SwatDB

Generated by Doxygen 1.8.13

Contents

1	Swa	tDB hea	ader files		1
2	Hier	archica	l Index		3
	2.1	Class	Hierarchy		3
3	Clas	ss Index			5
	3.1				5
4	Clas	ss Docu	mentation	1	7
	4.1	Bufferl	Manager C	Class Reference	7
		4.1.1	Detailed	Description	8
		4.1.2	Construc	ctor & Destructor Documentation	8
			4.1.2.1	BufferManager()	8
			4.1.2.2	~BufferManager()	9
		4.1.3	Member	Function Documentation	9
			4.1.3.1	allocatePage()	9
			4.1.3.2	createFile()	10
			4.1.3.3	deallocatePage()	10
			4.1.3.4	flushPage()	11
			4.1.3.5	getBufferState()	11
			4.1.3.6	getPage()	12
			4.1.3.7	releasePage()	12
			4.1.3.8	removeFile()	13
			4.1.3.9	setDirty()	13
	42	Buffer	Man Class	Reference	14

ii CONTENTS

	4.2.1	Detailed Description	 14
	4.2.2	Member Function Documentation	 15
		4.2.2.1 contains()	 15
		4.2.2.2 get()	 15
		4.2.2.3 insert()	 16
		4.2.2.4 remove()	 16
4.3	Buffers	State Struct Reference	 17
	4.3.1	Detailed Description	 17
	4.3.2	Member Data Documentation	 17
		4.3.2.1 clock_hand	 18
		4.3.2.2 dirty	 18
		4.3.2.3 pinned	 18
		4.3.2.4 ref_bit	 18
		4.3.2.5 total	 18
		4.3.2.6 unpinned	 18
		4.3.2.7 valid	 18
4.4	BufHas	sh Struct Reference	 19
	4.4.1	Detailed Description	 19
4.5	Catalo	g Class Reference	 19
	4.5.1	Detailed Description	 20
	4.5.2	Constructor & Destructor Documentation	 20
		4.5.2.1 Catalog()	 20
		4.5.2.2 ~Catalog()	 21
	4.5.3	Member Function Documentation	 21
		4.5.3.1 _saveDBStateToFile()	 21
		4.5.3.2 _setFile()	 21
		4.5.3.3 addEntry()	 22
		4.5.3.4 deleteEntry()	 23
		4.5.3.5 getFile()	 23
		4.5.3.6 getFileId()	 24

CONTENTS

4.5.3.8 getFileName() 4.5.3.9 getSchema() 4.5.3.10 getType() 4.6 CatalogEntry Struct Reference 4.6.1 Detailed Description 4.6.2 Member Data Documentation 4.6.2.1 alloced 4.6.2.2 entry_type 4.6.2.3 file 4.6.2.4 file_id 4.6.2.5 file_name 4.6.2.6 name 4.6.2.7 schema 4.6.2.8 valid 4.7 CorruptedDataHeapPage Class Reference 4.7.1 Detailed Description 4.8 Data Class Reference 4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/3] 4.8.2.3 Data() [3/3] 4.8.2.4 ~Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity() 4.8.3.2 getData()	. 24) .	∍lds()	getFile	3.7	4.5		
4.5.3.10 getType() . 4.6 CatalogEntry Struct Reference 4.6.1 Detailed Description 4.6.2 Member Data Documentation . 4.6.2.1 alloced . 4.6.2.2 entry_type . 4.6.2.3 file . 4.6.2.4 file_id . 4.6.2.5 file_name . 4.6.2.6 name . 4.6.2.7 schema . 4.6.2.8 valid . 4.7 CorruptedDataHeapPage Class Reference . 4.7.1 Detailed Description . 4.8 Data Class Reference . 4.8.1 Detailed Description . 4.8.2 Constructor & Destructor Documentation . 4.8.2.1 Data() [1/3] . 4.8.2.2 Data() [2/3] . 4.8.2.3 Data() [3/3] . 4.8.2.4 ~Data() . 4.8.3 Member Function Documentation .	. 25	 	 		 	 							ne()	∍Nam	getFile	3.8	4.5		
4.6.1 Detailed Description 4.6.2 Member Data Documentation 4.6.2.1 alloced 4.6.2.2 entry_type 4.6.2.3 file 4.6.2.4 file_id 4.6.2.5 file_name 4.6.2.6 name 4.6.2.7 schema 4.6.2.8 valid 4.7 CorruptedDataHeapPage Class Reference 4.7.1 Detailed Description 4.8 Data Class Reference 4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/3] 4.8.2.3 Data() [3/3] 4.8.2.4 ~Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity()	. 25	 	 		 	 							a()	hema	getScl	3.9	4.5		
4.6.1 Detailed Description 4.6.2 Member Data Documentation. 4.6.2.1 alloced 4.6.2.2 entry_type 4.6.2.3 file 4.6.2.4 file_id 4.6.2.5 file_name 4.6.2.6 name 4.6.2.7 schema 4.6.2.8 valid 4.7 CorruptedDataHeapPage Class Reference 4.7.1 Detailed Description 4.8 Data Class Reference 4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/3] 4.8.2.3 Data() [3/3] 4.8.2.4 ~Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity()	. 26)e() .	getTyp	3.10	4.5		
4.6.2.1 alloced 4.6.2.2 entry_type 4.6.2.3 file 4.6.2.4 file_id 4.6.2.5 file_name 4.6.2.6 name 4.6.2.7 schema 4.6.2.8 valid 4.7 CorruptedDataHeapPage Class Reference 4.7.1 Detailed Description 4.8 Data Class Reference 4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/3] 4.8.2.3 Data() [3/3] 4.8.2.4 ~Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity()	. 27	 	 		 	 							се	erenc	ct Refe	y Strı	gEnt	Catalo	4.6
4.6.2.1 alloced 4.6.2.2 entry_type 4.6.2.3 file 4.6.2.4 file_id 4.6.2.5 file_name 4.6.2.6 name 4.6.2.7 schema 4.6.2.8 valid 4.7 CorruptedDataHeapPage Class Reference 4.7.1 Detailed Description 4.8 Data Class Reference 4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/31] 4.8.2.3 Data() [3/31] 4.8.2.4 ~Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity()	. 27	 	 		 	 								tion	Descrip	ailed	De	4.6.1	
4.6.2.2 entry_type	. 27	 	 		 	 						ation	nent	ocum	Data Do	nber	Me	4.6.2	
4.6.2.3 file 4.6.2.4 file_id . 4.6.2.5 file_name . 4.6.2.6 name . 4.6.2.7 schema . 4.6.2.8 valid 4.7 CorruptedDataHeapPage Class Reference . 4.7.1 Detailed Description 4.8 Data Class Reference . 4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation . 4.8.2.1 Data() [1/3] . 4.8.2.2 Data() [2/3] . 4.8.2.3 Data() [3/3] . 4.8.2.4 ~Data() . 4.8.3 Member Function Documentation . 4.8.3.1 getCapacity() .	. 27	 	 		 	 								d	alloce	2.1	4.6		
4.6.2.4 file_id . 4.6.2.5 file_name . 4.6.2.6 name . 4.6.2.7 schema . 4.6.2.8 valid . 4.7 CorruptedDataHeapPage Class Reference . 4.7.1 Detailed Description . 4.8.1 Detailed Description . 4.8.2 Constructor & Destructor Documentation . 4.8.2.1 Data() [1/3] . 4.8.2.2 Data() [2/3] . 4.8.2.3 Data() [3/3] . 4.8.2.4 ~Data() . 4.8.3 Member Function Documentation . 4.8.3.1 getCapacity() .	. 28	 	 		 	 								type	entry_	2.2	4.6		
4.6.2.5 file_name. 4.6.2.6 name. 4.6.2.7 schema. 4.6.2.8 valid 4.7 CorruptedDataHeapPage Class Reference 4.7.1 Detailed Description 4.8 Data Class Reference 4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/3] 4.8.2.3 Data() [3/3] 4.8.2.4 ~Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity()	. 28	 	 		 	 									file .	2.3	4.6		
4.6.2.6 name	. 28	 	 		 	 									file_id	2.4	4.6		
4.6.2.7 schema	. 28	 	 		 	 								ame .	file_na	2.5	4.6		
4.6.2.8 valid 4.7 CorruptedDataHeapPage Class Reference 4.7.1 Detailed Description 4.8 Data Class Reference 4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/3] 4.8.2.3 Data() [3/3] 4.8.2.4 ~Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity()	. 28	 	 		 	 									name	2.6	4.6		
4.7.1 Detailed Description 4.8 Data Class Reference 4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/3] 4.8.2.3 Data() [3/3] 4.8.2.4 ∼Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity()	. 28	 	 		 	 								nа	schem	2.7	4.6		
4.7.1 Detailed Description 4.8 Data Class Reference 4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/3] 4.8.2.3 Data() [3/3] 4.8.2.4 ~Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity()	. 28	 	 		 	 									valid	2.8	4.6		
4.8 Data Class Reference 4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/3] 4.8.2.3 Data() [3/3] 4.8.2.4 ~Data() 4.8.2.4 ~Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity()	. 29	 	 		 	 				е.	enc	Refer	ass	e Cla	apPag	ataHe	otedE	Corru	4.7
4.8.1 Detailed Description 4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/3] 4.8.2.3 Data() [3/3] 4.8.2.4 ~Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity()	. 30	 	 		 	 								tion	Descrip	ailed	De	4.7.1	
4.8.2 Constructor & Destructor Documentation 4.8.2.1 Data() [1/3] 4.8.2.2 Data() [2/3] 4.8.2.3 Data() [3/3] 4.8.2.4 ~Data() 4.8.3 Member Function Documentation 4.8.3.1 getCapacity()	. 30	 	 		 	 									ence .	Refer	Class	Data (4.8
4.8.2.1 Data() [1/3]	. 30	 	 		 	 								tion	Descrip	ailed	De	4.8.1	
4.8.2.2 Data() [2/3]	. 31	 	 		 	 		1	tior	enta	ume	r Docı	ıcto	estru	or & D	struc	Со	4.8.2	
4.8.2.3 Data() [3/3]	. 31	 	 		 	 							3]	[1/:	Data()	2.1	4.8		
4.8.2.4 ~ Data()	. 31	 	 		 	 							3]	[2/:	Data()	2.2	4.8		
4.8.3 Member Function Documentation	. 32	 	 		 	 							3]	[3/:	Data()	2.3	4.8		
4.8.3.1 getCapacity()	. 32	 	 		 	 								a() .	\sim Data	2.4	4.8		
3-1-4-1-30	. 32	 	 		 	 					tion	nenta	ocun	n Do	unctio	nber	Me	4.8.3	
4.8.3.2 getData()	. 32	 	 		 	 							ty()	pacit	getCa	3.1	4.8		
	. 33	 	 		 	 								ta() .	getDa	3.2	4.8		
4.8.3.3 getSize()	. 33	 	 		 	 								:e() .	getSiz	3.3	4.8		
	. 33	 	 		 	 								e() .	setSiz	3.4	4.8		
4.8.3.4 setSize()	. 34	 	 		 	 						ce .	eren	Refe	Class	skMg	rrorD	DiskE	4.9

