Week2_Course_part1

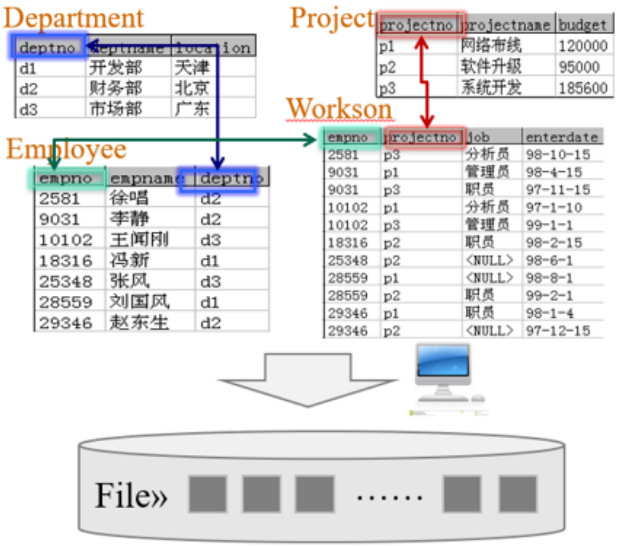
Principles of Data Layout and Index

Relation Model

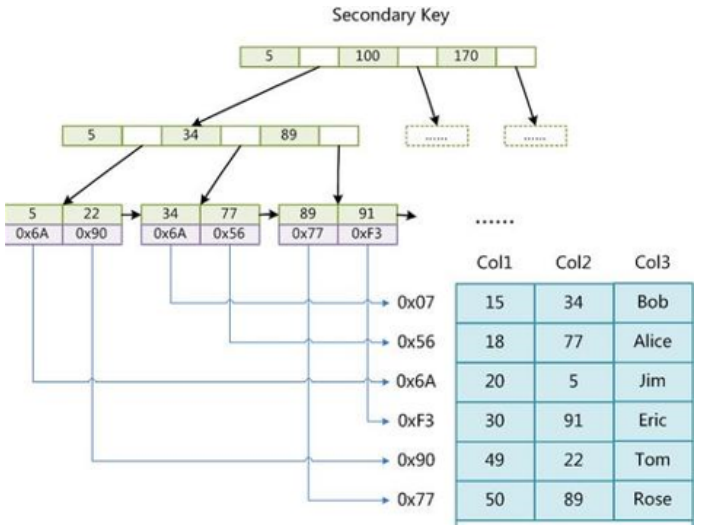
Database System - Nankai

Principles of Data Layout and Index

- How to lay out data on disk?



- How to find a record quickly?

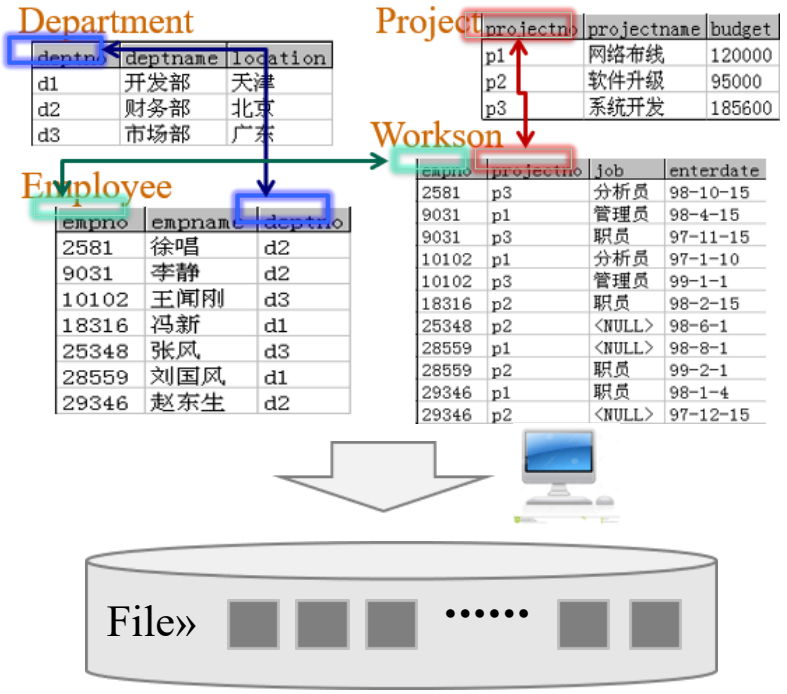


Database System - Nankai

Principles of Data Layout

- How to lay out data on disk
- How to move it to memory

Principles are rather simple, but there are lots of variations in the details



Database System - Nankai

主要讲授内容

Department

deptno	deptname	location
d1	开发部	天津
d2	财务部	北京
d3	市场部	广州

Project

projectno	projectname	budget
p1	网络布线	120000
p2	软件升级	95000
p3	系统开发	185600

Employee

empno	empname	deptno
2581	徐唱	d2
9031	李静	d2
10102	王闻刚	d3
18316	冯新	d1
25348	张风	d3
28559	刘国风	d1
29346	赵东生	d2

Workson

empno	projectno	job	enterdate
2581	p3	分析员	98-10-15
9031	p1	管理员	98-4-15
9031	p3	职员	97-11-15
10102	p1	分析员	97-1-10
10102	p3	管理员	99-1-1
18316	p2	职员	98-2-15
25348	p2	<NULL>	98-6-1
28559	p1	<NULL>	98-8-1
28559	p2	职员	99-2-1
29346	p1	职员	98-1-4
29346	p2	<NULL>	97-12-15

关系模式与关系

属性(Attribute) → 字段(Field)?

