EXEM technical report no.012

Ver. 2
2016.10.20

PostgreSQL Deep Internal

(PostgreSQL 9.4)

Research & Contents Team

Table of Agenda

01. PostgreSQL의 역사

02. PostgreSQL Architecture

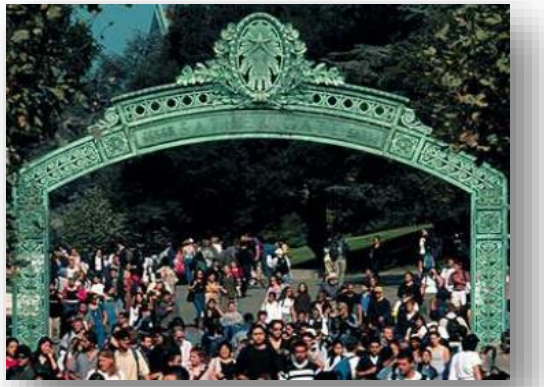
03. PostgreSQL MVCC

04. Page Layout

05. Vacuum

01. PostgreSQL의 역사

EXEM



The University of California at Berkeley

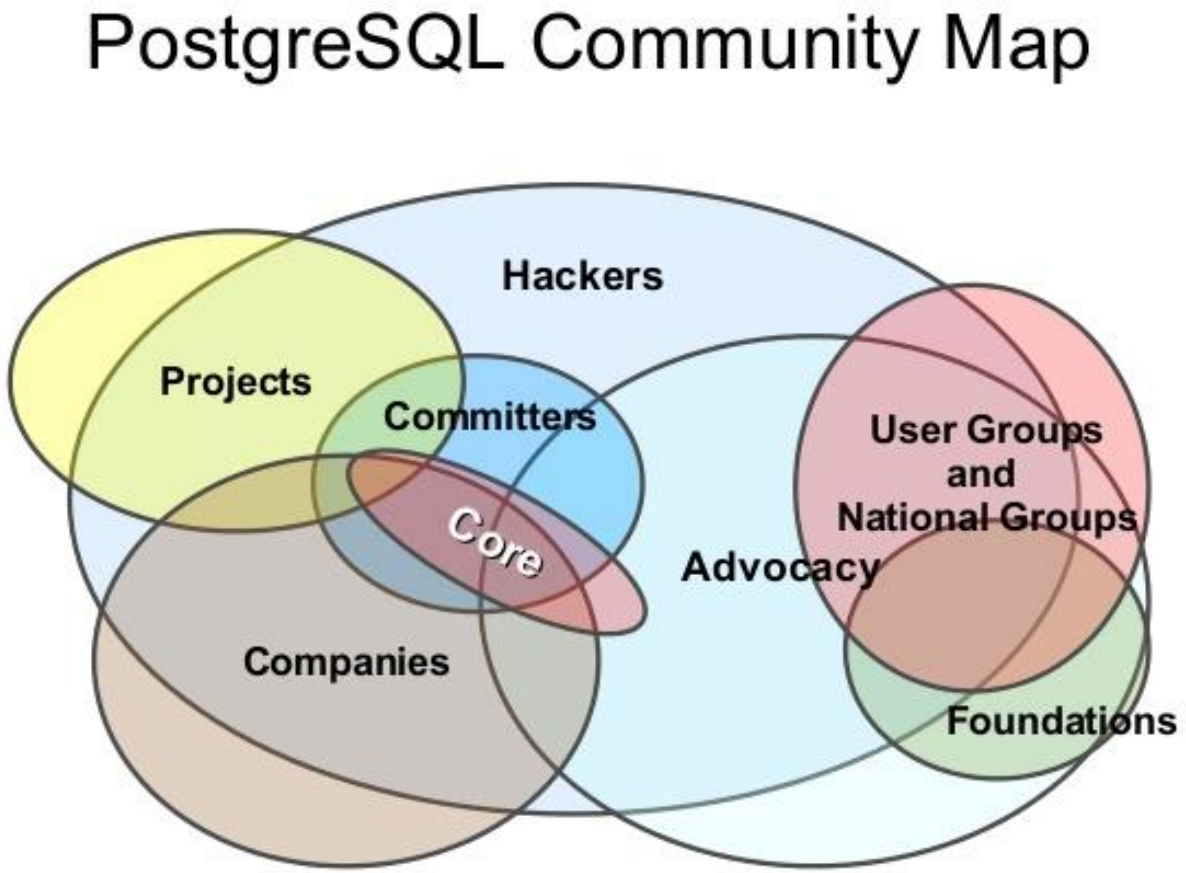
Michael Stonebraker

Jolly Chen
Andrew Yu

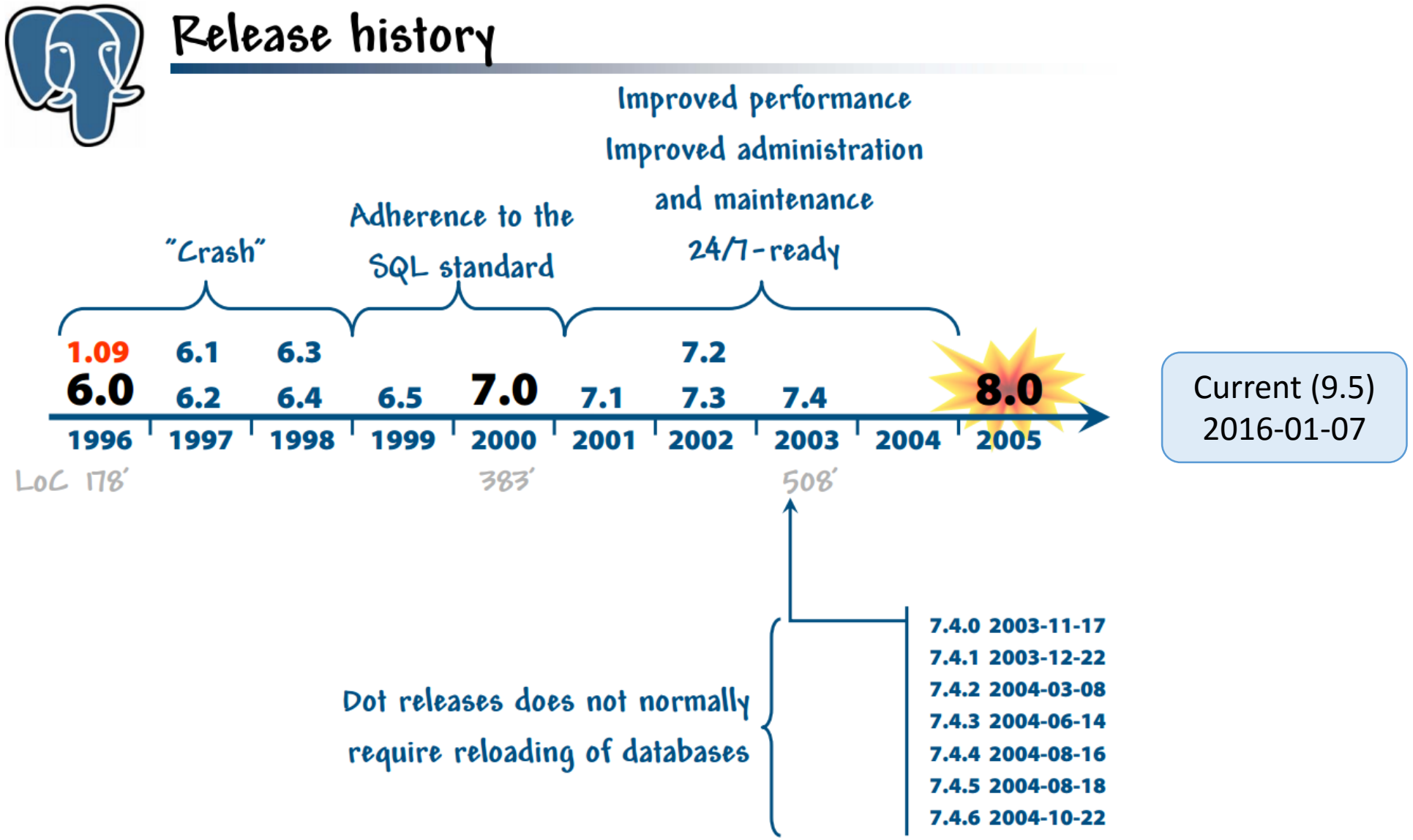
PostgreSQL Core Team

Year	Name	Developer	Features
1977-1985	Ingres	The University of California at Berkeley	• Research prototype • Relational Technologies • Established the company Ingres in 1980 • Purchased by Computer Associates in 1994
1986-1994	Postgres	Michael Stonebraker	• Research Prototype • Spawned Illustra • Purchased by Informix
1994-1995	Postgres95	Jolly Chen, Andrew Yu	• Added SQL • Spawned PostgreSQL
1996-	PostgreSQL(6.0)		• Renamed to PostgreSQL to reflect its support for SQL

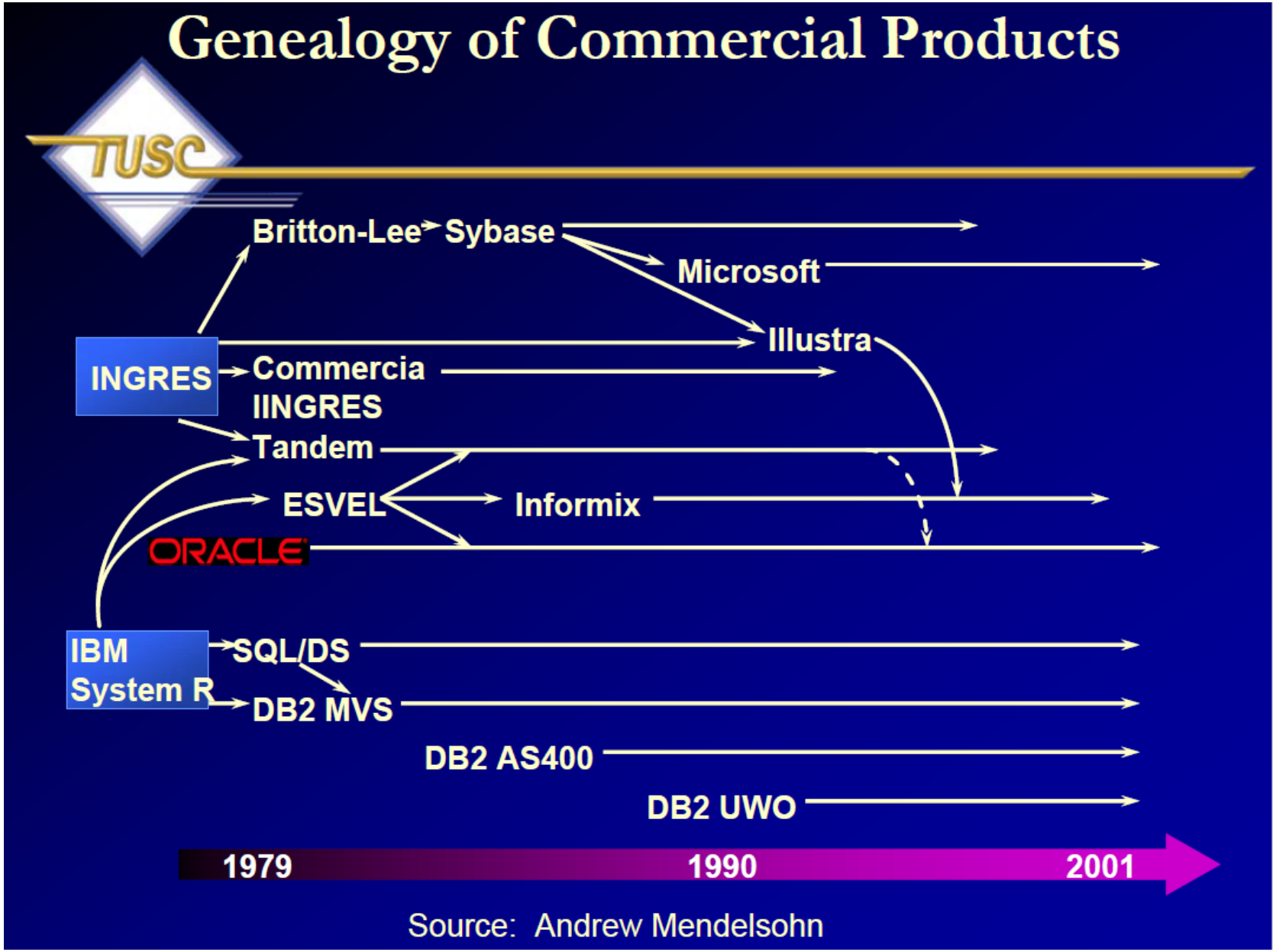
출처
- https://drive.google.com/folderview?id=0B4hWvkMs8fCPWG44dDBITHpMc0U&usp=sharing_eid&ts=5702089b&tid=0B4hWvkMs8fCPeU9oWEhHNnItVkk
- https://momjian.us/main/writings/pgsql/history.pdf



출처 http://www.slideshare.net/PGExperts/development-of-83-in-india?qid=5218f563-3be2-41bc-b54d-9f3146df1f1f&v=&b=&from_search=1



출처 get to know postgreSQL , Bruce Momjian



GreenPlum

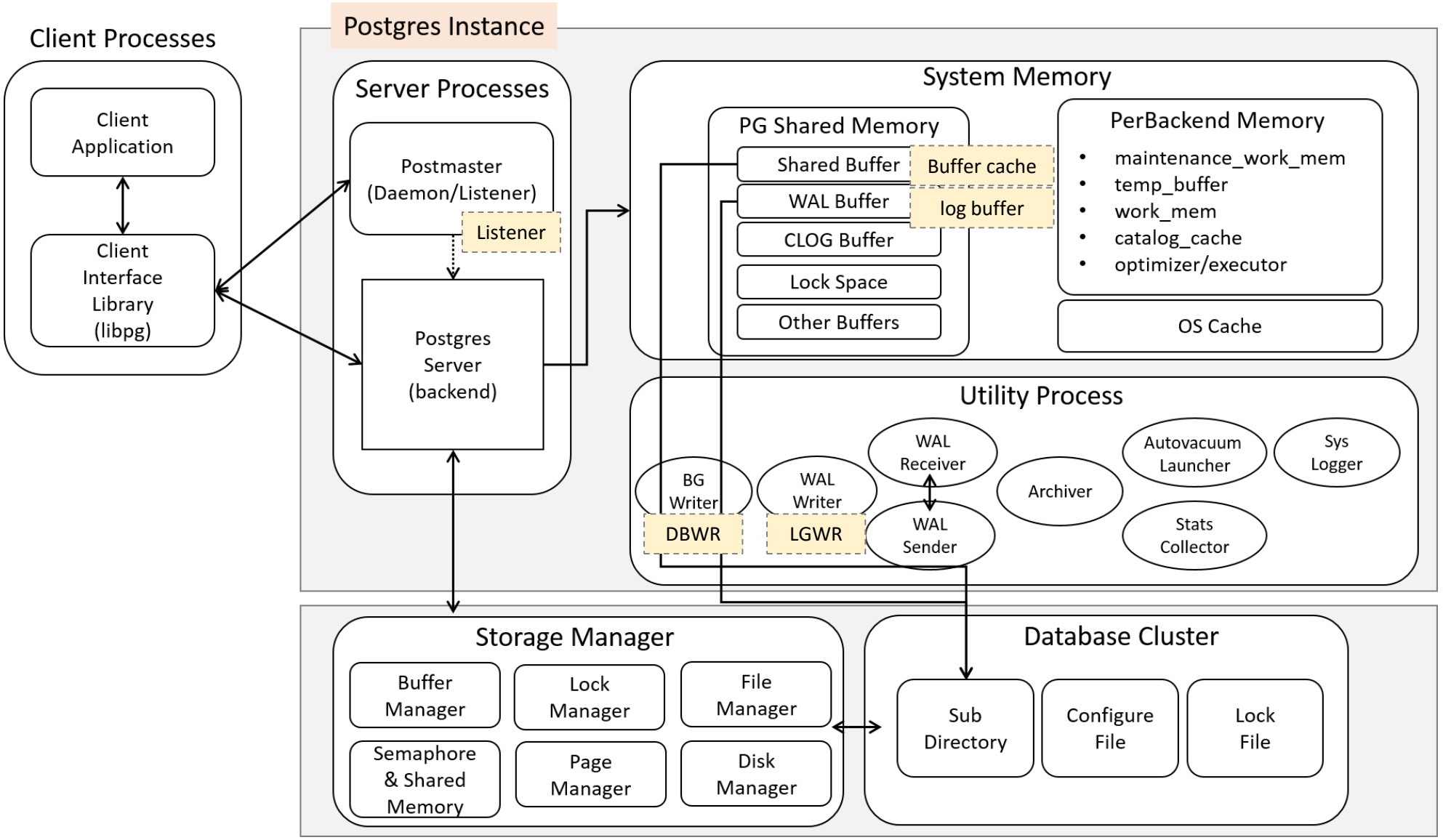
HigoDB (China)

출처 원저자 Andy Mendelsohn
how oracle rule db world, Rich Niemiec

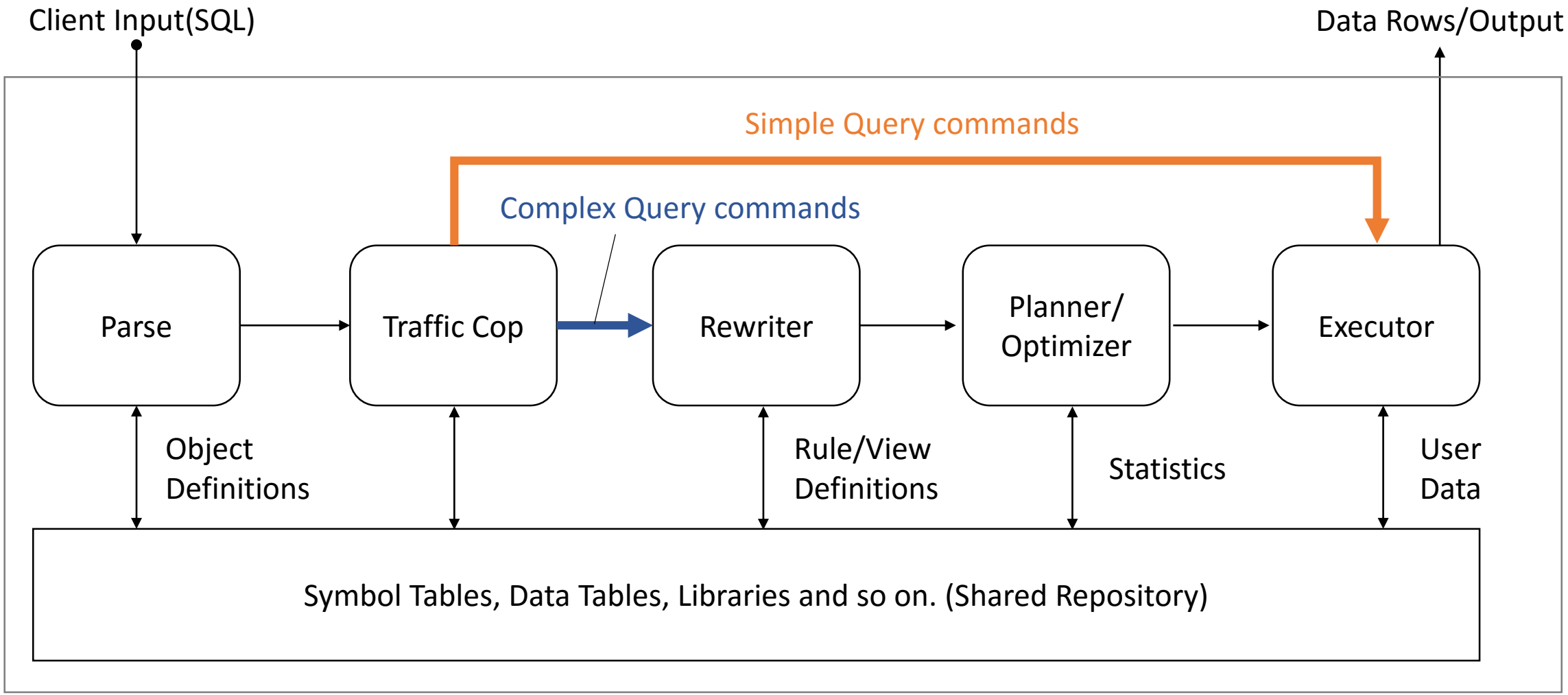
02. PostgreSQL Architecture

EXEM

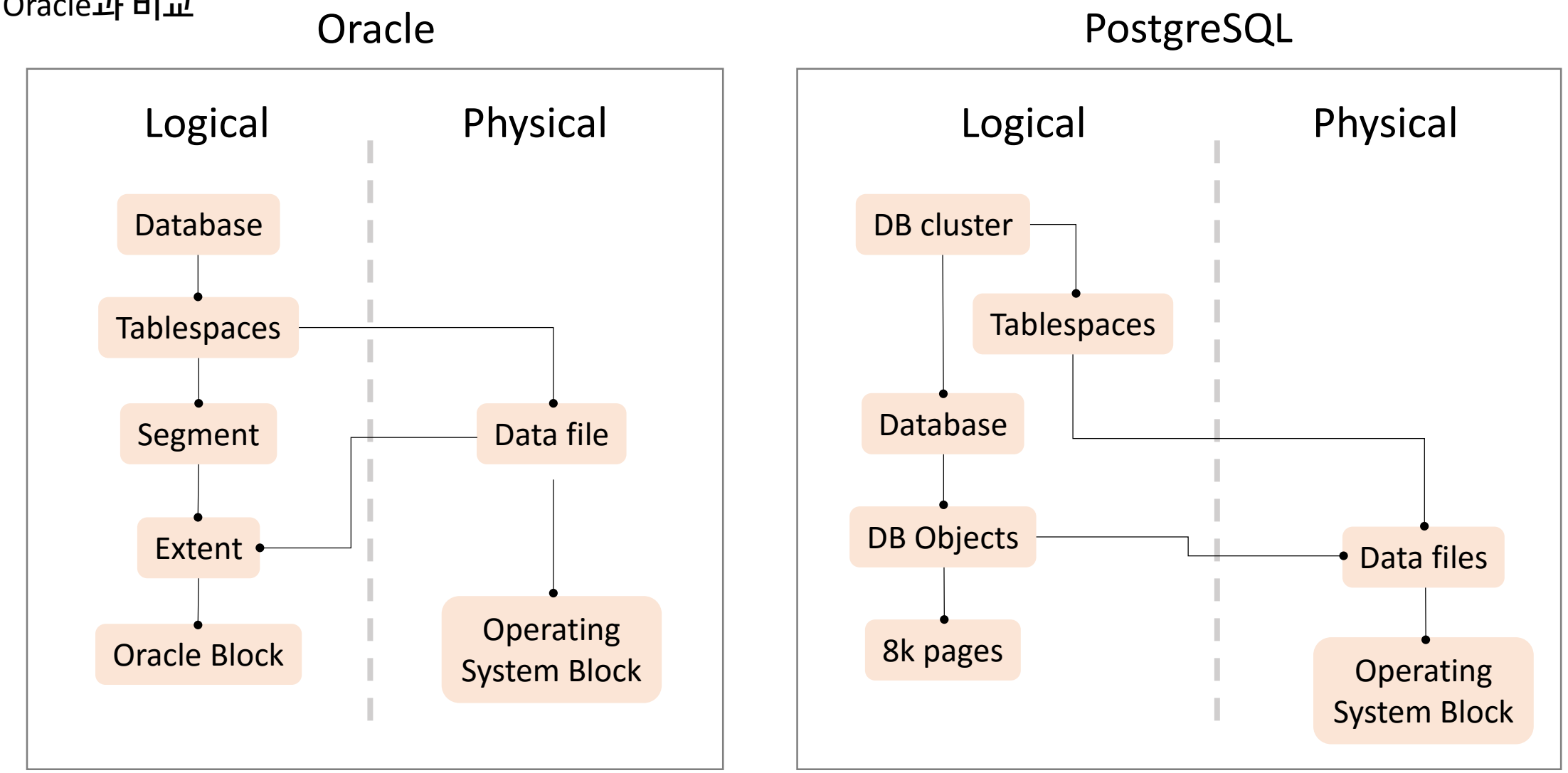
02. PostgreSQL Architecture



❖ Postgres server

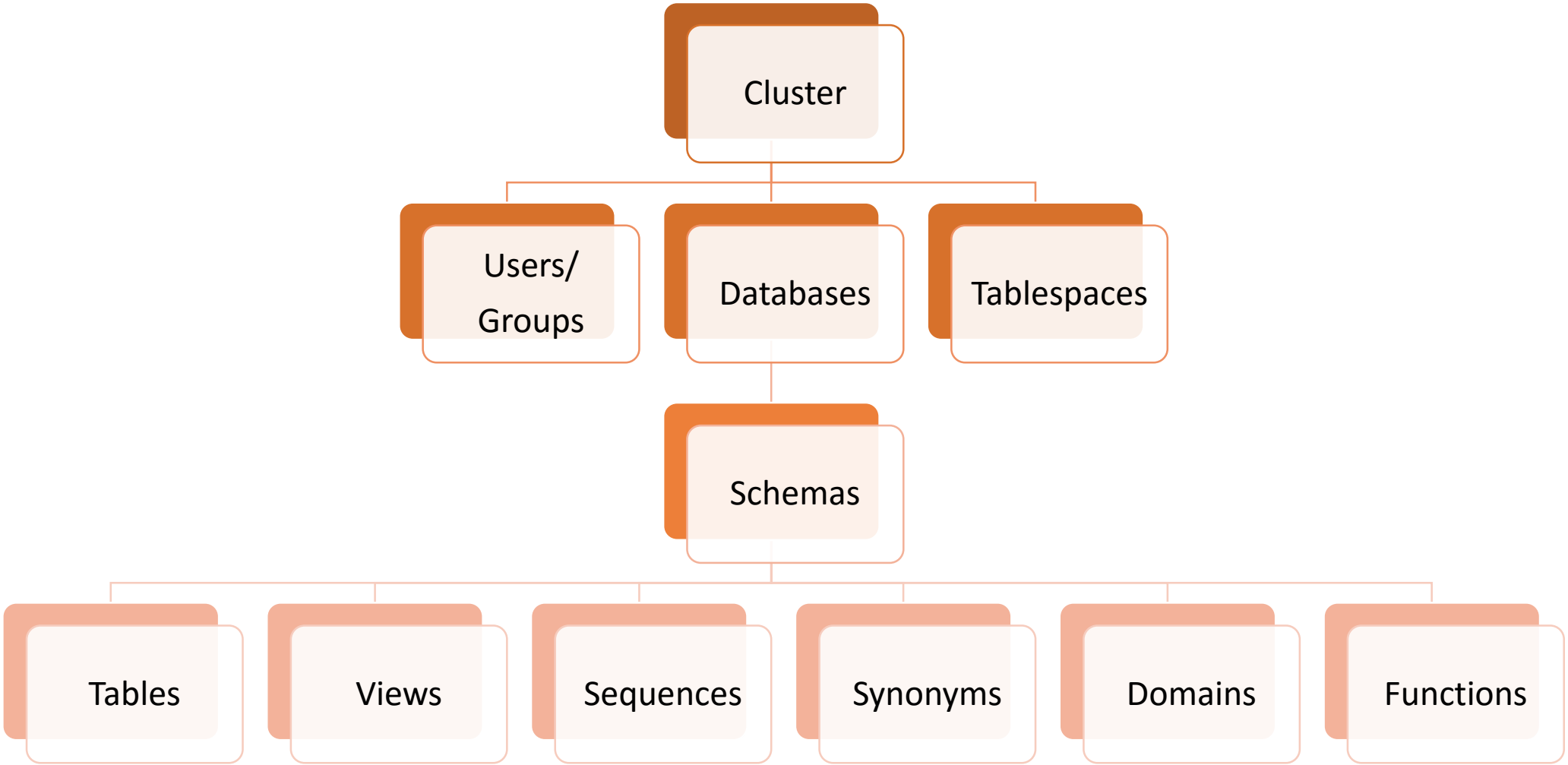


❖ Oracle과 비교

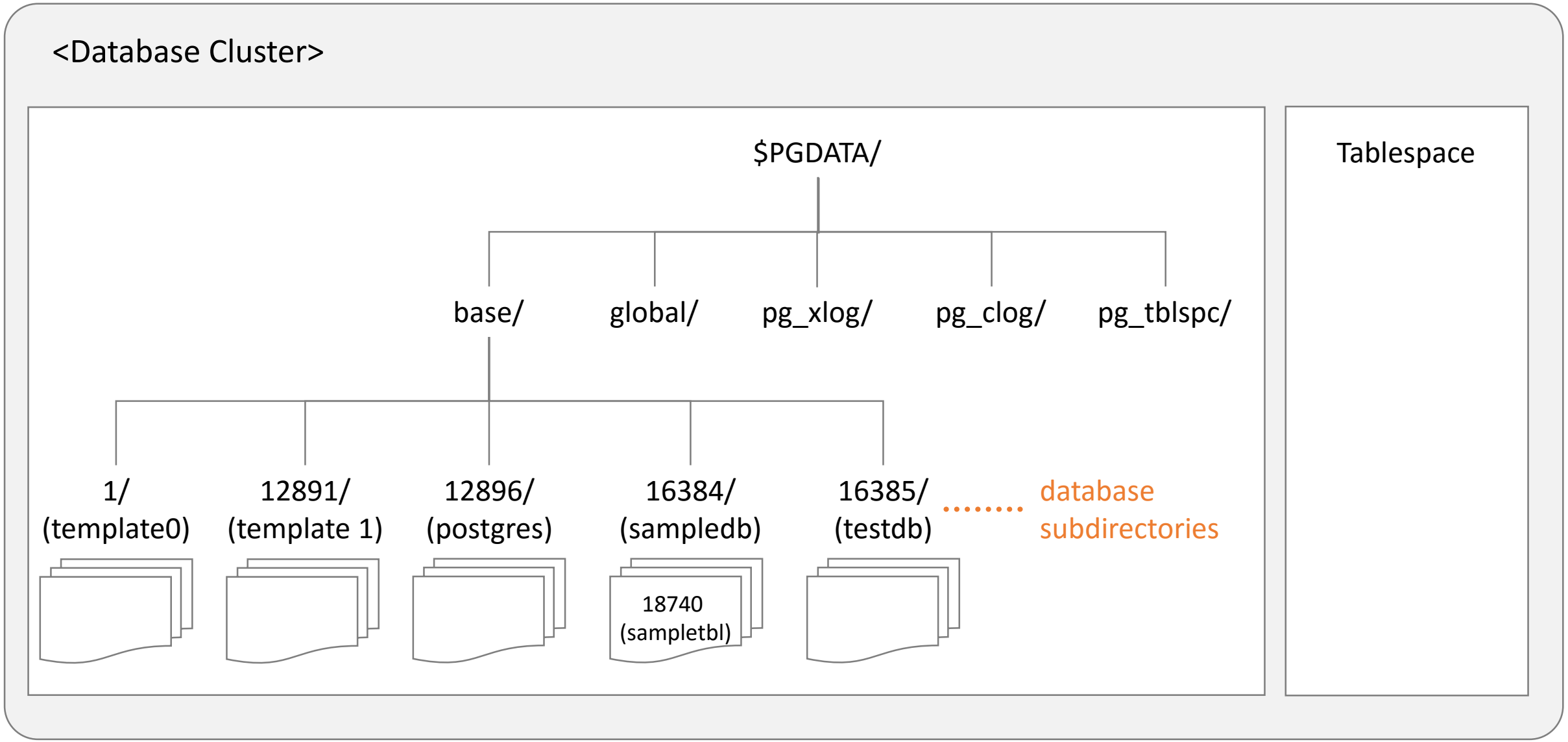


출처 http://www.p2d2.cz/files/postgres-for-oracle-dbas.pdf

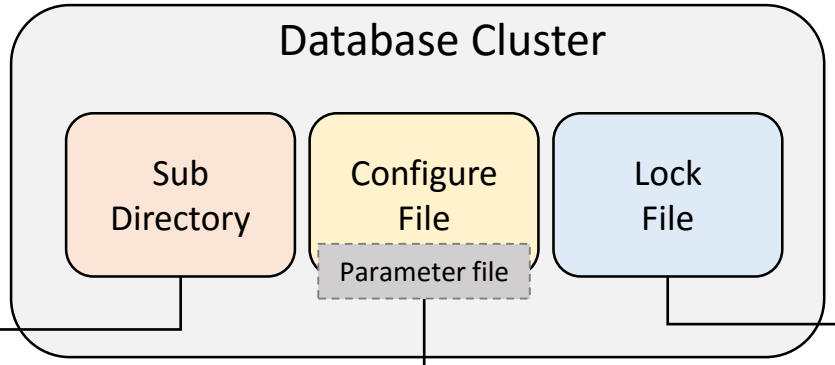
❖ Database 구조



출처 http://ktdsoss.tistory.com/403



❖ Database cluster



base	per database subdirectories global
global	cluster-wide information
pg_xlog	transaction logs (WAL files) Online Redo
pg_tblspc	symbolic link to tablespace location
pg_clog	transaction commit status data
pg_multixact	multitransaction status data
pg_notify	LISTEN/NOTIFY status data
pg_stat_tmp	temporary file for statistics subsystem
pg_subtrans	subtransaction data
pg_twophase	state files for prepared transactions

postgresql.conf	configuration parameter
pg_hba.conf	host based access file
pg_ident.conf	OS user mapping file

postmaster.pid	lock file with PID and shared memory segment
postmaster.opts	command line options the server was last started

03. PostgreSQL MVCC

EXEM

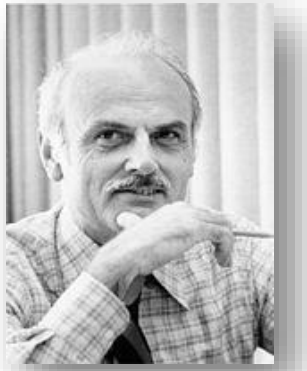
MVCC (Multi Version Concurrency Control)

Writing transactions does not prevent reading and vice versa, because each one sees its own version of the database.