iv CONTENTS

	4.9.1	Detailed Description	35
4.10	DiskMa	anager Class Reference	35
	4.10.1	Detailed Description	36
	4.10.2	Constructor & Destructor Documentation	36
		4.10.2.1 DiskManager()	36
		4.10.2.2 ~DiskManager()	37
	4.10.3	Member Function Documentation	37
		4.10.3.1 allocatePage()	37
		4.10.3.2 createFile()	38
		4.10.3.3 deallocatePage()	39
		4.10.3.4 getCapacity()	39
		4.10.3.5 getSize()	40
		4.10.3.6 isValidPage()	40
		4.10.3.7 printFile()	41
		4.10.3.8 readPage()	42
		4.10.3.9 removeFile()	42
		4.10.3.10 writePage()	43
4.11	Empty[DataHeapPage Class Reference	43
	4.11.1	Detailed Description	45
4.12	File Cla	ass Reference	45
	4.12.1	Detailed Description	46
	4.12.2	Constructor & Destructor Documentation	46
		4.12.2.1 File()	46
	4.12.3	Member Function Documentation	47
		4.12.3.1 _setMyFid()	47
		4.12.3.2 getHeaderId()	48
		4.12.3.3 getMyFid()	48
		4.12.3.4 getSchema()	48
	4.12.4	Member Data Documentation	48
		4.12.4.1 buf_mgr	49

CONTENTS

	4.12.4.2 catalog	49
	4.12.4.3 file_id	49
	4.12.4.4 header_id	49
	4.12.4.5 schema	49
4.13 FileAl	readyExistCat Class Reference	50
4.13.	1 Detailed Description	51
4.13.2	2 Constructor & Destructor Documentation	51
	4.13.2.1 FileAlreadyExistCat()	51
4.13.0	Member Function Documentation	51
	4.13.3.1 getFileName()	51
4.14 FileAl	readyExistDiskMgr Class Reference	52
4.14.	1 Detailed Description	53
4.14.2	2 Constructor & Destructor Documentation	53
	4.14.2.1 FileAlreadyExistDiskMgr()	53
4.15 FileId	AlreadyExistDiskMgr Class Reference	53
4.15.	1 Detailed Description	54
4.15.2	2 Constructor & Destructor Documentation	54
	4.15.2.1 FileIdAlreadyExistDiskMgr()	54
4.15.0	Member Function Documentation	55
	4.15.3.1 getFileId()	55
4.16 FileM	anager Class Reference	55
4.16.	1 Detailed Description	56
4.16.2	2 Constructor & Destructor Documentation	56
	4.16.2.1 FileManager()	56
	4.16.2.2 ~FileManager()	57
4.16.0	3 Member Function Documentation	57
	4.16.3.1 _closeAllFiles()	57
	4.16.3.2 _removeAllFiles()	57
	4.16.3.3 createRelation()	58
	4.16.3.4 getFile()	58

vi

	4.16.3.5 getRelation()	59
	4.16.3.6 removeFile()	59
	4.16.3.7 removeRelation()	60
4.17 FileNo	otFoundFileMgr Class Reference	61
4.17.1	Detailed Description	62
4.18 Frame	Class Reference	62
4.18.1	Detailed Description	62
4.18.2	Member Function Documentation	62
	4.18.2.1 loadFrame()	62
	4.18.2.2 resetFrame()	63
4.18.3	Friends And Related Function Documentation	63
	4.18.3.1 BufferManager	63
4.19 HeapF	File Class Reference	64
4.19.1	Detailed Description	65
4.19.2	Constructor & Destructor Documentation	65
	4.19.2.1 HeapFile()	65
	4.19.2.2 ~HeapFile()	66
4.19.3	Member Function Documentation	66
	4.19.3.1 createHeader()	66
	4.19.3.2 deleteRecord()	66
	4.19.3.3 flushHeader()	67
	4.19.3.4 getRecord()	68
	4.19.3.5 insertRecord()	68
	4.19.3.6 updateRecord()	69
4.20 HeapF	FileHeader Struct Reference	70
4.20.1	Detailed Description	70
4.20.2	Member Data Documentation	70
	4.20.2.1 free	70
	4.20.2.2 free_size	70
	4.20.2.3 full	71

CONTENTS vii

		4.20.2.4 full_size	71
		4.20.2.5 num_records	71
4.21	HeapFi	ileScanner Class Reference	71
	4.21.1	Detailed Description	71
	4.21.2	Constructor & Destructor Documentation	71
		4.21.2.1 HeapFileScanner()	72
		4.21.2.2 ~HeapFileScanner()	72
	4.21.3	Member Function Documentation	72
		4.21.3.1 getNext()	72
4.22	HeapPa	age Class Reference	73
	4.22.1	Detailed Description	74
	4.22.2	Constructor & Destructor Documentation	75
		4.22.2.1 HeapPage()	75
		4.22.2.2 ~HeapPage()	75
	4.22.3	Member Function Documentation	75
		4.22.3.1 deleteRecord()	75
		4.22.3.2 getFreeSpace()	76
		4.22.3.3 getNext()	76
		4.22.3.4 getPrev()	77
		4.22.3.5 getRecord()	77
		4.22.3.6 initializeHeader()	78
		4.22.3.7 insertRecord()	78
		4.22.3.8 isEmpty()	79
		4.22.3.9 isFull()	79
		4.22.3.10 setNext()	79
		4.22.3.11 setPrev()	80
		4.22.3.12 updateRecord()	80
4.23	HeapPa	ageHeader Struct Reference	81
	4.23.1	Detailed Description	81
	4.23.2	Member Data Documentation	81

viii CONTENTS

		4.23.2.1	capacity	y				 	 	 	 	٠.		81
		4.23.2.2	free_sp	ace_be	egin .			 	 	 	 			81
		4.23.2.3	free_sp	ace_er	nd			 	 	 	 			82
		4.23.2.4	next_pa	ıge				 	 	 	 			82
		4.23.2.5	prev_pa	age				 	 	 	 			82
		4.23.2.6	size					 	 	 	 			82
4.24	HeapP	ageScann	er Class	Refere	nce .			 	 	 	 			82
	4.24.1	Detailed I	Descripti	on				 	 	 	 			82
	4.24.2	Construc	tor & Des	structor	r <mark>Docu</mark> m	nentatio	on	 	 	 	 			83
		4.24.2.1	HeapPa	ageSca	inner()			 	 	 	 			83
	4.24.3	Member I	Function	Docum	nentatio	n		 	 	 	 			83
		4.24.3.1	getNext	i ()				 	 	 	 			83
		4.24.3.2	reset()					 	 	 	 			84
4.25	Insuffic	ientSpace	BufMgr (Class R	leferenc	е		 	 	 	 			84
	4.25.1	Detailed I	Descripti	on				 	 	 	 			85
4.26	Insuffic	ientSpace	DiskMgr	Class I	Referen	ice		 	 	 	 			85
	4.26.1	Detailed I	Descripti	on				 	 	 	 			87
	4.26.2	Construc	tor & Des	structor	r <mark>Docu</mark> m	nentatio	on	 	 	 	 			87
		4.26.2.1	Insuffici	entSpa	aceDiskl	Mgr()		 	 	 	 			87
	4.26.3	Member I	Function	Docum	nentatio	n		 	 	 	 			87
		4.26.3.1	getFileI	d()				 	 	 	 			87
4.27	Insuffic	ientSpace	HeapFile	Class	Referei	nce .		 	 	 	 			88
	4.27.1	Detailed I	Descripti	on				 	 	 	 			89
4.28	Insuffic	ientSpace	HeapPaç	ge Clas	ss Refer	ence.		 	 	 	 			89
	4.28.1	Detailed I	Descripti	on				 	 	 	 			90
4.29	Invalid	FileIdCat C	Class Ref	erence				 	 	 	 			90
	4.29.1	Detailed I	Descripti	on				 	 	 	 			91
	4.29.2	Construc	tor & Des	structor	r <mark>Docu</mark> m	nentatic	on	 	 	 	 			91
		4.29.2.1	InvalidF	ileldCa	at()			 	 	 	 			91
	4.29.3	Member I	Function	Docum	nentatio	n		 	 	 	 			92

CONTENTS

		4.29.3.1 getFileId()	92
4.30	Invalidi	FileIdDiskMgr Class Reference	92
	4.30.1	Detailed Description	93
	4.30.2	Constructor & Destructor Documentation	93
		4.30.2.1 InvalidFileIdDiskMgr()	93
	4.30.3	Member Function Documentation	94
		4.30.3.1 getFileId()	94
4.31	Invalid	FileIdHeapFile Class Reference	94
	4.31.1	Detailed Description	95
	4.31.2	Constructor & Destructor Documentation	95
		4.31.2.1 InvalidFileIdHeapFile()	95
	4.31.3	Member Function Documentation	96
		4.31.3.1 getFileId()	96
4.32	Invalid	FileTypeFileMgr Class Reference	96
	4.32.1	Detailed Description	97
	4.32.2	Constructor & Destructor Documentation	97
		4.32.2.1 InvalidFileTypeFileMgr()	97
	4.32.3	Member Function Documentation	98
		4.32.3.1 getFileType()	98
4.33	Invalidi	NameCat Class Reference	98
	4.33.1	Detailed Description	99
	4.33.2	Constructor & Destructor Documentation	99
		4.33.2.1 InvalidNameCat()	99
4.34	Invalid	PageIdBufMgr Class Reference	100
	4.34.1	Detailed Description	101
	4.34.2	Constructor & Destructor Documentation	101
		4.34.2.1 InvalidPageIdBufMgr()	101
	4.34.3	Member Function Documentation	101
		4.34.3.1 getPageId()	101
4.35	Invalid	PageNumDiskMgr Class Reference	102