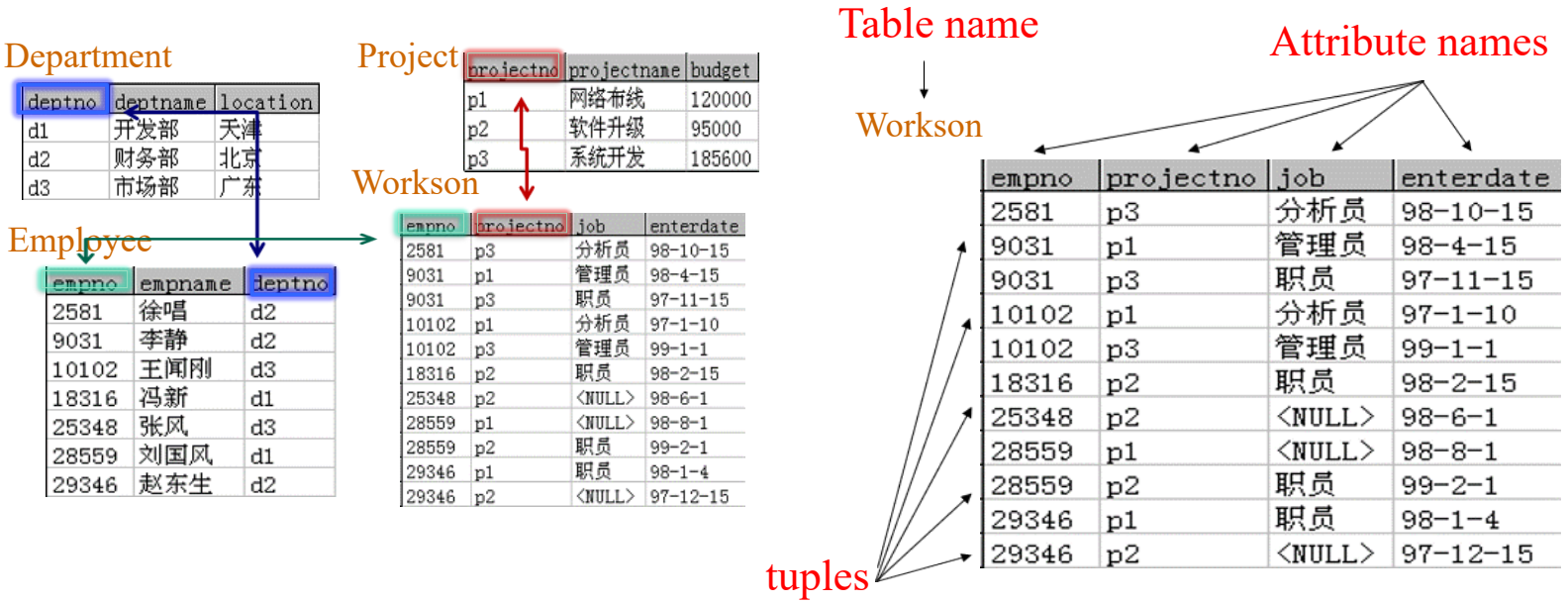
元组(tuple) → 记录(Record)?

记录集合 → 存储块（Block）

块的集合→ 存储关系(Relation)?

Database System - Nankai

The Relational Data Model



Tables → Relations ; Columns → Attributes ; Rows → Tuples ;

Schema (e.g.: Workson = (empno, projectno, job, enterdate))

Database System - Nankai

mathematical relations (i.e., sets)

$S\#: \{S_1, S_2, \ldots \ldots S_m\}$

$C\#: \{C_1, C_2, \ldots \ldots C_n\}$

$G: \{1,2, \ldots \ldots 100\}$

$SCGG = S\# \times C\# \times G \times G$

$SCGG: \{(S_1, C_1, 85,92), (S_1, C_1, 70,55), \ldots$
$\ldots (S_i, C_j, 60,74) \ldots\}$（$m \times n \times 100 \times 100$个元素）

$Relation: \quad SC \subseteq SCGG$

Database System - Nankai

Why Are They Called Relations?

Relation:

- *Each* attribute has a type, must be atomic type, called domain
- $D_1, D_2, \ldots, D_n$ are domains
- $R \subseteq D_1 \times \cdots \times D_n$

Example: *project* $\subseteq$ *String* $\times$ *String*$\times$ *Integer*

project

projectno	projectname	budget
p1	网络布线	120000
p2	软件升级	95000
p3	系统开发	185600

Tuple: $t \in R$

Example: t = ("p1", "网络布线", 95000)

Schema: associates labels to domains

Example: project: { [*projectno:string, projectname:string, budget:integer*] }

Database System - Nankai

Why Are They Called Relations?

Relations

projectno	projectname	budget
p1	网络布线	120000
p2	软件升级	95000
p3	系统开发	185600

Considered equivalent to...

{ (p1，网络布线，120000)，
(p2，软件升级， 95000)，
(p3，软件开发，185600) }

Relational database semantics are defined in terms of mathematical relations (i.e., sets)

Database System - Nankai

KEYS AND RELATIONS

- **Superkeys:** (超键、超码)
  set of attributes of table for which every row has distinct set of values
- **Candidate keys:** (候选键、候选码)
  "minimal" superkeys
- **Primary keys:** (主键、主码)
  DBA-chosen candidate key (marked in schema by underlining)