출처 https://en.wikipedia.org/wiki/Firebird_(database_server)#The_Multi-Generational_Architecture_.28MGA.29

1970년 E. F. Codd 박사의 논문

- The Data Independence
- Set Processing
- High level language



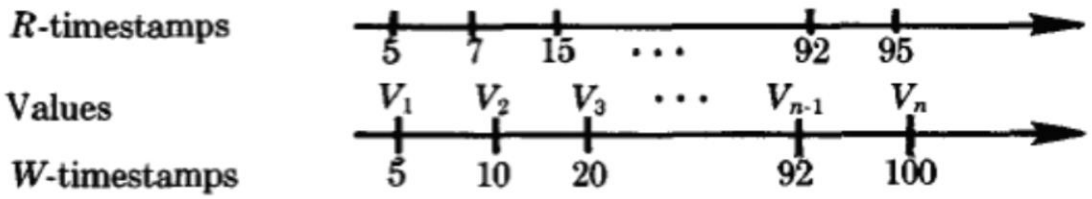
1978년 David Patric Reed 박사학위논문 MIT

Traditional approaches to the synchronization problems of shared data accessed by concurrently running computations have relied on mutual exclusion – the ability of one computation to stop the execution of other other computations that might access or change shared data accessed by that computation. Our approach is quite different. **We regard an object that is modifiable as a sequence of immutable versions. each version is the state of the object after an update is made to the object.**

1981년 PHILIP A. BERNSTEIN AND NATHAN GOODMAN 논문

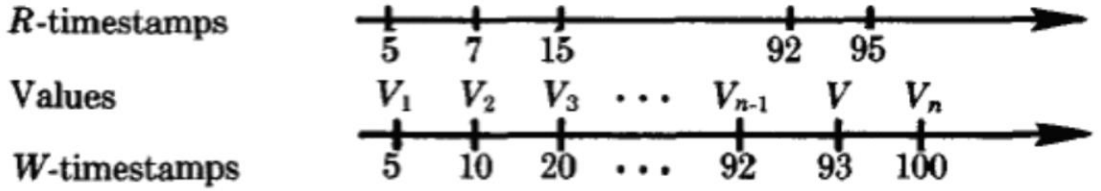
- Multiversion Reading and Writing

(b) Let us represent the R-timestamps of x similarly:



Let W be dm-write(x) with timestamp 93. Interval(W) = (93,100).

To process W we create a new version of x with that timestamp.



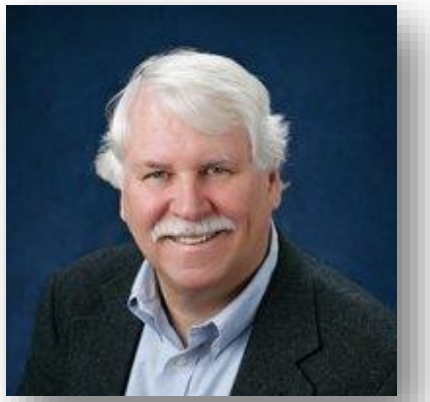
However, this new version "invalidates" the dm-read of part (a), because if the dm-read had arrived after the dm-write, it would have read value V instead of V_{n-1}. Therefore, we must reject the dm-write.

Figure 11. Multiversion reading and writing.

출처 - *Bernstein, Philip A.; Goodman, Nathan (1981). "Concurrency Control in Distributed Database Systems". ACM Computing Surveys.*

1984년 Jim Starkey

- The first shipping, commercial database software product featuring MVCC was Digital's VAX Rdb/ELN. The second was InterBase, both of which are still active, commercial products.
- Firebird – Multi Generation Architecture (추기형 아키텍처)



1986년 Bob Miner

- 1986년 Oracle version 6 에서 Rollback Segment 도입

출처 – https://en.wikipedia.org/wiki/Multiversion_concurrency_control

1999년 Mikheev, Vadim B.

- 1999년 Vadim 이 PostgreSQL 6.5 에 MVCC 아키텍쳐를 도입함

- Thomas Lockhart
- Jolly Chen
- Vadim Mikheev
- Jan Wieck
- Andrew Yu
- Tom Lane
- Bruce Momjian
- Marc Fournier

출처 get to know postgreSQL , Bruce Momjian

❖ MVCC에 대한 두 가지 접근법

접근법 1

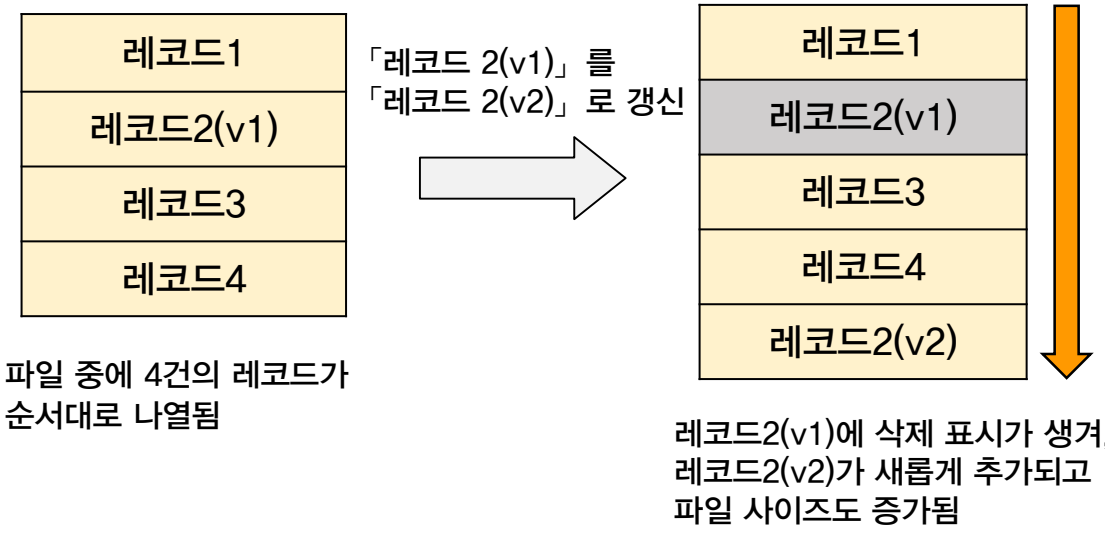
- PostgreSQL, Firebird/Interbase, SQL Server 해당
- 데이터베이스에 다중 버전의 레코드를 저장
- 더 이상 필요하지 않을 때 모아둔 레코드를 버림

접근법 2

- Oracle, MySQL/InnoDB 해당
- 최신 버전의 데이터만 데이터베이스 내에 저장
- 언두를 이용하여 이전 버전의 데이터를 재구성

1. First Approach

- 레코드 갱신처리 (UPDATE)

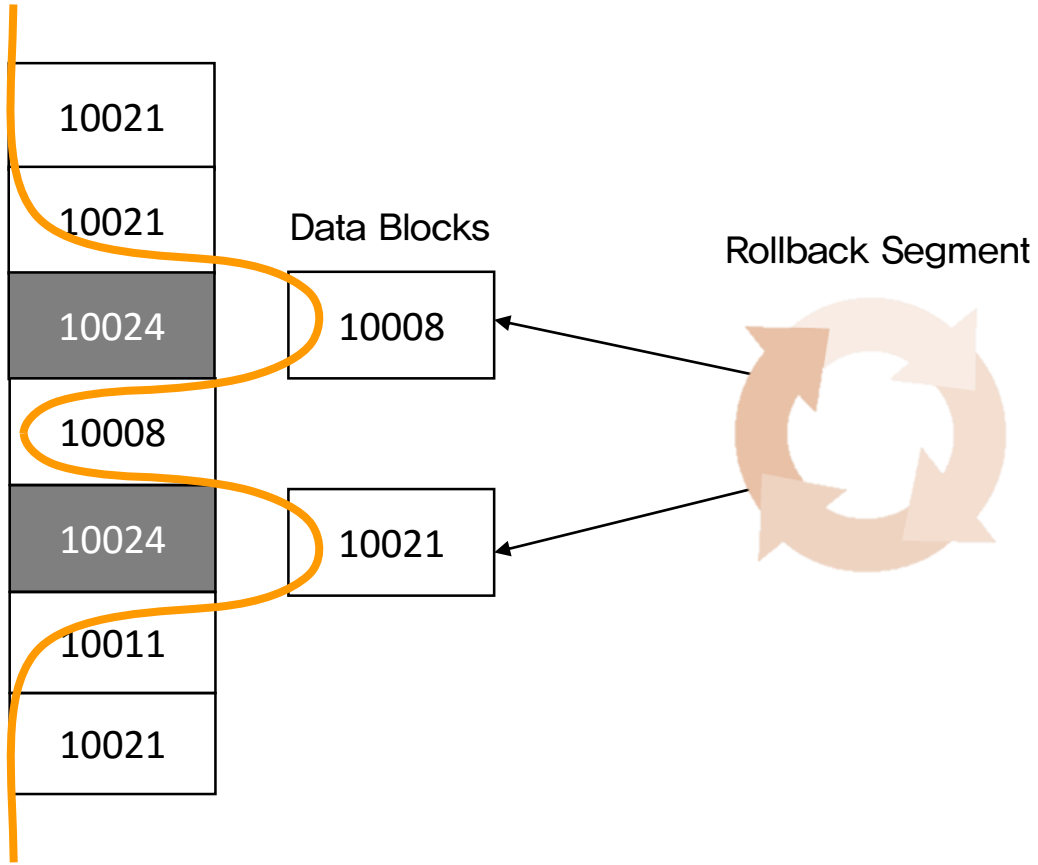


파일 중에 4건의 레코드가
순서대로 나열됨

레코드2(v1)에 삭제 표시가 생겨,
레코드2(v2)가 새롭게 추가되고
파일 사이즈도 증가됨

2. Second Approach

SELECT
(SCN 10023)

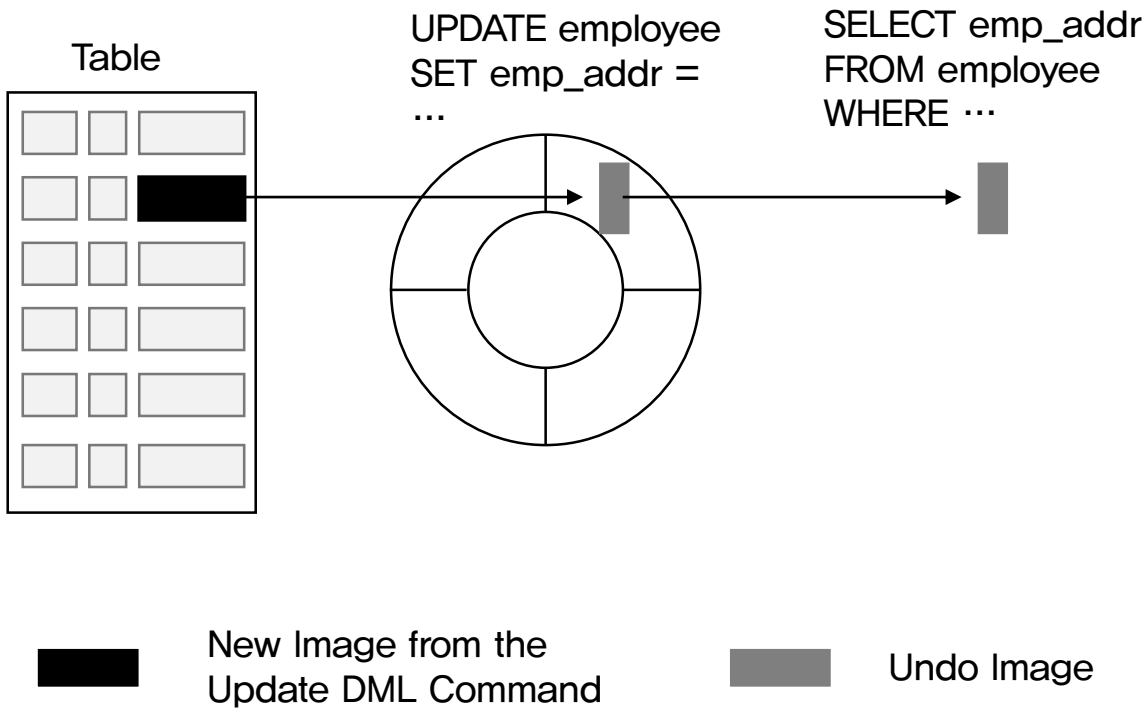


❖ PostgreSQL VS Oracle VS SQL Server

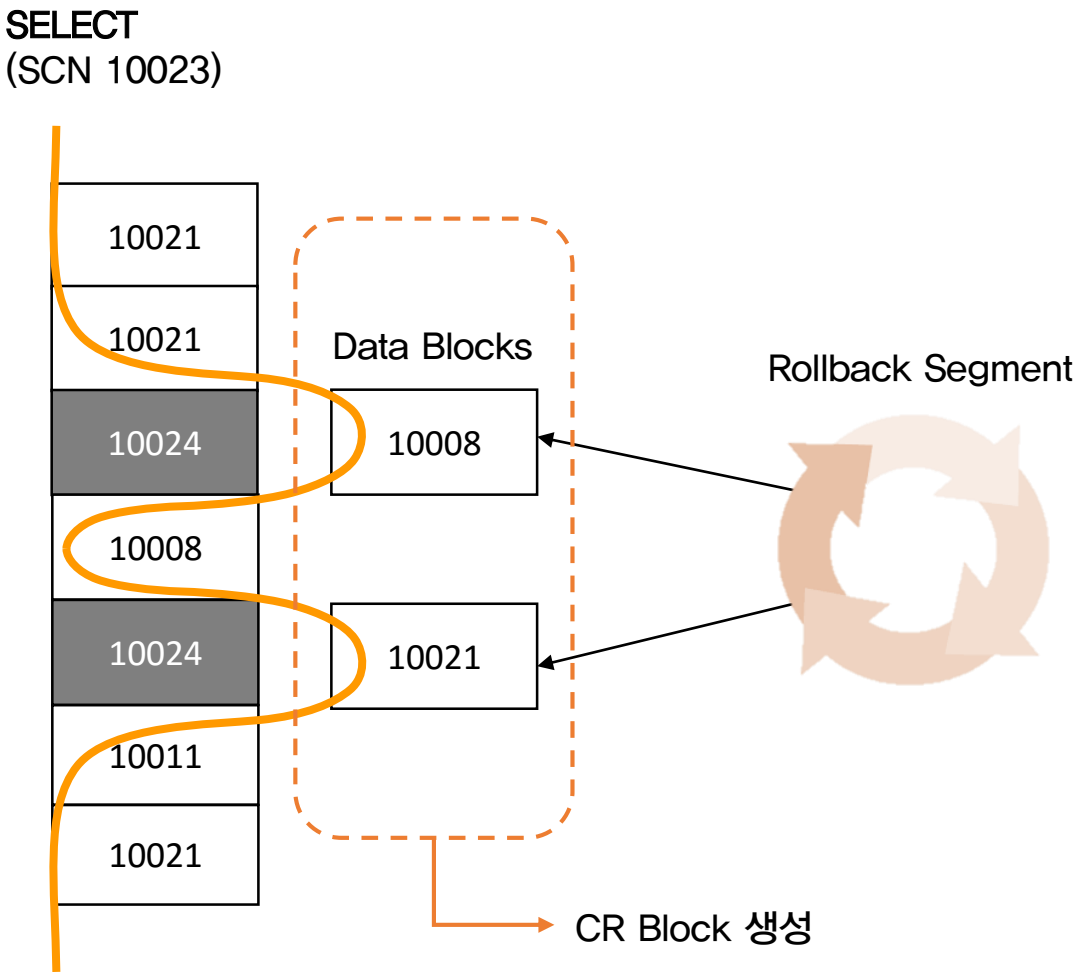
Database	PostgreSQL	Oracle	SQL Server
Storage for Old Versions	In the main segment (Heap/Index)	In the separate segment (Rollback Segment/Undo)	In the separate database (tempdb – known as version store)
Size of Tuple Header (bytes)	24	3	Fixed – 4 Variable – 14
Clean up	Vacuum	System Monitor Process (SMON)	Ghost Cleanup task

❖ Oracle

Read Consistency

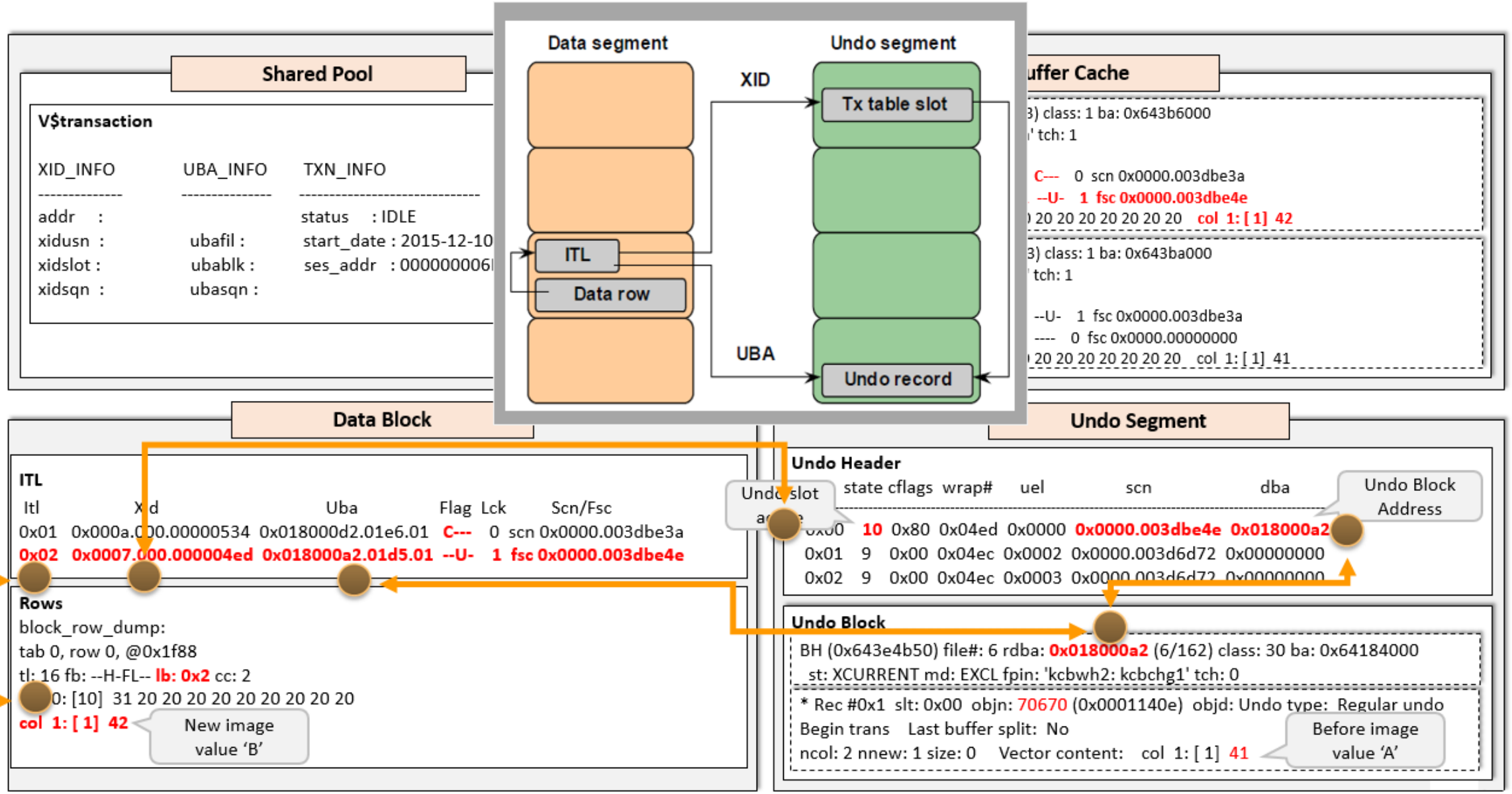


출처 http://www.siue.edu/~dbock/cmis565/module10-undo_data.htm



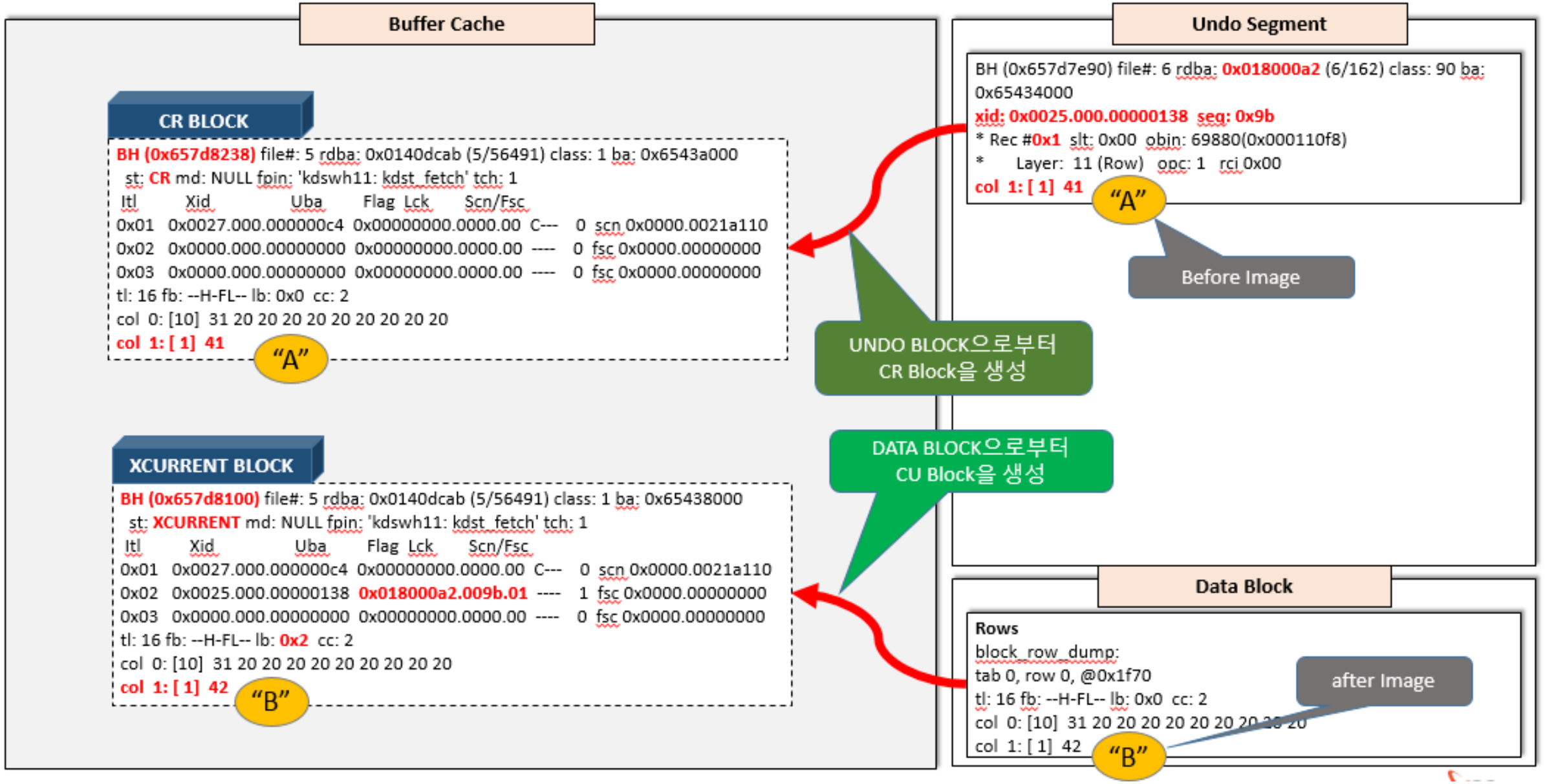
출처 https://riddle.ru/mirrors/oracledocs/server/scn73/ch1001.html

03. PostgreSQL MVCC – 2) MVCC에 대한 두 가지 접근법



EXEM

03. PostgreSQL MVCC - 2) MVCC에 대한 두 가지 접근법



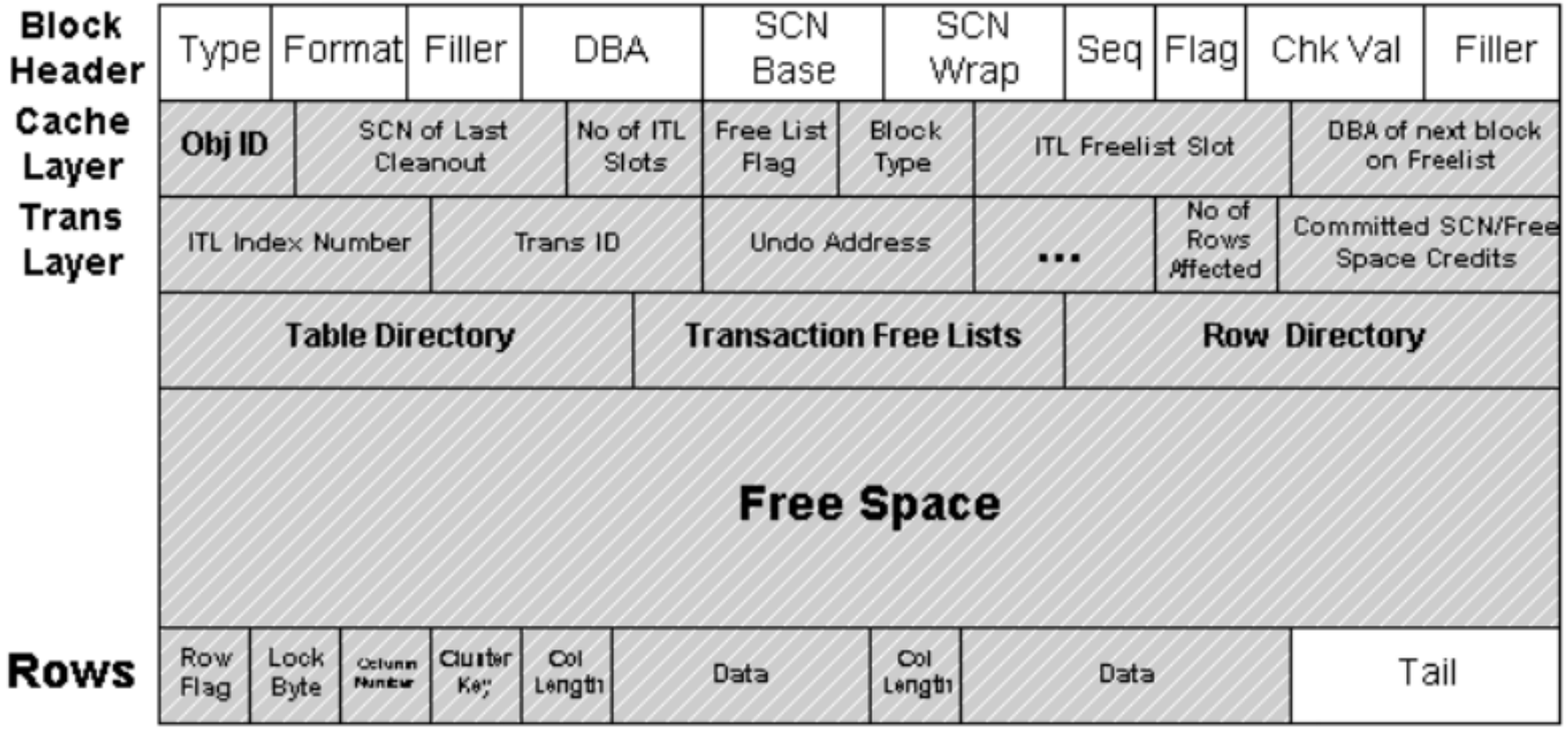
EXEM

03. PostgreSQL MVCC - 2) MVCC에 대한 두 가지 접근법

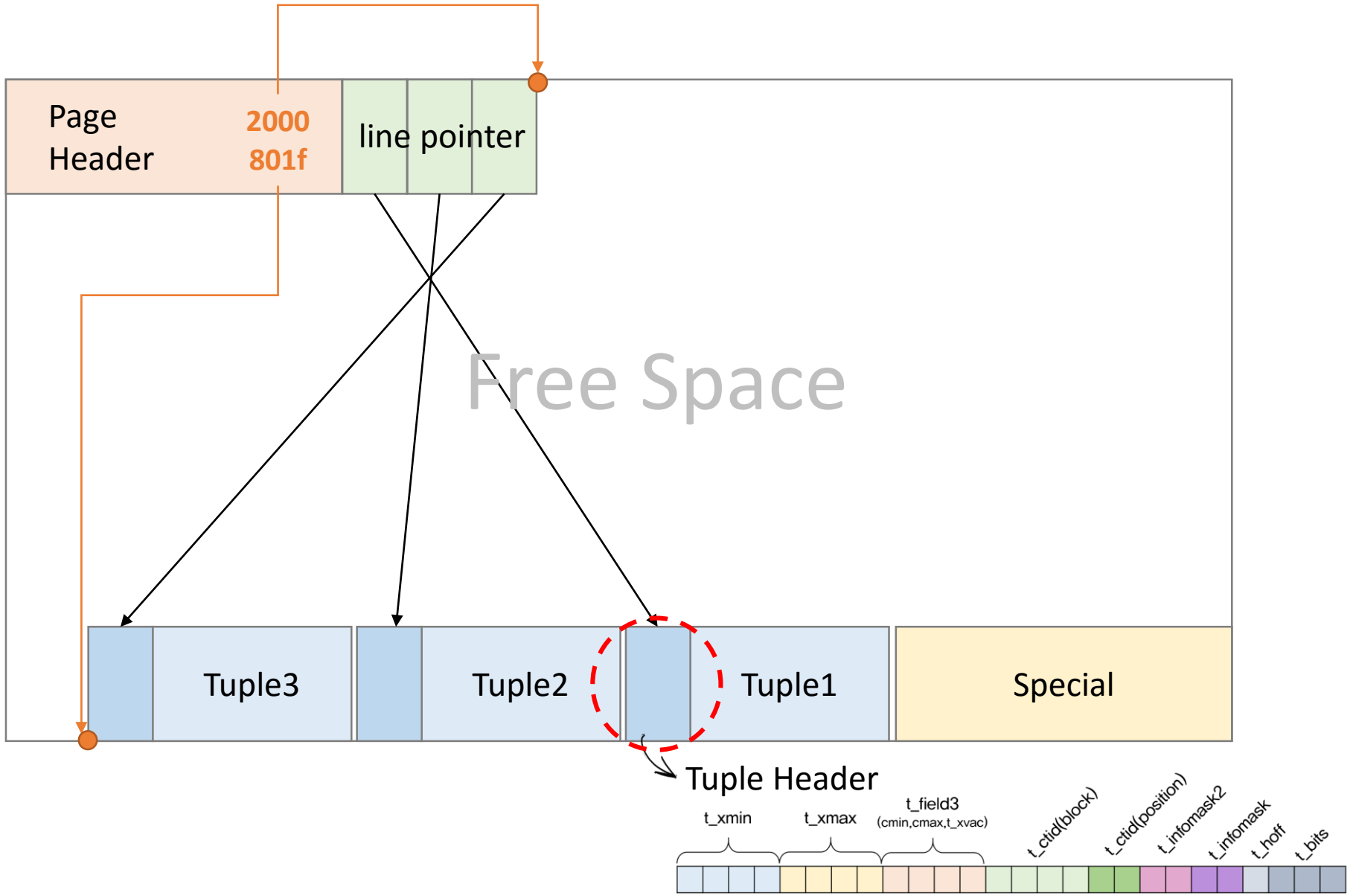
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Typ	Fmt	Filler		RDBA				SCNBase				SCN Wrap		Seq	Flg
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ChkVal		Filler		Ktbbhtyp				Ktbbhdsid,Obj id				csc			
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
ktbbhcsc				itc		flg	fsl	Bdba(next dba)				First Itl			
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
(xid + uba + scn/fsc)															
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
				Second itl (xid + uba + scn/fsc)											
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
xid												filler			
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
filelr				Flg	ntab	nrow		Frre=-1		fsbo		fseo		avsp	
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
tosp		offs		nrows		Offs 1		Offs 2							

8144	8145	8146	8147	8148	8149	8150	8151	8152	8153	8154	8155	8156	8157	8158	8159
8160	8161	8162	8163	8164	8165	8166	8167	8168	8169	8170	8171	8172	8173	8174	8175
8176	8177	8178	8179	8180	8181	8182	8183	8184	8185	8186	8187	8188	8189	8190	8191
												tailchk			

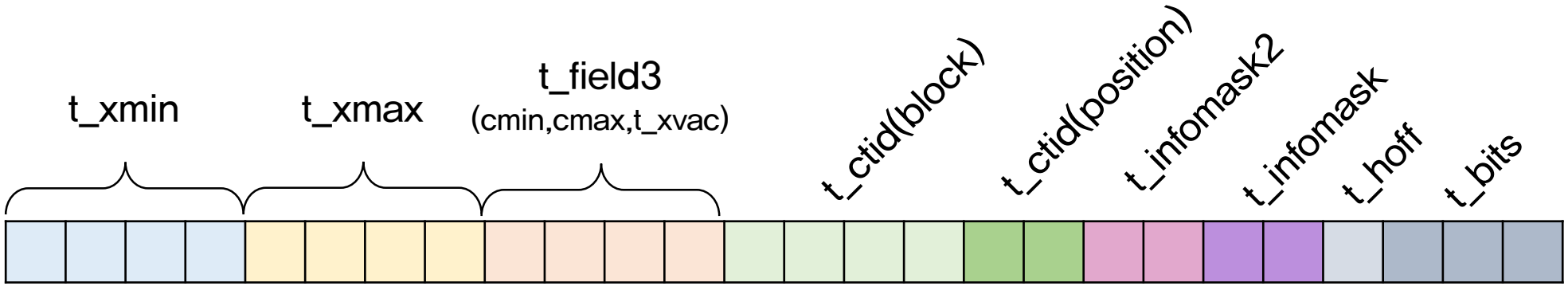
Block Layout



03. PostgreSQL MVCC – 2) MVCC에 대한 두 가지 접근법



EXEM



- t_xmin: insert 시의 XID
- t_xmax: delete 시의 XID
- t_field3: t_cid, t_xvac 로 구성
- t_cid: insert, delete command id (cmin, cmax)
- t_xvac: 이전 full vacuum의 트랜잭션 ID를 추적 가능
- t_ctid: 해당 튜플의 현재 버전을 가리키는 포인터

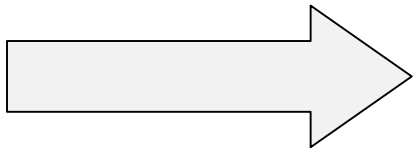
  block id data(4bytes) + offset number (2bytes)
- t_infomask2: 앞의 2자리는 변경 커맨드 정보, 뒤의 2자리는 컬럼 수
- t_infomask: commit 여부
- t_hoff: 튜플 헤더의 길이, 유저 데이터의 시작 주소
- t_bits: null 값의 variable length bitmap

❖ PostgreSQL – 레코드 갱신처리 (UPDATE)

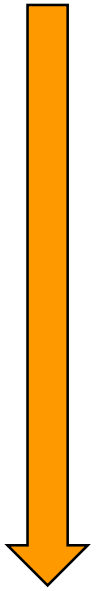
xmin	레코드1
xmin	레코드2 (v1)
xmin	레코드3
xmin	레코드4