CONTENTS

	4.35.1	Detailed Description	103
	4.35.2	Constructor & Destructor Documentation	103
		4.35.2.1 InvalidPageNumDiskMgr()	103
	4.35.3	Member Function Documentation	103
		4.35.3.1 getPageNum()	103
4.36	Invalid	SchemaHeapFile Class Reference	104
	4.36.1	Detailed Description	105
4.37	Invalid	SizeData Class Reference	105
	4.37.1	Detailed Description	106
4.38	Invalid	SlotIdHeapPage Class Reference	106
	4.38.1	Detailed Description	107
	4.38.2	Constructor & Destructor Documentation	107
		4.38.2.1 InvalidSlotIdHeapPage()	107
	4.38.3	Member Function Documentation	108
		4.38.3.1 getSlotId()	108
4.39	NameA	AlreadyExistCat Class Reference	108
	4.39.1	Detailed Description	109
	4.39.2	Constructor & Destructor Documentation	109
		4.39.2.1 NameAlreadyExistCat()	109
	4.39.3	Member Function Documentation	110
		4.39.3.1 getName()	110
4.40	Page C	Class Reference	110
	4.40.1	Detailed Description	111
	4.40.2	Member Function Documentation	111
		4.40.2.1 getData()	111
4.41	Pageld	Struct Reference	111
	4.41.1	Detailed Description	112
4.42	PageNo	otFoundBufMgr Class Reference	112
	4.42.1	Detailed Description	113
	4.42.2	Constructor & Destructor Documentation	113

CONTENTS xi

		4.42.2.1 PageNotFoundBufMgr()	3
	4.42.3	Member Function Documentation	14
		4.42.3.1 getPageId()	14
4.43	PageNo	otPinnedBufMgr Class Reference	14
	4.43.1	Detailed Description	15
	4.43.2	Constructor & Destructor Documentation	15
		4.43.2.1 PageNotPinnedBufMgr()	15
	4.43.3	Member Function Documentation	16
		4.43.3.1 getPageId()	16
4.44	PagePi	nnedBufMgr Class Reference	16
	4.44.1	Detailed Description	17
	4.44.2	Constructor & Destructor Documentation	17
		4.44.2.1 PagePinnedBufMgr()	17
	4.44.3	Member Function Documentation	18
		4.44.3.1 getPageId()	18
4.45	Record	Class Reference	18
	4.45.1	Detailed Description	19
	4.45.2	Constructor & Destructor Documentation	19
		4.45.2.1 Record()	19
		4.45.2.2 ~Record()	19
	4.45.3	Member Function Documentation	20
		4.45.3.1 compare()	20
		4.45.3.2 getRecordData()	20
		4.45.3.3 getSchema()	21
		4.45.3.4 setRecordData()	21
		4.45.3.5 setSchema()	21
4.46	Record	Ild Struct Reference	22
	4.46.1	Detailed Description	22
4.47	Schem	a Class Reference	22
	4.47.1	Detailed Description	23

xii CONTENTS

4.48 S	lotInfo	Struct Ref	erence				 	 	 	 	 	 123
4.	.48.1	Detailed D	escription				 	 	 	 	 	 123
4.	.48.2	Member D	ata Docume	entation .			 	 	 	 	 	 123
		4.48.2.1	length				 	 	 	 	 	 123
		4.48.2.2	offset				 	 	 	 	 	 124
4.49 S	watDE	3 Class Ref	erence				 	 	 	 	 	 124
4.	.49.1	Detailed D	escription				 	 	 	 	 	 124
4.	.49.2	Constructo	or & Destruc	ctor Docu	mentati	on	 	 	 	 	 	 125
		4.49.2.1	SwatDB() [1/2]			 	 	 	 	 	 125
		4.49.2.2	SwatDB() [2/2]			 	 	 	 	 	 125
		4.49.2.3	\sim SwatDB()				 	 	 	 	 	 125
4.	.49.3	Member F	unction Doc	cumentati	ion		 	 	 	 	 	 126
		4.49.3.1	setDestroy[OB()			 	 	 	 	 	 126
		4.49.3.2	setSaveDB	()			 	 	 	 	 	 126
4.50 S	watDE	BException	Class Refe	rence			 	 	 	 	 	 126
4.	.50.1	Detailed D	escription				 	 	 	 	 	 128
4.	.50.2	Constructo	or & Destruc	ctor Docu	mentati	on	 	 	 	 	 	 128
		4.50.2.1	SwatDBExc	ception()			 	 	 	 	 	 128
		4.50.2.2	\sim SwatDBE	xception(()		 	 	 	 	 	 128
4.	.50.3	Member F	unction Doc	cumentati	ion		 	 	 	 	 	 129
		4.50.3.1	what()				 	 	 	 	 	 129
4.	.50.4	Member D	ata Docume	entation .			 	 	 	 	 	 129
		4.50.4.1	message .				 	 	 	 	 	 129

Index

131

Chapter 1

SwatDB header files

These are .h files that can be included by any module in swatdb. They include .h interface files to the different layers and .h files that define common type definitions, typically used inter-layer. In test program, use these typedef types rather than the underlying base type to which they are currently defined.

2 SwatDB header files

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

BufferManager
BufferMap
BufferState
BufHash
Catalog
CatalogEntry
Data
DiskManager
exception
SwatDBException
CorruptedDataHeapPage
DiskErrorDiskMgr
EmptyDataHeapPage
FileAlreadyExistCat
FileAlreadyExistDiskMgr
FileIdAlreadyExistDiskMgr
FileNotFoundFileMgr
InsufficientSpaceBufMgr
InsufficientSpaceDiskMgr
InsufficientSpaceHeapFile
InsufficientSpaceHeapPage
InvalidFileIdCat
InvalidFileIdDiskMgr
InvalidFileIdHeapFile
InvalidFileTypeFileMgr
InvalidNameCat
InvalidPageIdBufMgr
InvalidPageNumDiskMgr
InvalidSchemaHeapFile
InvalidSizeData
InvalidSlotIdHeapPage
NameAlreadyExistCat
PageNotFoundBufMgr
PageNotPinnedBufMgr
PagePinnedBufMgr 116

Hierarchical Index

ile	 	45
HeapFile	 	64
FileManager	 	55
rame	 	62
HeapFileHeader	 	70
HeapFileScanner	 	71
HeapPageHeader	 	81
HeapPageScanner	 	82
Page	 1	10
HeapPage	 	73
Pageld	 1	11
Record	 1	18
RecordId	 1	22
Schema	 1	22
SlotInfo	 1	23
SwatDB	11	24

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

BufferManager	7
BufferMap	14
BufferState	17
BufHash	19
Catalog	19
CatalogEntry	27
CorruptedDataHeapPage	29
Data	30
DiskErrorDiskMgr	34
DiskManager	35
EmptyDataHeapPage	13
File	15
FileAlreadyExistCat	50
FileAlreadyExistDiskMgr	52
FileIdAlreadyExistDiskMgr	53
FileManager	55
FileNotFoundFileMgr	31
Frame	32
HeapFile 6	34
HeapFileHeader 7	70
HeapFileScanner	71
HeapPage	73
HeapPageHeader	31
HeapPageScanner	32
InsufficientSpaceBufMgr	34
InsufficientSpaceDiskMgr	35
InsufficientSpaceHeapFile	38
InsufficientSpaceHeapPage	39
InvalidFileIdCat	90
InvalidFileIdDiskMgr	92
InvalidFileIdHeapFile	94
InvalidFileTypeFileMgr	96
	98
InvalidPageIdBufMgr)0
InvalidPageNumDiskMgr 10	12

6 Class Index

alidSchemaHeapFile	 104
alidSizeData	 105
alidSlotIdHeapPage	 106
meAlreadyExistCat	 108
ge	 110
geld	 111
geNotFoundBufMgr	 112
geNotPinnedBufMgr	 114
gePinnedBufMgr	 116
cord	 118
cordld	 122
nema	
ıtlnfo	 123
atDB	 124
atDRException	126

Chapter 4

Class Documentation

4.1 BufferManager Class Reference

#include <bufmgr.h>

Public Member Functions

• BufferManager (DiskManager *disk mgr)

BufferManager constructor. Initializes the buf_pool and frame_table, and stores a pointer to SwatDB's DiskManager.

∼BufferManager ()

BufferManager destructor.

std::pair< Page *, PageId > allocatePage (FileId file_id)

Allocates a Page for the file of given Fileld. The Page is allocated both in the buffer pool, and on disk.

• void deallocatePage (PageId page_id)

Removes the Page of the given Pageld from the buffer pool, and deallocates the Page from the appropriate file on disk

Page * getPage (PageId page_id)

Gets Page by page_id, pins the Page, and returns a pointer to the Page object.

• void releasePage (PageId page_id, bool dirty)

Unpins a Page in the buffer pool.

void setDirty (PageId page_id)

Set the Page of the given Pageld dirty.

• void flushPage (PageId page_id)

Flushes the Page of the given Pageld to disk.

void createFile (FileId file_id)

Calls createFile() method on the DiskManager to create new Unix file that corresponds to the given FileId.

• void removeFile (FileId file id)

Calls removeFile() method on the DiskManager. Checks that none of the file's pages are pinned in the buffer pool. Removes any of the file's pages from the buffer pool before removing from disk.

· void clearBuffer ()

THIS METHOD IS FOR DEBUGGING ONLY. Clears the entire buffer pool, resetting all frames and removing pages from the buffer_map even if pinned. Does not flush any dirty pages either.

BufferState getBufferState ()

THIS METHOD IS FOR DEBUGGING ONLY. Returns the current state of the buffer pool.

· void printAllFrames ()

THIS METHOD IS FOR DEBUGGING ONLY. Prints Frame state of every Frame in the buffer pool, including pin count, valid bit, dirty bit, and ref_bit. If Page is valid, Pageld is printed.

void printValidFrames ()

THIS METHOD IS FOR DEBUGGING ONLY. Prints Frame state of every valid Frame in the buffer pool, including Pageld, pin count, valid bit, dirty bit, and ref_bit.