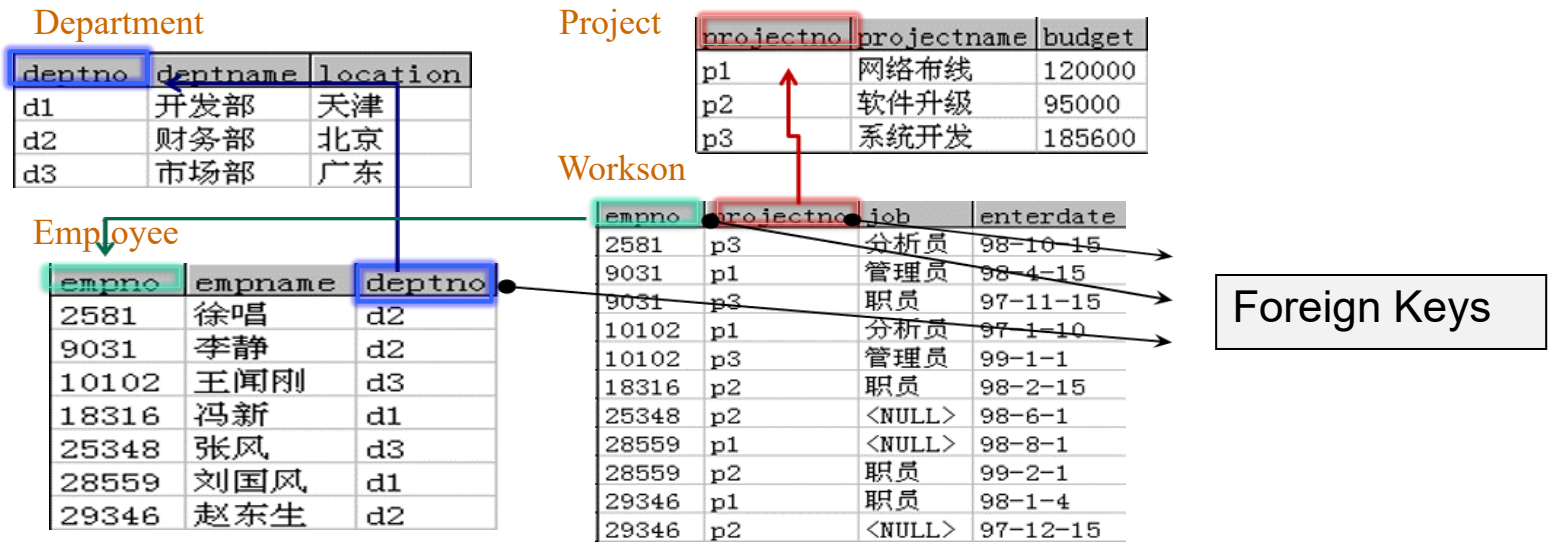
workson

empno	projectno	job	enterdate
2581	p3	分析员	98-10-15
9031	p1	管理员	98-4-15
9031	p3	职员	97-11-15
10102	p1	分析员	97-1-10
10102	p3	管理员	99-1-1
18316	p2	职员	98-2-15
25348	p2	<NULL>	98-6-1
28559	p1	<NULL>	98-8-1
28559	p2	职员	99-2-1
29346	p1	职员	98-1-4
29346	p2	<NULL>	97-12-15

e.g., workson = (empno, projectno, job, enterdate)

Database System - Nankai

Foreign Keys and Integrity



实体完整性(Entity Integrity) - 主键不允许为空值，也不允许出现重复值

参照完整性(Referential Integrity) - 外键如果取值，一定要取主键中出现的值

Database System - Nankai

The Relational Data Model- Summary

- Relational Data Model（关系数据模型）
- Relational Schema（关系模式）
- Relation（关系）
- Superkey（超键、超码）
- Candidate keys（候选键、候选码
- Primary keys（主键、主码）
- Entity Integrity (实体完整性)
- Referential Integrity (参照完整性)

Database System - Nankai

单选题 1分

互动交流一

关系表中的行（row）称为

A 属性

B 元组

C 关系

D 关系模式

workson

empno	projectno	job	enterdate
2581	p3	分析员	98-10-15
9031	p1	管理员	98-4-15
9031	p3	职员	97-11-15
10102	p1	分析员	97-1-10
10102	p3	管理员	99-1-1
18316	p2	职员	98-2-15
25348	p2	<NULL>	98-6-1
28559	p1	<NULL>	98-8-1
28559	p2	职员	99-2-1
29346	p1	职员	98-1-4
29346	p2	<NULL>	97-12-15

提交

se System - Nankai

单选题 1分

互动交流二

右表中employee(empno, empname,deptno)称为

A 属性

B 元组

C 关系

D 关系模式

employee

empno	empname	deptno
2581	徐唱	d2
9031	李静	d2
10102	王闻刚	d3
18316	冯新	d1
25348	张风	d3
28559	刘国风	d1
29346	赵东生	d2

提交

se System - Nankai

填空题 1分

互动交流三

以下关系表的候选码(Candidate keys)为 [填空1]

employee

empno	empname	deptno
2581	徐唱	d2
9031	李静	d2
10102	王闻刚	d3
18316	冯新	d1
25348	张风	d3
28559	刘国风	d1
29346	赵东生	d2

提交

Database System - Nankai

填空题 1分

互动交流四

下图关系表的候选码(Candidate keys)为 [填空1]

workson

empno	projectno	job	enterdate
2581	p3	分析员	98-10-15
9031	p1	管理员	98-4-15
9031	p3	职员	97-11-15
10102	p1	分析员	97-1-10
10102	p3	管理员	99-1-1
18316	p2	职员	98-2-15
25348	p2	<NULL>	98-6-1
28559	p1	<NULL>	98-8-1
28559	p2	职员	99-2-1
29346	p1	职员	98-1-4
29346	p2	<NULL>	97-12-15

提交

Database System - Nankai

填空题 3分

互动交流五

有三个关系表：

学生(学号，姓名，出生日期）；

课程（课号，课名，学分）；

选课（学号，课号，成绩）；

请问选课表的候选码有 [填空1]，选课表中有几个外键 [填空2]，外键分别为 [填空3]