파일 중에 4건의 레코드가 순서대로 나열됨

「레코드 2(v1)」를
「레코드 2(v2)」로 갱신



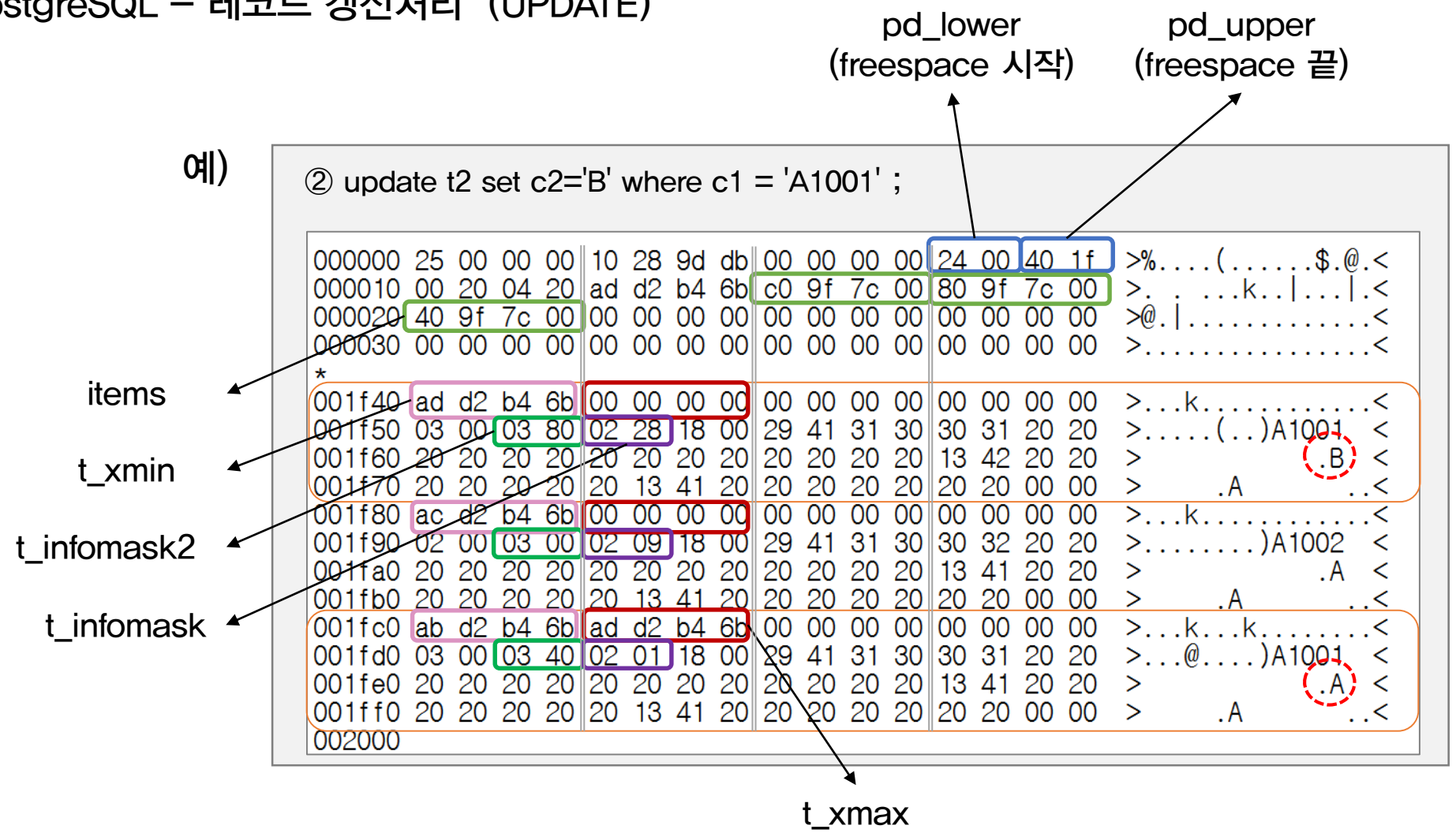
xmin	레코드1
xmax, infomask2, infomask	레코드2(v1)
xmin	레코드3
xmin	레코드4
xmin, infomask2, infomask	레코드2(v2)



오라클처럼 replace되는 것이 아니라
레코드2(v1)에 삭제 표시(xmax)가 생겨,
레코드2(v2)가 새롭게 추가되고 파일 사이즈도 증가됨

출처: http://pgsqldeepdive.blogspot.kr/2012/12/postgresql_16.html

❖ PostgreSQL – 레코드 갱신처리 (UPDATE)



- SCN은 커밋 시마다 순차적으로 증가하며, 6bytes 사용됨
- 초당 1만 번의 커밋이 발생한다고 하더라도 대략, 800년 이상 (2^{6*8} / (10,000 * 86,400 * 365))이 지나야 오버플로우 발생

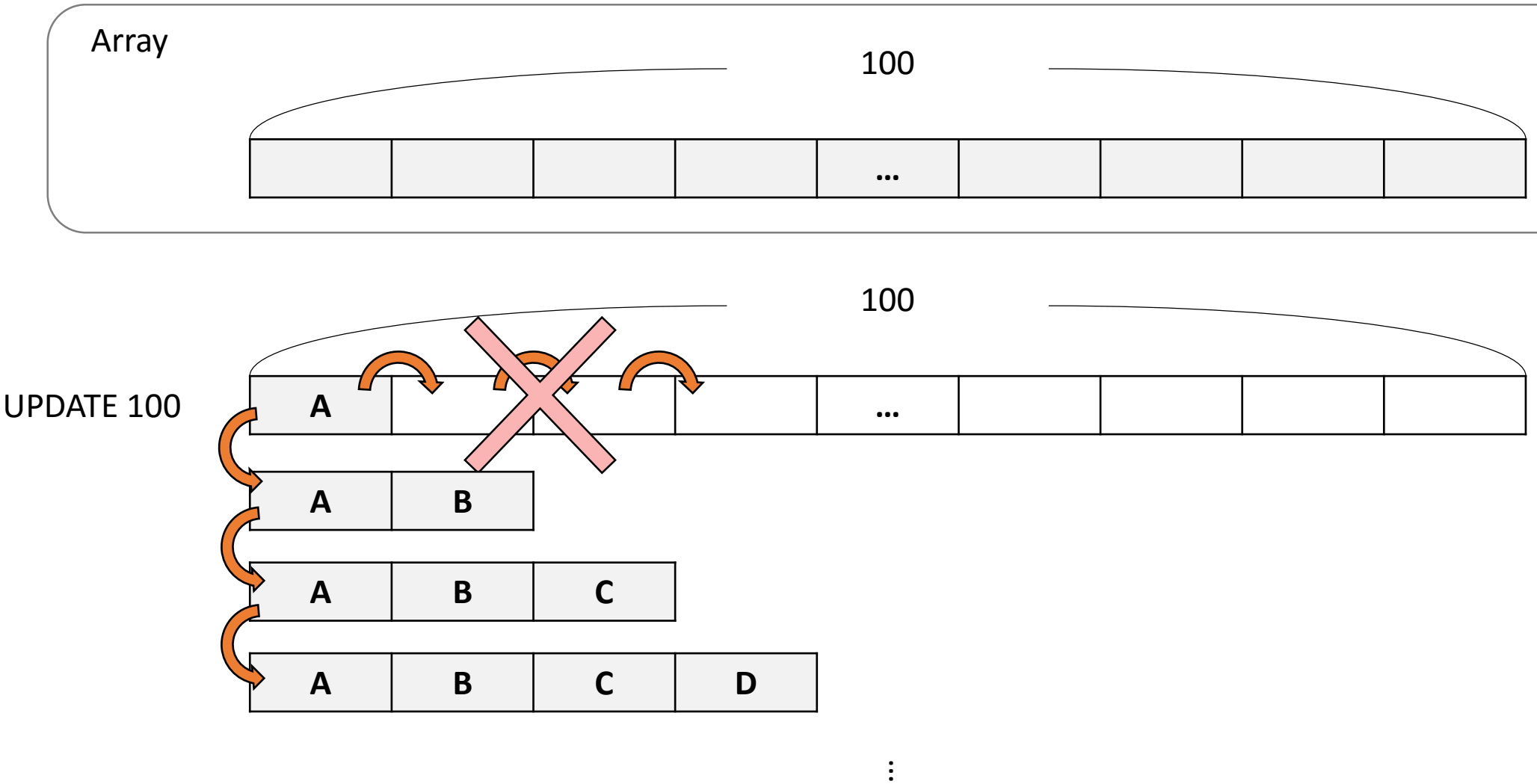
김시연, 최영준 『Transaction Internals in Oracle 10g R2』, 엑셈(2008)

ORACLE®

PostgreSQL

- 트랜잭션 ID는 32bit 정수형 크기
- 하나의 클러스터 기준으로 관리되기 때문에, 서버가 40억(2^{32}) 트랜잭션을 넘게 사용했다면 *트랜잭션 ID 겹침* 오류를 발생

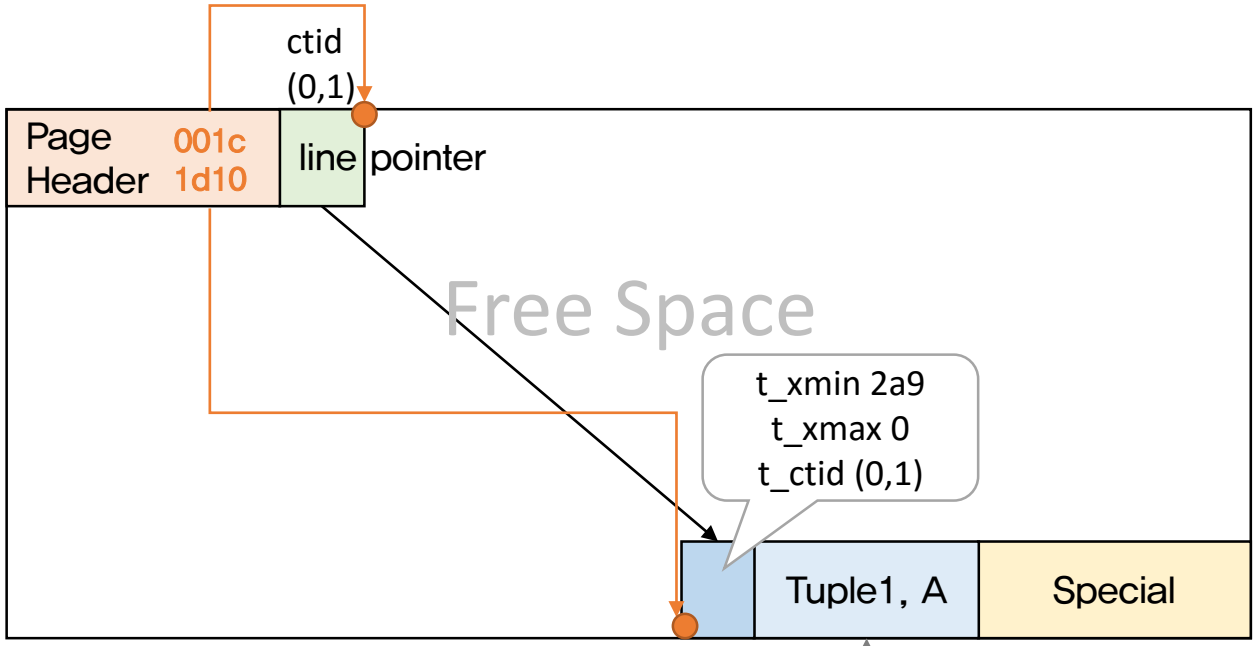
출처 http://postgresql.kr/docs/9.3/routine-vacuuming.html



- Insert (Tuple1, A)

세션0

① Begin;
Insert (Tuple1, A)
Commit;



세션1

② Read Only 후, Select

t_xmin 2a9
t_xmax 0
ctid (0,1)
Tuple 1, A

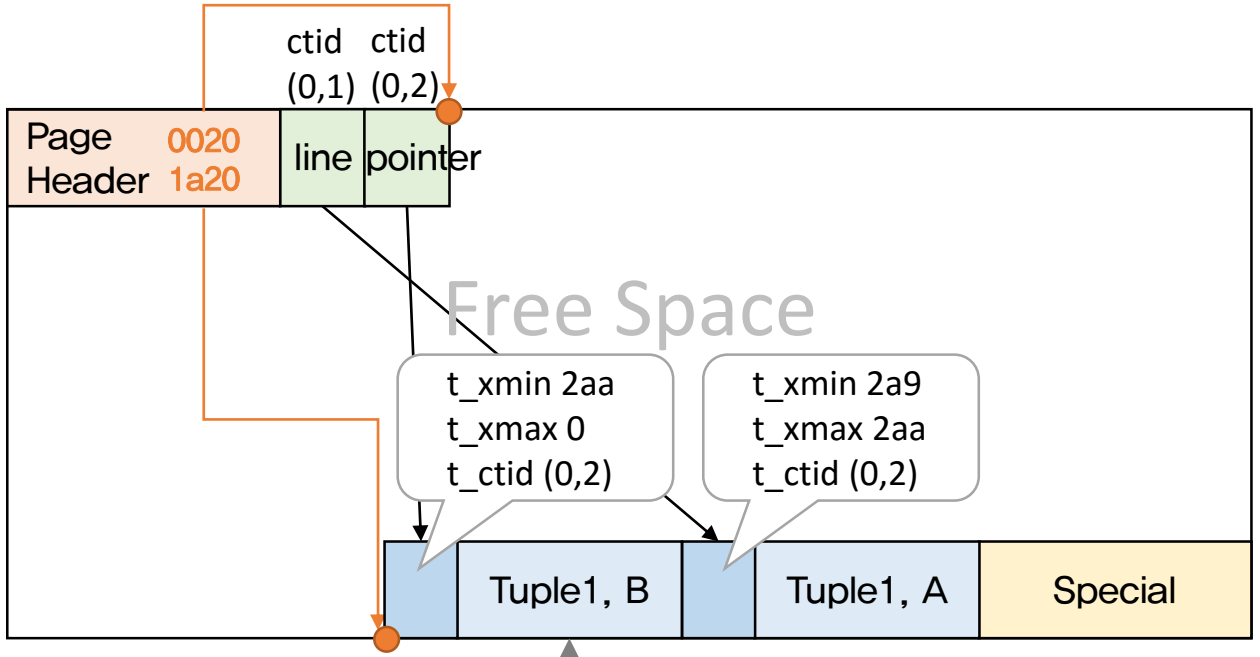
세션2

세션3

- Update (Tuple1, B)

세션0

① Begin;
Update (Tuple1, B)
Commit;



세션1

② Select

t_xmin 2a9
t_xmax 2aa
ctid (0,1)
Tuple 1, A

세션2

③ Read Only 후, Select

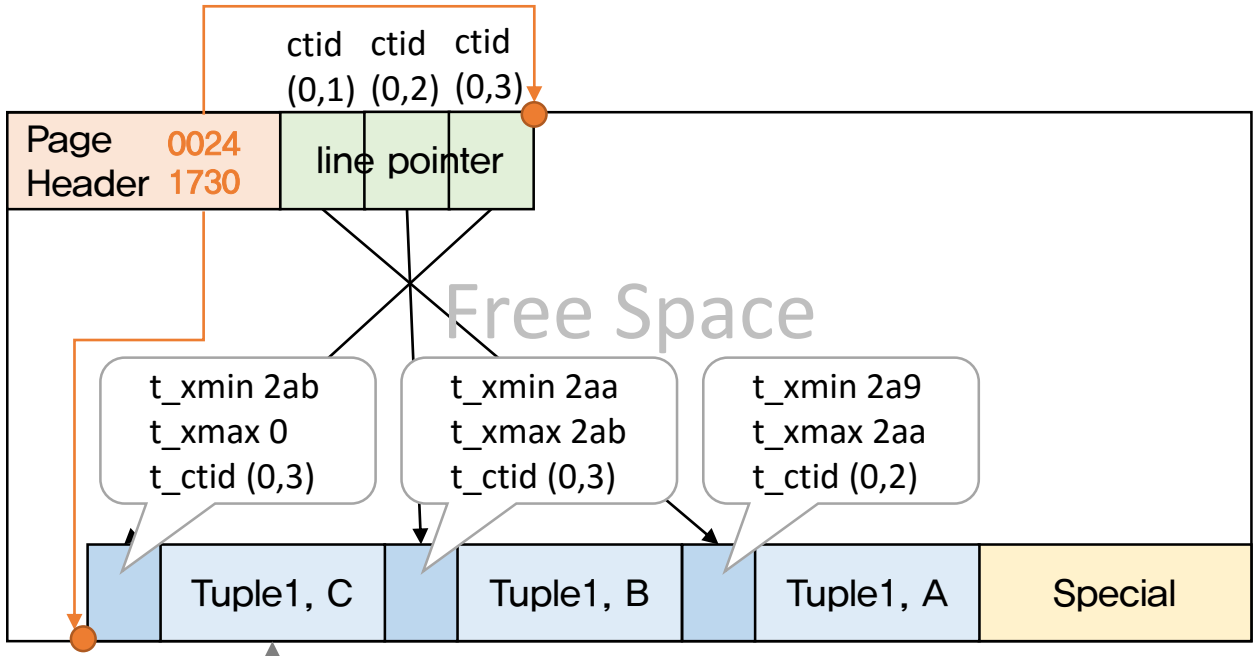
t_xmin 2aa
t_xmax 0
ctid (0,2)
Tuple 1, B

세션3

- Update (Tuple1, C)

세션0

① Begin;
Update (Tuple1, C)
Commit;



세션1

② Select

t_xmin 2a9
t_xmax 2aa
ctid (0,1)
Tuple 1, A

세션2

③ Select

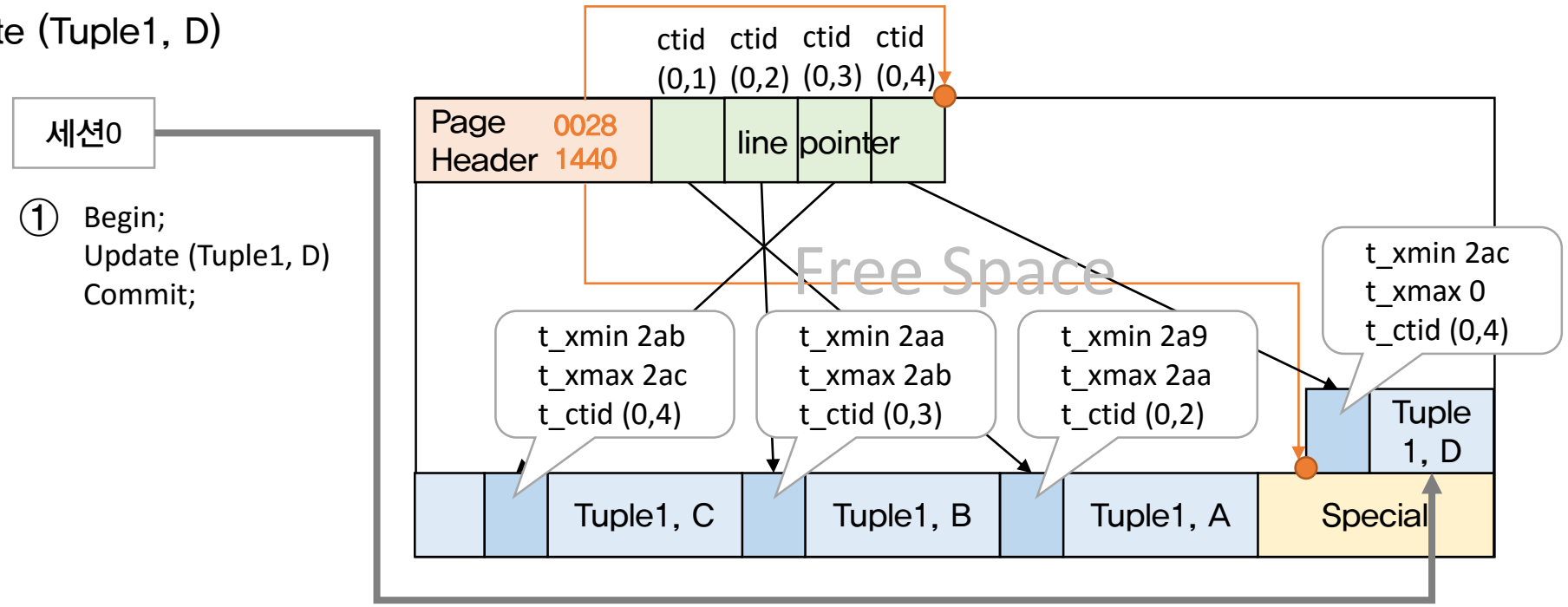
t_xmin 2aa
t_xmax 2ab
ctid (0,2)
Tuple 1, B

세션3

④ Read Only 후, Select

t_xmin 2ab
t_xmax 0
ctid (0,3)
Tuple 1, C

- Update (Tuple1, D)



세션1

② Select

t_xmin 2a9
t_xmax 2aa
ctid (0,1)
Tuple 1, A

세션2

③ Select

t_xmin 2aa
t_xmax 2ab
ctid (0,2)
Tuple 1, B

세션3

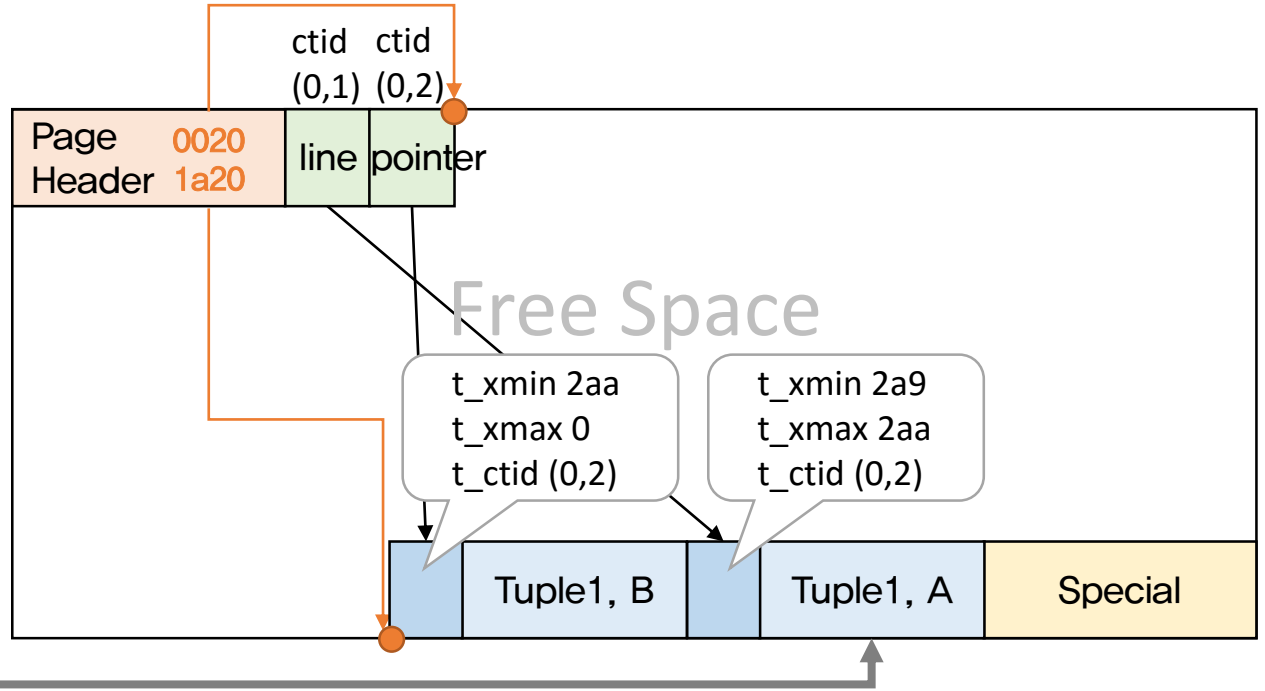
④ Select

t_xmin 2ab
t_xmax 2ac
ctid (0,3)
Tuple 1, C

- Update (Tuple1, B)

세션0

① Begin;
Update (Tuple1, B)
Commit은 안함



세션1

② Select

t_xmin 2a9
t_xmax 2aa
ctid (0,1)
Tuple 1, A

세션2

③ Update (Tuple1, C)

Lock

- Oracle
  - 시간 정보 “SCN”과 공간 정보 “XID”를 복합적으로 활용하여 트랜잭션을 처리함
  - 이전 이미지가 저장되어 있는 언두 레코드의 주소를 가리키는 “uba”
  - Commit SCN은 트랜잭션이 완료될 때 부여됨
- PostgreSQL
  - 다음 이미지를 가리키는 “t_ctid”
  - XID가 트랜잭션이 시작할 때 붙여지기 때문에 “Snapshot”이 필요
- MySQL
  - 이전 이미지를 가리키는 “roll pointer”
  - XID가 트랜잭션이 시작할 때 붙여지기 때문에 “Readview”가 필요

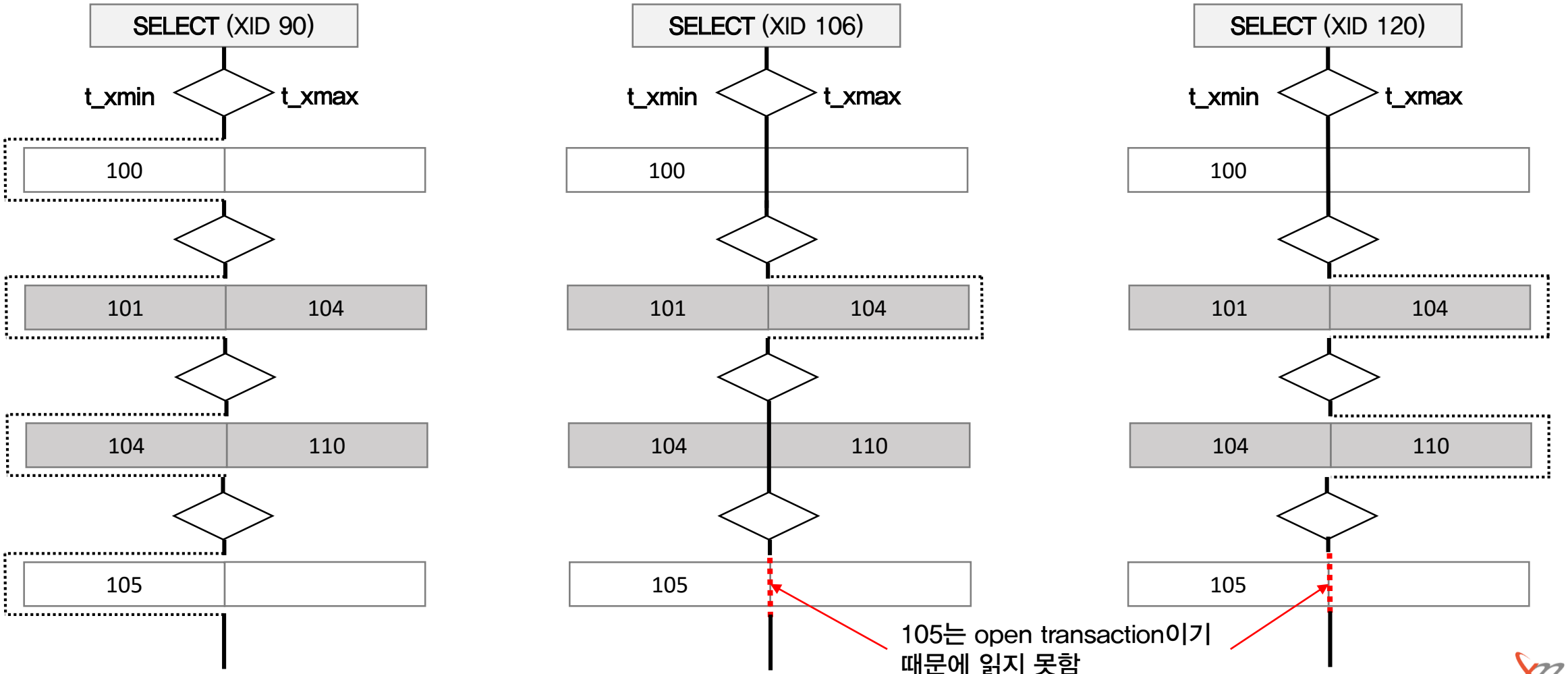
- Snapshot structure

```
typedef struct SnapshotData
{
    TransactionId xmin;          /* XID < xmin are visible to me */
    TransactionId xmax;          /* XID >= xmax are invisible to me */
    uint32        xcnt;          /* # of xact below */
    TransactionId *xip;          /* array of xacts in progress */
    ItemPointerData tid;         /* required for Dirty snapshot -:( */
} SnapshotData;
```

출처 - https://www.postgresql.org/files/developer/internalpics.pdf

- 시나리오

  Transaction Counter at query start: **90, 106, 120**
  Open Transaction: **105**



04. Page Layout

EXEM

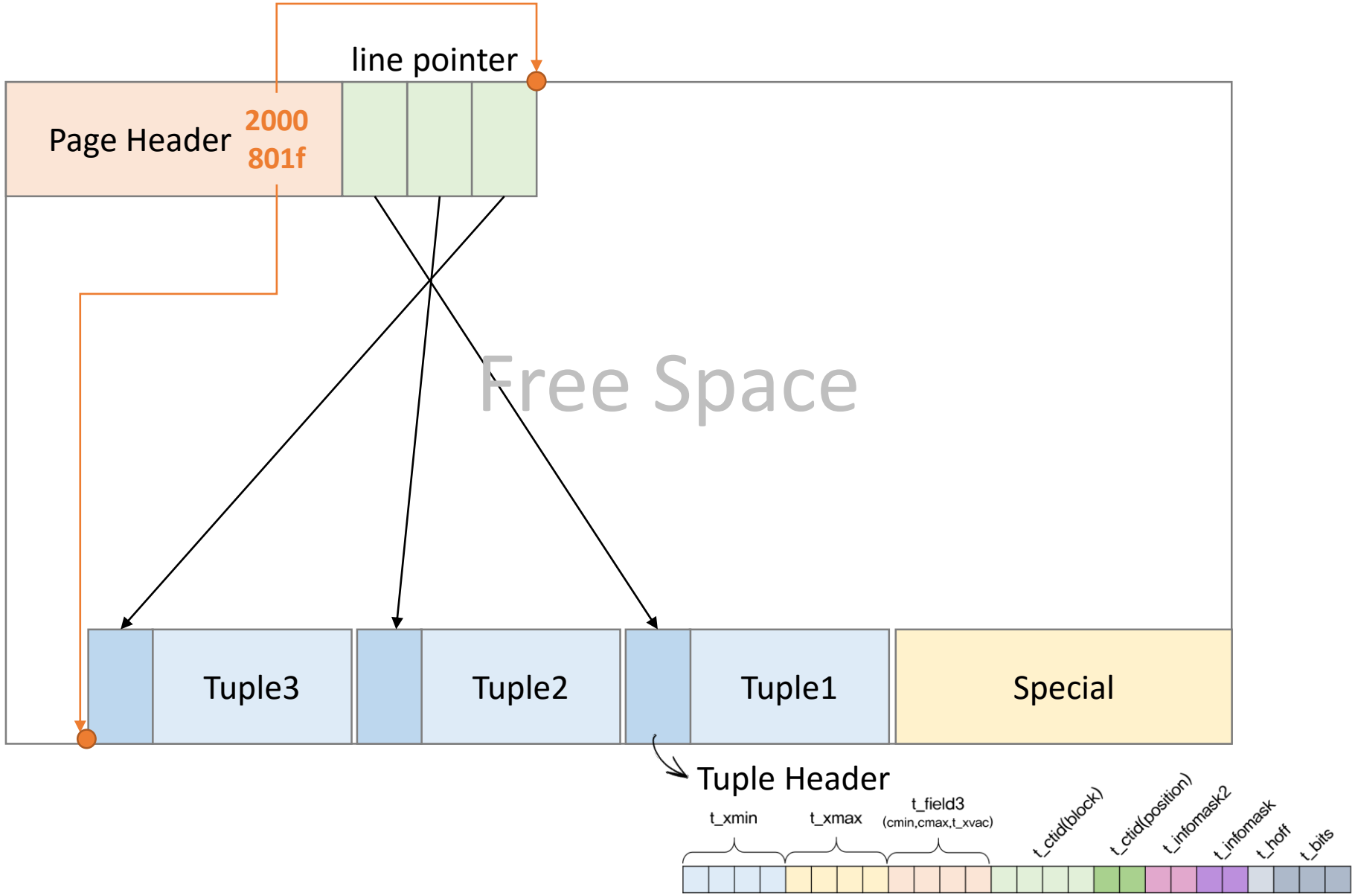


Table 63-2. Overall Page Layout

Item	Description
PageHeaderData	24 bytes long. Contains general information about the page, including free space pointers.
ItemIdData	Array of (offset,length) pairs pointing to the actual items. 4 bytes per item.
Free space	The unallocated space. New item pointers are allocated from the start of this area, new items from the end.
Items	The actual items themselves.
Special space	Index access method specific data. Different methods store different data. Empty in ordinary tables.