· void printFrame (FrameId frame id)

THIS METHOD IS FOR DEBUGGING ONLY. Prints Frame state of given Frameld, including pin count, valid bit, dirty bit, and ref_bit. If Page is valid, Pageld is printed.

void printPage (Pageld page id)

THIS METHOD IS FOR DEBUGGING ONLY. Prints Frame state of given Pageld, including Frameld, pin count, valid bit, dirty bit, and ref_bit. If Page is not in the buffer map, prints "Page Not Found".

void printBufferState ()

THIS METHOD IS FOR DEBUGGING ONLY. Prints current buffer state, including total number of pages, number of valid pages, number of pages, number of dirty pages, number of pages whose ref bit is set and the current clock hand position.

4.1.1 Detailed Description

SwatDb BufferManager Class. BufferManager manages in memory space of DBMS at page level granularity. At higher level, pages of data could be allocated, deallocated, retrieved to memory and fliushed to disk, using various methods.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BufferManager()

BufferManager constructor. Initializes the buf_pool and frame_table, and stores a pointer to SwatDB's DiskManager.

Precondition

disk mgr points to an initialized DiskManager object.

Postcondition

A BufferManager object will be initialized with an empty buffer pool. disk_mgr is set to the given DiskManager* and clock_hand is set to 0.

Parameters

disk_mgr	A pointer to SwatDB's DiskManager object. (DiskManager*).

4.1.2.2 \sim BufferManager()

```
BufferManager::\simBufferManager ( )
```

BufferManager destructor.

Precondition

None.

Postcondition

Every valid and dirty Page in buffer pool is written to disk.

4.1.3 Member Function Documentation

4.1.3.1 allocatePage()

Allocates a Page for the file of given FileId. The Page is allocated both in the buffer pool, and on disk.

Precondition

A valid FileId is provided and there is a free Page on disk or there is enough space in Unix file. There is also free space in the buffer pool, or a Page which can be evicted from the buffer pool.

Postcondition

A Frame is allocated, and a corresponding Page in the buffer pool is allocated. The Frame's page_id is set, valid is set to true, and the pin_count is set to 1. The Pageld is added to the BufferMap. Finally, a pair of a pointer to the allocated Page and Pageld is returned.

Parameters

file⊷	A FileId to which a Page should be allocated.
_id	

Returns

std::pair of Page* and PageId of the allocated Page.

Exceptions

InvalidFileIdDiskMgr	If page_id.file_id not valid.
InsufficientSpaceBufMgr	If there is not enough space in buffer pool.
InsufficientSpaceDiskMgr Generated by Doxygen	If there is not enough space in the Unix file.

4.1.3.2 createFile()

Calls createFile() method on the DiskManager to create new Unix file that corresponds to the given FileId.

Precondition

FileId is valid.

Postcondition

Unix file that corresponds to the file_id is created.

Parameters

file⊷	FileId of the file to be created.
_id	

See also

DiskManager::createFile()

4.1.3.3 deallocatePage()

Removes the Page of the given Pageld from the buffer pool, and deallocates the Page from the appropriate file on disk.

Precondition

A valid PageId of an unpinned Page is provided as a parameter.

Postcondition

If the Page is in the buffer pool, the Frame is reset, and the Page is removed from the buffer pool. The Page is deallocated from disk.

Parameters

page←	PageId of the Page to be deallocated
ld	

Exceptions

InvalidPageIdBufMgr	If page_id is not valid.				
PagePinnedBufMgr	If the Page is pinned.				

4.1.3.4 flushPage()

Flushes the Page of the given Pageld to disk.

Precondition

A Pageld of a pinned Page is provided as input.

Postcondition

If the Page is set dirty, the Page is written to disk through the disk_mgr. Page is still pinned.

Parameters

page←	PageId of the Page to set dirty.
_id	

Exceptions

PageNotFoundBufMgr	If page_id not in buf_map.
InvalidFileIdDiskMgr	If page_id.file_id not valid.
InvalidPageNumDiskMgr	If page_id.page_num not valid.

4.1.3.5 getBufferState()

```
BufferState BufferManager::getBufferState ( )
```

THIS METHOD IS FOR DEBUGGING ONLY. Returns the current state of the buffer pool.

See also

BufferState

4.1.3.6 getPage()

Gets Page by page_id, pins the Page, and returns a pointer to the Page object.

Precondition

A Pageld of an allocated Page is provided as a parameter and buffer pool is not full of pinned pages.

Postcondition

If the page_id is in buf_map, the Page is pinned and its pointer is returned. Else, a Frame is allocated in the buffer pool according to the page replacement policy, the Page is read from disk_mgr into the buffer pool, and the page_id, pin count, and valid bit are set. Page* is returned.

Parameters

page←	A PageId corresponding to the pointer to be returned.
_id	

Returns

Pointer to the Page with page_id.

Exceptions

InvalidPageIdBufMgr	If page_id is not valid.
InsufficientSpaceBufMgr	If buffer pool is full.

4.1.3.7 releasePage()

Unpins a Page in the buffer pool.

Precondition

A PageId of a pinned Page is provided as input. The Page is in the buffer pool and is pinned by the executing thread/process.

Postcondition

The pin count of the Page is decremented. ref_bit is set to true. Dirty bit is set if the dirty parameter is true.

Parameters

page←	PageId of the Page to be released.
_id	

Exceptions

PageNotPinnedBufMgr	If Page is not pinned. (pin_count is 0).
PageNotFoundBufMgr	If page_id is not in buf_map.

4.1.3.8 removeFile()

Calls removeFile() method on the DiskManager. Checks that none of the file's pages are pinned in the buffer pool. Removes any of the file's pages from the buffer pool before removing from disk.

Precondition

A valid FileId is given as a parameter. None of the file's pages are pinned in the buffer pool.

Postcondition

If the file has pages in the buffer pool, the corresponding frames are reset and pages are removed from buf_map. The file is removed from disk via DiskManager->removeFile().

Parameters

file⊷	FileId of the file to be removed.
_id	

Exceptions

```
PagePinnedBufMgr If the are pinned pages of file_id.
```

See also

DiskManager::removeFile()

4.1.3.9 setDirty()

Set the Page of the given Pageld dirty.

Precondition

A Pageld of a pinned Page is provided as input.

Postcondition

The Page is set dirty.

Parameters

page←	PageId of the Page to set dirty.
_id	

Exceptions

otFoundBufMgr If page_id is not in the buffer pool.

The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/bufmgr.h

4.2 BufferMap Class Reference

```
#include <bufmgr.h>
```

Public Member Functions

• BufferMap ()

Constructor for BufferMap.

∼BufferMap ()

Destructor for BufferMap.

Frameld get (Pageld page_id)

Returns Frameld corresponding to the given Pageld.

bool contains (PageId page_id)

Returns true if the map contains the given Pageld, else false.

• void insert (Pageld page_id, Frameld frame_id)

Inserts the pair < page_id, frame_id> into the map.

void remove (PageId page_id)

Removes the key-value pair corresponding to the given Pageld from the map.

4.2.1 Detailed Description

BufferMap is a wrapper class for std::unordered_map<PageId, FrameId> that maps PageIds to a Frame index in the buffer pool. Has different method names from std::unordered_map. Have get(), contains(), insert(), and remove() methods.

4.2.2 Member Function Documentation

4.2.2.1 contains()

Returns true if the map contains the given Pageld, else false.

Precondition

A lock is held on the map. A Pageld is provided.

Postcondition

Returns true if the map contains the Pageld, else false. Lock is still held on the map.

Parameters

page←	A Pageld to be searched in the map.
_id	

Returns

bool indicating whether the given Pageld exists in the BufferMap.

4.2.2.2 get()

Returns Frameld corresponding to the given Pageld.

Precondition

A lock is held on the map. This map contains the given Pageld.

Postcondition

Returns the corresponding Frameld. Lock is still held on the map.

Parameters

page←	A Pageld for which the corresponding Frameld will be returned.	
_id		

Returns

Frameld which correspond to the given Pageld.

Exceptions

4.2.2.3 insert()

Inserts the pair <page_id, frame_id> into the map.

Precondition

A lock is held on the map. A Pageld and Frameld are provided as input.

Postcondition

If the map contains page_id, then the Frameld in the Frameld will be updated. Else a new < Pageld, Frameld > pair is added to the map. Lock is still held on the map.

Parameters

page_id	A Pageld key.
frame←	A Frameld value.
_id	

4.2.2.4 remove()

Removes the key-value pair corresponding to the given Pageld from the map.

Precondition

A lock is held on the map. The given Pageld is in the map.

Postcondition

The key-value pair searched by page_id is removed from the map.

Parameters

page←	The Pageld key for the key-value pair to be removed.
_id	

Exceptions

PageIdNotFoundBufMgr	If page_id is not in the map.
----------------------	-------------------------------

The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/bufmgr.h

4.3 BufferState Struct Reference

```
#include <bufmgr.h>
```

Public Attributes

- std::uint32_t total
- std::uint32_t valid
- std::uint32_t pinned
- std::uint32_t unpinned
- std::uint32_t dirty
- std::uint32_t ref_bit
- std::uint32_t clock_hand

4.3.1 Detailed Description

THIS STRUCT IS FOR DEBUGGING ONLY. Struct that represents the state of the buffer pool.

4.3.2 Member Data Documentation

4.3.2.1 clock_hand

```
std::uint32_t BufferState::clock_hand
```

The current position of the clock hand.

4.3.2.2 dirty

```
std::uint32_t BufferState::dirty
```

The number of dirty pages in the buffer pool.

4.3.2.3 pinned

```
std::uint32_t BufferState::pinned
```

The number of pinned pages in the buffer pool.

4.3.2.4 ref_bit

```
std::uint32_t BufferState::ref_bit
```

The number of pages that have ref_bit set in the buffer pool.

4.3.2.5 total

```
std::uint32_t BufferState::total
```

The total number of pages in the buffer pool.

4.3.2.6 unpinned

```
std::uint32_t BufferState::unpinned
```

The number of unpinned pages in the buffer pool.

4.3.2.7 valid

```
std::uint32_t BufferState::valid
```

The number of valid pages in the buffer pool.