提交

Database System - Nankai

Week2_Course_part2

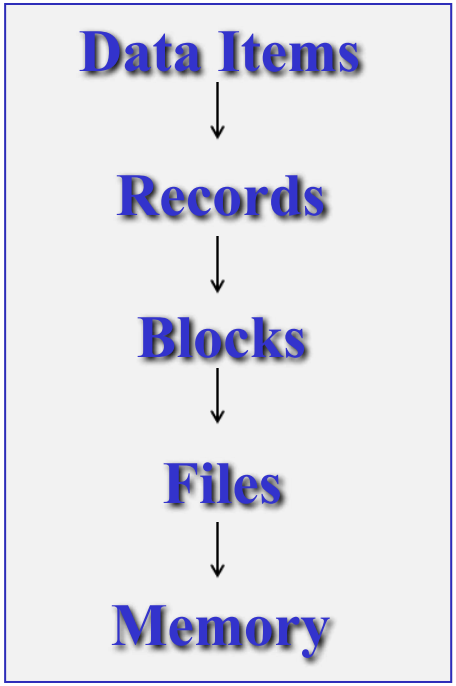
Principles of Data Layout and Index

Data Layout - Data Items & Records

Database System - Nankai

Principles of Data Layout

- Attributes of relational tuples (or objects) represented by sequences of bytes called fields (e.g.，p1，网络布线，120000)
- Fields grouped together into records
  - representation of tuples or objects
- Records stored in blocks
- File: collection of blocks that forms a relation (or the extent of an object class)

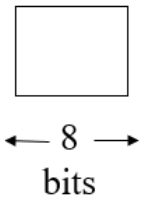


Database System - Nankai

What are the data items we want to store?

- a salary, a name
- a date, a picture
- ...

⇨ What we have available: Bytes



Database System - Nankai

To represent:

- Integer (short): 2 bytes (~ -32000…+32000)
  e.g., 35 is

00000000	00100011

- Real, floating point
  - *n* bits for mantissa, *m* for exponent….

- arithmetic interpretation by hardware

To represent:

- Characters

  various coding schemes suggested,

  most popular is ASCII

  Example (8 bit ASCII):

```
A:    01000001
a:    01100001
5:    00110101
LF:   00001010
```

To represent:

- Boolean

  e.g., TRUE | 1111 1111

  FALSE | 0000 0000

- Application specific

  e.g., RED → 1 GREEN → 2

  BLUE → 3 YELLOW → 4 …

⇨ Can we use less than 1 byte/code?

Yes, but only if desperate...

Database System - Nankai

To represent:

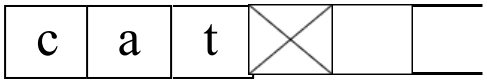
- Dates, e.g.:
  - Integer: # days since Jan 1, 1900
  - 8 chars: YYYYMMDD
  - 7 chars: YYYYDDD
  - 10 chars: YYYY-MM-DD (SQL2)

    (not YYMMDD! Why?)
- Time, e.g.
  - Integer: seconds since midnight
  - chars: HH:MM:SS[.FF…] (SQL2)

To represent:

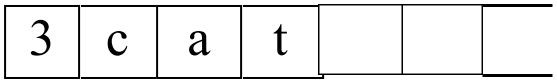
- String of characters
  - Null terminated

    e.g.,

  - Length given

    e.g.,

  - Fixed length

- **Bag of bits**

Record - Collection of related fields

E.g.: Employee record:

name field,

salary field,

date-of-hire field, ...

empno	projectno	job	enterdate
2581	p3	分析员	98-10-15
9031	p1	管理员	98-4-15
9031	p3	职员	97-11-15
10102	p1	分析员	97-1-10
10102	p3	管理员	99-1-1
18316	p2	职员	98-2-15
25348	p2	<NULL>	98-6-1
28559	p1	<NULL>	98-8-1
28559	p2	职员	99-2-1
29346	p1	职员	98-1-4
29346	p2	<NULL>	97-12-15

Types of records:

- **Main choices:**
  - **FIXED vs VARIABLE FORMAT**
  - **FIXED vs VARIABLE LENGTH**

Database System - Nankai

Fixed Format and Length

A SCHEMA (not record) contains following information

- number of fields
- type of each field
- order in record
- meaning of each field

Example: fixed format and length

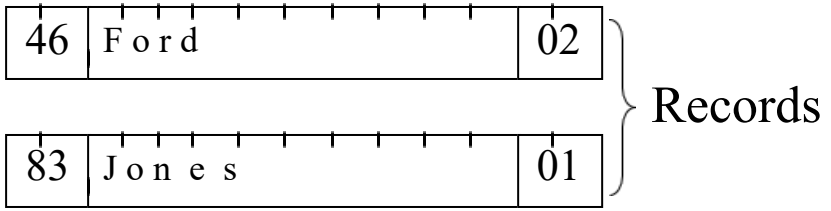
Employee record

(1) E#, 2 byte integer

(2) E.name, 10 char.