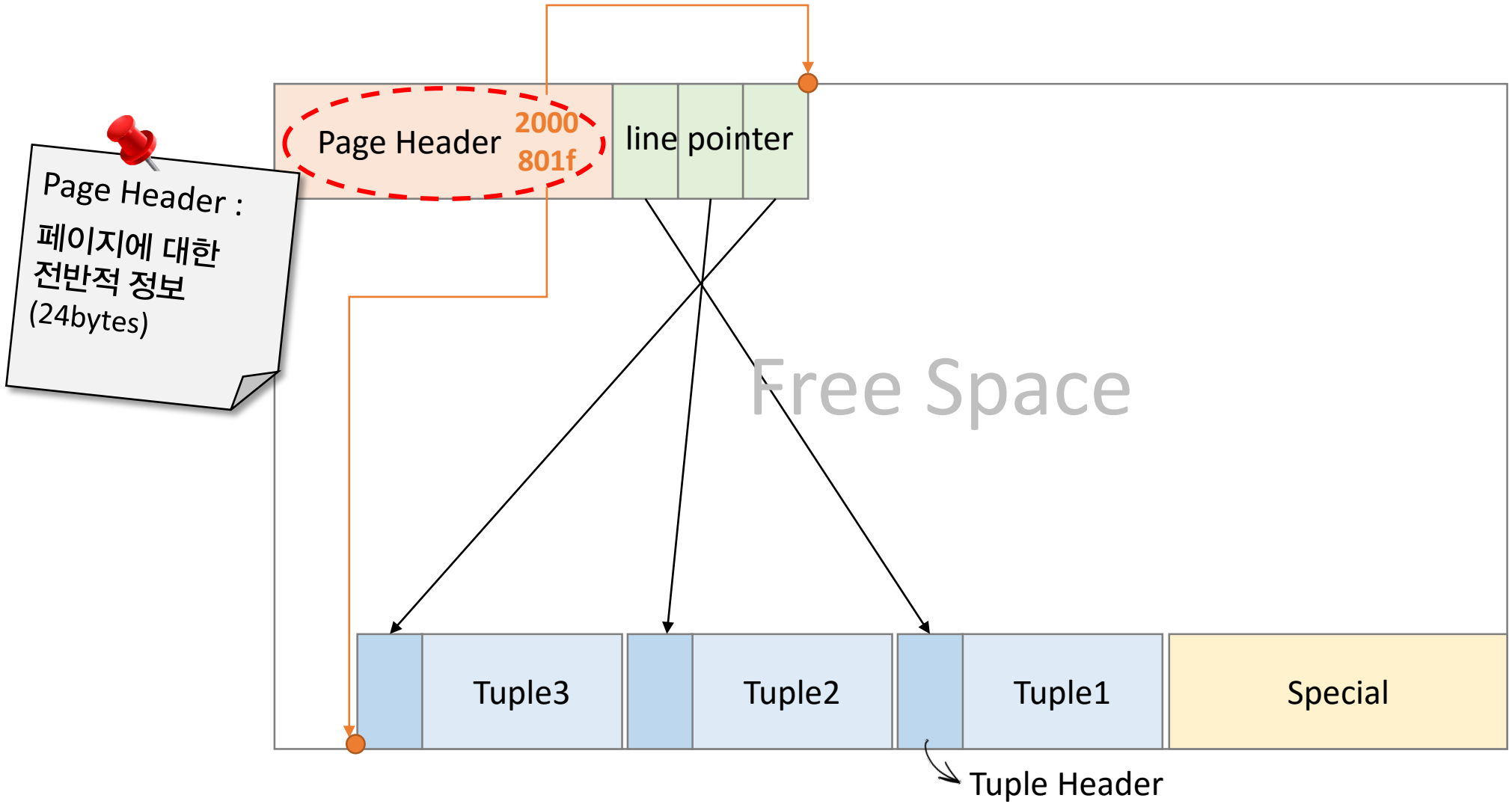
❖ Dump

```
000000 25 00 00 00 38 26 9d db 00 00 00 00 20 00 80 1f  >%...8&..... ...<
000010 00 20 04 20 00 00 00 00 c0 9f 7c 00 80 9f 7c 00  >. . ......|...|.<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001f80 ac d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001f90 02 00 03 00 02 08 18 00 29 41 31 30 30 32 20 20  >........)A1002  <
001fa0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20  >             .A  <
001fb0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A       ..<
001fc0 ab d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001fd0 01 00 03 00 02 08 18 00 29 41 31 30 30 31 20 20  >........)A1001  <
001fe0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20  >             .A  <
001ff0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A       ..<
002000
```

```
(postgres@[local]:5432) [postgres] > select xmin,xmax,* from t2;
    xmin    | xmax |          c1          |    c2    |    c3
------------+------+----------------------+----------+----------
 1807012524 |    0 | A1002                | A        | A
 1807012525 |    0 | A1001                | B        | A
(2 rows)
```

- Page Header Data: 페이지에 대한 전반적 정보
- Item Id Data(line pointer): 실제 아이템을 가리키는 포인터로 offset, length에 대한 정보, 아이템 당 4bytes
- Free space: 할당되지 않은 공간
- Items: 실제 아이템에 대한 정보
- Special space: 일반 테이블에서는 빈 공간

04. Page Layout – 2) Page Header Data Layout



EXEM

Table 63-3. PageHeaderData Layout

Field	Type	Length	Description
pd_lsn	PageXLogRecPtr	8 bytes	LSN: next byte after last byte of xlog record for last change to this page
pd_checksum	uint16	2 bytes	Page checksum
pd_flags	uint16	2 bytes	Flag bits
pd_lower	LocationIndex	2 bytes	Offset to start of free space
pd_upper	LocationIndex	2 bytes	Offset to end of free space
pd_special	LocationIndex	2 bytes	Offset to start of special space
pd_pagesize_version	uint16	2 bytes	Page size and layout version number information
pd_prune_xid	TransactionId	4 bytes	Oldest unpruned XMAX on page, or zero if none

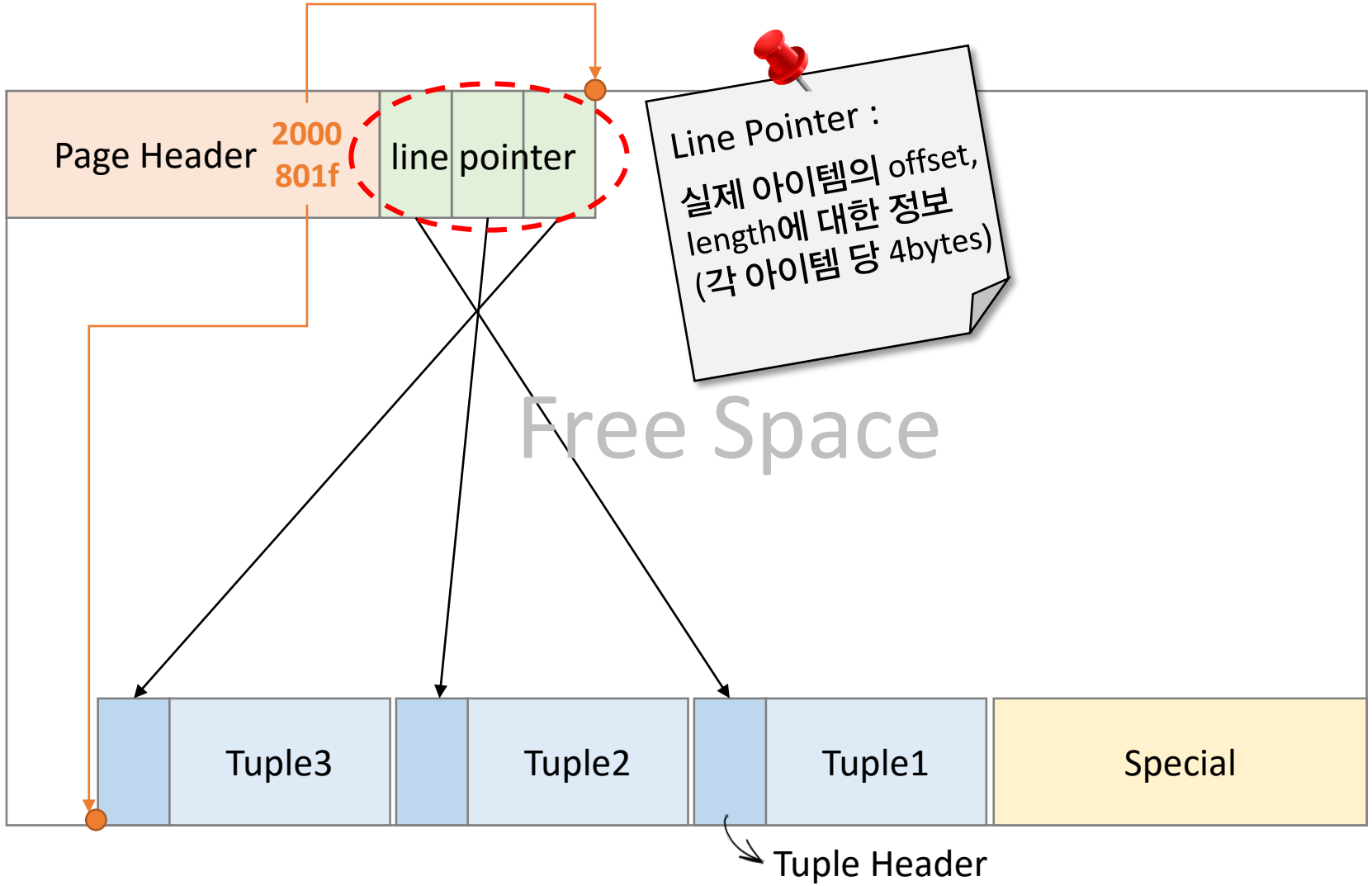
All the details can be found in `src/include/storage/bufpage.h`.

❖ Dump

```
000000 25 00 00 00 38 26 9d db 00 00 00 00 20 00 80 1f  >%...8&..... ...<
000010 00 20 04 20 00 00 00 00 c0 9f 7c 00 80 9f 7c 00  >. . ......|...|.<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001f80 ac d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001f90 02 00 03 00 02 08 18 00 29 41 31 30 30 32 20 20  >........)A1002  <
001fa0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20  >             .A  <
001fb0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A       ..<
001fc0 ab d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001fd0 01 00 03 00 02 08 18 00 29 41 31 30 30 31 20 20  >........)A1001  <
001fe0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20  >             .A  <
001ff0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A       ..<
002000
```

```
(postgres@[local]:5432) [postgres] > select xmin,xmax,* from t2;
    xmin    | xmax |          c1          |    c2    |    c3
------------+------+----------------------+----------+----------
 1807012524 |    0 | A1002                | A        | A
 1807012525 |    0 | A1001                | B        | A
(2 rows)
```

- pd_lsn: 이 페이지의 가장 마지막 변경에 대한 xlog record
- pd_checksum: 페이지 체크섬
- pd_flags: 플래그 비트
- pd_lower: freespace의 시작
- pd_upper: free space의 끝이자 items의 시작
- pd_special: special space의 시작
- pd_pagesize_version: 페이지 크기 (2004: 8196 = 8k)
- pd_prune_xid: xmax중 가장 오래된 값(작은 값)



```
typedef struct ItemIdData
{
    unsigned    lp_off:15,      /* offset to tuple (from start of page) */
                lp_flags:2,     /* state of item pointer, see below */
                lp_len:15;      /* byte length of tuple */
} ItemIdData;
```

- lp(line pointer): 블록 내의 아이템 offset ID
- lp_off: 튜플의 offset (주소)
- lp_flags: 아이템 포인터의 상태
  - lp_unused: 0
  - lp_normal: 1
  - lp_redirect: 2
  - lp_dead: 3
- lp_len: 튜플의 길이

❖ Dump

00 7c 9f c0

```
000000 25 00 00 00 38 26 9d db 00 00 00 00 20 00 80 1f  >%...8&..... ...<
000010 00 20 04 20 00 00 00 00 c0 9f 7c 00 80 9f 7c 00  >. . ......|...|.<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001f80 ac d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001f90 02 00 03 00 02 08 18 00 29 41 31 30 30 32 20 20  >........)A1002  <
001fa0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20  >             .A  <
001fb0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A       ..<
001fc0 ab d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001fd0 01 00 03 00 02 08 18 00 29 41 31 30 30 31 20 20  >........)A1001  <
001fe0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20  >             .A  <
001ff0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A       ..<
002000
```

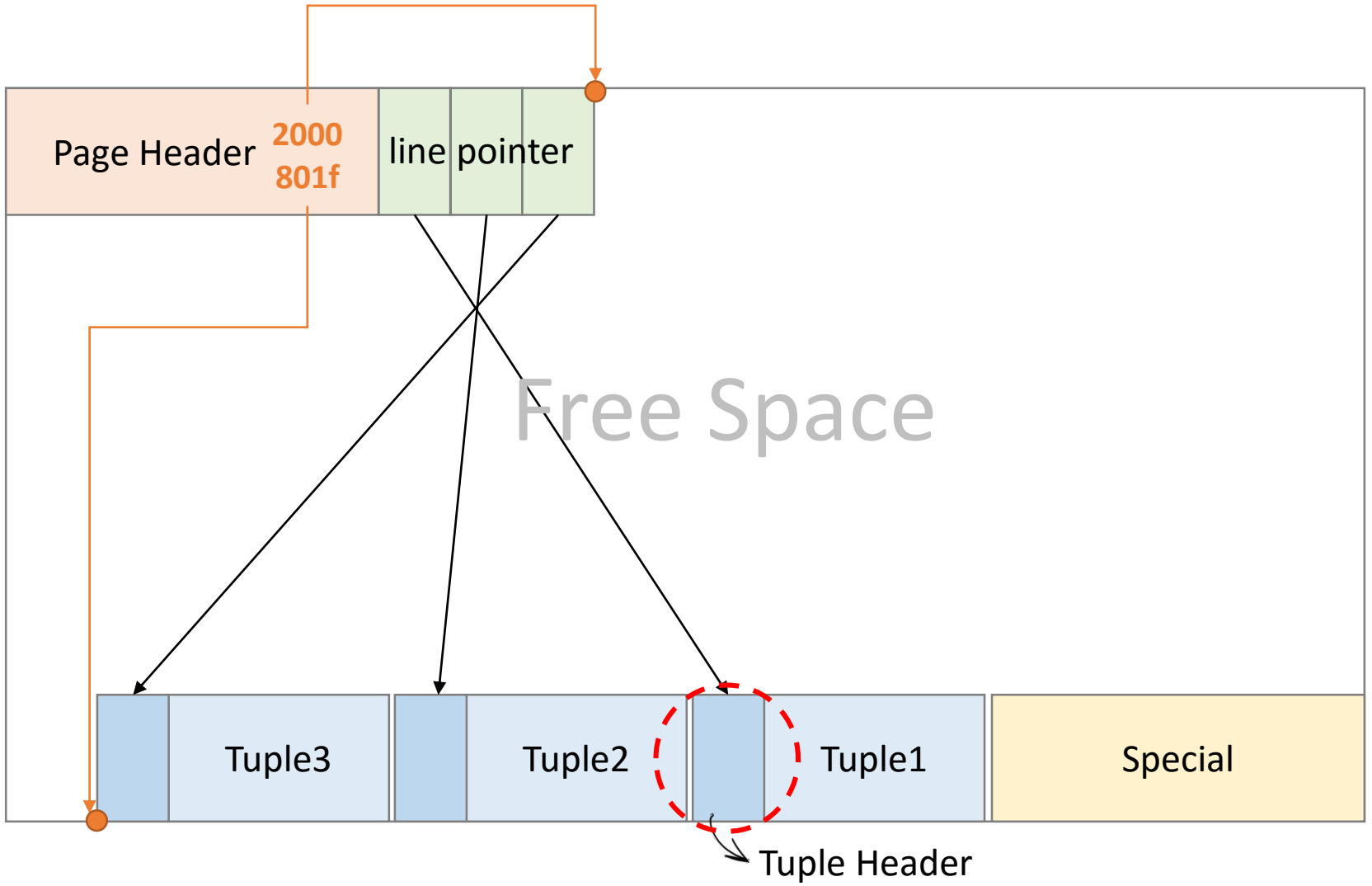
‘00 7c 9f c0’= 0 0 7 c 9 f c 0

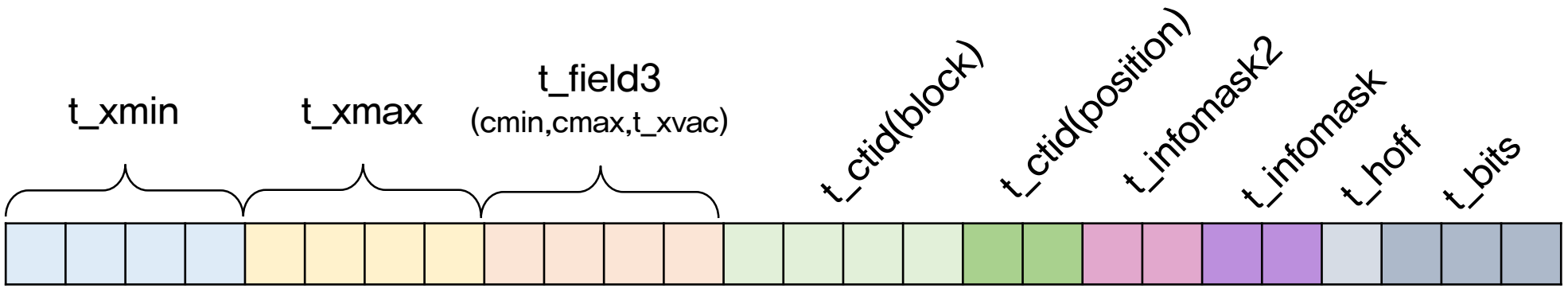
0000 0000 0111 1100 1001 1111 1100 0000

- lp_len (15 bit): 0000 0000 0111 110
- lp_flags (2 bit): 0
- lp_off (15 bit): 1001 1111 1100 000

- 실제 아이템을 가리키는 Item Id Data는 lp_off(15bit), lp_flags(2bit), lp_len(15bit) 총 4byte로 구성되어 있다.
- 예를 들어, Item Id Data인 ‘c0 9f 7c 00’ 은 순서를 거꾸로 하여 ‘00 7c 9f c0’ 에서 뒤의 4자리에서 맨 앞의 1bit 를 제외한 ‘1fc0’ 을 통해 Item을 찾아갈 수 있다.

04. Page Layout – 4) Heap Tuple Header Data Layout





- t_xmin: insert 시의 XID
- t_xmax: delete 시의 XID
- t_field3: t_cid, t_xvac 로 구성
- t_cid: insert, delete command id (cmin, cmax)
- t_xvac: 이전 full vacuum의 트랜잭션 ID를 추적 가능
- t_ctid: 해당 튜플의 현재 버전을 가리키는 포인터

  block id data(4bytes) + offset number (2bytes)
- t_infomask2: 앞의 2자리는 변경 커맨드 정보, 뒤의 2자리는 컬럼 수
- t_infomask: commit 여부
- t_hoff: 튜플 헤더의 길이, 유저 데이터의 시작 주소
- t_bits: null 값의 variable length bitmap

t_xmin	t_xmax	t_field3 t_cid/t_xvac	t_ctid 중 block id data
4bytes	4bytes	4bytes	4bytes

t_ctid 중 offset	t_infomas k2	t_infomask	t_hoff	사용자
2 bytes	2 bytes	2 bytes	1 bytes	9bytes

예)

```
001380 69 31 00 00 6b 31 00 00 00 00 00 00 00 00 00 00  >i1..k1..........<
001390 64 00 01 20 02 01 18 00 11 31 39 39 20 20 20 20  >d.. .....199   <
```

04. Page Layout – 4) Heap Tuple Header Data Layout

Table 63-4. HeapTupleHeaderData Layout

Field	Type	Length	Description
t_xmin	TransactionId	4 bytes	insert XID stamp
t_xmax	TransactionId	4 bytes	delete XID stamp
t_cid	CommandId	4 bytes	insert and/or delete CID stamp (overlays with t_xvac)
t_xvac	TransactionId	4 bytes	XID for VACUUM operation moving a row version
t_ctid	ItemPointerData	6 bytes	current TID of this or newer row version
t_infomask2	uint16	2 bytes	number of attributes, plus various flag bits
t_infomask	uint16	2 bytes	various flag bits
t_hoff	uint8	1 byte	offset to user data

All the details can be found in src/include/access/htup_details.h.

❖ Dump

00 7c 9f 80

```
000000 25 00 00 00 38 26 9d db 00 00 00 00 20 00 80 1f  >%...8&..... ...<
000010 00 20 04 20 00 00 00 00 c0 9f 7c 00 80 9f 7c 00  >. . ......|...|.<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001f80 ac d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001f90 02 00 03 00 02 08 18 00 29 41 31 30 30 32 20 20  >........)A1002  <
001fa0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20  >             .A  <
001fb0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A       ..<
001fc0 ab d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001fd0 01 00 03 00 02 08 18 00 29 41 31 30 30 31 20 20  >........)A1001  <
001fe0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20  >             .A  <
001ff0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A       ..<
002000
```

❖ SELECT * from heap_page_items(get_raw_page('t2', 0));

- heap_page_items : 테이블 내의 데이터
- get_raw_page : 페이지를 나타내는 byte array

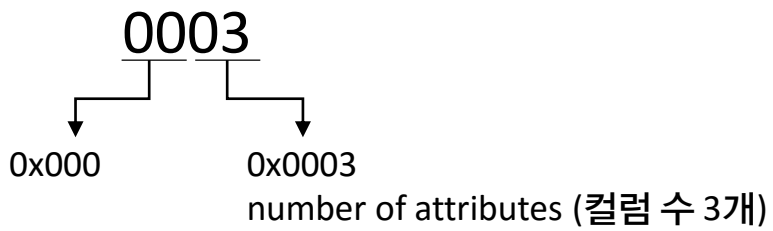
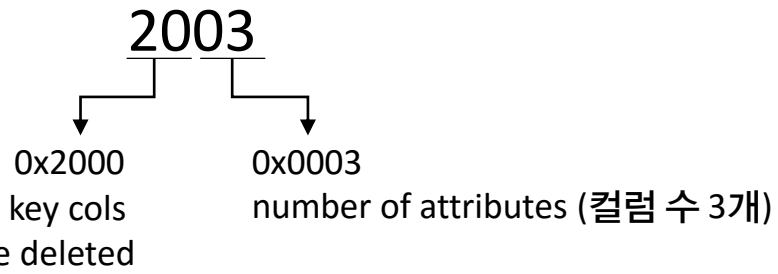
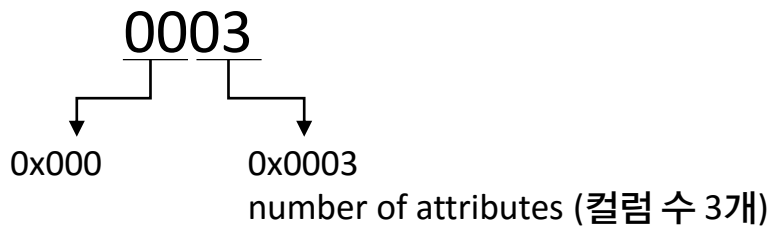
```
(postgres@[local]:5432) [postgres] > SELECT * from heap_page_items(get_raw_page('t2', 0));
 lp | lp_off | lp_flags | lp_len |   t_xmin   | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
----+--------+----------+--------+------------+--------+----------+--------+-------------+------------+--------+--------+-------
  1 |   8128 |        1 |     62 | 1807012523 |      0 |        0 | (0,1)  |           3 |       2050 |     24 | NULL   |  NULL
  2 |   8064 |        1 |     62 | 1807012524 |      0 |        0 | (0,2)  |           3 |       2050 |     24 | NULL   |  NULL
(2 rows)
```


EXEM

❖ t_infomask2

Name	Body	Description
HEAP_NATTS_MASK	0x07FF	11 bits for number of attributes
	0x1800	bits 0x1800 are available
HEAP_KEYS_UPDATED	0x2000	tuple was updated and key cols modified, or tuple deleted
HEAP_HOT_UPDATED	0x4000	tuple was HOT-updated
HEAP_ONLY_TUPLE	0x8000	this is heap-only tuple
HEAP2_XACT_MASK	0xE000	visibility-related bits

❖ t_infomask2

Command		Dump	Meaning
commit	insert	0003	
	delete	2003	
rollback		0003	

❖ t_infomask

Name	Body	Description
HEAP_HASNULL	0x0001	has null attribute(s)
HEAP_HASVARWIDTH	0x0002	has variable-width attribute(s)
HEAP_HASEXTERNAL	0x0004	has external stored attribute(s)
HEAP_HASOID	0x0008	has an object-id field
HEAP_XMAX_KEYSHR_LOCK	0x0010	xmax is a key-shared locker
HEAP_COMBOCID	0x0020	t_cid is a combo cid
HEAP_XMAX_EXCL_LOCK	0x0040	xmax is exclusive locker
HEAP_XMAX_LOCK_ONLY	0x0080	xmax, if valid, is only a locker

❖ t_infomask

Name	Body	Description
HEAP_XMIN_COMMITTED	0x0100	t_xmin committed
HEAP_XMIN_INVALID	0x0200	t_xmin invalid/aborted
HEAP_XMIN_FROZEN	0X0300	(HEAP_XMIN_COMMITTED | HEAP_XMIN_INVALID)
HEAP_XMAX_COMMITTED	0x0400	t_xmax committed
HEAP_XMAX_INVALID	0x0800	t_xmax invalid/aborted
HEAP_XMAX_IS_MULTI	0x1000	t_xmax is a MultiXactId
HEAP_UPDATED	0x2000	this is UPDATEd version of row
HEAP_MOVED_OFF	0x4000	• moved to another place by pre-9.0 • VACUUM FULL; kept for binary • upgrade support
HEAP_MOVED_IN	0x8000	• moved from another place by pre-9.0 • VACUUM FULL; kept for binary • upgrade support
HEAP_XACT_MASK	0xFFF0	visibility-related bits

❖ t_infomask

Command		Dump	Meaning
commit	insert	0902	0902 0x0900 (0x0100 + 0x0800) xmin committed + xmax invalid 0x0002 variable-width attributes
	delete	0502	0502 0x0500 (0x0100 + 0x0400) xmin committed + xmax committed 0x0002 variable-width attributes
	update	delete : 0502 insert : 2902	2902 0x2900 (0x2000 + 0x0900) updated version of row + insert와 동일 0x0002 variable-width attributes

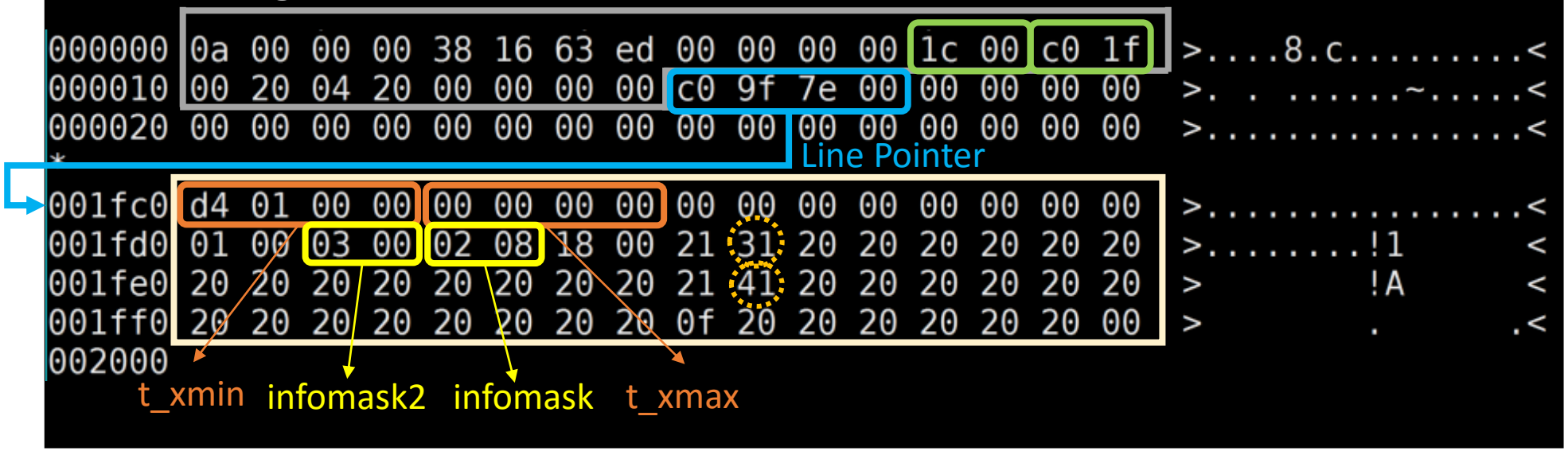
❖ t_infomask

Command	Dump	Meaning
rollback	0A02	0A02 0x0A00 (0x0200 + 0x0800) xmin invalid + xmax invalid 0x0002 variable-width attributes
cf) null이 있는 경우	insert : 0903 rollback : 0A03	0A03 0x0A00 (0x0200 + 0x0800) xmin invalid + xmax invalid 0x0003 (0x0001 + 0x0002) has null attribute(s) + variable-width attributes

```
create table myt1 ( c1 char(15),c2 char(15),c3 char(6) );
CREATE TABLE
```

insert 1건 (1,A,), commit NO

```
insert into myt1 select generate_series(1,1),'A',' ';
INSERT 0 1
Time: 0.770 ms
select * from myt1;
       c1        |       c2        |   c3
-----------------+-----------------+--------
 1               | A               |
(1 row)
```



insert 1건 (1,A,), commit NO

```
insert into myt1 select generate_series(1,1),'A',' ';
INSERT 0 1
Time: 0.770 ms
select * from myt1;
       c1        |       c2        |   c3
-----------------+-----------------+-------
 1               | A               |
(1 row)

000000 0a 00 00 00 38 16 63 ed 00 00 00 00 1c 00 c0 1f  >....8.c.........<
000010 00 20 04 20 00 00 00 00 c0 9f 7e 00 00 00 00 00  >. . ......~.....<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001fc0 d4 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001fd0 01 00 03 00 02 08 18 00 21 31 20 20 20 20 20 20  >........!1      <
001fe0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001ff0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
002000
```

- pd_lower 001c
- pd_upper 1fc0
- line pointer 007e9fc0

- t_xmin 000001d4
- t_xmax 00000000
- t_infomask2 0003
- t_infomask **0802** 8 : xmax invalid/ aborted

commit YES

```
commit;
COMMIT
Time: 1.305 ms
select * from myt1;
       c1        |       c2        |   c3
-----------------+-----------------+-------
 1               | A               |
(1 row)

000000 0a 00 00 00 38 16 63 ed 00 00 00 00 1c 00 c0 1f  >....8.c.........<
000010 00 20 04 20 00 00 00 00 c0 9f 7e 00 00 00 00 00  >. . ......~.....<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001fc0 d4 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001fd0 01 00 03 00 02 09 18 00 21 31 20 20 20 20 20 20  >........!1      <
001fe0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001ff0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
002000
```

- pd_lower 001c
- pd_upper 1fc0
- line pointer 007e9fc0

- t_xmin 000001d4
- t_xmax 00000000
- t_infomask2 0003
- t_infomask **0902** 8 : xmax invalid/ aborted
1 : xmin committed

EXEM

❖ t_infomask

	xmax invalid/aborted	xmax committed	xmin invalid/aborted	xmin committed
0000	2^3	2^2	2^1	2^0
insert commit NO	1	0	0	0
insert commit YES	1	0	0	1
delete commit NO	0	0	0	1
delete commit YES	0	1	0	1

cf) update 2xxx
frozen bit XbXX

insert 1건 (2,A,) commit YES

- pd_lower 0020
- pd_upper 1f80
- line pointer 007e9fc0
  007e9f80

(2,A,)

- t_xmin 000001d5
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

(1,A,)

- t_xmin 000001d4
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

```
insert into myt1 select generate_series(2,2),'A',' ';
INSERT 0 1
Time: 1.563 ms

COMMIT
Time: 0.134 ms
select * from myt1;
       c1        |       c2        |   c3
-----------------+-----------------+--------
 1               | A               |
 2               | A               |
(2 rows)

000000 0a 00 00 00 28 18 63 ed 00 00 00 00 20 00 80 1f  >....(.c..... ...<
000010 00 20 04 20 00 00 00 00 c0 9f 7e 00 80 9f 7e 00  >. . ......~...~.<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001f80 d5 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001f90 02 00 03 00 02 09 18 00 21 32 20 20 20 20 20 20  >........!2      <
001fa0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001fb0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
001fc0 d4 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001fd0 01 00 03 00 02 09 18 00 21 31 20 20 20 20 20 20  >........!1      <
001fe0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001ff0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
002000
```

insert 1건 (3,A,) commit YES

- pd_lower 0024
- pd_upper 1f40
- line pointer 007e9fc0
  007e9f80
  007e9f40

(3,A,)
- t_xmin 000001d6
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

(2,A,)
- t_xmin 000001d5
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

(1,A,)
- t_xmin 000001d4
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

```
insert into myt1 select generate_series(3,3),'A',' ';
INSERT 0 1
Time: 1.413 ms
COMMIT
Time: 0.135 ms
select * from myt1;
       c1        |       c2        |   c3
-----------------+-----------------+--------
 1               | A               |
 2               | A               |
 3               | A               |
(3 rows)
000000 0a 00 00 00 f8 19 63 ed 00 00 00 00 24 00 40 1f  >......c.....$.@.<
000010 00 20 04 20 00 00 00 00 c0 9f 7e 00 80 9f 7e 00  >. . ......~...~.<
000020 40 9f 7e 00 00 00 00 00 00 00 00 00 00 00 00 00  >@.~.............<
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001f40 d6 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001f50 03 00 03 00 02 09 18 00 21 33 20 20 20 20 20 20  >........!3      <
001f60 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001f70 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
001f80 d5 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001f90 02 00 03 00 02 09 18 00 21 32 20 20 20 20 20 20  >........!2      <
001fa0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001fb0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
001fc0 d4 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001fd0 01 00 03 00 02 09 18 00 21 31 20 20 20 20 20 20  >........!1      <
001fe0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001ff0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
002000
```

delete 1건 (3,A,) commit NO

- pd_lower 0024
- pd_upper 1f40
- line pointer 007e9fc0
  007e9f80
  007e9f40

(3,A,)

- t_xmin 000001d6
- t_xmax 000001d7
- t_infomask2 2003
- t_infomask 0102

(2,A,)

- t_xmin 000001d5
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

(1,A,)

- t_xmin 000001d4
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

```
delete from myt1 where c1= '3';
DELETE 1
Time: 1.113 ms
select * from myt1;
       c1        |        c2        |   c3
-----------------+------------------+--------
 1               | A                |
 2               | A                |
(2 rows)

000000 0a 00 00 00 c8 1b 63 ed 00 00 00 00 24 00 40 1f  >......c.....$.@.<
000010 00 20 04 20 d7 01 00 00 c0 9f 7e 00 80 9f 7e 00  >. . ......~...~.<
000020 40 9f 7e 00 00 00 00 00 00 00 00 00 00 00 00 00  >@.~.............<
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001f40 d6 01 00 00 d7 01 00 00 00 00 00 00 00 00 00 00  >................<
001f50 03 00 03 20 02 01 18 00 21 33 20 20 20 20 20 20  >... ....!3      <
001f60 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001f70 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .     .<
001f80 d5 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001f90 02 00 03 00 02 09 18 00 21 32 20 20 20 20 20 20  >........!2      <
001fa0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001fb0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .     .<
001fc0 d4 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001fd0 01 00 03 00 02 09 18 00 21 31 20 20 20 20 20 20  >........!1      <
001fe0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001ff0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .     .<
002000
```

delete 1건 (3,A,) commit YES

- pd_lower 0024
- pd_upper 1f40
- line pointer 007e9fc0
007e9f80
007e9f40

(3,A,)

- t_xmin 000001d6
- t_xmax 000001d7
- t_infomask2 2003
- t_infomask 0502

(2,A,)

- t_xmin 000001d5
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

(1,A,)

- t_xmin 000001d4
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

```
COMMIT
Time: 1.677 ms
select * from myt1;
       c1        |       c2        |   c3
-----------------+-----------------+--------
 1               | A               |
 2               | A               |
(2 rows)

000000 0a 00 00 00 c8 1b 63 ed 00 00 00 00 24 00 40 1f  >......c.....$.@.<
000010 00 20 04 20 d7 01 00 00 c0 9f 7e 00 80 9f 7e 00  >. . ......~...~.<
000020 40 9f 7e 00 00 00 00 00 00 00 00 00 00 00 00 00  >@.~.............<
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001f40 d6 01 00 00 d7 01 00 00 00 00 00 00 00 00 00 00  >................<
001f50 03 00 03 20 02 05 18 00 21 33 20 20 20 20 20 20  >... ....!3      <
001f60 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001f70 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
001f80 d5 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001f90 02 00 03 00 02 09 18 00 21 32 20 20 20 20 20 20  >........!2      <
001fa0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001fb0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
001fc0 d4 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001fd0 01 00 03 00 02 09 18 00 21 31 20 20 20 20 20 20  >........!1      <
001fe0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001ff0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
002000
```

insert 1건(3,A,) commit YES

(3,A,)