The documentation for this struct was generated from the following file:

/home/koh2/swatdb/SwatDB/include/bufmgr.h

4.4 BufHash Struct Reference

```
#include <bufmgr.h>
```

Public Member Functions

• std::size_t operator() (const PageId &page_id) const

4.4.1 Detailed Description

Hash function for BufferMap.

The documentation for this struct was generated from the following file:

· /home/koh2/swatdb/SwatDB/include/bufmgr.h

4.5 Catalog Class Reference

```
#include <catalog.h>
```

Public Member Functions

Catalog ()

creates an empty Catalog (should only be one in the system)

Catalog (std::string db metadata file)

create a catalog from existing DB state stored in meta data file

∼Catalog ()

Destroys the Catalog object.

• FileId addEntry (std::string name, Schema *schema, File *file, CatType type, std::string file_name)

Adds an entry to the catalog with a defined schema. May be called to create an entry for an index or relation as the SwatDB instance runs, or may be called as a SwatDB instance is booted from a saved state.

void deleteEntry (FileId file_id)

Deletes an entry from the database.

• std::string getFileName (FileId file_id)

Returns the filename associated with the given FileId.

FileId getFileId (std::string name)

Returns the FileId associated with the File identified by the given relation/index name.

File * getFile (FileId file_id)

Returns the File object associated with the given FileId.

Schema * getSchema (FileId file_id)

Returns the schema of the requested file.

CatType getType (FileId file_id)

Returns the type of the requested file.

std::vector< FileId > getFileIds ()

Gets the set of valid FileIDs in the system.

Protected Member Functions

```
    void _setFile (FileId file_id, File *file_ptr)
        Set the File * field of a Catalog entry.
    void _saveDBStateToFile (std::string db_metadata_filename)
        Valled by SwatDB on shutdown to save DB meta data state to a file.
```

Friends

- class SwatDB
- · class FileManager

4.5.1 Detailed Description

SwatDB Catalog Class: defines the interface to the part of SwatDB that keeps track of information about Relations and Indices in the system. High-level layers may add index and relation entries, and examine their schema for query processing. Low level layers may access some internal information about a Relation or Index through methods provided by the Catalog class. A Relation and Index is uniquely identified in the system by its FileId value.

The Catalog class is the only one that needs to know the format of a saved DB metadata file. It is the class that reads and parses the DB metadata as part of initing the SwatDB state to an existing DB, and also the one that saves Catalog state to a DB metadata file on shutdown. The main SwatDB class controls invoking the Catalog class constructor to init the catalog from a saved DB metadata file or to init an empty Catalog on start-up. It also is the only class that can invoke the Catalog's _saveDBStateToFile method, which it may do on shutdown of the DB. Currently, the metadata file format is the following (NOTE: TODO missing is representation of entries schema in metadata): num_entries 1st entry's Relation or Index name (string) 1st entry's File or Relation Type (int (its CatType value)) 1st entry's Disk file name (string) 2nd entry's Relation of Index name (string) ... FileIds are unique for a full execution of SwatDB regardless of if relations and indices are created or deleted at runtime. They do not persist across two separate boots of SwatDB.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Catalog()

create a catalog from existing DB state stored in meta data file

Parameters

db_metadata_file | file containing metadata information about the DB with which to initialize the Catalog

4.5.2.2 \sim Catalog()

```
Catalog::\simCatalog ( )
```

Destroys the Catalog object.

NOTE: The destructor does not determine whether the state of the SwatDB should be saved or not in the current implementation.

4.5.3 Member Function Documentation

4.5.3.1 _saveDBStateToFile()

Valled by SwatDB on shutdown to save DB meta data state to a file.

Parameters



4.5.3.2 _setFile()

```
void Catalog::_setFile (
          FileId file_id,
          File * file_ptr ) [protected]
```

Set the File * field of a Catalog entry.

This is called by the FileManager constructor when SwatDB is booted and inited from exising db state.

Precondition

file_id is a valid FileId and its Catalog entry has a nullptr value for its file field.

Postcondition

The file field of the Catalog entry for file_id is set to file_ptr

Parameters

file_id	A valid FileId.
file_ptr	A pointer to a valid File object

Exceptions

InvalidFileIdCat	if the FileId is not valid.
FileAlreadyExistCat	if the file entry already has a valid File *

4.5.3.3 addEntry()

Adds an entry to the catalog with a defined schema. May be called to create an entry for an index or relation as the SwatDB instance runs, or may be called as a SwatDB instance is booted from a saved state.

Precondition

An entry with matching name, type and file_name does not already exist in the system.

Postcondition

A new file or index has been added to the system, the passed File object's file_id field is set by this method since a file's id is determined by adding an entry for it to the Catalog.

Parameters

name	The name of the relation.
schema	The schema object assciated with the relation (may be null).
file	A pointer to the File object associated with the relation/index (may be null).
type	The type of entry (index or relation).
file_name	The name of the file into which the DiskManager stores the relation.

Returns

FileID of the added entry.

Exceptions

FileAlreadyExistCat	if the file_name already exists in the database.
RelationAlreadyExistCat	if a relation named name already exists in the databse.
IndexAlreadyExistsCat	if an index named name already exists in the database.

4.5.3.4 deleteEntry()

Deletes an entry from the database.

Precondition

A valid FileId is provided as input.

Postcondition

The relation or index identified by the given FileId is removed from the database, along with all other stored data associated with it.

Parameters

file⊷	The FileID of the file/index to remove from the database.	l
_id		

Exceptions

```
InvalidFileIdCat if the FileId is not valid.
```

4.5.3.5 getFile()

Returns the File object associated with the given FileId.

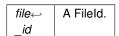
Precondition

The given FileId is valid.

Postcondition

The File object associated with the given FileId in the system is returned.

Parameters



Returns

File object associated with the given FileId.

Exceptions

```
InvalidFileIdCat if the given FileId is not valid.
```

4.5.3.6 getFileId()

Returns the FileId associated with the File identified by the given relation/index name.

Precondition

A File identified by the given name exists in the SwatDB system.

Postcondition

FileId of the File identified by the given relation/index name is is returned.

Parameters

file⊷	A FileId.
id	

Returns

FileId of the File identified by the given relation/index name.

Exceptions

InvalidNameCat	if the given name is not valid.
----------------	---------------------------------

4.5.3.7 getFileIds()

```
std::vector<FileId> Catalog::getFileIds ( )
```

Gets the set of valid FileIDs in the system.

Returns

std::vector<FileId> vector of FileIDs in the system. The vector is empty if there are no files in the database.

4.5.3.8 getFileName()

Returns the filename associated with the given FileId.

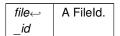
Precondition

The given FileId is valid.

Postcondition

The filename associated with the given FileId in the system is returned.

Parameters



Returns

std::string of the file name associated with the given FileId.

Exceptions

```
InvalidFileIdCat if the given FileId is not valid.
```

4.5.3.9 getSchema()

Returns the schema of the requested file.

Precondition

A valid FileId is provided as input.

Postcondition

The schema pointer of the requested file is returned. (note: this points to the same Schema object as in the catalog—it is not a copy).

Parameters

file⊷	A valid FileId.
_id	

Returns

Schema* associated with the file identified by the given FileId.

Exceptions

InvalidFileIdCat if t	the FileId is not valid.
-----------------------	--------------------------

4.5.3.10 getType()

Returns the type of the requested file.

Precondition

A valid FileId is provided as input.

Postcondition

The type of the file with this FileId is returned.

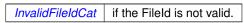
Parameters

file⊷	A valid FileId.
_id	

Returns

CatType value of file identified by the given FileId.

Exceptions



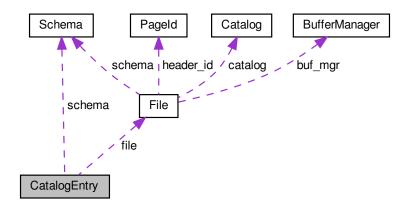
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/catalog.h

4.6 CatalogEntry Struct Reference

#include <catalog.h>

Collaboration diagram for CatalogEntry:



Public Attributes

- · FileId file id
- std::string name
- CatType entry_type
- Schema * schema
- File * file
- std::string file_name
- bool valid
- bool alloced

4.6.1 Detailed Description

Struct for an entry in the catalog. There is one entry for each relation and index in the system.

4.6.2 Member Data Documentation

4.6.2.1 alloced

bool CatalogEntry::alloced

Set to true if this entry has been allocated for use.

```
4.6.2.2 entry_type
CatType CatalogEntry::entry_type
The type of entry in the catalog (Relation or Index).
4.6.2.3 file
File* CatalogEntry::file
A pointer to the SwatDB file object (e.g. HeapFile).
4.6.2.4 file id
FileId CatalogEntry::file_id
The entry's FileId used as a system-wide identifier for this relation or index.
4.6.2.5 file_name
std::string CatalogEntry::file_name
The name of the underlying unix file for the file object.
4.6.2.6 name
std::string CatalogEntry::name
The name of the index or relation.
4.6.2.7 schema
Schema* CatalogEntry::schema
A pointer to the schema object of this entry.
4.6.2.8 valid
```

Set to true if this entry has valid contents.

bool CatalogEntry::valid

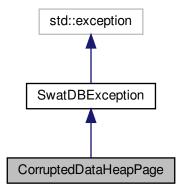
The documentation for this struct was generated from the following file:

/home/koh2/swatdb/SwatDB/include/catalog.h

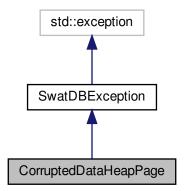
4.7 CorruptedDataHeapPage Class Reference

#include <exceptions.h>

Inheritance diagram for CorruptedDataHeapPage:



Collaboration diagram for CorruptedDataHeapPage:



Public Member Functions

• CorruptedDataHeapPage ()

Constructor.

∼CorruptedDataHeapPage () throw ()

Destructor.

Additional Inherited Members

4.7.1 Detailed Description

CorruptedDataHeapPage is thrown by HeapPage if data on HeapPage is corrupted.

The documentation for this class was generated from the following file:

· /home/koh2/swatdb/SwatDB/include/exceptions.h

4.8 Data Class Reference

```
#include <data.h>
```

Public Member Functions

• Data ()=delete

Disable base constructor.

• Data (const Data &other)=delete

Disable copy constructor.

• Data & operator= (const Data &other)=delete

Disable copy assignment constructor.

Data (std::uint32_t size, const char *other_data)

Constructor with given char array and size.