(3) Dept, 2 byte code

} Schema

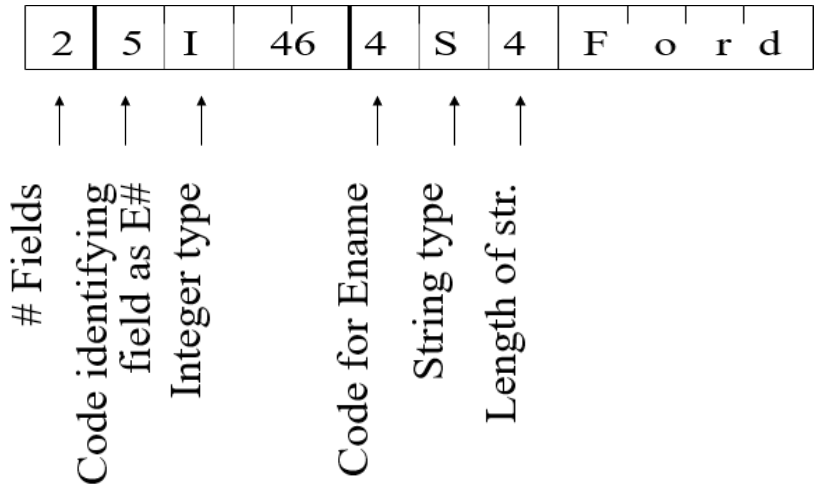


} Records

Database System - Nankai

Variable format

- Record itself contains format; "Self Describing"



A	B	C	D	E	F	...		X	Y
1									
2			0.5						
3								0.7	
4									
...					12				
...									

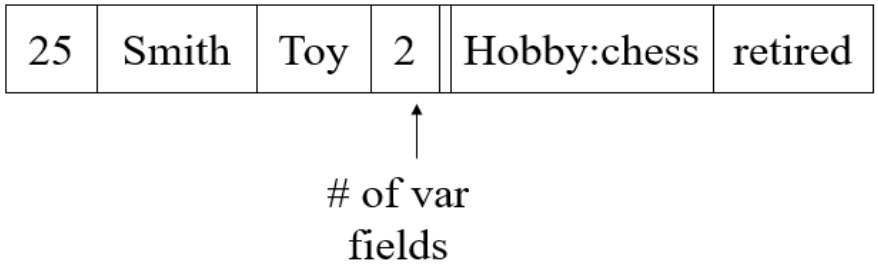
Field name codes could also be strings, i.e. **tags**
(→ XML as a data interchange format)

Database System - Nankai

Variant between FIXED/VAR format

- Hybrid format
  - one part is fixed, other variable

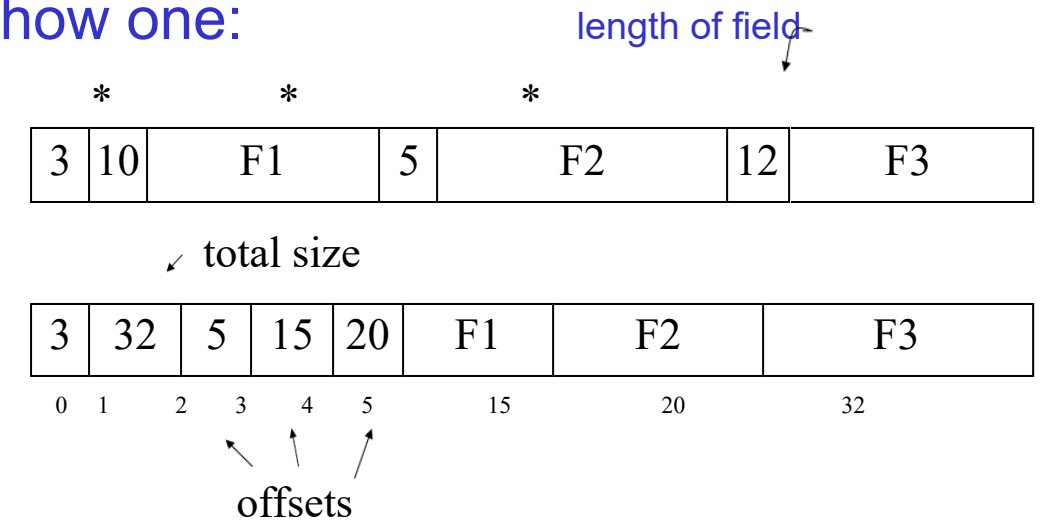
E.g.: All employees have E#, name, dept;
Other fields vary.



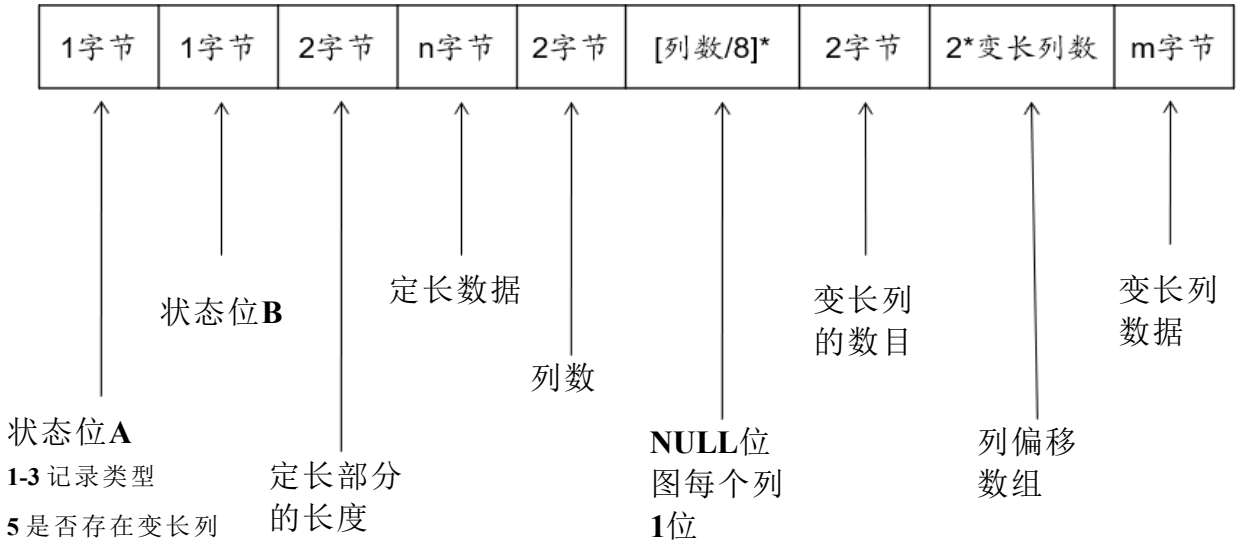
Database System - Nankai

Many variations in internal organization of record

Just to show one:



