file.

An **index file** consists of records (called **index entries**) of the form

**Ordered indices/ Hash**

**Primary index:** in a sequentially ordered file, the index whose search key specifies the sequential order of the file.

* + Also called **clustering index**

**Secondary index**:an index whose search key specifies an order different from the sequential order of the file. Also called non-clustering index**.**

**Index-sequential file:** ordered sequential file with a primary index.

**Dense index** — Index record appears for every search-key value in the file (not every record).

**Sparse** Index: contains index records for only some search-key values

A **bucket** is a unit of storage containing one or more records

**closed hashing. open hashing**

Annotated expression specifying detailed evaluation strategy is called an **evaluation-plan**.

**Runs-------------好像在另外一门课也见过，什么意思来着。**

Partitioning is said to be **skewed**

**Hash-table overflow**

**Query Optimization**

Generating logically equivalent expressions using **equivalence rules**

Choosing the cheapest plan based on **estimated cost**

Two relational algebra expressions are said to be **equivalent** if on every legal database instance the two expressions generate the same set of tuples

Systems may use ***heuristics***to reduce the number of choices that must be made in a cost-based fashion.

**Heuristic optimization transforms the query-tree by using a set of rules that typically (but not in all cases) improve execution performance**:

The **selectivity** of a condition θ*i* is the probability that a tuple in the relation *r* satisfies θ*i* .

**A transaction** is a unit of program execution that accesses and possibly updates various data items.To preserve the integrity of data the database system must ensure

**Atomicity Consistency Isolation Durability**

**Active Partially committed Failed Aborted Committed Rolled back**

The **recovery-management** component of a database system implements the support for atomicity and durability

***shadow-database*** scheme

**Schedule** – a sequences of instructions that specify the chronological order in which instructions of concurrent transactions are executed

**serial schedule *equivalent* Schedule Recoverableschedule**

**conflict serializability view serializability**

**conflict equivalent***.* ***view equivalent***

Every view serializable schedule that is not conflict serializable has **blind writes**

**Cascading rollback Cascadelessschedules**

**Data items** can be locked in two **modes**

***exclusive (X) mode shared (S) mode***

**deadlock *Deadlock prevention***

**wait-die scheme — non-preemptive**

**wound-wait scheme — preemptive**

**Timeout-Based Schemes**

**Starvation** is also possible if concurrency control manager is badly designed

**Granularity of locking** (level in tree where locking is done):

**fine granularity** (lower in tree):

**coarse granularity** (higher in tree):

**A log** is kept on stable storage.

**The deferred database** modification scheme records all modifications to the log, but defers all the writes to after **partial commit.**

**The immediate database modification scheme** allows database modification to be output to the database while the transaction is still in **active state.**

**Undo *redo must be* idempotent**

Streamline recovery procedure by periodically performing **checkpointing** in addition to the previous technique**s**