Data (std::uint32_t size, std::uint32_t capacity)

Constructor with given size and capacity.

• Data (std::uint32_t capacity)

Constructor with given capacity.

• ~Data ()

Destructor.

• char * getData ()

Getter for data char array.

• std::uint32_t getSize ()

Getter for size.

void setSize (std::uint32_t new_size)

Setter for size.

• std::uint32_t getCapacity ()

Getter for capacity.

4.8.1 Detailed Description

SwatDB Data Class. Data a is class that allows storing and moving data of specified size in a more convenient way. A lot of times used as a serialized object that could be given structure by storing it as data member of another object with appropriate methods.

4.8 Data Class Reference 31

4.8.2 Constructor & Destructor Documentation

Constructor with given char array and size.

Precondition

Valid char* and size are provided.

Postcondition

size bytes long char array is dynamically allocated and the size number of bytes are copied from the given char* pointer. size and capacity are set to the given size.

Constructor with given size and capacity.

Precondition

None.

Postcondition

capacity bytes long char array is dynamically allocated and capacity is set to the given capacity. size is set to the given size.

Exceptions

InvalidSizeData If size is greater than capacity.

```
4.8.2.3 Data() [3/3]
Data::Data (
               std::uint32_t capacity )
Constructor with given capacity.
Precondition
      None.
Postcondition
      capacity bytes long char array is dynamically allocated and capacity is set to the given capacity. size is set to
      0.
4.8.2.4 \sim Data()
Data::~Data ( )
Destructor.
Precondition
      Data object is valid.
Postcondition
     data array is deallocated.
4.8.3 Member Function Documentation
4.8.3.1 getCapacity()
std::uint32_t Data::getCapacity ( )
Getter for capacity.
Precondition
      None.
Postcondition
     capacity of the Data object is returned.
Returns
```

capacity of Data.

4.8 Data Class Reference 33

```
4.8.3.2 getData()
char* Data::getData ( )
Getter for data char array.
Precondition
     None.
Postcondition
     char* to data array is returned.
Returns
     char* to data array.
4.8.3.3 getSize()
std::uint32_t Data::getSize ( )
Getter for size.
Precondition
     None.
Postcondition
     size is returned.
Returns
     size of Data.
4.8.3.4 setSize()
void Data::setSize (
             std::uint32_t new_size )
Setter for size.
Precondition
     None.
Postcondition
     size is set.
```

Parameters

size New size to be set to.

Exceptions

InvalidSizeData	If new_size is greater than capacity.
-----------------	---------------------------------------

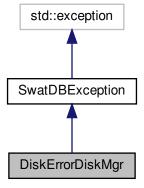
The documentation for this class was generated from the following file:

· /home/koh2/swatdb/SwatDB/include/data.h

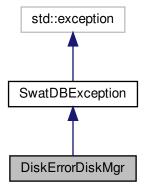
4.9 DiskErrorDiskMgr Class Reference

#include <exceptions.h>

Inheritance diagram for DiskErrorDiskMgr:



Collaboration diagram for DiskErrorDiskMgr:



Public Member Functions

- DiskErrorDiskMgr ()
 - Constructor.
- ∼DiskErrorDiskMgr () throw ()

Destructor.

Additional Inherited Members

4.9.1 Detailed Description

DiskErrorDiskMgr is thrown by DiskManager if there was error during file operations.

The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

4.10 DiskManager Class Reference

#include <diskmgr.h>

Public Member Functions

DiskManager (Catalog *catalog)

DiskManager constructor. Uses Catalog object pointer and a vector of FileIds to initialize file_map.

∼DiskManager ()

DiskManager destructor. All pages have been written to disk prior to this call.

• void createFile (FileId file id)

Creates a new Unix file, opens the fstream, initializes DiskFileInfo object, and adds < FileId, DiskFileInfo*> pair to fileMap. Adds header data to Unix file via SerializedFileInfo.

• void removeFile (FileId file id)

Deletes the Unix file corresponding to file_id, and removes the <FileId, DiskFileInfo*> key-value pair from fileMap.

PageId allocatePage (FileId file_id)

Allocates a Page to the file which corresponds to file_id.

void deallocatePage (PageId page id)

Deallocates page by adding its offest to unused_pages in the appropriate DiskFileInfo struct.

void readPage (PageId page_id, Page *page)

Reads the page data from the Unix file into the Page object pointer.

void writePage (PageId page id, Page *page)

Writes the page data of the given Page object at the right offset in the appropriate Unix file.

bool isValidPage (PageId page_id)

Checks if the page of a given pageld is valid.

void printFile (FileId file_id)

THIS METHOD IS FOR DEBUGGING ONLY. Prints contents of a file inlcuding FileId, size, capacity, and contents of each page.

std::uint32 t getCapacity (FileId file id)

Get method for the capacity of a file.

std::uint32_t getSize (FileId file_id)

Get method for size of a file.

4.10.1 Detailed Description

SwatDB DiskManager Class. DiskManager manages page level disk operations of SwatDB, including writing, reading, allocating, and deallocating pages. As SwatDB is built for pedagogical reason, this layer is built on top of regular Unix file system, rather than raw device/DIRECT_IO.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 DiskManager()

DiskManager constructor. Uses Catalog object pointer and a vector of FileIds to initialize file_map.

Precondition

Valid Catalog* is passed as an input.

Postcondition

A vector of FileIds are retrieved from the Catalog and appropriate fstream is opened for each FileId. Then a DiskManager object is initialized and each <FileId, DiskFileInfo*> pair is added to file map.

Parameters

catalog	A pointer to the DBMS's catalog object (Catalog*).
	· · - · · · · · · · · · · · · · ·

Exceptions

```
DiskErrorDiskMgr if file operation fails
```

4.10.2.2 \sim DiskManager()

```
DiskManager::\simDiskManager ( )
```

DiskManager destructor. All pages have been written to disk prior to this call.

Precondition

None

Postcondition

The metadata for each relation/index file is written to the appropriate Unix file via _flushDiskFileInfo().

See also

DiskManager::_flushDiskFileInfo()

4.10.3 Member Function Documentation

4.10.3.1 allocatePage()

Allocates a Page to the file which corresponds to file_id.

Precondition

A FileId in file_map is provided as input and there is enough space in the corresponding Unix file.

Postcondition

If unused_pages set of the corresponding DiskFileInfo struct is not empty, then a PageNum in the set is popped, and is used with the file_id to construct an appropriate PageId, which is returned. If the unused pages set of the corresponding DiskFileInfo struct is empty, and capacity is less than MAX_PAGE_NUM, then it is used with the file_id to construct an appropriate PageId, which is returned. Capacity is incremented and the size of underlying Unix file is increased by PAGE_SIZE. The updated metadata about the file is not immediately written to the file to minimize disk I/O.

Parameters

file⇔	A FileId corresponding to the file in which a page will be allocated.
_id	

Returns

Pageld of the allocated page.

Exceptions

InsufficientSpaceDiskMgr	If file_id is in file_map and of the file is equal to MAX_CAPCITY.
InvalidFileIdDiskMgr	If file_id is not in file_map.
DiskErrorDiskMgr	If file operation fails.

4.10.3.2 createFile()

Creates a new Unix file, opens the fstream, initializes DiskFileInfo object, and adds <FileId, DiskFileInfo*> pair to fileMap. Adds header data to Unix file via SerializedFileInfo.

Precondition

A FileId that is added to catalog and is not in file_map.

Postcondition

A new Unix file is created, the fstream is opened, DiskFileInfo is initialized for that file and <FileId, DiskFile \leftarrow Info*> pair is added to file_map.

Parameters

file⊷	A FileId of the file to be created.
_id	

Exceptions

FileIdAlreadyExistDiskMgr	If file_id is already in file_map.
FileAlreadyExistDiskMgr	If Unix file already exists.
DiskErrorDiskMgr	If file operation fails.

4.10.3.3 deallocatePage()

Deallocates page by adding its offest to unused_pages in the appropriate DiskFileInfo struct.

Precondition

PageId of an allocated Page is provided as input.

Postcondition

page_id.page_num is added to the unused_pages set in the appropriate DiskFileInfo struct. The updated metadata about the file is not immediately written to the file due to performance reasons.

Parameters

page←	A PageId corresponding to the page to be deallocated.
_id	

Exceptions

InvalidPageNumDiskMgr	If page_id.page_num is out of range or if the page_id.page_num is in unused_pages.
InvalidFileIdDiskMgr	If page_id.file_id is not in file_map.

4.10.3.4 getCapacity()

Get method for the capacity of a file.

Precondition

A FileId in file_map is provided as input.

Postcondition

capacity of the corresponding file is returned.

Parameters

file←	A FileId corresponding to the Unix file to get capacity.
_id	

Returns

capacity of the corresponding file.

Exceptions

```
InvalidFileIdDiskMgr If file_id is not in file_map.
```

4.10.3.5 getSize()

Get method for size of a file.

Precondition

A FileId in file_map is provided as input.

Postcondition

size of the corresponding file is returned.

Parameters

file⊷	A FileId corresponding to the Unix file to get size.
_id	

Returns

size of the corresponding file.

Exceptions

```
InvalidFileIdDiskMgr If file_id is not in file_map.
```

4.10.3.6 isValidPage()

Checks if the page of a given pageld is valid.

Precondition

PageId of a Page is provided as input.

Postcondition

If page_id.file_id is not in file_map or page_id.page_num is out of range or is in unused_pages, false is returned. Otherwise true is returned.

Parameters

page⊷	A Pageld for validity to be checked.
_id	

Returns

bool indicating whether the page is valid.

4.10.3.7 printFile()

THIS METHOD IS FOR DEBUGGING ONLY. Prints contents of a file inlcuding FileId, size, capacity, and contents of each page.

Precondition

A FileId in file_map is provided as input.

Postcondition

The content of the corresponding file,including size, capacity and content of each page is printed.

Parameters

file⊷	A FileId corresponding to the Unix file to be printed
_id	

Exceptions

InvalidFileIdDiskMgr	If file_id is not in file_map.	
DiskErrorDiskMgr	If file operation fails.	

4.10.3.8 readPage()

Reads the page data from the Unix file into the Page object pointer.

Precondition

PageId of an allocated Page, and a valid Page object pointer are provided as input.

Postcondition

The data of the page is read from the Unix file into the Page object.