SQL Server 数据行结构



Database System - Nankai

Data Items & Records - Summary

Data Items – 数值型、日期型、字符型……

Records – 格式：固定/可变，长度：固定/可变

Blocks – 下一节介绍

Database System - Nankai

填空题 1分

互动交流一

学生记录采用固定格式和定长记录方式存储，如右图所示。存储100个这样的学生记录，占磁盘空间 [填空1] 个字节。

record: fixed format and length

student record

(1) sno, char(8)

(2) name, char(10)

(3) birth, date --(3 bytes)

提交

se System - Nankai

单选题 1分

互动交流二

State if the following statements are true or false:

Fixed-size records should always be fixed-format

A true

B false

提交

Database System - Nankai

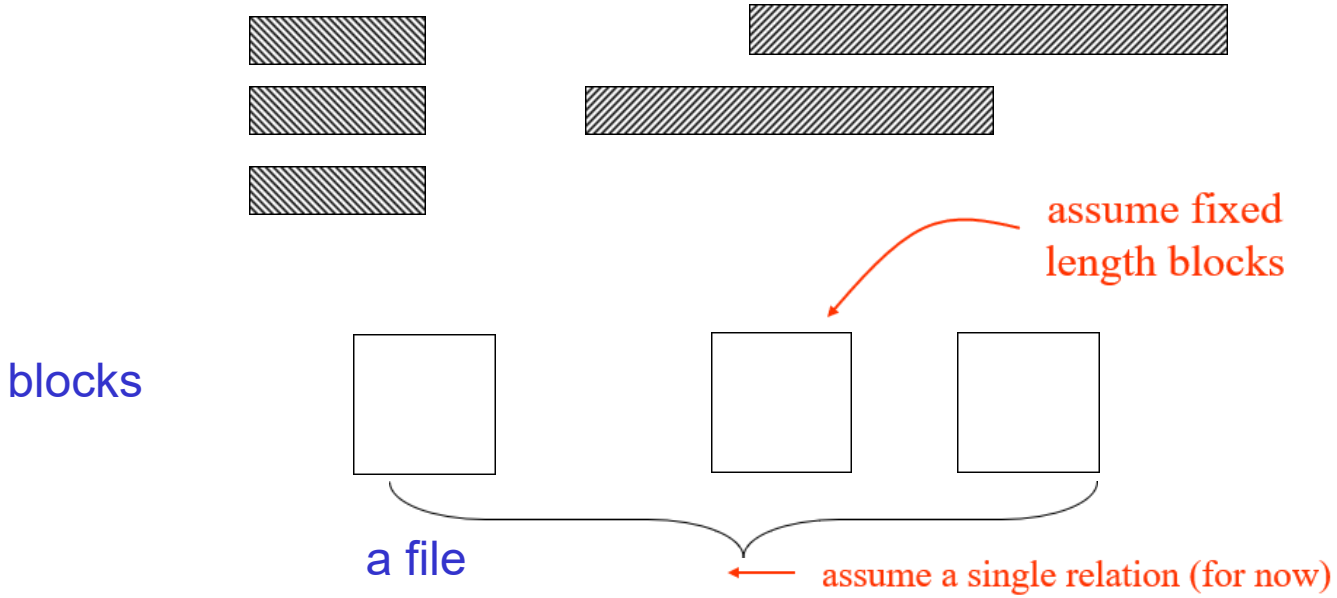
Week2_Course_part3

Principles of Data Layout and Index

Data Layout - Blocks & Column Store

Database System - Nankai

Next: placing records into blocks



Database System - Nankai

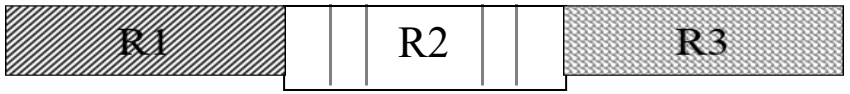
Issues in storing records in blocks:

(1) separating records
(2) spanned vs. unspanned
(3) mixed record types – clustering
(4) split records
(5) sequencing
(6) addressing records

Database System - Nankai

(1) Separating records

Block



(a) fixed size recs. -> no need to separate

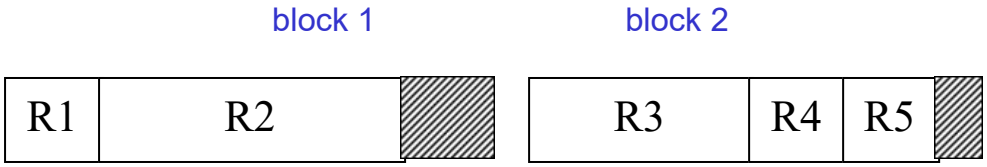
(b) special marker

(c) give record lengths (or offsets)

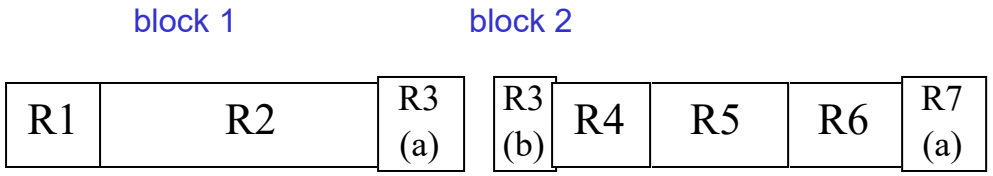
- within each record
- in block header (see later)