- t_xmin 000001d8
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

(3,A,)

- t_xmin 000001d6
- t_xmax 000001d7
- t_infomask2 2003
- t_infomask 0502

(2,A,)

- t_xmin 000001d5
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

(1,A,)

- t_xmin 000001d4
- t_xmax 00000000
- t_infomask2 0003
- t_infomask 0902

```
insert into myt1 select generate_series(3,3),'A',' ';
INSERT 0 1
Time: 382.225 ms
commit;
COMMIT
Time: 0.187 ms
select * from myt1;
 c1 |  c2  |  c3
----+------+-------
  1 | A    |
  2 | A    |
  3 | A    |
(3 rows)
```

- pd_lower 0028
- pd_upper 1f00
- line pointer 007e9fc0 007e9f80 007e9f40 007e9f00

```
000000 0a 00 00 00 40 1e 63 ed 00 00 00 00 28 00 00 1f  >....@.c.....(...<
000010 00 20 04 20 d7 01 00 00 c0 9f 7e 00 80 9f 7e 00  >. . ......~...~.<
000020 40 9f 7e 00 00 9f 7e 00 00 00 00 00 00 00 00 00  >@.~...~.........<
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001f00 d8 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001f10 04 00 03 00 02 09 18 00 21 33 20 20 20 20 20 20  >........!3      <
001f20 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001f30 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 00 20  >         .     .<
001f40 d6 01 00 00 d7 01 00 00 00 00 00 00 00 00 00 00  >................<
001f50 03 00 03 20 02 05 18 00 21 33 20 20 20 20 20 20  >... ....!3      <
001f60 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001f70 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 00 20  >         .     .<
001f80 d5 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001f90 02 00 03 00 02 09 18 00 21 32 20 20 20 20 20 20  >........!2      <
001fa0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001fb0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 00 20  >         .     .<
001fc0 d4 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001fd0 01 00 03 00 02 09 18 00 21 31 20 20 20 20 20 20  >........!1      <
001fe0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001ff0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 00 20  >         .     .<
002000
```

update 1건(3,B,) commit NO

- pd_lower 002c
- pd_upper 1ec0
- line pointer 007e9fc0
  007e9f80
  007e9f40
  007e9f00
  007e9ec0

(3,B,)	•	t_xmin	000001d9
	•	t_xmax	00000000
	•	t_infomask2	8003
	•	t_infomask	2802
(3, A,) update	•	t_xmin	000001d8
	•	t_xmax	000001d9
	•	t_infomask2	4003
	•	t_infomask	0102
(3,A,) delete	•	t_xmin	000001d6
	•	t_xmax	000001d7
	•	t_infomask2	2003
	•	t_infomask	0502

```
update myt1 set c2 = 'B' where c1 = '3';
UPDATE 1
Time: 0.332 ms
select * from myt1;
 c1 |  c2  |  c3
----+------+------
 1  | A    |
 2  | A    |
 3  | B    |
(3 rows)
000000 0a 00 00 00 b8 20 63 ed 00 00 00 00 2c 00 c0 1e  >..... c.....,...<
000010 00 20 04 20 d7 01 00 00 c0 9f 7e 00 80 9f 7e 00  >. . ......~...~.<
000020 40 9f 7e 00 00 9f 7e 00 c0 9e 7e 00 00 00 00 00  >@.~...~...~.....<
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001ec0 d9 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001ed0 05 00 03 80 02 28 18 00 21 33 20 20 20 20 20 20  >.....(..!3      <
001ee0 20 20 20 20 20 20 20 20 21 42 20 20 20 20 20 20  >        !B      <
001ef0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .     .<
001f00 d8 01 00 00 d9 01 00 00 00 00 00 00 00 00 00 00  >................<
001f10 05 00 03 40 02 01 18 00 21 33 20 20 20 20 20 20  >...@....!3      <
001f20 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001f30 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .     .<
001f40 d6 01 00 00 d7 01 00 00 00 00 00 00 00 00 00 00  >................<
001f50 03 00 03 20 02 05 18 00 21 33 20 20 20 20 20 20  >... ....!3      <
001f60 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001f70 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .     .<
001f80 d5 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001f90 02 00 03 00 02 09 18 00 21 32 20 20 20 20 20 20  >........!2      <
001fa0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001fb0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .     .<
001fc0 d4 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001fd0 01 00 03 00 02 09 18 00 21 31 20 20 20 20 20 20  >........!1      <
001fe0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001ff0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .     .<
002000
```

update 1건(3,B,) commit YES

- pd_lower 002c
- pd_upper 1ec0
- line pointer 007e9fc0
  007e9f80
  007e9f40
  007e9f00
  007e9ec0

(3,B,)	t_xmin	000001d9
	t_xmax	00000000
	t_infomask2	0803
	t_infomask	2902
(3,A,) update	t_xmin	000001d8
	t_xmax	000001d9
	t_infomask2	4003
	t_infomask	0502
(3,A,) delete	t_xmin	000001d6
	t_xmax	000001d7
	t_infomask2	2003
	t_infomask	0502

```
commit;
COMMIT
Time: 0.966 ms
select * from myt1;
      c1        |       c2        |   c3
----------------+-----------------+-------
 1              | A               |
 2              | A               |
 3              | B               |
(3 rows)

000000 0a 00 00 00 b8 20 63 ed 00 00 00 00 2c 00 c0 1e  >..... c.....,...<
000010 00 20 04 20 d7 01 00 00 c0 9f 7e 00 80 9f 7e 00  >. . ......~...~.<
000020 40 9f 7e 00 00 9f 7e 00 c0 9e 7e 00 00 00 00 00  >@.~...~...~.....<
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001ec0 d9 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001ed0 05 00 03 80 02 29 18 00 21 33 20 20 20 20 20 20  >.....)..!3      <
001ee0 20 20 20 20 20 20 20 20 21 42 20 20 20 20 20 20  >        !B      <
001ef0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
001f00 d8 01 00 00 d9 01 00 00 00 00 00 00 00 00 00 00  >................<
001f10 05 00 03 40 02 05 18 00 21 33 20 20 20 20 20 20  >...@....!3      <
001f20 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001f30 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
001f40 d6 01 00 00 d7 01 00 00 00 00 00 00 00 00 00 00  >................<
001f50 03 00 03 20 02 05 18 00 21 33 20 20 20 20 20 20  >... ....!3      <
001f60 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001f70 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
001f80 d5 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001f90 02 00 03 00 02 09 18 00 21 32 20 20 20 20 20 20  >........!2      <
001fa0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001fb0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
001fc0 d4 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
001fd0 01 00 03 00 02 09 18 00 21 31 20 20 20 20 20 20  >........!1      <
001fe0 20 20 20 20 20 20 20 20 21 41 20 20 20 20 20 20  >        !A      <
001ff0 20 20 20 20 20 20 20 20 0f 20 20 20 20 20 20 00  >         .      .<
002000
```

Question

만약 1억 건을 update 한 후에 Commit을 한다면,

1억 건에 대해서 모두 완료하기까지 시간이 얼마나 소요될까?

05. Vacuum

1) Vacuum 정의 및 필요성
2) Vacuum 실행 구조
3) 표준 Vacuum VS Vacuum full
4) XID Wraparound 와 Freeze
5) VM과 FSM 구조 분석
6) Autovacuum

EXEM

‘Multi Generation Architecture’의 단점은?

데이터의 변경이 빈번할 경우, 파일 사이즈가 커져 심각한 성능저하의 원인이 될 수 있다!

그렇다면 이를 해결해주는 기능은?

VACUUM!

05. Vacuum – 1) Vacuum 정의 및 필요성

1. Vacuum의 정의

PostgreSQL에서 특정 튜플을 update 하거나 delete 한다고 해서 해당 영역이 자동으로 재사용되거나 사라지지 않는다.

이렇게 오래된 영역을 정리하여 공간을 반환하는 명령어가 Vacuum이다. ≒ **디스크 조각 모음**

2. Vacuum의 필요성

1) 변경 및 삭제된 자료들이 차지하고 있는 디스크 공간을 확보하기 위해서
2) transaction id가 겹침(wraparound)으로 인해 오래된 자료의 손실이 발생하는 것을 방지하기 위해서
3) Query Planner가 사용할 자료의 통계 정보를 갱신하기 위해서
4) 실자료 지도(visibility map, VM)의 정보를 갱신하기 위해서
   (VM은 인덱스 전용의 검색 성능을 향상시키기 위해 사용함)

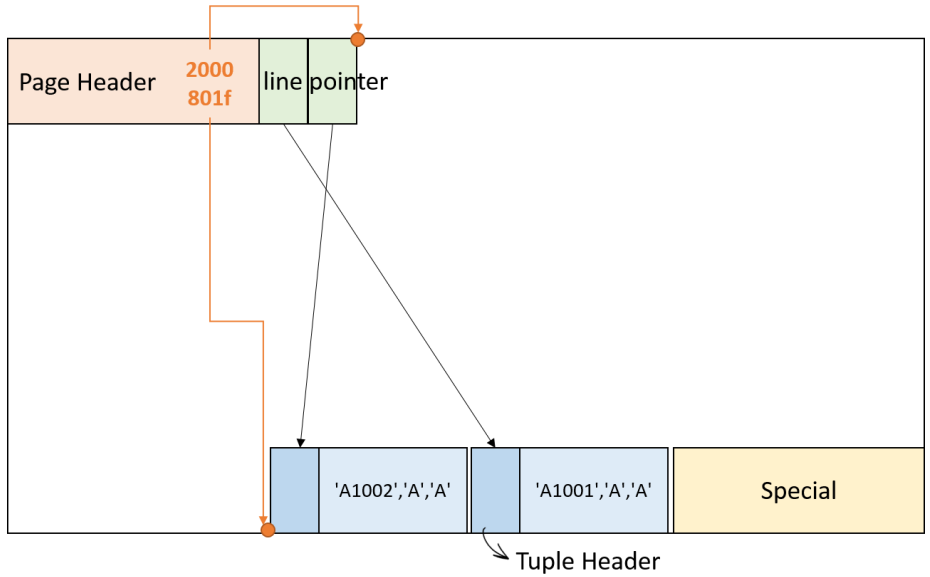
VM (Visibility Map)

dead tuple의 존재 여부를 알려주는 정보

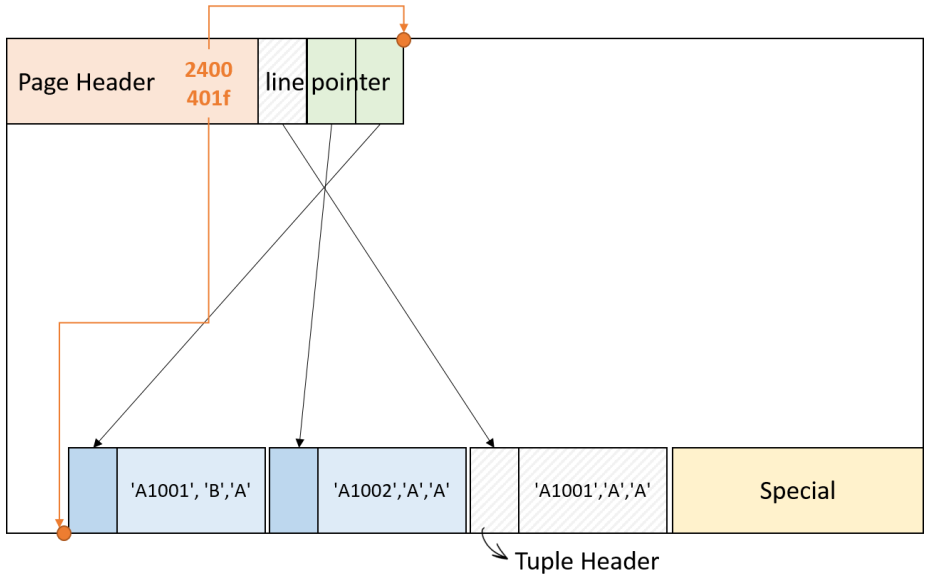
- dead tuple이 포함되지 않은 경우 : 1
- dead tuple이 포함되거나 확실하지 않은 경우 : 0

05. Vacuum – 2) Vacuum 실행 구조

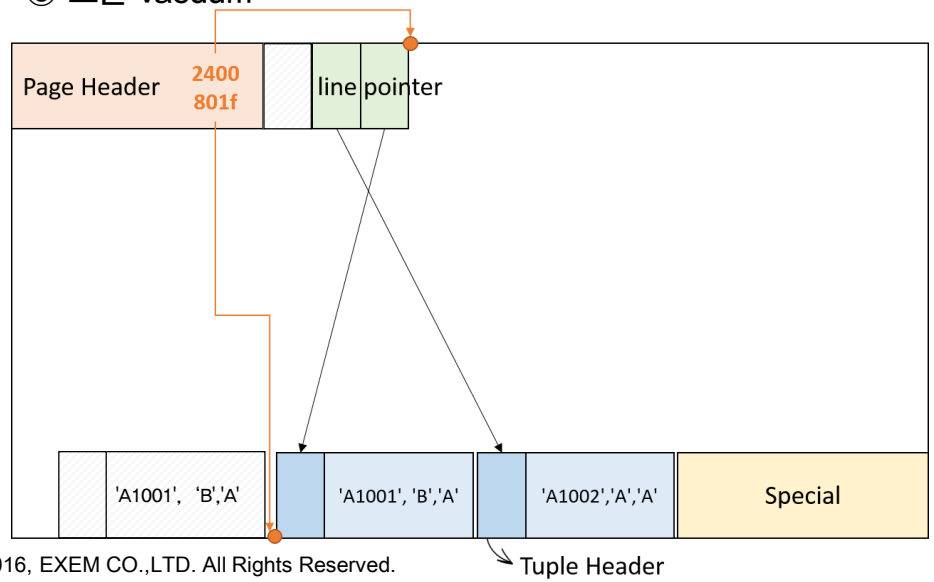
① insert into t2 values('A1001','A','A');
insert into t2 values('A1002','A','A');



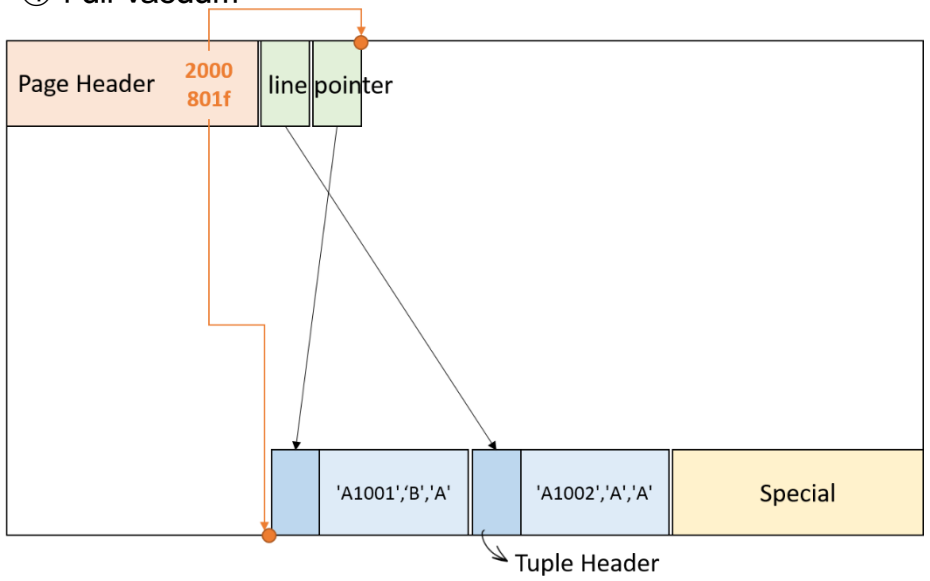
② update t2 set c2='B' where c1 = 'A1001' ;



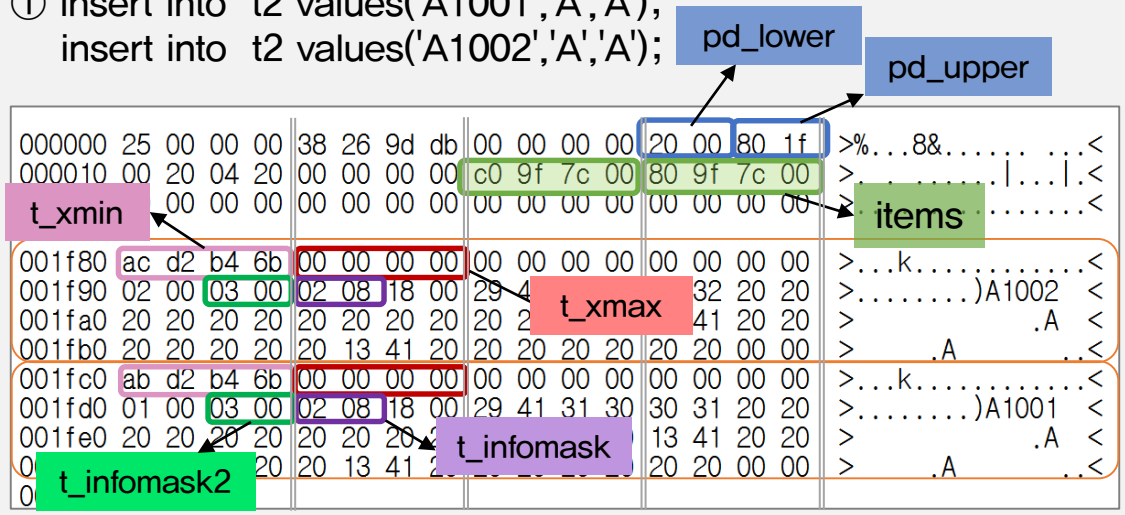
③ 표준 vacuum



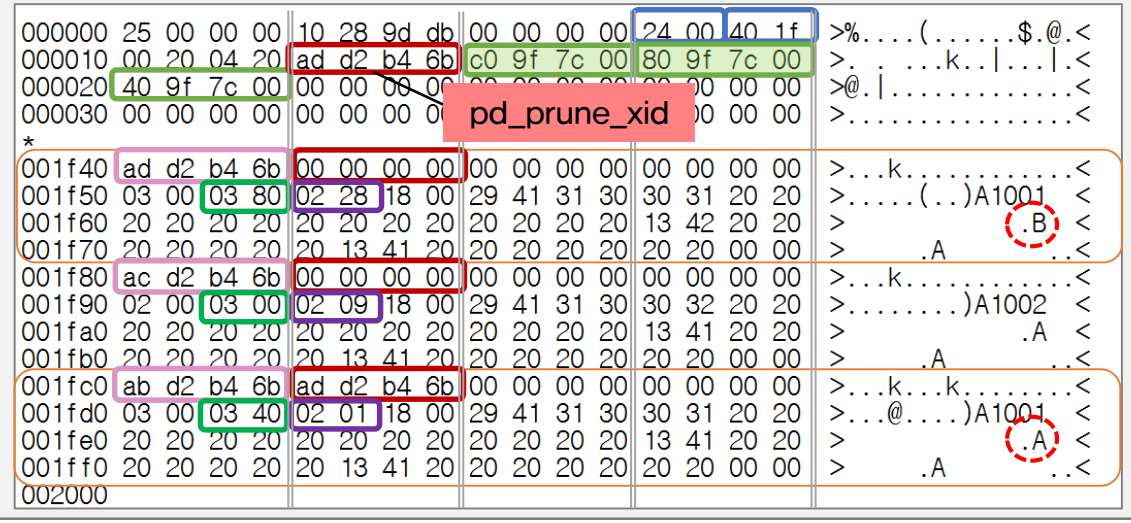
④ Full vacuum



① insert into t2 values('A1001','A','A');
insert into t2 values('A1002','A','A');

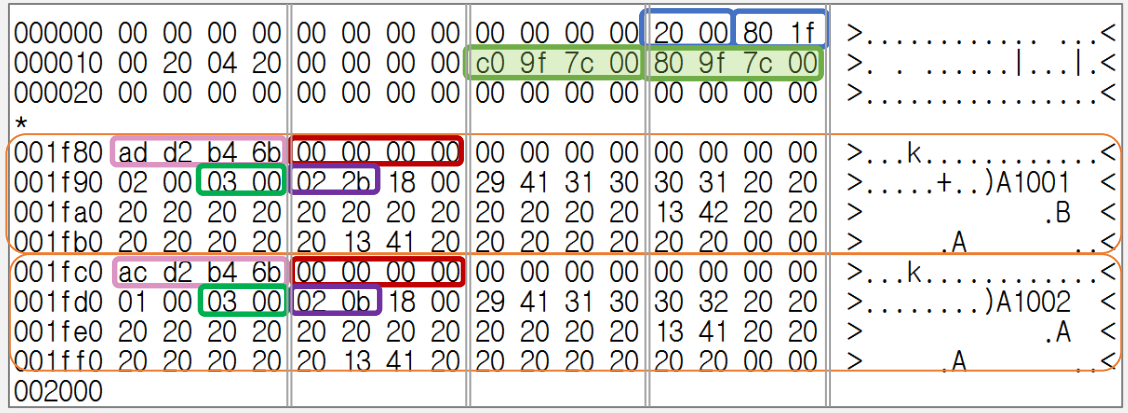


② update t2 set c2='B' where c1 = 'A1001' ;

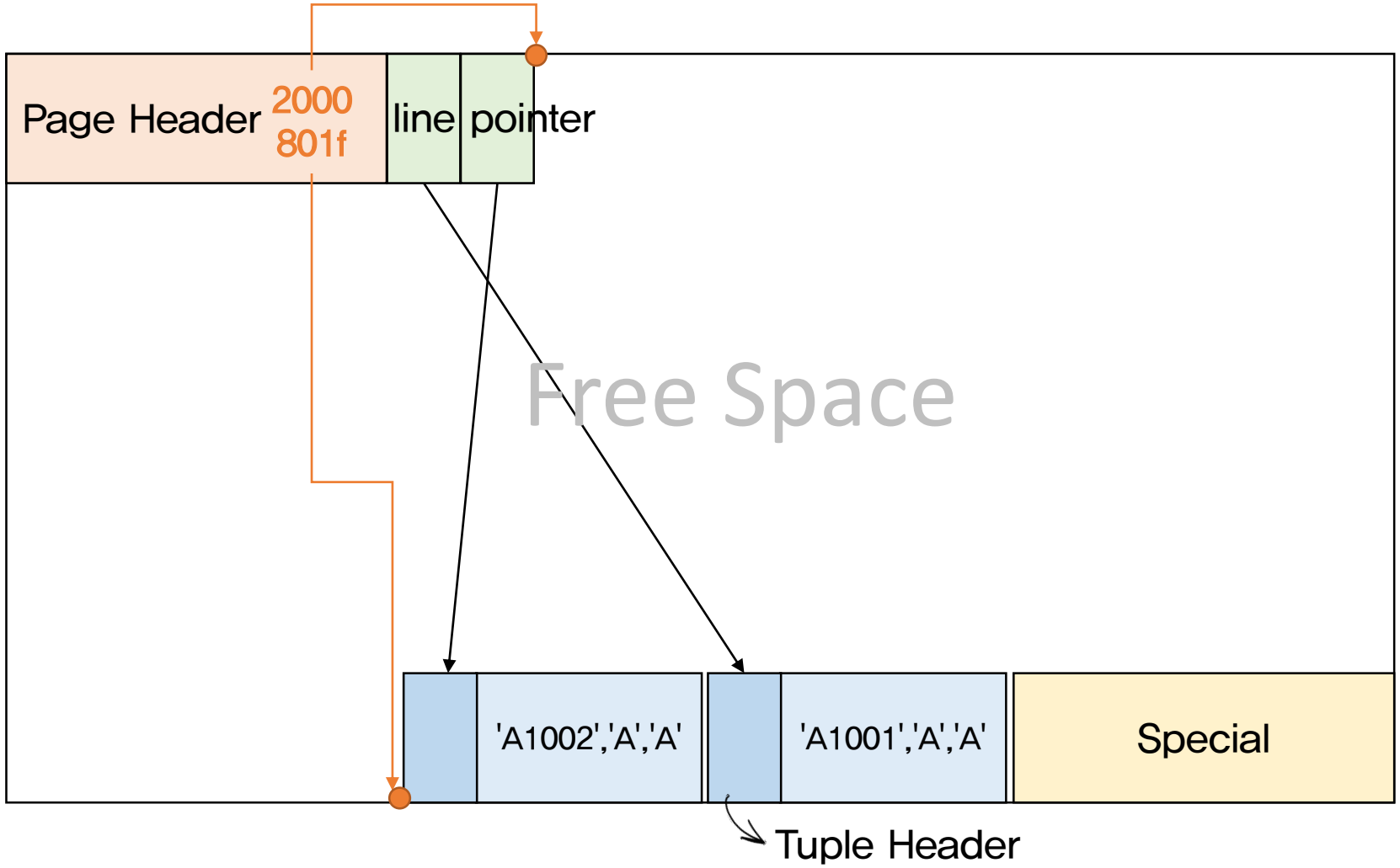


③ 표준 vacuum

④ Full vacuum



- insert into t2 values('A1001','A','A');
- insert into t2 values('A1002','A','A');



create table t2 (c1 char(19),c2 char(8),c3 char(8));

① insert into t2 values('A1001','A','A');
insert into t2 values('A1002','A','A');

```
000000 25 00 00 00 38 26 9d db 00 00 00 00 20 00 80 1f  >%...8&..... ...<
000010 00 20 04 20 00 00 00 00 c0 9f 7c 00 80 9f 7c 00  >. . ......|...|.<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001f80 ac d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001f90 02 00 03 00 02 08 18 00 29 41 31 30 30 32 20 20  >........)A1002  <
001fa0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20  >             .A  <
001fb0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A        ..<
001fc0 ab d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001fd0 01 00 03 00 02 08 18 00 29 41 31 30 30 31 20 20  >........)A1001  <
001fe0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20  >             .A  <
001ff0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A        ..<
002000
```

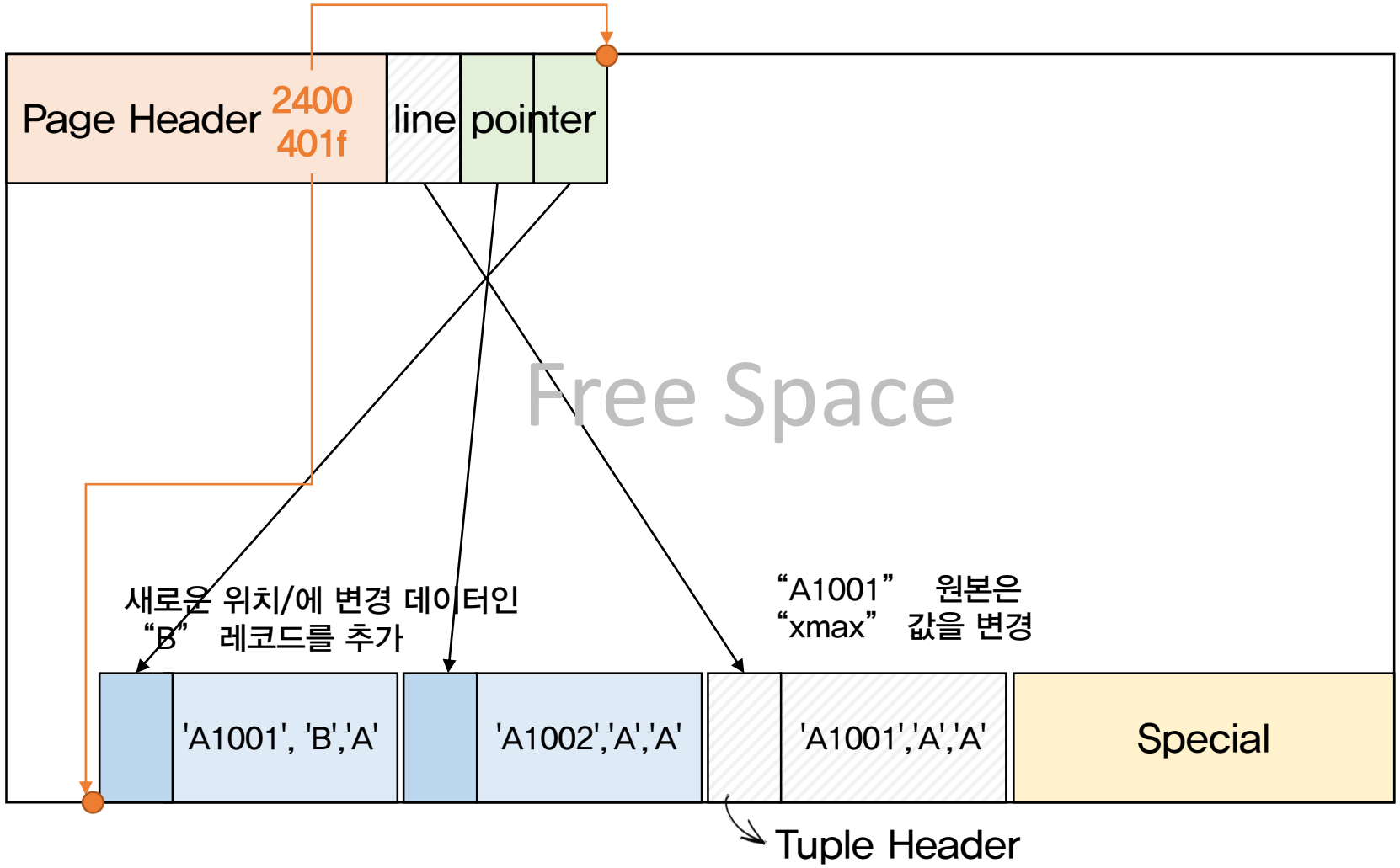
- extension module

create extension pageinspect;

- SELECT * from heap_page_items(get_raw_page('t2', 0));

```
(postgres@[local]:5432) [postgres] > SELECT * from heap_page_items(get_raw_page('t2', 0));
 lp | lp_off | lp_flags | lp_len |   t_xmin   | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
----+--------+----------+--------+------------+--------+----------+--------+-------------+------------+--------+--------+-------
  1 |   8128 |        1 |     62 | 1807012523 |      0 |        0 | (0,1)  |           3 |       2050 |     24 | NULL   |  NULL
  2 |   8064 |        1 |     62 | 1807012524 |      0 |        0 | (0,2)  |           3 |       2050 |     24 | NULL   |  NULL
(2 rows)
```

- update t2 set c2='B' where c1 = 'A1001' ;



② update t2 set c2='B' where c1 = 'A1001' ;

```
000000 25 00 00 00 10 28 9d db 00 00 00 00 24 00 40 1f >%....(......$.@.<
000010 00 20 04 20 ad d2 b4 6b c0 9f 7c 00 80 9f 7c 00 >. . ...k..|...|.<
000020 40 9f 7c 00 00 00 00 00 00 00 00 00 00 00 00 00 >@.|.............<
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*

001f80 ac d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00 >...k............<
001f90 02 00 03 00 02 09 18 00 29 41 31 30 30 32 20 20 >........)A1002  <
001fa0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20 >             .A  <
001fb0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00 >     .A       ..<
001fc0 ab d2 b4 6b ad d2 b4 6b 00 00 00 00 00 00 00 00 >...k...k........<
001fd0 03 00 03 40 02 01 18 00 29 41 31 30 30 31 20 20 >...@....)A1001  <
001fe0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20 >             .A  <
001ff0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00 >     .A       ..<
002000
```

"A1001" 원본은 "xmax" 값을 변경

- SELECT * from heap_page_items(get_raw_page('t2', 0));

```
(postgres@[local]:5432) [postgres] > SELECT * from heap_page_items(get_raw_page('t2', 0));
 lp | lp_off | lp_flags | lp_len |   t_xmin   |   t_xmax   | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
----+--------+----------+--------+------------+------------+----------+--------+-------------+------------+--------+--------+-------
  1 |   8128 |        1 |     62 | 1807012523 | 1807012525 |        0 | (0,3)  |       16387 |        258 |     24 | NULL   |  NULL
  2 |   8064 |        1 |     62 | 1807012524 |          0 |        0 | (0,2)  |           3 |       2306 |     24 | NULL   |  NULL
  3 |   8000 |        1 |     62 | 1807012525 |          0 |        0 | (0,3)  |       32771 |      10242 |     24 | NULL   |  NULL
(3 rows)
```

- select xmin,xmax,ctid, * from t2;

```
    xmin    | xmax | ctid  |        c1          |    c2    |    c3
------------+------+-------+--------------------+----------+----------
 1807012524 |    0 | (0,2) | A1002              | A        | A
 1807012525 |    0 | (0,3) | A1001              | B        | A
(2 rows)
```

"A1001" item 의 xmin 변경됨

② update t2 set c2='B' where c1 = 'A1001' ;

```
000000 25 00 00 00 10 28 9d db 00 00 00 00 24 00 40 1f >%....(......$.@.<
000010 00 20 04 20 ad d2 b4 6b c0 9f 7c 00 80 9f 7c 00 >. . ...k..|...|.<
000020 40 9f 7c 00 00 00 00 00 00 00 00 00 00 00 00 00 >@.|.............<
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
001f40 ad d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00 >...k............<
001f50 03 00 03 80 02 28 18 00 29 41 31 30 30 31 20 20 >.....(..)A1001  <
001f60 20 20 20 20 20 20 20 20 20 20 20 20 13 42 20 20 >             .B  <
001f70 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00 >     .A         ..<
001f80 ac d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00 >...k............<
001f90 02 00 03 00 02 09 18 00 29 41 31 30 30 32 20 20 >........)A1002  <
001fa0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20 >             .A  <
001fb0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00 >     .A         ..<
001fc0 ab d2 b4 6b ad d2 b4 6b 00 00 00 00 00 00 00 00 >...k...k........<
001fd0 03 00 03 40 02 01 18 00 29 41 31 30 30 31 20 20 >...@....)A1001  <
001fe0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20 >             .A  <
001ff0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00 >     .A         ..<
002000
```

새로운 위치에 변경 데이터인 “B” 레코드를 추가

“A1001” 원본은 “xmax” 값을 변경

- SELECT * from heap_page_items(get_raw_page('t2', 0));

```
(postgres@[local]:5432) [postgres] > SELECT * from heap_page_items(get_raw_page('t2', 0));
 lp | lp_off | lp_flags | lp_len |   t_xmin   |   t_xmax   | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
----+--------+----------+--------+------------+------------+----------+--------+-------------+------------+--------+--------+-------
  1 |   8128 |        1 |     62 | 1807012523 | 1807012525 |        0 | (0,3)  |       16387 |        258 |     24 | NULL   | NULL
  2 |   8064 |        1 |     62 | 1807012524 |          0 |        0 | (0,2)  |           3 |       2306 |     24 | NULL   | NULL
  3 |   8000 |        1 |     62 | 1807012525 |          0 |        0 | (0,3)  |       32771 |      10242 |     24 | NULL   | NULL
(3 rows)
```

- select xmin,xmax,ctid, * from t2;

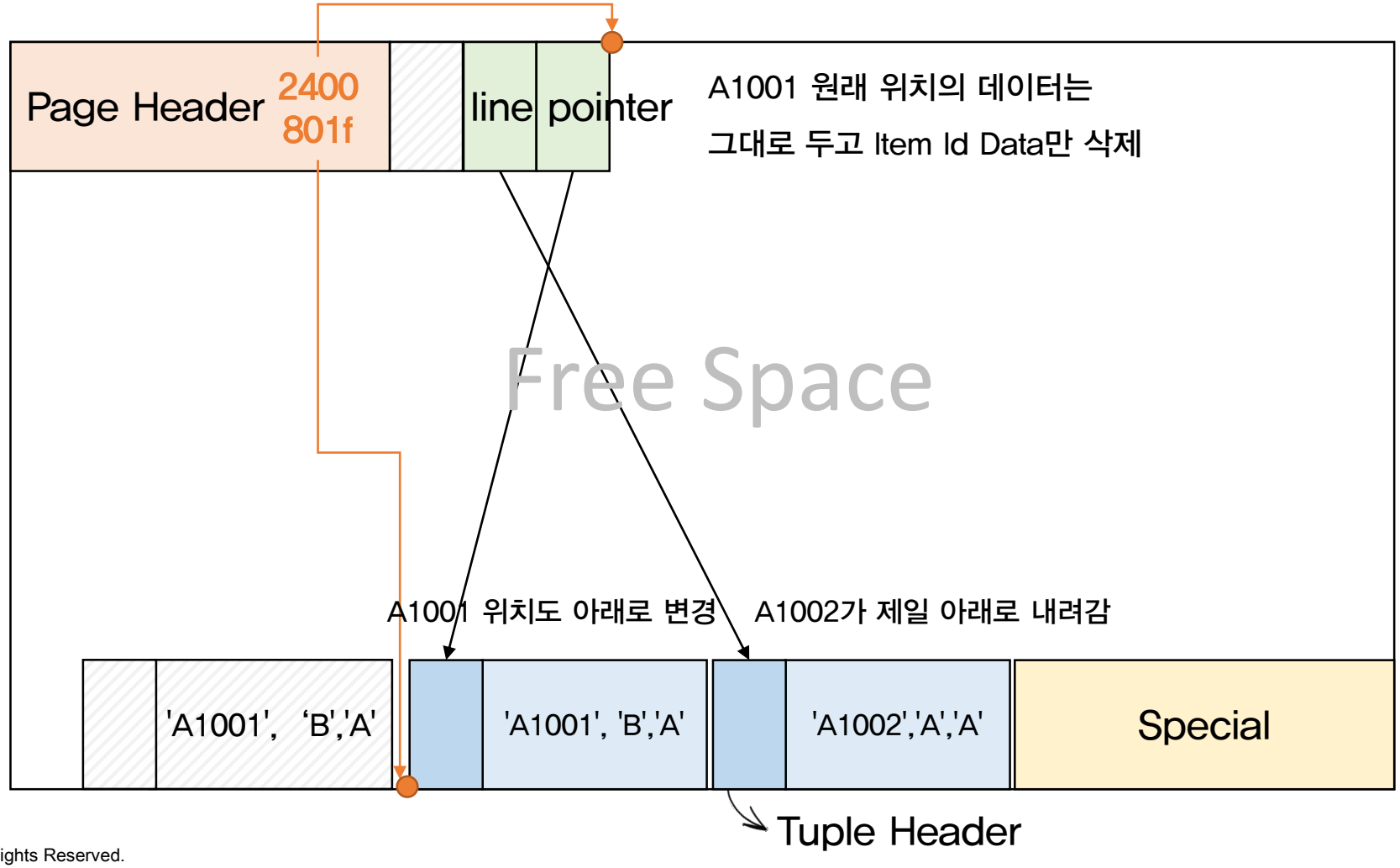
```
    xmin    | xmax | ctid  |        c1          |    c2     |    c3
------------+------+-------+--------------------+-----------+----------
 1807012524 |    0 | (0,2) | A1002              | A         | A
 1807012525 |    0 | (0,3) | A1001              | B         | A
(2 rows)
```

“A1001” item 의 xmin 변경됨

- 표준 Vacuum

  A의 line pointer는 삭제되나 공간은 남아있고, items에는 공간도 삭제된다. free space가 증가된다.