Parameters

page⊷ _id	A Pageld of the page to be read from Unix file.
page	A Page pointer to be initialized.

Exceptions

InvalidFileIdDiskMgr	If page_id.file_id is not in file_map.
InvalidPageNumDiskMgr	If page_id.page_num is out of range or if the page_id.page_num is in unused_pages.
DiskErrorDiskMgr	If file operation fails.

4.10.3.9 removeFile()

Deletes the Unix file corresponding to file_id, and removes the <FileId, DiskFileInfo*> key-value pair from fileMap.

Precondition

A FileId in file_map is provided as input.

Postcondition

The corresponding file's fstream is closed, the Unix file is removed, and the <FileId, DiskFileInfo*> pair is removed from file map.

Parameters

file⊷	A file_id of the file to be removed.
_id	

Exceptions

```
InvalidFileIdDiskMgr If file_id is not in file_map.
```

4.10.3.10 writePage()

Writes the page data of the given Page object at the right offset in the appropriate Unix file.

Precondition

PageId of an allocated Page, and a valid Page reference are provided as input.

Postcondition

The Page object data is written to the Unix file at the right offset.

Parameters

page← _id	A Pageld to write Page object data in the appropriate Unix file.	
page	A Page* object containing data to be written to the appropriate Unix file.	

Exceptions

InvalidFileIdDiskMgr	If page_id.file_id is not in file_map.
InvalidPageNumDiskMgr	If page_id.page_num is out of range or if the page_id.page_num is in unused_pages.
DiskErrorDiskMgr	If file operation fails.

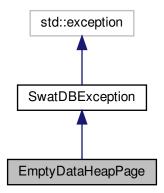
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/diskmgr.h

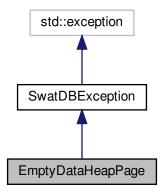
4.11 EmptyDataHeapPage Class Reference

#include <exceptions.h>

Inheritance diagram for EmptyDataHeapPage:



Collaboration diagram for EmptyDataHeapPage:



Public Member Functions

• EmptyDataHeapPage ()

Constructor.

~EmptyDataHeapPage () throw ()

Destructor.

Additional Inherited Members

4.12 File Class Reference 45

4.11.1 Detailed Description

EmptyDataHeapPage is thrown by HeapPage if the record to be inserted has data field with size 0.

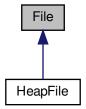
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

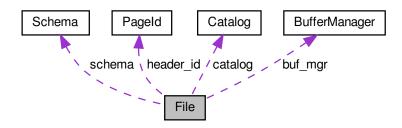
4.12 File Class Reference

#include <file.h>

Inheritance diagram for File:



Collaboration diagram for File:



Public Member Functions

• File (Catalog *catalog, BufferManager *buf mgr, Schema *schema)

Constructor. Initializes common state associated with every file, including catalog, buf_mgr, and schema.

virtual ~File ()

Destructor. Data members are cleaned up, but none of the dynamically allocated data members are not deleted. The underlying Unix file is not deleted.

• virtual void createHeader ()=0

Allocates and initializes the header. Is a virtual method to be overridden at each derived class.

virtual void flushHeader ()=0

Flushes Header Page to disk. Is a virtual method to be overridden at each derived class.

• FileId getMyFid ()

Returns the FileId of the File.

Schema * getSchema ()

Returns the schema of the File.

Pageld getHeaderld ()

Returns Header Pageld.

Protected Member Functions

void <u>setMyFid</u> (FileId file_id)

Sets the file_id and the schema fields of this File.

Protected Attributes

- · FileId file id
- Catalog * catalog
- BufferManager * buf_mgr
- Schema * schema
- · Pageld header id

Friends

- · class Catalog
- class FileManager

4.12.1 Detailed Description

SwatDB File Class. The base class for all file-type objects in the system. A File is used to represent a relation or index in the system. This base class inludes state and methods that are common to every type of file in the system.

4.12.2 Constructor & Destructor Documentation

Constructor. Initializes common state associated with every file, including catalog, buf_mgr, and schema.

4.12 File Class Reference 47

Parameters

catalog	Pointer to the SwatDB Catalog. Needed for getting file and idex relation files and schema.
buf_mgr	Pointer to the SwatDB Buffer Manager. Needed for de/allocating Pages, for getting and flushing Pages.
schema.	Pointer to Schema for the File.

Precondition

Input paramaters (pointers) are all valid.

Postcondition

The data members of File are initialized. file_id and header_id, however, are not set.

4.12.3 Member Function Documentation

4.12.3.1 _setMyFid()

Sets the file_id and the schema fields of this File.

NOTE: this is called by

- 1. Catalog.createFile so that a File object can be added to the Catalog as part of an entry before the Catalog has determined its fileId (our solution to a chicken and egg problem).
- 2. FileManager._loadFile via its constructor when initializing SwatDB from saved db state, we add entries to the Catalog without Files, and the File manager creates File * for them and updates each file's fid field after creation (could just have another constructor to handle this case)

Precondition

The given FileId is valid.

Postcondition

file_id is set to the given value.

Parameters

file←	The File's FileId.
id	

4.12.3.2 getHeaderId() PageId File::getHeaderId () Returns Header Pageld. Precondition None. Postcondition Header Pageld of the File is returned. Returns The Header Pageld. 4.12.3.3 getMyFid() FileId File::getMyFid () Returns the Fileld of the File. Returns The FileId of the File. 4.12.3.4 getSchema() Schema* File::getSchema () Returns the schema of the File. Returns schema of the File.

4.12.4 Member Data Documentation

4.12 File Class Reference 49

```
4.12.4.1 buf_mgr
BufferManager* File::buf_mgr [protected]
Pointer to the SwatDB BufferManager
4.12.4.2 catalog
Catalog* File::catalog [protected]
Pointer to the SwatDB Catalog
4.12.4.3 file_id
FileId File::file_id [protected]
The File's FileId.
4.12.4.4 header_id
PageId File::header_id [protected]
Pageld for the header Page of the File.
4.12.4.5 schema
Schema* File::schema [protected]
```

Pointer to the Schema associated with this File. (this is just for fast access to a file's Schema vs. going through the Catalog every time this is needed)

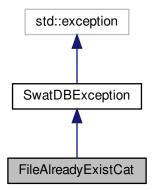
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/file.h

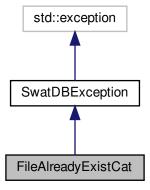
4.13 FileAlreadyExistCat Class Reference

#include <exceptions.h>

Inheritance diagram for FileAlreadyExistCat:



Collaboration diagram for FileAlreadyExistCat:



Public Member Functions

• FileAlreadyExistCat (const std::string &filename)

Constructor.

~FileAlreadyExistCat () throw ()

Destructor.

• std::string getFileName () const throw ()

Returns the File name of the file that already exists.

Additional Inherited Members

4.13.1 Detailed Description

Catalog Exceptions. FileAlreadyExistCat is thrownn by Catalog on index or relation create if the underlying file name for the index or relation is already used.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 FileAlreadyExistCat()

Constructor.

Parameters

file. File name of the file that already exists.

4.13.3 Member Function Documentation

4.13.3.1 getFileName()

```
std::string FileAlreadyExistCat::getFileName ( ) const throw )
```

Returns the File name of the file that already exists.

Returns

File name of the file that already exists.

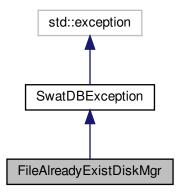
The documentation for this class was generated from the following file:

/home/koh2/swatdb/SwatDB/include/exceptions.h

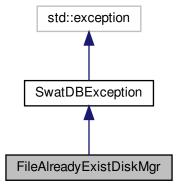
4.14 FileAlreadyExistDiskMgr Class Reference

#include <exceptions.h>

Inheritance diagram for FileAlreadyExistDiskMgr:



Collaboration diagram for FileAlreadyExistDiskMgr:



Public Member Functions

• FileAlreadyExistDiskMgr (const std::string &file)

Constructor.

• \sim FileAlreadyExistDiskMgr () throw ()

Destructor.

• std::string getFileName () const throw ()

Returns the file name of the file that already exists.

Additional Inherited Members

4.14.1 Detailed Description

FileAlreadyExistDiskMgr is thrown by DiskManager if Unix file already exists.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 FileAlreadyExistDiskMgr()

Constructor.

Parameters

file. File name of the file that already exists.

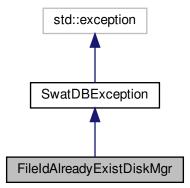
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

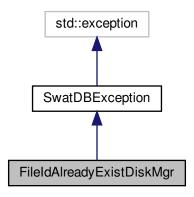
4.15 FileIdAlreadyExistDiskMgr Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for FileIdAlreadyExistDiskMgr:



Collaboration diagram for FileIdAlreadyExistDiskMgr:



Public Member Functions

• FileIdAlreadyExistDiskMgr (FileId file_id)

Constructor.

• \sim FileIdAlreadyExistDiskMgr () throw ()

Destructor.

• FileId getFileId () const throw ()

Returns the FileId of the file that already exists.

Additional Inherited Members

4.15.1 Detailed Description

FileIdAlreadyExistDiskMgr is thrown by DiskManager if the file identified by FileId already exists.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 FileIdAlreadyExistDiskMgr()

```
\label{eq:file_dal_readyExistDiskMgr::FileIdAlreadyExistDiskMgr (file_id) [explicit]} FileIdAlreadyExistDiskMgr (file_id) [explicit]
```

Constructor.

Parameters

file_id. FileId of the file	that already exists.
-----------------------------	----------------------

4.15.3 Member Function Documentation

4.15.3.1 getFileId()

```
FileId FileIdAlreadyExistDiskMgr::getFileId ( ) const throw )
```

Returns the FileId of the file that already exists.

Returns

FileId of the file that already exists.

The documentation for this class was generated from the following file:

· /home/koh2/swatdb/SwatDB/include/exceptions.h

4.16 FileManager Class Reference

```
#include <filemgr.h>
```

Public Member Functions

FileManager (Catalog *cat, BufferManager *buf_mgr, bool load=false)

Creates SwatDB FileManager, the interface to the File layer.

∼FileManager ()

Destructor.