(2) Spanned vs. Unspanned

- Unspanned: records are within one block



- Spanned: records span block boundaries



Database System - Nankai

Spanned vs. unspanned

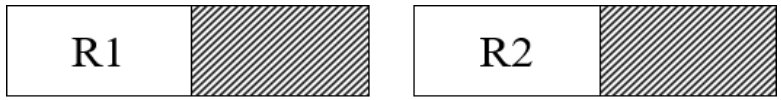
- Unspanned is much simpler, but may waste space...
- Spanned necessary if

  record size > block size

  (e.g., fields containing large "BLOB"s

  for, say, MPEG video clips)

10^6 records
each of size 2,050 bytes (fixed)
block size = 4096 bytes



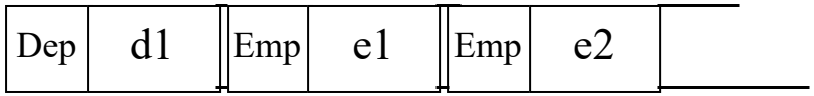
2050 bytes wasted 2046 2050 bytes wasted 2046

Space used about 4×10^9 B, about half wasted

(3) Mixed record types

- Mixed - records of different types
  (e.g. DEPT, EMPLOYEE)
  allowed in same block

e.g., a block:



Why do we want to mix?

Answer: CLUSTERING

Records that are frequently accessed together should be in the same block

Database System - Nankai

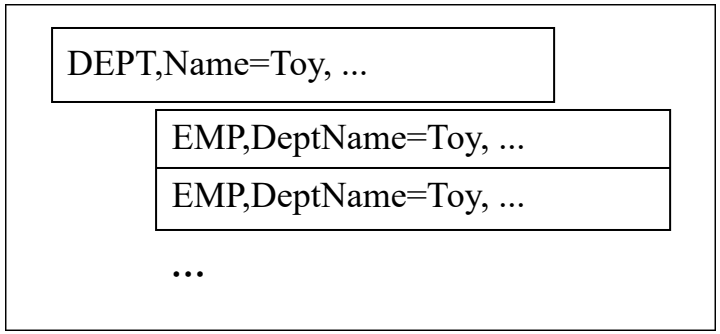
Example

Q1: select DEPT.Name, EMP.Name, ...

from DEPT, EMP

where DEPT. Name = EMP.DeptName

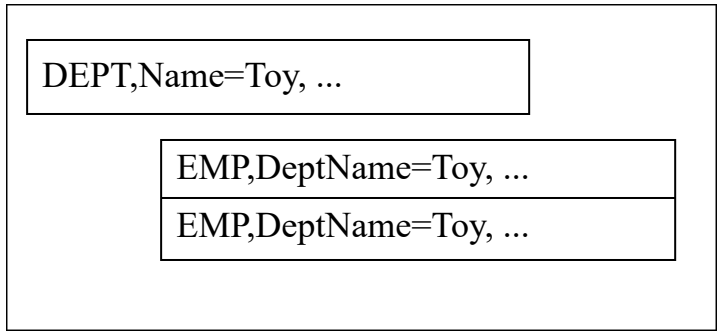
a block



- If Q1 frequent, clustering is good
- But consider Q2:

```
SELECT *
FROM DEPT
```