  그러나 items에서 실제로 데이터있는 것처럼 보인다.



③ 표준 vacuum

```
000000 25 00 00 00 38 2a 9d db 00 00 04 00 24 00 80 1f >%...8*......$...<
000010 00 20 04 20 00 00 00 00 03 00 01 00 c0 9f 7c 00 >. . ..........|.<
000020 80 9f 7c 00 00 00 00 00 00 00 00 00 00 00 00 00 >..|.............<
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
001f40 ad d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00 >...k............<
001f50 03 00 03 80 02 29 18 00 29 41 31 30 30 31 20 20 >.....)..)A1001  <
001f60 20 20 20 20 20 20 20 20 20 20 20 20 13 42 20 20 >             .B  <
001f70 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00 >     .A       ..<
```

A1001 원래 위치의 데이터는 그대로 두고 Item Id Data만 삭제

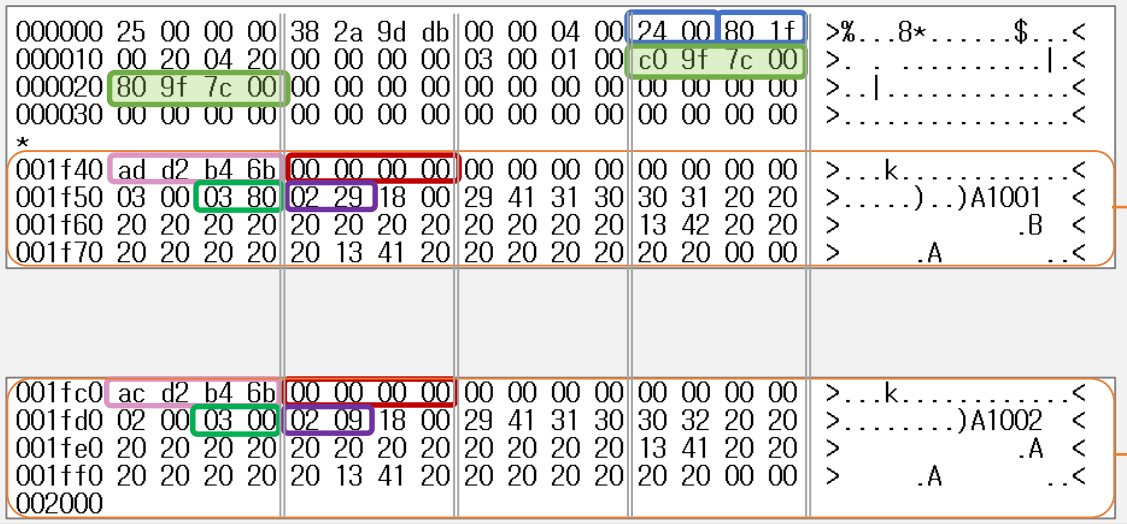
- SELECT * from heap_page_items(get_raw_page('t2', 0));

```
(postgres@[local]:5432) [postgres] > SELECT * from heap_page_items(get_raw_page('t2', 0));
 lp | lp_off | lp_flags | lp_len |   t_xmin   | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
----+--------+----------+--------+------------+--------+----------+--------+-------------+------------+--------+--------+-------
  1 |      3 |        2 |      0 |       NULL |   NULL |     NULL | NULL   |        NULL |       NULL |   NULL | NULL   |  NULL
  2 |   8128 |        1 |     62 | 1807012524 |      0 |        0 | (0,2)  |           3 |       2306 |     24 | NULL   |  NULL
  3 |   8064 |        1 |     62 | 1807012525 |      0 |        0 | (0,3)  |       32771 |      10498 |     24 | NULL   |  NULL
(3 rows)
```

- select xmin,xmax,ctid, * from t2;

```
    xmin    | xmax | ctid  |         c1          |    c2    |    c3
------------+------+-------+---------------------+----------+----------
 1807012524 |    0 | (0,2) | A1002               | A        | A
 1807012525 |    0 | (0,3) | A1001               | B        | A
(2 rows)
```

③ 표준 vacuum



A1001 원래 위치의 데이터는 그대로 두고 Item Id Data만 삭제

A1002가 제일 아래로 내려감

- SELECT * from heap_page_items(get_raw_page('t2', 0));

```
(postgres@[local]:5432) [postgres] > SELECT * from heap_page_items(get_raw_page('t2', 0));
 lp | lp_off | lp_flags | lp_len |   t_xmin   | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
----+--------+----------+--------+------------+--------+----------+--------+-------------+------------+--------+--------+-------
  1 |      3 |        2 |      0 |       NULL |   NULL |     NULL | NULL   |        NULL |       NULL |   NULL | NULL   |  NULL
  2 |   8128 |        1 |     62 | 1807012524 |      0 |        0 | (0,2)  |           3 |       2306 |     24 | NULL   |  NULL
  3 |   8064 |        1 |     62 | 1807012525 |      0 |        0 | (0,3)  |       32771 |      10498 |     24 | NULL   |  NULL
(3 rows)
```

- select xmin,xmax,ctid, * from t2;

```
    xmin    | xmax | ctid  |         c1          |    c2     |    c3
------------+------+-------+---------------------+-----------+----------
 1807012524 |    0 | (0,2) | A1002               | A         | A
 1807012525 |    0 | (0,3) | A1001               | B         | A
(2 rows)
```

③ 표준 vacuum

```
000000 25 00 00 00 38 2a 9d db 00 00 04 00 24 00 80 1f >%...8*......$...<
000010 00 20 04 20 00 00 00 00 03 00 01 00 c0 9f 7c 00 >. . ..........|.<
000020 80 9f 7c 00 00 00 00 00 00 00 00 00 00 00 00 00 >..|.............<
000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
001f40 ad d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00 >...k............<
001f50 03 00 03 80 02 29 18 00 29 41 31 30 30 31 20 20 >.....)..)A1001  <
001f60 20 20 20 20 20 20 20 20 20 20 20 20 13 42 20 20 >             .B  <
001f70 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00 >     .A       ..<
001f80 ad d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00 >...k............<
001f90 03 00 03 80 02 29 18 00 29 41 31 30 30 31 20 20 >.....)..)A1001  <
001fa0 20 20 20 20 20 20 20 20 20 20 20 20 13 42 20 20 >             .B  <
001fb0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00 >     .A       ..<
001fc0 ac d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00 >...k............<
001fd0 02 00 03 00 02 09 18 00 29 41 31 30 30 32 20 20 >........)A1002  <
001fe0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20 >             .A  <
001ff0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00 >     .A       ..<
002000
```

- A1001 원래 위치의 데이터는 그대로 두고 Item Id Data만 삭제
- A1001 위치도 아래로 변경
- A1002가 제일 아래로 내려감

- SELECT * from heap_page_items(get_raw_page('t2', 0));

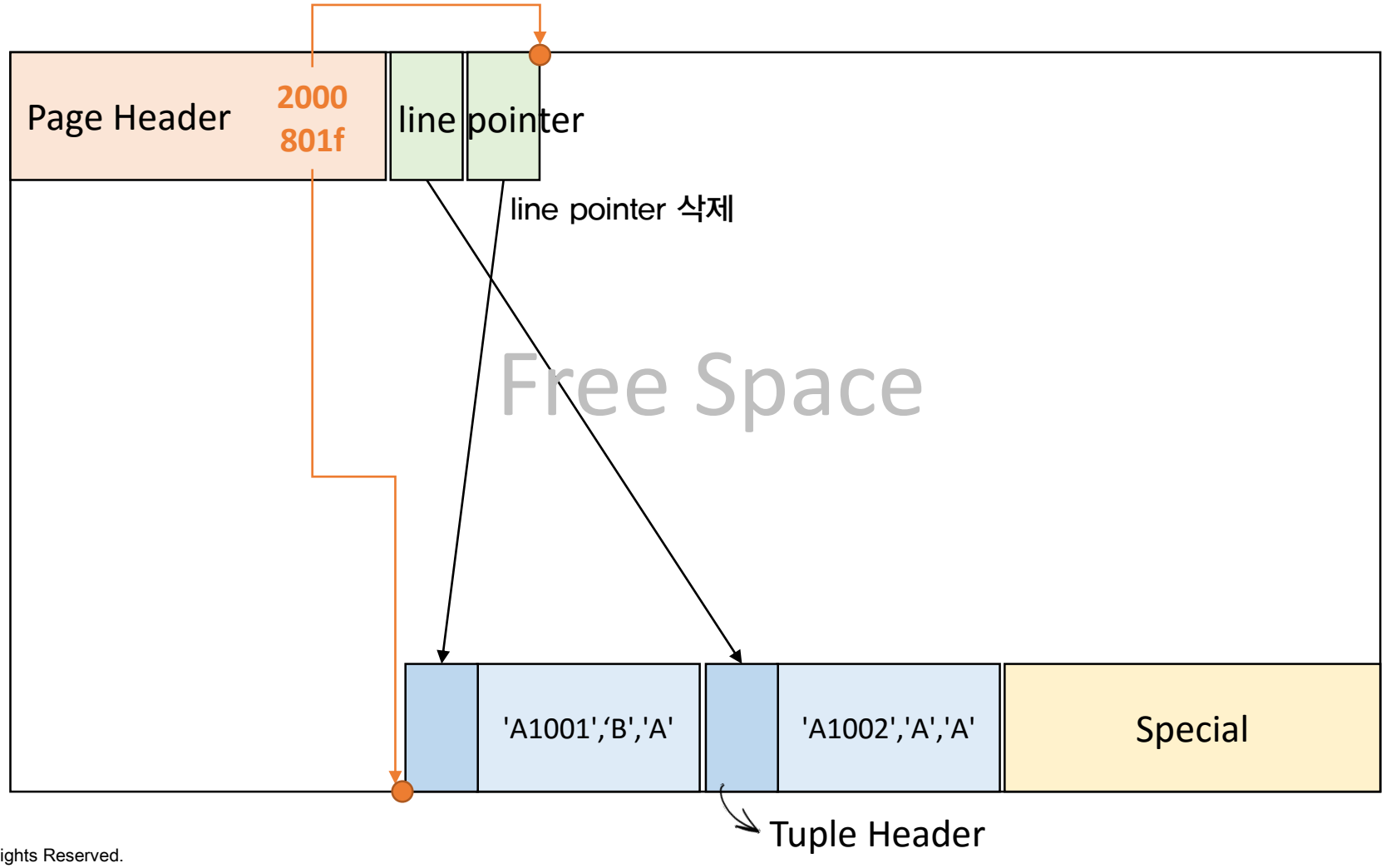
```
(postgres@[local]:5432) [postgres] > SELECT * from heap_page_items(get_raw_page('t2', 0));
 lp | lp_off | lp_flags | lp_len |   t_xmin   | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
----+--------+----------+--------+------------+--------+----------+--------+-------------+------------+--------+--------+-------
  1 |      3 |        2 |      0 |       NULL |   NULL |     NULL | NULL   |        NULL |       NULL |   NULL | NULL   |  NULL
  2 |   8128 |        1 |     62 | 1807012524 |      0 |        0 | (0,2)  |           3 |       2306 |     24 | NULL   |  NULL
  3 |   8064 |        1 |     62 | 1807012525 |      0 |        0 | (0,3)  |       32771 |      10498 |     24 | NULL   |  NULL
(3 rows)
```

- select xmin,xmax,ctid, * from t2;

```
    xmin    | xmax | ctid  |         c1          |    c2    |    c3
------------+------+-------+---------------------+----------+----------
 1807012524 |    0 | (0,2) | A1002               | A        | A
 1807012525 |    0 | (0,3) | A1001               | B        | A
(2 rows)
```

- Full Vacuum

  Full Vacuum을 하면 삭제되지 않았던 line pointer의 공간도 삭제된다.



④ Full vacuum

```
000000 00 00 00 00 00 00 00 00 00 00 00 00 20 00 80 1f  >................ ...<
000010 00 20 04 20 00 00 00 00 c0 9f 7c 00 80 9f 7c 00  >. . ........|...|.<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
001f80 ad d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001f90 02 00 03 00 02 2b 18 00 29 41 31 30 30 31 20 20  >.....+..)A1001  <
001fa0 20 20 20 20 20 20 20 20 20 20 20 20 13 42 20 20  >             .B  <
001fb0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A       ..<
001fc0 ac d2 b4 6b 00 00 00 00 00 00 00 00 00 00 00 00  >...k............<
001fd0 01 00 03 00 02 0b 18 00 29 41 31 30 30 32 20 20  >........)A1002  <
001fe0 20 20 20 20 20 20 20 20 20 20 20 20 13 41 20 20  >             .A  <
001ff0 20 20 20 20 20 13 41 20 20 20 20 20 20 20 00 00  >     .A       ..<
002000
```

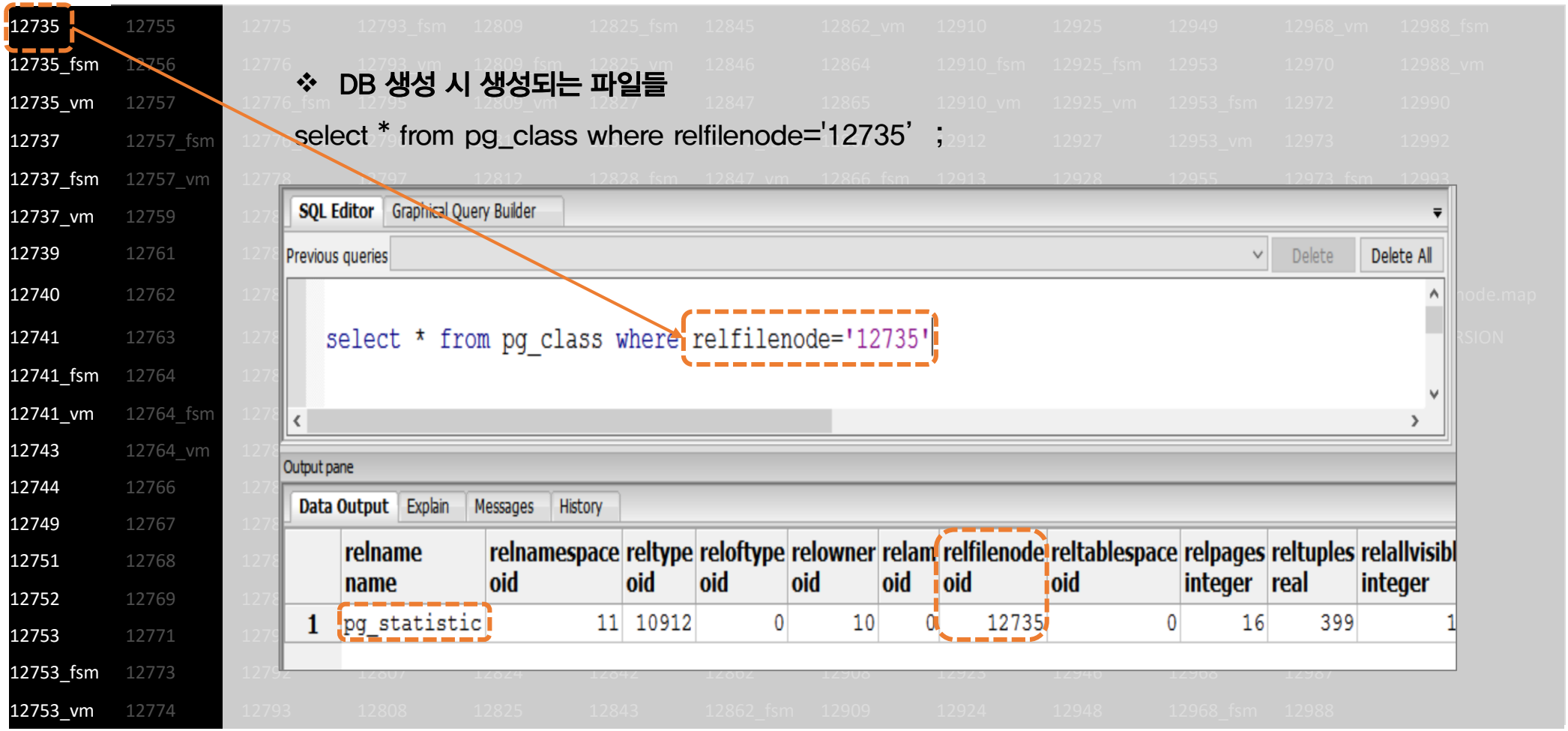
line pointer 삭제

- select xmin,xmax,ctid, * from t2;

```
    xmin    | xmax | ctid  |         c1         |    c2    |    c3
------------+------+-------+--------------------+----------+----------
 1807012524 |    0 | (0,1) | A1002              | A        | A
 1807012525 |    0 | (0,2) | A1001              | B        | A
(2 rows)
```

	표준 vacuum	Vacuum full
처리 방식	• 다른 자료가 저장될 수 있도록 빈 공간으로 표시 • OS 입장에서는 디스크의 여유 공간 확보가 불가	• 새 파일에 저장하는 방식 (pg_class의 relfilenode값이 변경) • OS 입장에서 디스크의 여유 공간 확보가 가능 • 최적의 물리적 크기로 테이블 생성
처리 속도	Vacuum full 보다는 시간이 적게 걸림	처리 속도가 매우 느려 시간이 오래 걸림
Lock 여부	여러 다른 작업들과 함께 사용 가능 select, update, insert, delete (단, ALTER TABLE는 안됨)	해당 테이블에 배타적 잠금을 지정하기 때문에 어떤 작업도 함께 사용할 수 없음

❖ DB 생성 후, 무엇이 생성되는가?



❖ 표준 Vacuum, Full Vacuum 후, 각각의 결과

Vacuum 전

```
+ ls -l /usr/local/pgsql/data/base/1303366/1303367
-rw------- 1 postgres postgres 8192 Oct 13 10:49 /usr/local/pgsql/data/base/1303366/1303367
```

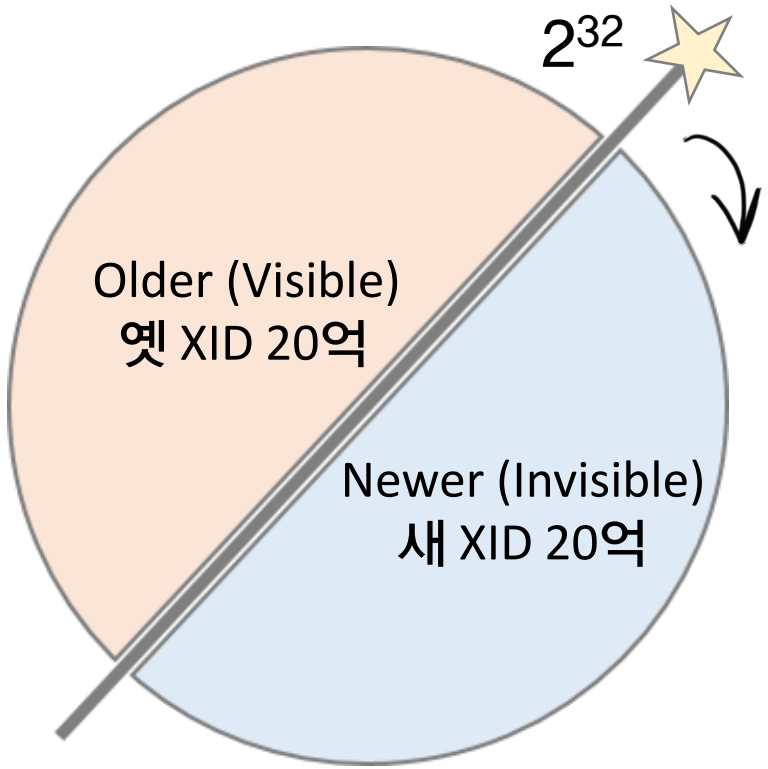
표준 Vacuum

```
vacuum myt;
VACUUM
+ ls -l /usr/local/pgsql/data/base/1303366/1303367 /usr/local/pgsql/data/base/1303366/1303367_fsm /usr/local/pgsql/data/base/1303366/1303367_vm
-rw------- 1 postgres postgres  8192 Oct 13 10:49 /usr/local/pgsql/data/base/1303366/1303367
-rw------- 1 postgres postgres 24576 Oct 13 10:49 /usr/local/pgsql/data/base/1303366/1303367_fsm
-rw------- 1 postgres postgres  8192 Oct 13 10:49 /usr/local/pgsql/data/base/1303366/1303367_vm
select pg_relation_filepath('public.myt') ;
 pg_relation_filepath
----------------------
 base/1303366/1303367
(1 row)
```

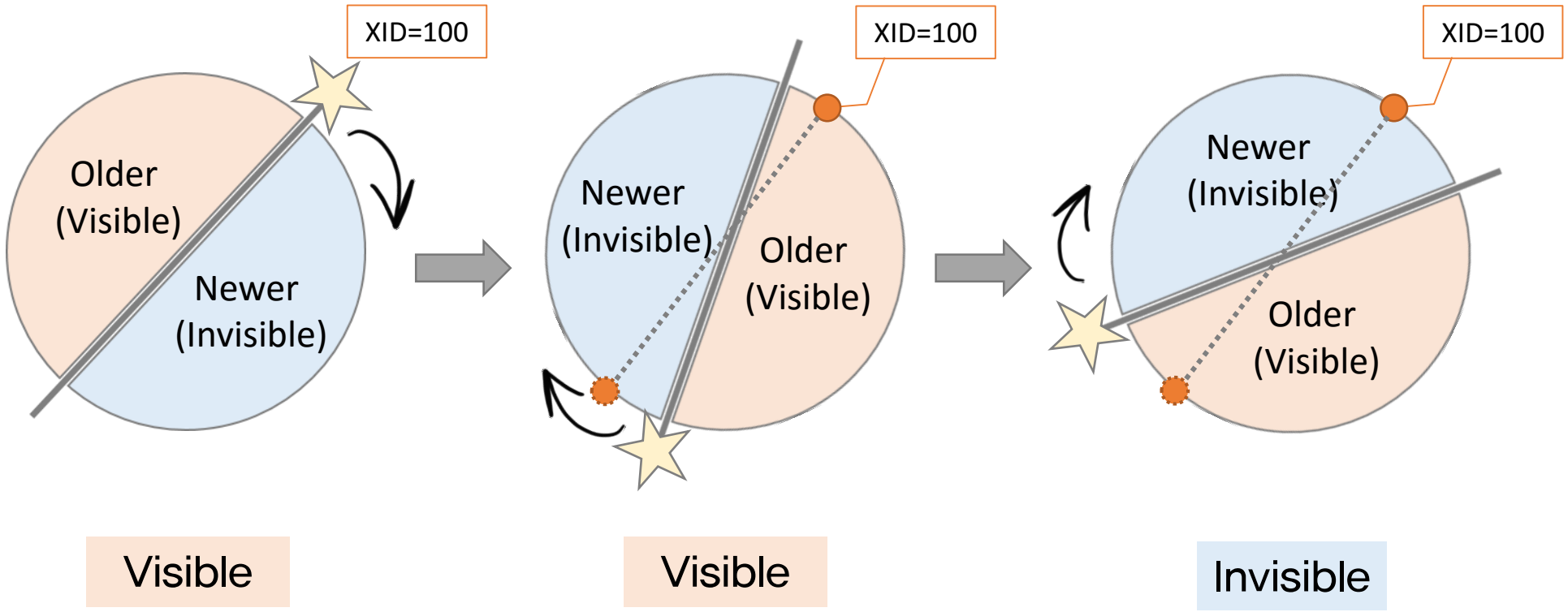
Full Vacuum

```
vacuum full myt;
VACUUM
+ ls -l /usr/local/pgsql/data/base/1303366/1303370
-rw------- 1 postgres postgres 8192 Oct 13 10:49 /usr/local/pgsql/data/base/1303366/1303370
select pg_relation_filepath('public.myt') ;
 pg_relation_filepath
----------------------
 base/1303366/1303370
(1 row)
```

- 트랜잭션에 대한 MVCC 기법:
  XID(트랜잭션 ID)를 숫자로 처리하고 비교
- 2^{32} 까지 표현이 가능 (대략 40억)
- 옛 XID(20억) + 새 XID(20억)
- 옛 XID 20억은 보이고, 새 XID 20억은 안보이게 처리
- 계속 순환하면서 사용

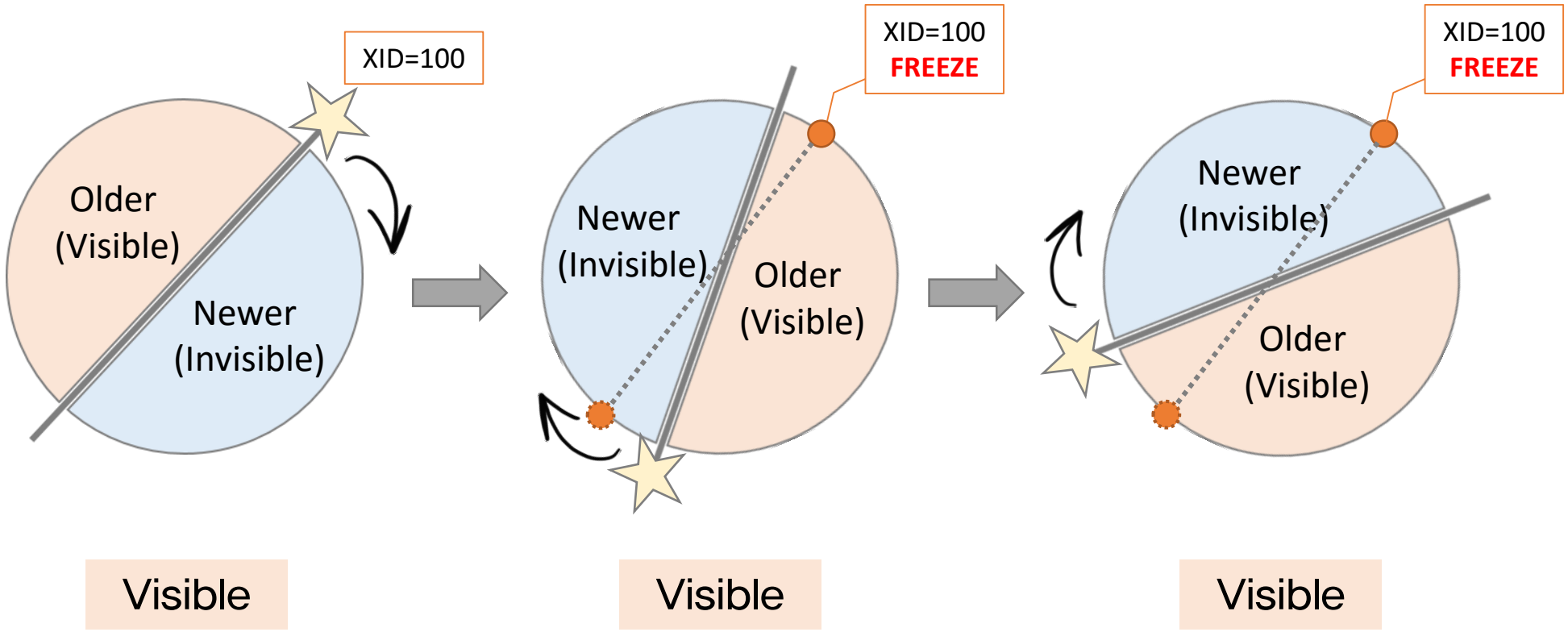


출처: http://www.slideshare.net/pgdayasia/introduction-to-vacuum-freezing-and-xid



출처: http://www.slideshare.net/hadoopxnttdata/postgresql-xid-wraparound-another-issue

- XID wraparound 때문에 오래된 데이터에 대한 손실 발생
- Insert-only 테이블에서만 발생

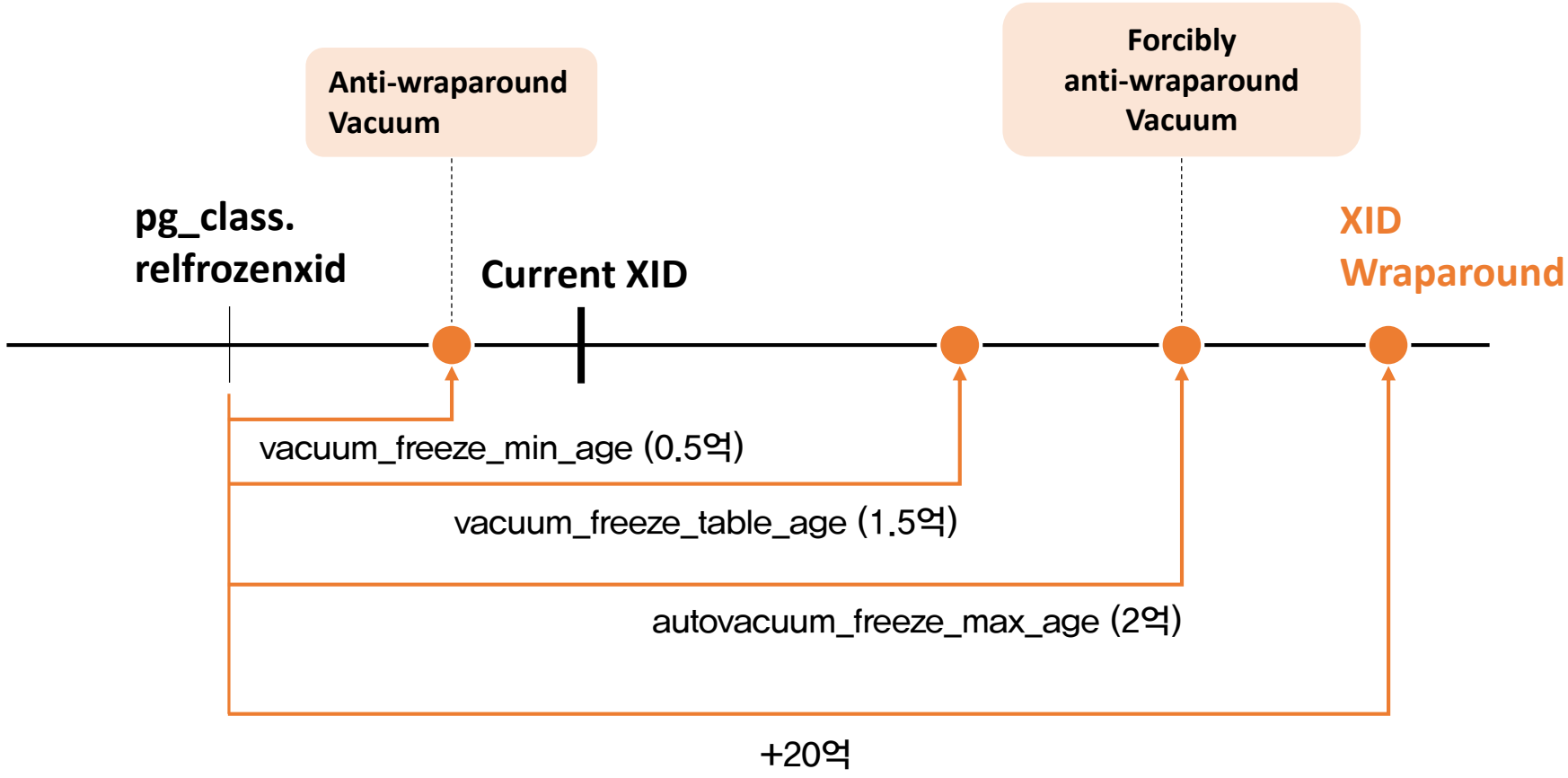


출처: http://www.slideshare.net/hadoopxnttdata/postgresql-xid-wraparound-another-issue