If Q2 is frequent, clustering is counter-productive



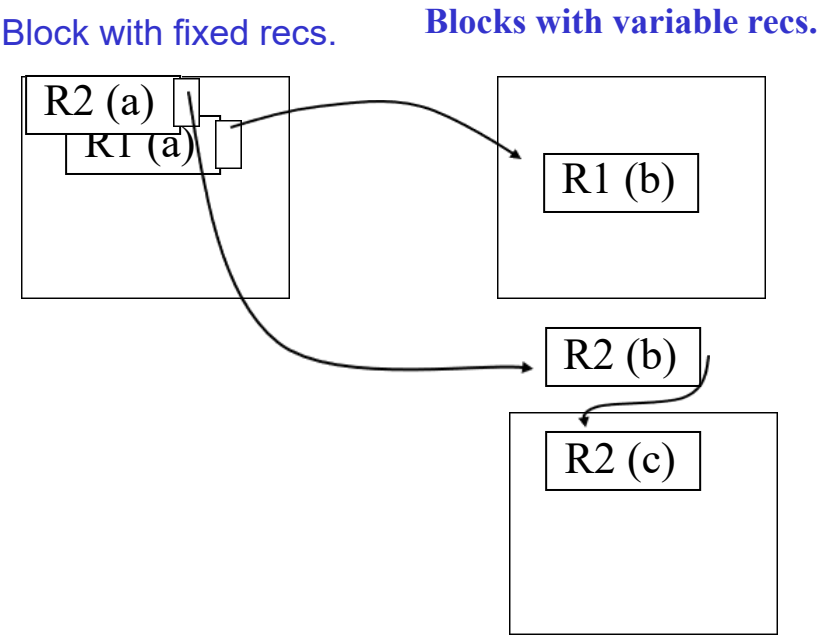
Database System - Nankai

(4) Split records

Typically for hybrid format

Fixed part in one block

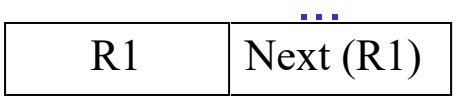
Variable part in another block



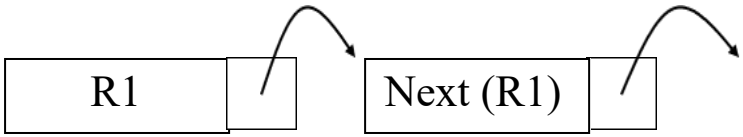
Database System - Nankai

Sequencing Options

(a) Next record physically contiguous



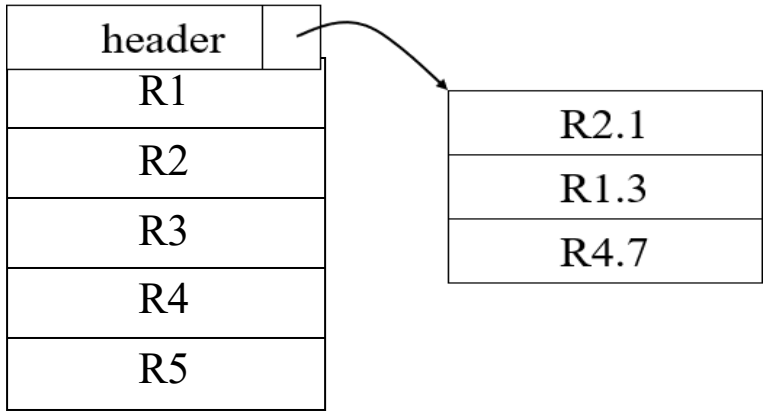
(b) Linked



Sequencing Options

(c) Overflow area

Records in sequence



Ex #1 Indirection in block

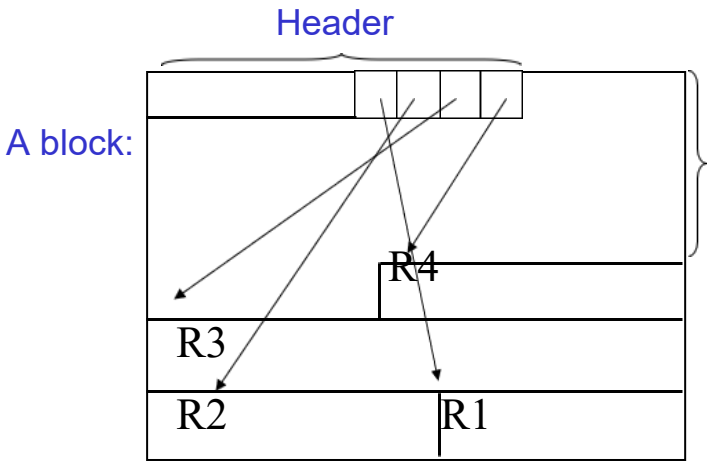
Block header May contain:

- File ID (or RELATION or DB ID); the ID of this block - Record directory; Pointer to free space

- Type of block (e.g. contains records of type 4; is overflow, ...)

- Pointer to other blocks "like it" (say, if part of an index structure)

- Timestamp ...



Database System - Nankai

SQL Server 的数据页

- 数据页是8K，包括三部分：页头、数据行和行偏移数组
- 页头96个字节
  - pageID，nextpage，prevpage，objID，lsn，slotCnt，level，indexID，freeData，pminlen……
- 数据行，最大长度8060个字节
  - 数据行不能跨页存储
- 行偏移数组，每行2个字节
- DBCC PAGE

Database System - Nankai

Row vs Column Store (Knowledge expansion)

- So far we assumed that fields of a record are stored contiguously (row store)...
- Another option is to store like fields together (column store)

Row Store

- Example: Order consists of
  - id, cust, prod, store, price, date, qty

id1	cust1	prod1	store1	price1	date1	qty1
id2	cust2	prod2	store2	price2	date2	qty2
id3	cust3	prod3	store3	price3	date3	qty3

Database System - Nankai

Column Store

- Example: Order consists of
  - id, cust, prod, store, price, date, qty

id1	cust1
id2	cust2
id3	cust3
id4	cust4
...	...

id1	prod1
id2	prod2
id3	prod3
id4	prod4
...	...

id1	price1	qty1
id2	price2	qty2
id3	price3	qty3
id4	price4	qty4
...	...	...

ids may or may not be stored explicitly

Database System - Nankai

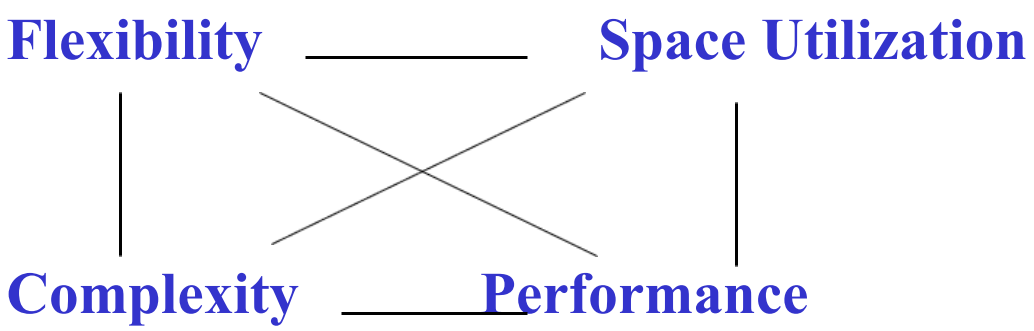
Row vs Column Store

- Advantages of Column Store
  - more compact storage (fields need not start at byte boundaries)
  - efficient reads on data mining operations
- Advantages of Row Store
  - writes (multiple fields of one record)more efficient
  - efficient reads for record access (OLTP)

- Mike Stonebreaker, Elizabeth (Betty) O'Neil, Pat O’ Neil, Xuedong Chen, et al. " C-Store: A Column-oriented DBMS," Presented at the 31st VLDB Conference, September 2005.

Data Layout - Summary

- About 10,000,000 ways to organize my data on disk...
  Which one is right for me?



Data Items

Records

Blocks

请同学们在学堂云上完成本节课后作业！！！