- 튜플을 “FREEZE”로 표시하고, 이 FrozenXID는 일반 XID 비교 대상에서 항상 제외되어 항상 보여짐
- 20억 트랜잭션이 생기기 전에 FrozenXID 즉, 영구 보관용 자료로 변경하면 XID 겹침 오류 방지
- 이러한 변경 작업(data freezing)은 VACUUM FREEZE 명령으로 처리

Vacuum Count = 0

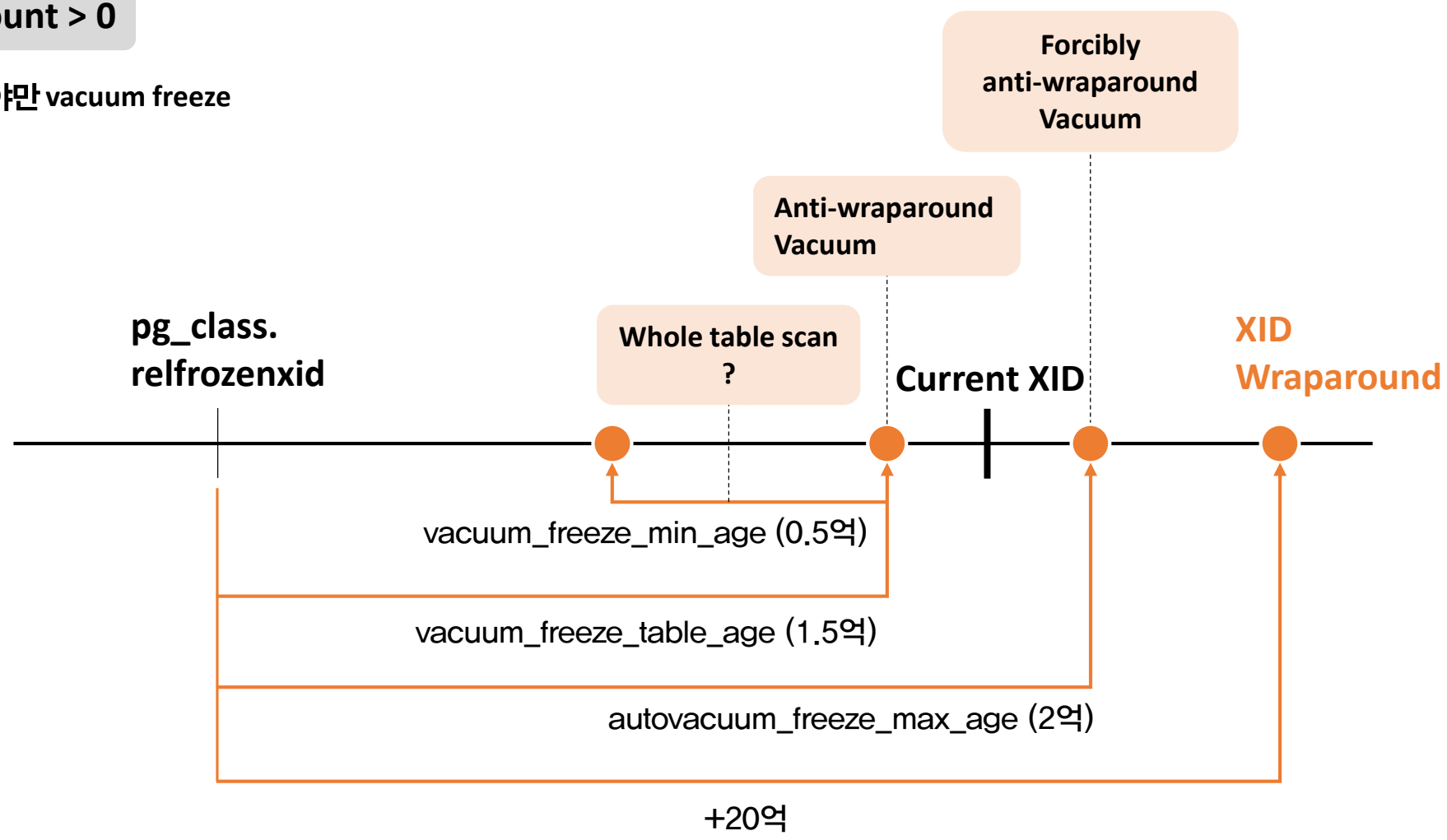
0.5 억이 넘어야만 vacuum freeze



출처: http://www.slideshare.net/pgdayasia/introduction-to-vacuum-freezing-and-xid

Vacuum Count > 0

1.5 억이 넘어야만 vacuum freeze



출처: http://www.slideshare.net/pgdayasia/introduction-to-vacuum-freezing-and-xid

❖ Vacuum Count = 0 시나리오

- create table t2 (c1 char(15),c2 char(15),c3 char(689));
- INSERT INTO T2 SELECT 'Tuple'||generate_series(1001,2000),'A',' ';

- psql –x < ./fsm_vm/tab.sql

```
[postgres@mylinux 2.pg]$ psql -x < ./fsm_vm/tab.sql
Timing is on.
-[ RECORD 1 ]----+------------------------------
relname          | t2
relfilenode      | 16393
frozenxid_h      | 00000710
frozenxid_d      | 1808
age              | 3
last_vacuum      |
last_autovacuum  |
last_autoanalyze | 2016-10-20 09:31:23.255484+09
relpages         | 100
reltuples        | 1000
n_tup_ins        | 1000
n_tup_upd        | 0
n_tup_del        | 0
seq_scan         | 2

Time: 16.396 ms
```

```
 lp | lp_off | lp_flags | lp_len |  t_xmin  |  t_xmax  | t_ctid | t_infomask2 | t_infomask
----+--------+----------+--------+----------+----------+--------+-------------+------------
  1 | 1d10   |        1 | 02ed   | 00000711 | 00000000 | (0,1)  | 0003        | 0902
  2 | 1a20   |        1 | 02ed   | 00000711 | 00000000 | (0,2)  | 0003        | 0902
  3 | 1730   |        1 | 02ed   | 00000711 | 00000000 | (0,3)  | 0003        | 0902
  4 | 1440   |        1 | 02ed   | 00000711 | 00000000 | (0,4)  | 0003        | 0902
  5 | 1150   |        1 | 02ed   | 00000711 | 00000000 | (0,5)  | 0003        | 0902
  6 | 0e60   |        1 | 02ed   | 00000711 | 00000000 | (0,6)  | 0003        | 0902
  7 | 0b70   |        1 | 02ed   | 00000711 | 00000000 | (0,7)  | 0003        | 0902
  8 | 0880   |        1 | 02ed   | 00000711 | 00000000 | (0,8)  | 0003        | 0902
  9 | 0590   |        1 | 02ed   | 00000711 | 00000000 | (0,9)  | 0003        | 0902
 10 | 02a0   |        1 | 02ed   | 00000711 | 00000000 | (0,10) | 0003        | 0902
(10 rows)
```

- Jump Current Xid ➔ 52,428,800 (pg_resetxlog)
- psql –x < ./fsm_vm/tab.sql

```
[postgres@mylinux 2.pg]$ psql -x < ./fsm_vm/tab.sql
Timing is on.
-[ RECORD 1 ]----+------------------------------
relname          | t2
relfilenode      | 16393
frozenxid_h      | 00000710
frozenxid_d      | 1808
age              | 52426993
last_vacuum      |
last_autovacuum  |
last_autoanalyze | 2016-10-20 09:31:23.255484+09
relpages         | 100
reltuples        | 1000
n_tup_ins        | 1000
n_tup_upd        | 0
n_tup_del        | 0
seq_scan         | 2

Time: 16.309 ms
```

❖ Vacuum Count = 0 시나리오

- update t2 set c2='B' where c1 >= 'Tuple1700' ;
- psql –x < ./fsm_vm/tab.sql

```
[postgres@mylinux 2.pg]$ psql -c "update t2 set c2='B' where c1 >= 'Tuple1700' ;"
UPDATE 301
[postgres@mylinux 2.pg]$ psql -x < ./fsm_vm/tab.sql
Timing is on.
-[ RECORD 1 ]----+------------------------------
relname          | t2
relfilenode      | 16393
frozenxid_h      | 00250f82
frozenxid_d      | 2428802
age              | 50000001
last_vacuum      |
last_autovacuum  | 2016-10-20 09:34:37
last_autoanalyze | 2016-10-20 09:34:37.336421+09
relpages         | 131
reltuples        | 1000
n_tup_ins        | 1000
n_tup_upd        | 301
n_tup_del        | 0
seq_scan         | 3

Time: 14.566 ms
```

- ./fsm_vm/pageinspect.sql

```
> \\i ./fsm_vm/pageinspect.sql
> !
Timing is on.
    lsn     | lower | upper | prune_xid
------------+-------+-------+-----------
 0/207BF50 | 0040  | 02a0  | 00000000
(1 row)

Time: 5.487 ms
 lp | lp_off | lp_flags | lp_len |  t_xmin  |  t_xmax  | t_ctid | t_infomask2 | t_infomask
----+--------+----------+--------+----------+----------+--------+-------------+------------
  1 | 1d10   |        1 | 02ed   | 00000711 | 00000000 | (0,1)  | 0003        | 0b02
  2 | 1a20   |        1 | 02ed   | 00000711 | 00000000 | (0,2)  | 0003        | 0b02
  3 | 1730   |        1 | 02ed   | 00000711 | 00000000 | (0,3)  | 0003        | 0b02
  4 | 1440   |        1 | 02ed   | 00000711 | 00000000 | (0,4)  | 0003        | 0b02
  5 | 1150   |        1 | 02ed   | 00000711 | 00000000 | (0,5)  | 0003        | 0b02
  6 | 0e60   |        1 | 02ed   | 00000711 | 00000000 | (0,6)  | 0003        | 0b02
  7 | 0b70   |        1 | 02ed   | 00000711 | 00000000 | (0,7)  | 0003        | 0b02
  8 | 0880   |        1 | 02ed   | 00000711 | 00000000 | (0,8)  | 0003        | 0b02
  9 | 0590   |        1 | 02ed   | 00000711 | 00000000 | (0,9)  | 0003        | 0b02
 10 | 02a0   |        1 | 02ed   | 00000711 | 00000000 | (0,10) | 0003        | 0b02
(10 rows)

Time: 1.689 ms
```

❖ Vacuum Count 〉 0 시나리오

- create table t2 (c1 char(15),c2 char(15),c3 char(689));
- INSERT INTO T2 SELECT 'Tuple'||generate_series(1001,2000),'A',' ';

- psql –x 〈 ./fsm_vm/tab.sql

```
[postgres@mylinux 2.pg]$ psql -x < ./fsm_vm/tab.sql
Timing is on.
-[ RECORD 1 ]----+------------------------------
relname          | t2
relfilenode      | 16393
frozenxid_h      | 00000710
frozenxid_d      | 1808
age              | 3
last_vacuum      |
last_autovacuum  |
last_autoanalyze | 2016-10-20 09:37:43.640506+09
relpages         | 100
reltuples        | 1000
n_tup_ins        | 1000
n_tup_upd        | 0
n_tup_del        | 0
seq_scan         | 2

Time: 15.607 ms
```

- psql –c "vacuum freeze t2;"
- psql –x 〈 ./fsm_vm/tab.sql

```
[postgres@mylinux 2.pg]$ psql -x < ./fsm_vm/tab.sql
Timing is on.
-[ RECORD 1 ]----+------------------------------
relname          | t2
relfilenode      | 16393
frozenxid_h      | 00000713
frozenxid_d      | 1811
age              | 0
last_vacuum      | 2016-10-20 09:38:22
last_autovacuum  |
last_autoanalyze | 2016-10-20 09:37:43.640506+09
relpages         | 100
reltuples        | 1000
n_tup_ins        | 1000
n_tup_upd        | 0
n_tup_del        | 0
seq_scan         | 2

Time: 20.123 ms
```

❖ Vacuum Count 〉 0 시나리오

- Jump Current Xid ➔ 52,428,800 (pg_resetxlog)
- psql –x 〈 ./fsm_vm/tab.sql

```
[postgres@mylinux 2.pg]$ psql -x < ./fsm_vm/tab.sql
Timing is on.
-[ RECORD 1 ]----+----------------------------
relname          | t2
relfilenode      | 16393
frozenxid_h      | 00000713
frozenxid_d      | 1811
age              | 52426990
last_vacuum      | 2016-10-20 09:38:22
last_autovacuum  |
last_autoanalyze | 2016-10-20 09:37:43.640506+09
relpages         | 100
reltuples        | 1000
n_tup_ins        | 1000
n_tup_upd        | 0
n_tup_del        | 0
seq_scan         | 2
```

- psql –c "update t2 set c2='B' where c1 〉 'Tuple1700' ;"
- psql –x 〈 ./fsm_vm/tab.sql

```
[postgres@mylinux 2.pg]$ psql -x < ./fsm_vm/tab.sql
Timing is on.
-[ RECORD 1 ]----+----------------------------
relname          | t2
relfilenode      | 16393
frozenxid_h      | 00000713
frozenxid_d      | 1811
age              | 52426992
last_vacuum      | 2016-10-20 09:38:22
last_autovacuum  | 2016-10-20 09:41:05
last_autoanalyze | 2016-10-20 09:41:05.291362+09
relpages         | 130
reltuples        | 1000
n_tup_ins        | 1000
n_tup_upd        | 300
n_tup_del        | 0
seq_scan         | 3

Time: 38.554 ms
```

❖ Vacuum Count 〉 0 시나리오

- Jump Current Xid ➔ 157,286,400 (pg_resetxlog)
- psql –x 〈 ./fsm_vm/tab.sql

```
[postgres@mylinux 2.pg]$ psql -x < ./fsm_vm/tab.sql
Timing is on.
-[ RECORD 1 ]----+-----------------------------
relname          | t2
relfilenode      | 16393
frozenxid_h      | 00000713
frozenxid_d      | 1811
age              | 157284590
last_vacuum      | 2016-10-20 09:38:22
last_autovacuum  | 2016-10-20 09:41:05
last_autoanalyze | 2016-10-20 09:41:05.291362+09
relpages         | 130
reltuples        | 1000
n_tup_ins        | 1000
n_tup_upd        | 300
n_tup_del        | 0
seq_scan         | 3

Time: 33.891 ms
```

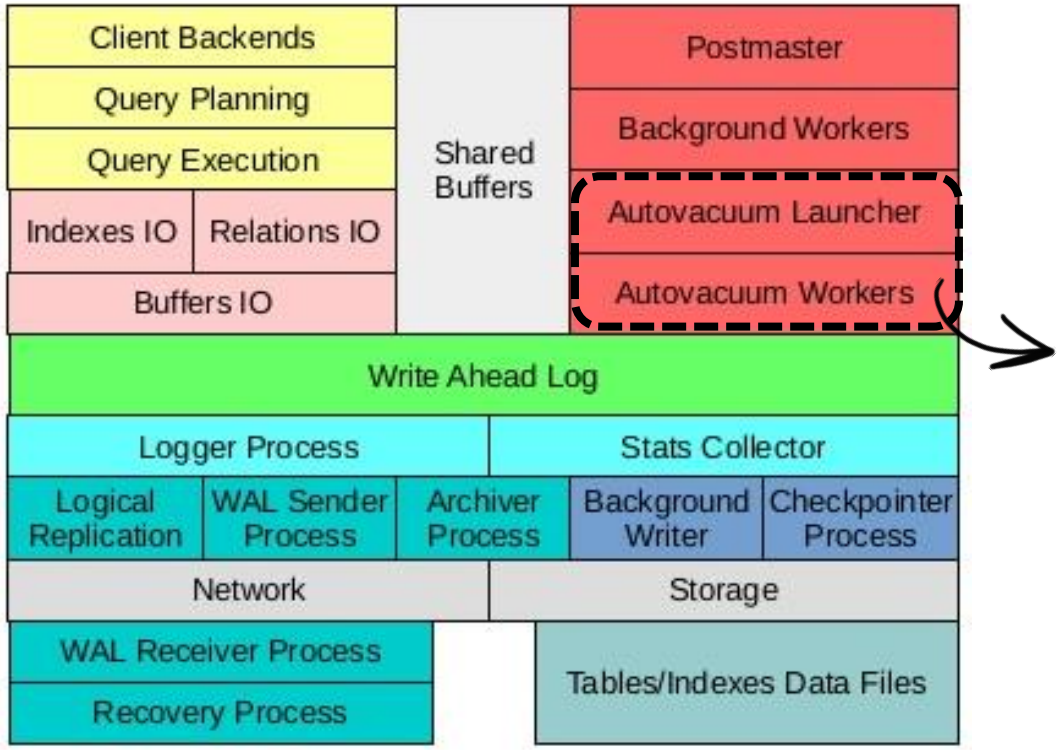
- psql –c "update t2 set c2='C' where c1 〉 'Tuple1700' ;"
- psql –x 〈 ./fsm_vm/tab.sql

```
[postgres@mylinux 2.pg]$ psql -x < ./fsm_vm/tab.sql
Timing is on.
-[ RECORD 1 ]----+-----------------------------
relname          | t2
relfilenode      | 16393
frozenxid_h      | 06650f82
frozenxid_d      | 107286402
age              | 50000001
last_vacuum      | 2016-10-20 09:38:22
last_autovacuum  | 2016-10-20 09:44:04
last_autoanalyze | 2016-10-20 09:44:04.400175+09
relpages         | 100
reltuples        | 1000
n_tup_ins        | 1000
n_tup_upd        | 600
n_tup_del        | 0
seq_scan         | 4

Time: 27.508 ms
```

AUTOVACUUM이란? VACUUM 명령과 ANALYZE 명령을 주기적으로 자동 실행

- 주기적인 표준 Vacuum 작업을 하여 빈 공간을 확보하여 최대한 Vacuum Full 작업을 방지
- 최소의 디스크를 쓰는 것이 아닌 최적의 디스크 공간만 사용



- **Autovacuum launcher**
Autovacuum_naptime 값으로 지정한 초 간격으로 한 번에 하나의 데이터베이스를 작업할 수 있도록 worker 프로세스의 실행 시간을 관리

- **Autovacuum workers**
Autovacuum_max_workers worker 프로세스 최대 개수

출처: http://www.slideshare.net/alexeylesovsky/deep-dive-into-postgresql-statistics-60849690

- Autovacuum 데몬은 예상치 못한 상황에 대해 자동으로 Vacuum 작업을 진행하여 위와 같은 문제를 피할 수 있다.
- 정확한 데이터베이스 사용량을 파악하지 않고 autovacuum을 끄면 안 된다.
- Autovacuum 데몬을 사용하지 않는다면, 해당 DB 서버에서 사용하고 있는 모든 DB에 대해서 Vacuum 작업을 해야 한다.
- Autovacuum 데몬은 테이블 자료 추가/변경/삭제 된 경우, 무조건 작업의 고려 대상으로 판단한다.
  (단, 임시 테이블은 대상에서 제외)

05. Vacuum – 5) Autovacuum

```
(postgres@[local]:5432) [postgres] > select name, setting, context from pg_settings where category ~'Autovacuum';
                name                 |  setting  |  context
-------------------------------------+-----------+------------
 autovacuum                          | on        | sighup
 autovacuum_analyze_scale_factor     | 0.1       | sighup
 autovacuum_analyze_threshold        | 50        | sighup
 autovacuum_freeze_max_age           | 200000000 | postmaster
 autovacuum_max_workers              | 3         | postmaster
 autovacuum_multixact_freeze_max_age | 400000000 | postmaster
 autovacuum_naptime                  | 60        | sighup
 autovacuum_vacuum_cost_delay        | 20        | sighup
 autovacuum_vacuum_cost_limit        | -1        | sighup
 autovacuum_vacuum_scale_factor      | 0.2       | sighup
 autovacuum_vacuum_threshold         | 20        | sighup
(11 rows)
```

autovacuum_analyze_scale_factor
한 테이블 내에서 autovacuum_analyze_threshold 값이 초과되어 analyze 작업을 시작하려고 할 때, 추가적으로 테이블 크기의 변화량 (기본값은 0.1 = 테이블 크기의 10%)

autovacuum_vacuum_threshold / autovacuum_analyze_threshold
한 테이블 내에서 업데이트/삭제된 튜플이 설정 값만큼 된다면 vacuum/anaylze 작업하는 수행, 작업 시작할 최소 변경 수 (기본 값 50)

autovacuum_freeze_max_age
한 테이블 내에서 마지막 vacuum 작업으로 pg_class.relfrozenxid 값이 지정된 뒤로, 이 설정 값보다 더 커지면 vacuum 작업 수행

autovacuum_naptime
autovacuum 데몬에 대한 “activity rounds 간격” 즉, autovacuum 데몬이 한번 작업하고 쉬는 시간 (기본 값 1분 – 초(秒)로 설정)

autovacuum_vacuum_cost_delay
autovacuum 데몬의 초과 비용 사용시, 멈추는 최대 시간 (기본 값 20 millisecond)

autovacuum_vacuum_cost_limit
autovacuum 데몬이 사용할 수 있는 최대 비용 (기본 값 –1)

autovacuum_vacuum_scale_factor
한 테이블 내에서 autovacuum_vacuum_threshold 값이 초과되어 vacuum 작업을 시작하려고 할 때, 추가적으로 테이블 크기의 변화량 (기본값은 0.2 = 테이블 크기의 20%)

❖ Autovacuum의 작업 기준

1. 테이블 나이가 autovacuum_freeze_max_age 설정으로 지정한 트랜잭션 수보다 많다면, 그 테이블은 무조건 Vacuum 작업
    - 테이블의 나이: relfrozenxid 칼럼 값을 age () 함수로 조사한 값

2. 테이블의 자료가 변경되어, Vacuum 임계치를 초과했다면, Vacuum 작업
    * Vacuum 임계치 = Vacuum 초기 임계치 + Vacuum 배율값 * 튜플수
        - Vacuum 초기 임계치: **autovacuum_vacuum_threshold**
        - Vacuum 배율값: **autovacuum_vacuum_scale_factor**
        - 튜플수: pg_class.reltuples

3. 테이블의 자료가 변경되어, Analyze 임계치를 초과했다면, Analyze 작업
    * Analyze 임계치 = Analyze 초기 임계치 + Analyze 배율값 * 튜플수
        - Analyze 초기 임계치: **autovacuum_analyze_threshold**
        - Analyze 배율값: **autovacuum_analyze_scale_factor**
        - 튜플수: pg_class.reltuples

autovacuum_vacuum_scale_factor = 0.2
autovacuum_vacuum_threshold = 50

1. 1000건 insert

```
insert into t2 SELECT 'A' || generate_series(1000,1000+1000-1),'A','A';
INSERT 0 1000
Time: 48.101 ms
 relname | relfilenode | frozenxid | age | last_vacuum | last_autovacuum | last_autoanalyze | pages | tuples | n_ins | n_upd | n_del | seq_scan
---------+-------------+-----------+-----+-------------+-----------------+------------------+-------+--------+-------+-------+-------+----------
 t2      |       16436 | 0000075a  |   2 |             |                 |                  |     0 |      0 |  1000 |     0 |     0 |        1
(1 row)

Time: 19.266 ms
```

2. autovacuum_naptime=60s 이상 wait 후 확인 : autovacuum **NOT OK**, autoanalyze **OK**.

```
 relname | relfilenode | frozenxid | age | last_vacuum | last_autovacuum |  last_autoanalyze   | pages | tuples | n_ins | n_upd | n_del | seq_scan
---------+-------------+-----------+-----+-------------+-----------------+---------------------+-------+--------+-------+-------+-------+----------
 t2      |       16436 | 0000075a  |   3 |             |                 | 2016-10-14 14:36:26 |     9 |   1000 |  1000 |     0 |     0 |        1
(1 row)
```

autovacuum_vacuum_scale_factor = 0.2
autovacuum_vacuum_threshold = 50

3. autovacuum_vacuum_scale_factor=0.2 인 200건 update → wait 후 확인 : autovacuum **NOT OK.**

```
update t2 set c2 = 'B' where c1 < 'A1200' ;
UPDATE 200
Time: 15.445 ms
 relname | relfilenode | frozenxid | age | last_vacuum | last_autovacuum |  last_autoanalyze   | pages | tuples | n_ins | n_upd | n_del | seq_scan
---------+-------------+-----------+-----+-------------+-----------------+---------------------+-------+--------+-------+-------+-------+----------
 t2      |       16436 | 0000075a  |   5 |             |                 | 2016-10-14 14:38:26 |    10 |   1000 |  1000 |   200 |     0 |        3
(1 row)
```

4. autovacuum_vacuum_threshold=50 인 50건 update → wait 후 확인 : autovacuum **NOT OK.**

```
update t2 set c2 = 'C' where c1 < 'A1050' ;
UPDATE 50
Time: 7.749 ms
 relname | relfilenode | frozenxid | age | last_vacuum | last_autovacuum |  last_autoanalyze   | pages | tuples | n_ins | n_upd | n_del | seq_scan
---------+-------------+-----------+-----+-------------+-----------------+---------------------+-------+--------+-------+-------+-------+----------
 t2      |       16436 | 0000075a  |   6 |             |                 | 2016-10-14 14:38:26 |    10 |   1000 |  1000 |   250 |     0 |        5
(1 row)
```

autovacuum_vacuum_scale_factor = 0.2
autovacuum_vacuum_threshold = 50

5. Only 1건 update → wait 후 확인 : autovacuum **OK.**

```
update t2 set c2 = 'D' where c1 < 'A1001' ;
UPDATE 1
Time: 6.466 ms
 relname | relfilenode | frozenxid | age | last_vacuum |   last_autovacuum    |  last_autoanalyze    | pages | tuples | n_ins | n_upd | n_del | seq_scan
---------+-------------+-----------+-----+-------------+----------------------+----------------------+-------+--------+-------+-------+-------+----------
 t2      |       16436 | 0000075a  |   7 |             | 2016-10-14 14:40:26 | 2016-10-14 14:38:26 |    11 |   1000 |  1000 |   251 |     0 |        7
(1 row)
```

❖ autovacuum vacuum 대상 조건 :

변경건수(insert 제외) 〉 (reltuples * 0.2(autovacuum_vacuum_scale_factor)) + 50 (autovacuum_vacuum_scale_factor)

autovacuum_analyze_scale_factor = 0.1
autovacuum_analyze_threshold = 50

1. 1000건 insert

```
insert into  t2 SELECT 'A' || generate_series(1000,1000+1000-1),'A','A';
INSERT 0 1000
Time: 4.422 ms
 relname | relfilenode | relfrozenxid | age | last_vacuum | last_autovacuum | last_autoanalyze | reltuples | n_tup_ins | n_tup_upd | n_tup_del
---------+-------------+--------------+-----+-------------+-----------------+------------------+-----------+-----------+-----------+-----------
 t2      |      136794 |   1807012750 |   2 | NULL        | NULL            | NULL             |         0 |      1000 |         0 |         0
(1 row)
```

2. autovacuum_naptime=60s 이상 wait 후 확인 : autoanalyze **OK.**

```
 relname | relfilenode | relfrozenxid | age | last_vacuum | last_autovacuum |       last_autoanalyze        | reltuples | n_tup_ins | n_tup_upd | n_tup_del
---------+-------------+--------------+-----+-------------+-----------------+-------------------------------+-----------+-----------+-----------+-----------
 t2      |      136794 |   1807012750 |   3 | NULL        | NULL            | 2016-07-14 14:30:09.897825+09 |      1000 |      1000 |         0 |         0
(1 row)
```

autovacuum_analyze_scale_factor = 0.1
autovacuum_analyze_threshold = 50

3. autovacuum_analyze_scale_factor=0.1 인 100건 insert → wait 후 확인 : autoanalyze **NOT OK.**

```
 insert into  t2 SELECT 'A' || generate_series(2000,2000+100-1),'A','A';
INSERT 0 100
Time: 1.122 ms
 relname | relfilenode | relfrozenxid | age | last_vacuum | last_autovacuum |       last_autoanalyze        | reltuples | n_tup_ins | n_tup_upd | n_tup_del
---------+-------------+--------------+-----+-------------+-----------------+-------------------------------+-----------+-----------+-----------+-----------
 t2      |      136794 |   1807012750 |   4 | NULL        | NULL            | 2016-07-14 14:30:09.897825+09 |      1000 |      1000 |         0 |         0
(1 row)
```

4. autovacuum_analyze_threshold=50 인 50건 insert → wait 후 확인 : autoanalyze **NOT OK.**

```
 insert into  t2 SELECT 'A' || generate_series(3000,3000+ 50-1),'A','A';
INSERT 0 50
Time: 0.818 ms

 relname | relfilenode | relfrozenxid | age | last_vacuum | last_autovacuum |       last_autoanalyze        | reltuples | n_tup_ins | n_tup_upd | n_tup_del
---------+-------------+--------------+-----+-------------+-----------------+-------------------------------+-----------+-----------+-----------+-----------
 t2      |      136794 |   1807012750 |   5 | NULL        | NULL            | 2016-07-14 14:30:09.897825+09 |      1000 |      1100 |         0 |         0
(1 row)
```

autovacuum_analyze_scale_factor = 0.1
autovacuum_analyze_threshold = 50

5. Only 1건 insert → wait 후 확인 : autoanalyze OK.

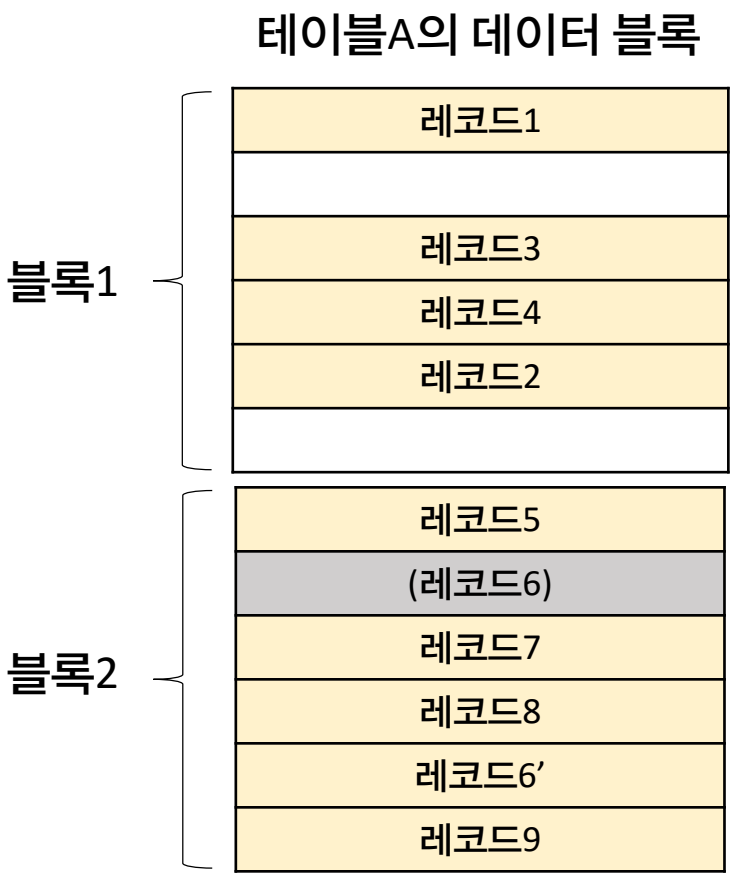
```
insert into t2 SELECT 'A' || generate_series(4000,4000      ),'A','A';
INSERT 0 1
Time: 0.669 ms
 relname | relfilenode | relfrozenxid | age | last_vacuum | last_autovacuum |       last_autoanalyze        | reltuples | n_tup_ins | n_tup_upd | n_tup_del
---------+-------------+--------------+-----+-------------+-----------------+-------------------------------+-----------+-----------+-----------+-----------
 t2      |      136794 |   1807012750 |   6 | NULL        | NULL            | 2016-07-14 14:30:09.897825+09 |      1000 |      1150 |         0 |         0
(1 row)
```

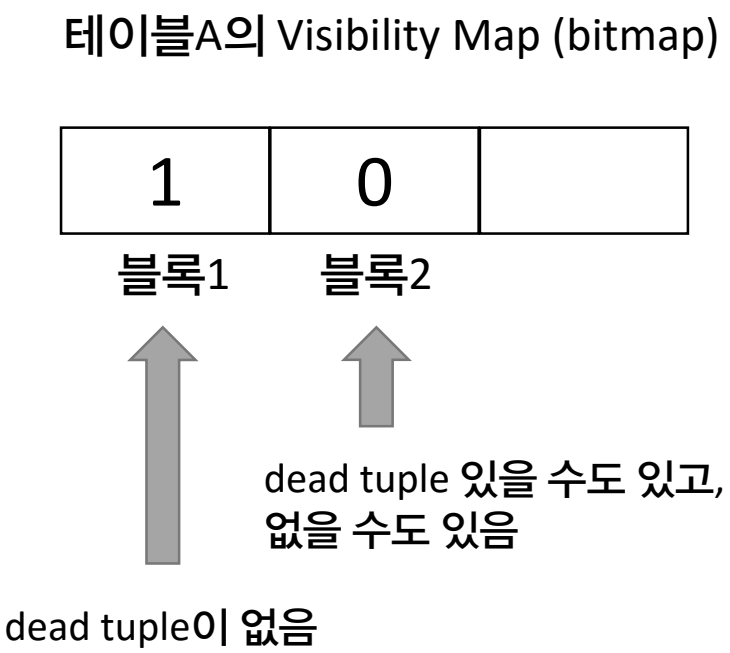
❖ autovacuum analyze 대상 조건 :

변경건수(insert포함) 〉 (reltuples * 0.1(autovacuum_analyze_scale_factor)) + 50 (autovacuum_analyze_scale_factor)

Visibility Map (≒ bitmap index)

- 각각의 heap relation은 VM을 통해 어떤 페이지가 모두 active 상태의 트랜잭션으로 보이는 튜플만 포함하는 지를 추적
- 메인 릴레이션 데이터를 따라서 저장이 되는데, filenode 번호 뒤에 ' _vm' 이라는 접미사가 붙는 형태로 생성
- 예를 들면, 릴레이션의 filenode가 ‘123’이면, vm은 ‘123_vm’ 형태로 생성
- 한 heap page당 1bit씩 저장
- bit는 vacuum 작업에 의해 설정이 되며, 페이지의 데이터 변경 작업에 의해서 clear될 수 있음





출처: http://www.postgresqlinternals.org/index.php/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:4_9.jpg

- 블록 1은 dead tuple 없는 '1'을 나타내므로 vacuum 작업이 필요 없다.
- 블록 2는 삭제된 튜플이 있기 때문에 '0'이 표시된다.
- 업데이트시에 새로 추가된 튜플은 dead tuple이 없지만 0으로 표시된다.
- Index-Only Scan시에 블록1의 튜플은 모두 존재한다고 알고있다.

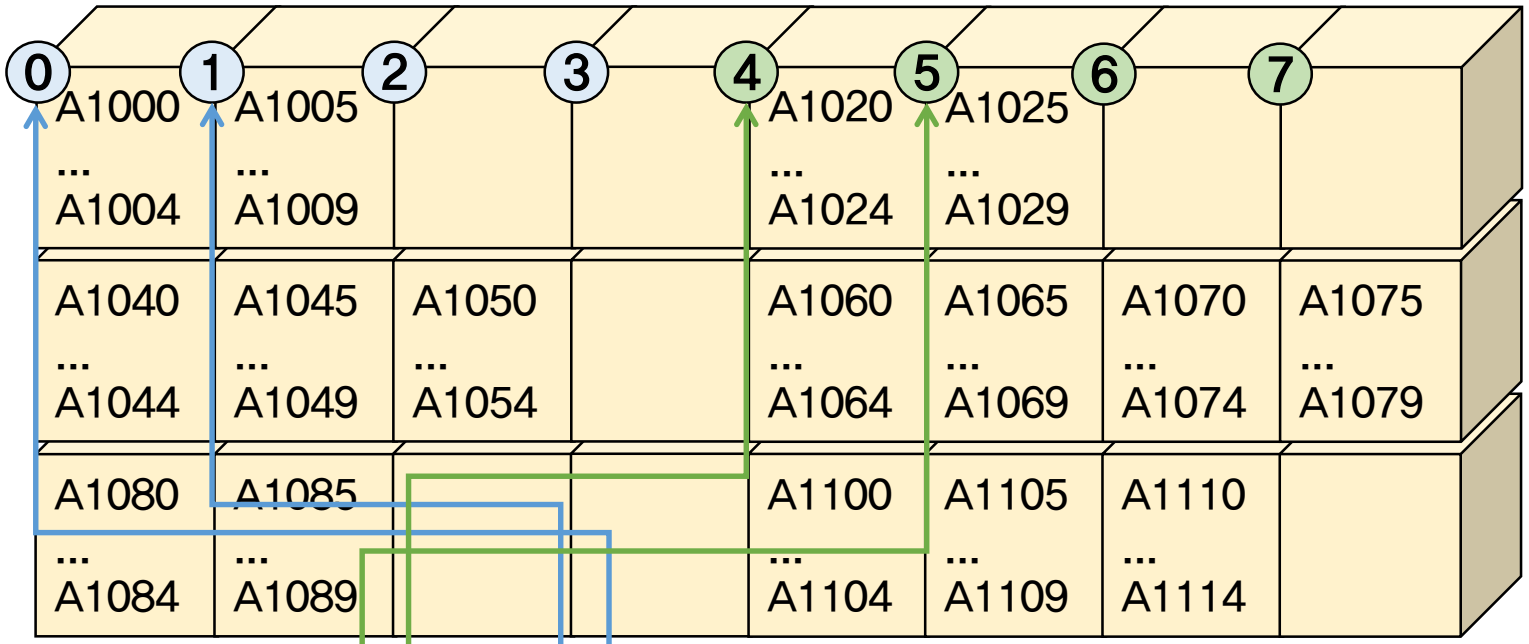
❖ VM Layout

```
Time: 0.541 ms
delete from t2 where c1 = 'A1000' ;
DELETE 1
Time: 1.858 ms
checkpoint ;
CHECKPOINT
Time: 100.459 ms
₩!     sh $(cat tmp1.sh | grep dump_od )
-rw------- 1 postgres postgres 8192 Aug  2 15:10 /pgdata/9.4/base/13003/153276_vm
000000  25  00  00  00  90  36  6c  dd  00  00  00  00  18  00  00  20 >%....6l........ <
000010  00  20  04  20  00  00  00  00  fe  ff  ff  00  00  00  00  00 >. . ............<
000020  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00 >................<
*
002000
```

VM Header

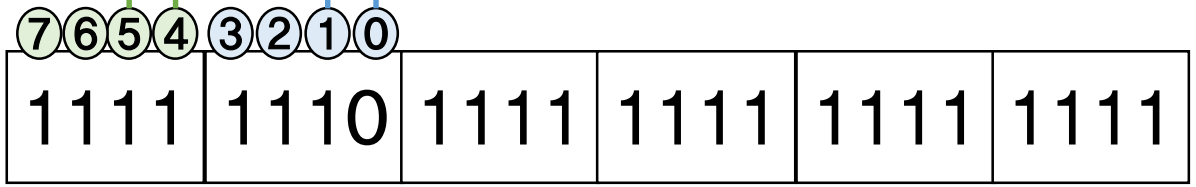
1. The visibility map simply stores one bit per heap page.
2. Visibility map bits are only set by vacuum, but are cleared by any data-modifying operations on a page.

*** Byte 단위로 한 bit씩 Little-endian 순서로 저장 (하위 비트부터 저장)**



25	00	00	00	90	36	6c	dd	00	00	00	00	18	00	00	20
00	20	04	20	00	00	00	00	fe	ff	ff	00	00	00	00	00

16진수를 2진수로 변환



```
insert into  t2 SELECT 'A' || generate_series(1000,1000+120-1),'A','A';
vacuum t2 ;
```

```
000000  25 00 00 00 90 36 6c dd 00 00 00 00 18 00 00 20  >%....6l........ <
000010  00 20 04 20 00 00 00 00 ff ff ff 00 00 00 00 00  >. . ............<
000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
002000
```

1. t2 테이블에 120건 insert (24페이지 생성, 1 페이지당 5건)
2. vacuum 실행
3. vm 페이지 Dump → 전부 visible 이므로 “1111 1111 1111 1111 1111 1111” (ff ff ff – hexa)

```
delete from t2 where c1 = 'A1000' ;
```

```
select ctid from t2 where c1 = 'A1000' ;
 ctid
-------
 (0,1)
(1 row)

Time: 0.541 ms
delete from t2 where c1 = 'A1000' ;
DELETE 1
Time: 1.858 ms
checkpoint ;
CHECKPOINT
Time: 100.459 ms
#!     sh $(cat tmp1.sh | grep dump_od )
-rw------- 1 postgres postgres 8192 Aug  2 15:10 /pgdata/9.4/base/13003/153276_vm
000000 25 00 00 00 90 36 6c dd 00 00 00 00 18 00 00 20 >%....6l........ <
000010 00 20 04 20 00 00 00 00 fe ff ff 00 00 00 00 00 >. . ............<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
002000
```

1. 0번 페이지에서 1번째 레코드 삭제 * ctid (0,1) : 0번 페이지에서 1번째 레코드
2. vm 페이지 Dump → 0번 페이지에 dead tuple이 발생해서 1번 Byte의 8번 Bit가 clear

“1111 1110 1111 1111 1111 1111” (fe ff ff – hexa)

```
delete from t2 where c1 = 'A1025' ;
```

```
select ctid from t2 where c1 = 'A1025' ;
 ctid
-------
 (5,1)
(1 row)

Time: 2.322 ms
delete from t2 where c1 = 'A1025' ;
DELETE 1
Time: 1.380 ms
checkpoint ;
CHECKPOINT
Time: 101.235 ms
#!     sh $(cat tmp1.sh | grep dump_od )
-rw------- 1 postgres postgres 8192 Aug  2 15:10 /pgdata/9.4/base/13003/153276_vm
000000  25  00  00  00  90  36  6c  dd  00  00  00  00  18  00  00  20 >%....6l........ <
000010  00  20  04  20  00  00  00  00  de  ff  ff  00  00  00  00  00 >. . ............<
000020  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00 >................<
*
002000
```

1. 5번 페이지에서 1번째 레코드 삭제
2. vm 페이지 Dump → 5번 페이지에 dead tuple이 발생해서 1번 Byte의 3번 Bit가 clear

“1101 1110 1111 1111 1111 1111”(de ff ff – hexa)

```
delete from t2 where c1 = 'A1050' ;
```

```
select ctid from t2 where c1 = 'A1050' ;
  ctid
--------
 (10,1)
(1 row)

Time: 1.945 ms
delete from t2 where c1 = 'A1050' ;
DELETE 1
Time: 0.662 ms
checkpoint ;
CHECKPOINT
Time: 101.430 ms
#!     sh $(cat tmp1.sh | grep dump_od )
-rw------- 1 postgres postgres 8192 Aug  2 15:10 /pgdata/9.4/base/13003/153276_vm
000000  25 00 00 00 90 36 6c dd 00 00 00 00 18 00 00 20 >%....6l........ <
000010  00 20 04 20 00 00 00 00 de fb ff 00 00 00 00 00 >. . ............<
000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
002000
```

1. 10번 페이지에서 1번째 레코드 삭제
2. vm 페이지 Dump → 10번 페이지에 dead tuple이 발생해서 2번 Byte의 6번 Bit가 clear

“1101 1110 1111 1011 1111 1111”(de fb ff – hexa)

```
delete from t2 where c1 = 'A1075' ;
```

```
select ctid from t2 where c1 = 'A1075' ;
  ctid
--------
 (15,1)
(1 row)

Time: 2.045 ms
delete from t2 where c1 = 'A1075' ;
DELETE 1
Time: 0.755 ms
checkpoint ;
CHECKPOINT
Time: 101.237 ms
#!     sh $(cat tmp1.sh | grep dump_od )
-rw------- 1 postgres postgres 8192 Aug  2 15:10 /pgdata/9.4/base/13003/153276_vm
000000  25 00 00 00 90 36 6c dd 00 00 00 00 18 00 00 20 >%....6l........ <
000010  00 20 04 20 00 00 00 00 de 7b ff 00 00 00 00 00 >. . .....{......<
000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
002000
```

1. 15번 페이지에서 1번째 레코드 삭제
2. vm 페이지 Dump → 15번 페이지에 dead tuple 이 발생해서 2번 Byte의 1번 Bit가 clear

"1101 1110 0111 1011 1111 1111" (de 7b ff – hexa)

```
delete from t2 where c1 in ( 'A1080','A1085' ) ;
```

```
select ctid from t2 where c1 in ( 'A1080','A1085' ) ;
  ctid
--------
 (16,1)
 (17,1)
(2 rows)

Time: 2.038 ms
delete from t2 where c1 in ( 'A1080','A1085' ) ;
DELETE 2
Time: 0.748 ms
checkpoint ;
CHECKPOINT
Time: 101.391 ms
#!     sh $(cat tmp1.sh | grep dump_od )
-rw------- 1 postgres postgres 8192 Aug  2 15:10 /pgdata/9.4/base/13003/153276_vm
000000  25  00  00  00  90  36  6c  dd  00  00  00  00  18  00  00  20 >%....6l........ <
000010  00  20  04  20  00  00  00  00  de  7b  fc  00  00  00  00  00 >. . .....{......<
000020  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00 >................<
*
```

1. 16번 페이지에서 1번째 레코드, 17번 페이지에서 1번째 레코드를 삭제
2. vm 페이지 Dump → 16,17번 페이지에 dead tuple이 발생해서 3번 Byte의 7,8번 Bit가 clear

“1101 1110 0111 1011 1111 1100”(de 7b fc – hexa)

```
delete from t2 where c1 in ( 'A1100','A1105','A1110' ) ;
```

```
select ctid from t2 where c1 in ( 'A1100','A1105','A1110') ;
  ctid
--------
 (20,1)
 (21,1)
 (22,1)
(3 rows)

Time: 2.173 ms
delete from t2 where c1 in ( 'A1100','A1105','A1110') ;
DELETE 3
Time: 0.797 ms
checkpoint ;
CHECKPOINT
Time: 101.042 ms
#!     sh $(cat tmp1.sh | grep dump_od )
-rw------- 1 postgres postgres 8192 Aug  2 15:10 /pgdata/9.4/base/13003/153276_vm
000000  25  00  00  00  90  36  6c  dd  00  00  00  00  18  00  00  20 >%....6l........ <
000010  00  20  04  20  00  00  00  00  de  7b  8c  00  00  00  00  00 >. . .....{......<
000020  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00 >................<
```

1. 20번 페이지에서 1번째 레코드, 21번 페이지에서 1번째 레코드, 22번 페이지에서 1번째 레코드 삭제
2. vm 페이지 Dump → 20,21,22번 페이지에 dead tuple이 발생해서 3번 Byte의 2,3,4번 Bit가 clear

“1101 1110 0111 1011 1000 1100”(de 7b 8c – hexa)

9.6 Visibility Map

65.4. Visibility Map

Each heap relation has a Visibility Map (VM) to keep track of which pages contain only tuples that are known to be visible to all active transactions; it also keeps track of which pages contain only unfrozen tuples. It's stored alongside the main relation data in a separate relation fork, named after the filenode number of the relation, plus a _vm suffix. For example, if the filenode of a relation is 12345, the VM is stored in a file called 12345_vm, in the same directory as the main relation file. Note that indexes do not have VMs.

The visibility map stores two bits per heap page. The first bit, if set, indicates that the page is all-visible, or in other words that the page does not contain any tuples that need to be vacuumed. This information can also be used by **index-only scans** to answer queries using only the index tuple. The second bit, if set, means that all tuples on the page have been frozen. That means that even an anti-wraparound vacuum need not revisit the page.

The map is conservative in the sense that we make sure that whenever a bit is set, we know the condition is true, but if a bit is not set, it might or might not be true. Visibility map bits are only set by vacuum, but are cleared by any data-modifying operations on a page.

The pg_visibility module can be used to examine the information stored in the visibility map.

1. PostgreSQL 9.6 버전에서 frozen bit 추가
2. Anti-wraparound vacuum 작업할때 '1'로 setting 된 페이지는 Skip .

   장점 : vacuum 작업성능 개선

   단점 : vm 파일사이즈 2배 증가

9.6 Visibility Map

```
vacuum t2 ;
VACUUM
Time: 18.776 ms
checkpoint ;
CHECKPOINT
Time: 101.989 ms
-rw------- 1 postgres postgres 8192 Oct 18 15:56 /pgdata/9.6/base/13269/16456_vm
000000 00 00 00 00 a8 b9 19 07 00 00 00 00 18 00 00 20  >.............. <
000010 00 20 04 20 00 00 00 00 55 55 55 55 55 55 00 00  >. . ....UUUUUU..<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
002000
```

Vacuum 작업완료후 vm 확인결과
55(Hexa Code) ➔ 01010101

```
vacuum freeze t2 ;
VACUUM
Time: 16.465 ms
checkpoint ;
CHECKPOINT
Time: 101.596 ms
-rw------- 1 postgres postgres 8192 Oct 18 15:56 /pgdata/9.6/base/13269/16456_vm
000000 00 00 00 00 f0 f1 1c 07 00 00 00 00 18 00 00 20  >.............. <
000010 00 20 04 20 00 00 00 00 ff ff ff ff ff ff 00 00  >. . ............<
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*
002000
```

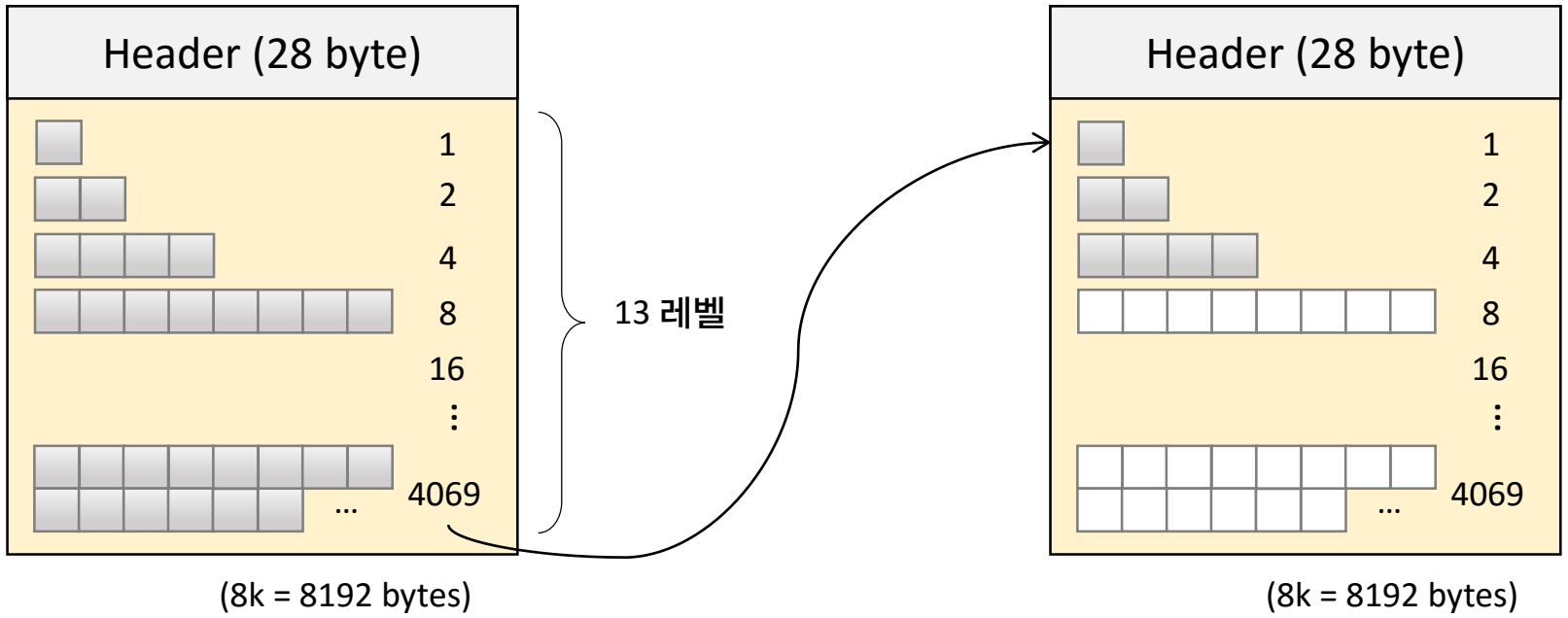
Vacuum Freeze 작업완료후 vm 확인결과

ff(Hexa Code) ➔ 11111111

Free Space Map (≒ Freelist)

- 각 heap과 index relation은 fsm을 통해 릴레이션의 사용 가능한 공간을 추적
- 메인 릴레이션 데이터를 따라서 저장이 되는데, filenode 번호 뒤에 ' _fsm' 이라는 접미사가 붙는 형태로 생성

  ex) 릴레이션의 filenode가 '123'이면, fsm은 '123_fsm' 형태로 생성
- 각 fsm 페이지는 이진 트리로 구성되어 있으며 각 노드당 1byte
- 각 리프 노드는 heap page를 나타내거나 하위 레벨의 fsm 페이지를 나타냄
- 하위 레벨의 fsm 페이지에 각 heap 페이지의 사용 가능한 공간이 저장
- 상위 레벨은 하위 레벨로부터 모은 정보이기 때문에 루트는 최대값이 저장

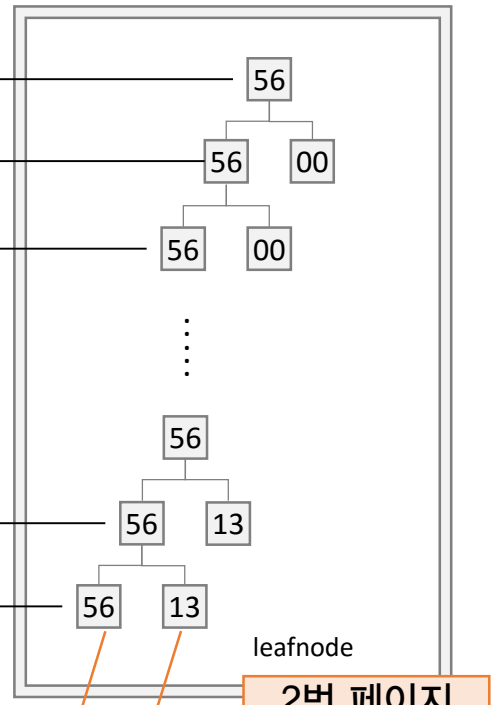


- 28 (Header) + 8191(13 레벨의 총합) = 8219 bytes
- 8219 – 8192(8k) = 27 bytes 초과
- 마지막 레벨인 4096 bytes 중 27 bytes 모자람 = 4069 bytes 즉, 페이지당 관리할 수 있는 블록의 수 4069개
- 만약 FSM이 초과된다면 페이지가 하나 더 생김

❖ FSM Layout

FSM Header

```
004000  00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 20 >. . . . . <
004010  00 20 04 20 00 00 00 00 00 00 00 00 56 56 00 56 >.  .  . . . . . VV.V<
004020  00 00 00 56 00 00 00 00 00 00 00 56 00 00 00 00 >...V. . . . . V. . . .<
004030  00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >. . . . . V. . . .<
004040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
004050  00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >. . . . . V. . . .<
004060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
004090  00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >. . . . . V. . . .<
0040a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
004110  00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >. . . . . V. . . .<
004120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
004210  00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >. . . . . V. . . .<
004220  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
004410  00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >. . . . . V. . . .<
004420  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
004810  00 00 00 00 00 00 00 00 00 00 00 00 56 13 00 00 00 >. . . . . V. . . .<
004820  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
005010  00 00 00 00 00 00 00 00 00 00 00 00 56 13 13 00 00 >. . . . . V. . . .<
005020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
006000
```



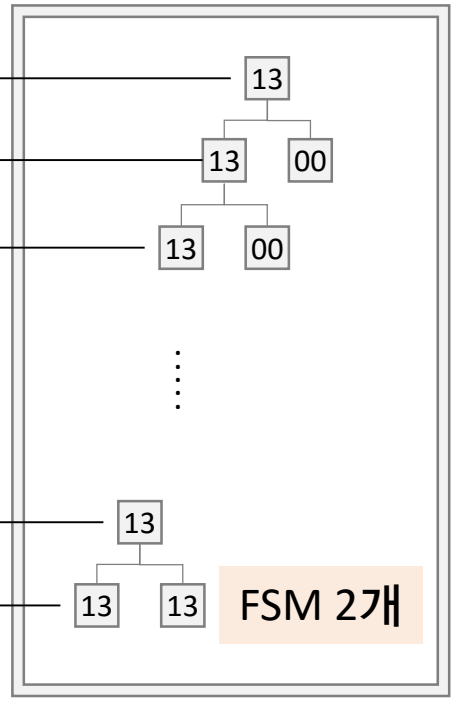
2번 페이지
Free space

1번 페이지
Free space

- 블록마다 1byte를 사용해서 공간을 BLCKSZ/ 256으로 나눈 값 유지 (1byte = 8bit = 2^8 =256)
- FSM value = Free space / (BLCKSZ/ 256) ex) 1024 / (8192/256) = 32

- CREATE TABLE
- INSERT 0 21

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
004000 00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 20 >.............. <
004010 00 20 04 20 00 00 00 00 00 00 00 00 13 13 00 13 >. . ............<
004020 00 00 00 13 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004030 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
004050 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
004090 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
0040a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
004110 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
004210 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
004410 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004420 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
004810 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
005010 00 00 00 00 00 00 00 00 00 00 00 13 13 00 00 00 >.............. <
005020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
006000
```

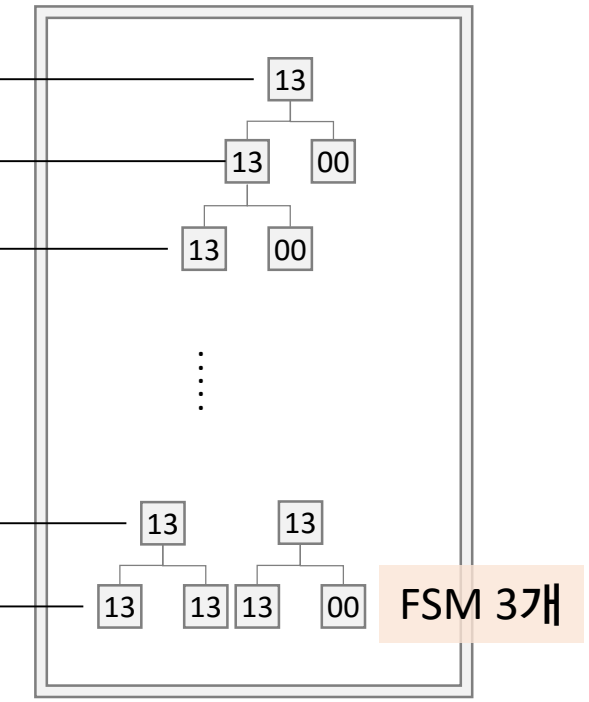


- 1 페이지에 7건의 레코드가 적재된다.
- 21건의 레코드를 insert하면 3 페이지가 생성되고, FSM에는 2개만 보인다.

05. Vacuum – 6) VM과 FSM 구조 분석

VACUUM

```
004000 00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 20 >. . . . . <
004010 00 20 04 20 00 00 00 00 00 00 00 00 13 13 00 13 >. . . . . <
004020 00 00 00 13 00 00 00 00 00 00 00 13 00 00 00 00 >. . . . . <
004030 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >. . . . . <
004040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
004050 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >. . . . . <
004060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
004090 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >. . . . . <
0040a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
004110 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >. . . . . <
004120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
004210 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >. . . . . <
004220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
004410 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >. . . . . <
004420 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
004810 00 00 00 00 00 00 00 00 00 00 00 13 13 00 00 00 >. . . . . <
004820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
005010 00 00 00 00 00 00 00 00 00 00 00 13 13 13 00 00 >. . . . . <
005020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >. . . . . <
*
006000
```



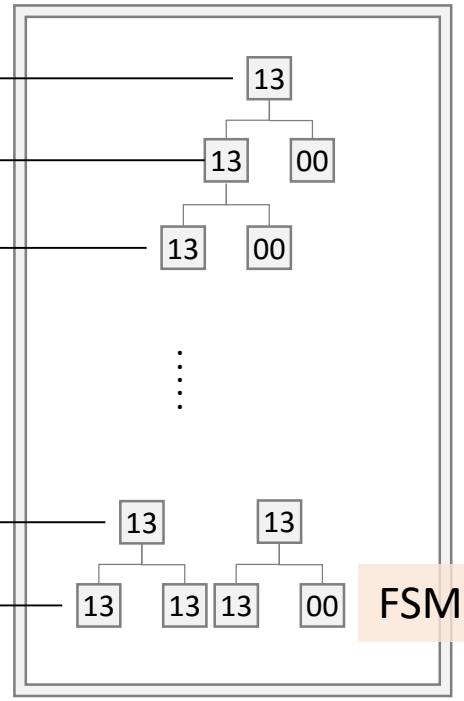
FSM 3개

- Vacuum 작업이 처리되면 FSM에 반영이 된다.

EXEM

DELETE 2

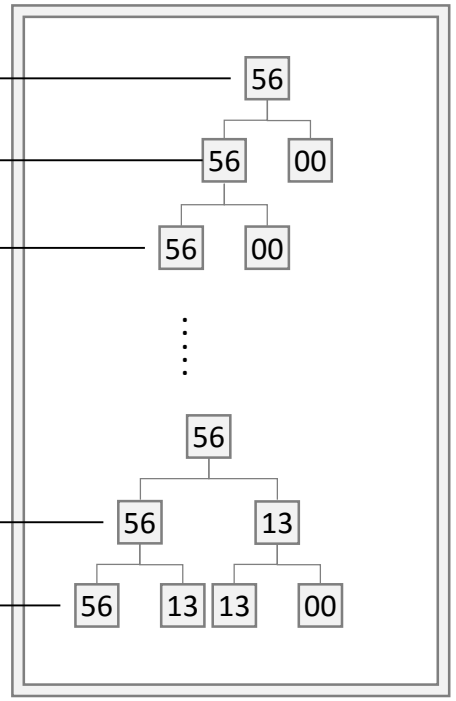
```
004000 00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 20 >.............. <
004010 00 20 04 20 00 00 00 00 00 00 00 00 13 13 00 13 >. . ............<
004020 00 00 00 13 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004030 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
004050 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
004090 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
0040a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
004110 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
004210 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
004410 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 00 >.............. <
004420 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
004810 00 00 00 00 00 00 00 00 00 00 00 13 13 00 00 00 >.............. <
004820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
005010 00 00 00 00 00 00 00 00 00 00 00 13 13 13 00 00 >.............. <
005020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.............. <
*
006000
```



- 2건을 delete 하면, FSM에 반영이 안 된다.

VACUUM

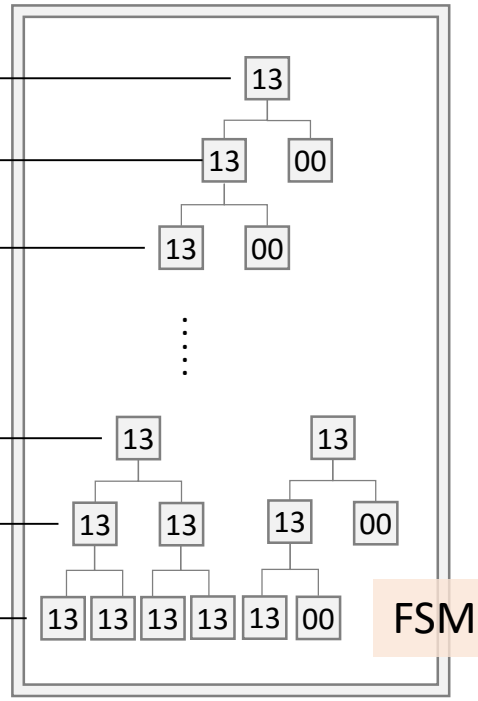
```
004000 00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 20 >..............  <
004010 00 20 04 20 00 00 00 00 00 00 00 00 56 56 00 56 >. . ........VV.V<
004020 00 00 00 56 00 00 00 00 00 00 00 56 00 00 00 00 >...V.......V....<
004030 00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >...........V....<
004040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
004050 00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >...........V....<
004060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004090 00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >...........V....<
0040a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004110 00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >...........V....<
004120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004210 00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >...........V....<
004220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004410 00 00 00 00 00 00 00 00 00 00 00 56 00 00 00 00 >...........V....<
004420 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004810 00 00 00 00 00 00 00 00 00 00 00 56 13 00 00 00 >...........V....<
004820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
005010 00 00 00 00 00 00 00 00 00 00 00 56 13 13 00 00 >...........V....<
005020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
006000
```



- Vacuum 작업이 처리되면 FSM에 반영이 된다.

UPDATE 19

```
004000 00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 20 >.............. <
004010 00 20 04 20 00 00 00 00 01 00 00 00 13 13 00 13 >. . ............<
004020 00 00 00 13 00 00 00 00 00 00 00 00 13 00 00 00 >................<
004030 00 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 >................<
004040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
004050 00 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 >................<
004060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004090 00 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 >................<
0040a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004110 00 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 >................<
004120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004210 00 00 00 00 00 00 00 00 00 00 00 00 13 00 00 00 >................<
004220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004410 00 00 00 00 00 00 00 00 00 00 00 00 13 13 00 00 00 >................<
004420 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004810 00 00 00 00 00 00 00 00 00 00 00 00 13 13 13 00 00 >................<
004820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
005010 00 00 00 00 00 00 00 00 00 00 00 00 13 13 13 13 13 >................<
005020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
006000
```



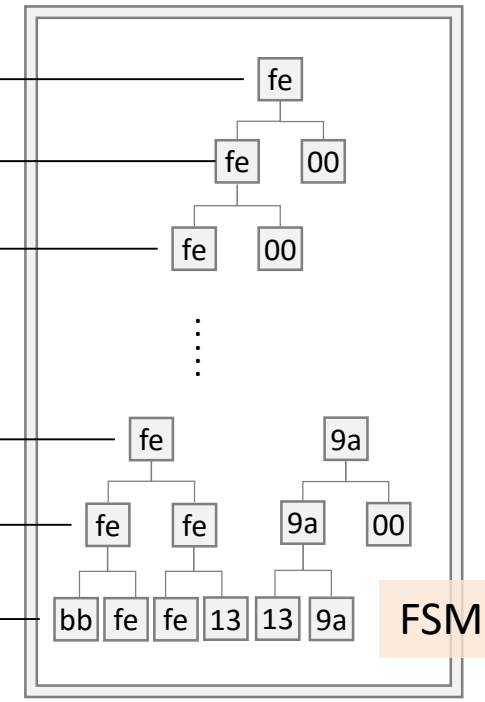
FSM 5개

- 19건을 Update 하면 페이지가 6개로 늘어나는데, FSM에는 반영이 되지 않는다. (5개)

05. Vacuum – 6) VM과 FSM 구조 분석

VACUUM

```
004000 00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 20 >.............. <
004010 00 20 04 20 00 00 00 00 00 00 00 00 fe fe 00 fe >. . ............<
004020 00 00 00 fe 00 00 00 00 00 00 00 fe 00 00 00 00 >................<
004030 00 00 00 00 00 00 00 00 00 00 00 fe 00 00 00 00 >................<
004040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
004050 00 00 00 00 00 00 00 00 00 00 00 fe 00 00 00 00 >................<
004060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004090 00 00 00 00 00 00 00 00 00 00 00 fe 00 00 00 00 >................<
0040a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004110 00 00 00 00 00 00 00 00 00 00 00 fe 00 00 00 00 >................<
004120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004210 00 00 00 00 00 00 00 00 00 00 00 fe 00 00 00 00 >................<
004220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004410 00 00 00 00 00 00 00 00 00 00 00 fe 9a 00 00 00 >................<
004420 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
004810 00 00 00 00 00 00 00 00 00 00 00 fe fe 9a 00 00 >................<
004820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
005010 00 00 00 00 00 00 00 00 00 00 00 bb fe fe 13 13 >................<
005020 9a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
005030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >................<
*
006000
```



FSM 6개

- Vacuum 작업이 처리되면 FSM에 반영이 된다. (6개 확인)

EXEM

Q & A

cafe.naver.com/playexem

• EXEM Research & Contents Team

Youtube	https://www.youtube.com/channel/UC5wKR_-A0eL_Pn_EMzoauJg
Tudou (中)	http://www.tudou.com/home/maxgauge/

NAVER	http://cafe.naver.com/playexem
ITPUB (中)	http://blog.itpub.net/31135309/
Wordpress	https://playexem.wordpress.com/
Slideshare	http://www.slideshare.net/playexem

교육 문의: 연구컨텐츠팀 김숙진

edu@ex-em.com

Thank You

cafe.naver.com/playexem