• FileId createRelation (std::string name, Schema *schema, CatType type, std::string file_name)

Creates a new Relation in the system. Adds an entry for the new file to the Catalog and creates the underlying storage for it with the DiskManager.

void removeRelation (std::string name)

Deletes relation from the system and removes its underlying storage from the database.

• void removeFile (FileId file_id)

Deletes a File or Index from the system and removes its underlying storage from the database.

File * getRelation (std::string name)

Retuns File object identified by the relation name.

File * getFile (FileId file_id)

Retuns File object identified by the given FileId.

Protected Member Functions

• void _removeAllFiles ()

Removes all Files in the system and their underlying storage from the database.

void _closeAllFiles ()

Flushes and closes all underlying files in the system, saving db state.

Friends

· class SwatDB

4.16.1 Detailed Description

SwatDB FileManager Class. The interface to the file layer of the system: manages Files and Indexes.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 FileManager()

Creates SwatDB FileManager, the interface to the File layer.

NOTE: load paramameter has a defaulf value of false, which means that the constructor can be called just passing argument values for the first 2 parameter when a false value for load is desired.

Precondition

Input parameter objects (cat and buf_mgr) are valid.

Postcondition

If load is true, FileManager is constructed for SwatDB tat is loaded from existing files on disk. File objects are created for every relation and index file in the system and corresponding Catalog entries are updated with them. If load is false, FileManager is constructed for a new empty SwatDB.

Parameters

cat	Pointer to the SwatDB Catalog.	
buf_mgr	buf_mgr Pointer to the SwatDB BufferManager.	
load	bool indicating whether Files shoud be loaded from the existing SwatDB.	

4.16.2.2 \sim FileManager()

```
FileManager::~FileManager ()
```

Destructor.

SwatDB controls how all the database state is saved/removed at shut down and deletes the BufferManager and Catalog objects in its destructor

4.16.3 Member Function Documentation

4.16.3.1 _closeAllFiles()

```
void FileManager::_closeAllFiles ( ) [protected]
```

Flushes and closes all underlying files in the system, saving db state.

NOTE: This method is called on shutdown of the SwatDB, saving the db state.

Precondition

None.

Postcondition

Header page of each File is flushed to disk, but the underlying Unix files are not removed, effectively saving the current db state to disk.

4.16.3.2 _removeAllFiles()

```
void FileManager::_removeAllFiles ( ) [protected]
```

Removes all Files in the system and their underlying storage from the database.

NOTE: This method is called on shutdown of the SwatDB, removing the db state.

Precondition

None. $\mbox{\em Opost Underlying files}$ that store the db state of the $\mbox{\em SwatDB}$ are removed.

4.16.3.3 createRelation()

```
FileId FileManager::createRelation (
    std::string name,
    Schema * schema,
    CatType type,
    std::string file_name )
```

Creates a new Relation in the system. Adds an entry for the new file to the Catalog and creates the underlying storage for it with the DiskManager.

Precondition

A file with the requested relation or index name and type and underlying file name is not already defined in the system.

Postcondition

A new file or index of the specified type is added to the system. A new file or index object is created, its entry is added to the SwatDB catalog, and a Unix file to store its data is created on disk by the DiskManager.

Parameters

name	The Relation name in the system.	
schema	A pointer to a Schema object associated with the file. (for RelationFile types).	
type	type The specific type of file or index to create.	
file_name	The name of the disk file in which the new file object will be stored.	

Returns

FileId of the newly created file or index, or INVALID_FILE_ID if an error occurs.

Exceptions

FileAlreadyExistCat	if the file_name already exists in the database.
RelationAlreadyExistCat	if a relation named name already exists in the database.
IndexAlreadyExistsCat	if an index named name already exists in the database.
InvalidFileTypeFileMgr	if the passed CatType is invalid or not supported.

4.16.3.4 getFile()

Retuns File object identified by the given FileId.

Precondition

The File, identified by the FileId already exists.

Postcondition

File* is returned.

Parameters

file⊷	Fileld of the file to be returned.
_id	

Exceptions

FileNotFoundFileMgr if the file identified by the FileId does not exist

4.16.3.5 getRelation()

Retuns File object identified by the relation name.

Precondition

The realtion of the given name already exists.

Postcondition

File* is returned.

Parameters

name Relation ame of the relation file to be re	e returned.
---	-------------

Exceptions

```
FileNotFoundFileMgr if the file identified by the name does not exist.
```

4.16.3.6 removeFile()

Deletes a File or Index from the system and removes its underlying storage from the database.

Precondition

A valid FileId is provided as input.

Postcondition

The file identified by the given FileId is removed from the database.

Parameters

file⊷	FileId of the file to be deleted from the database.
id	

Exceptions

```
FileNotFoundFileMgr if the given FileId does not exist in the system.
```

4.16.3.7 removeRelation()

Deletes relation from the system and removes its underlying storage from the database.

Precondition

A valid relation name is provided as input.

Postcondition

The relation of the given name is removed from the database.

Parameters

name	Name of the relation to be deleted from the database.

Exceptions

FileNotFoundFileMgr	if the file identified by the FileId does not exist.

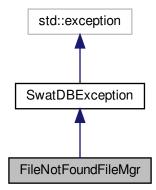
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/filemgr.h

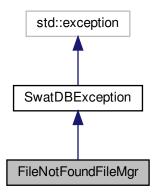
4.17 FileNotFoundFileMgr Class Reference

#include <exceptions.h>

Inheritance diagram for FileNotFoundFileMgr:



Collaboration diagram for FileNotFoundFileMgr:



Public Member Functions

• FileNotFoundFileMgr ()

Constructor.

• \sim FileNotFoundFileMgr () throw ()

Destructor.

Additional Inherited Members

4.17.1 Detailed Description

FileNotFoundFileMgr is thrown by FileManager if file is not found.

The documentation for this class was generated from the following file:

· /home/koh2/swatdb/SwatDB/include/exceptions.h

4.18 Frame Class Reference

```
#include <bufmgr.h>
```

Public Member Functions

• Frame ()

Constructor. Calls resetFrame to reset the Frame.

• ∼Frame ()

Destructor.

void resetFrame ()

Resets the metadata of the Frame.

void loadFrame (PageId page_id)

Updates the Frame data according to the loaded Page.

Friends

· class BufferManager

4.18.1 Detailed Description

Frame Class which holds metadata about corresponding Page in the buffer pool.

4.18.2 Member Function Documentation

4.18.2.1 loadFrame()

Updates the Frame data according to the loaded Page.

Precondition

None.

Postcondition

Frame data is updated. pin_count is set to 1, page_id is set to page_id parameter, and valid is set to true.

Parameters

page⊷	PageId of the Page that is loaded to the Frame
_id	

4.18.2.2 resetFrame()

```
void Frame::resetFrame ( )
```

Resets the metadata of the Frame.

Precondition

None.

Postcondition

page_id is set to INVALID_PAGE_ID. pin_count is set to 0. valid, dirty, and ref_bit are all set to false.

4.18.3 Friends And Related Function Documentation

4.18.3.1 BufferManager

```
friend class BufferManager [friend]
```

BufferManager has access to private data members of each Frame.

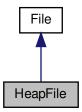
The documentation for this class was generated from the following file:

 $\bullet \ /home/koh2/swatdb/SwatDB/include/bufmgr.h$

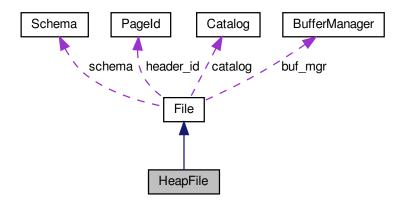
4.19 HeapFile Class Reference

#include <heapfile.h>

Inheritance diagram for HeapFile:



Collaboration diagram for HeapFile:



Public Member Functions

- HeapFile (Catalog *catalog, BufferManager *buf_mgr, Schema *schema)
 - Constructor for HeapFile class.
- ∼HeapFile ()

HeapFile destructor.

- void createHeader ()
 - Allocates and initializes the header Page of the file.
- · void flushHeader ()

Flushes header Page to disk.

• RecordId insertRecord (Record record)

Inserts a Record into the HeapFile.

void getRecord (RecordId record_id, Record *record)

Sets the record_data of the given Record pointer to the data corresponding to the given Recordld.

void updateRecord (RecordId record_id, Record record)

Updates the record data in the HeapFile identified by the given RecordId to the data in the provided Record.

void deleteRecord (RecordId record_id)

Deletes the Record identified by the given RecordId.

HeapFileHeader getHeader ()

THIS METHOD IS FOR DEBUGGING ONLY. Returns the current HeapFileHeader.

Friends

· class HeapFileScanner

Additional Inherited Members

4.19.1 Detailed Description

SwatDB HeapFile Class. Represents heap file in the system. It consists of doubly linked list of HeapPage and is an unsorted collection of records. Provides various methods, including inserting, modifying and retrieving records.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 HeapFile()

Constructor for HeapFile class.

Precondition

A valid Catalog pointer and BufferManager pointer are provided as inputs.

Postcondition

HeapFile object is constructed. Catalog and BufferManager data members are set to the provided inputs. Other values are initialized after construction.

Parameters

catalog	Pointer to the SwatDB Catalog object.
buf_mgr	Pointer to the SwatDB BufferManager object.
schema	Pointer to this file's Schema.

Generated by Doxygen

4.19.2.2 ∼HeapFile()

```
HeapFile::~HeapFile ( ) [inline]
```

HeapFile destructor.

Precondition

None.

Postcondition

The HeapFile object is destroyed. The file on disk is not removed.

4.19.3 Member Function Documentation

4.19.3.1 createHeader()

```
void HeapFile::createHeader ( ) [virtual]
```

Allocates and initializes the header Page of the file.

Precondition

There is sufficient space for a Page in buffer pool. The file with the corresponding FileId is already created.

Postcondition

A Page is allocated and the header_id field is initialized to the PageId of the allcated Page. The free and full fields are initialized to INVALID_PAGE_NUM and free_size and full_size are initialized to 0.

Exceptions

InsufficientSpaceBufMgr If there is not enough space in the bufferpool.

Implements File.

4.19.3.2 deleteRecord()

Deletes the Record identified by the given RecordId.

Precondition

A valid RecordId is provided.

Postcondition

Deletes the Record identified by the given Recordld. If the Page was in the list of full pages, and it is no longer full after deletion, then the Page is moved to the list of free pages. If the Page is empy after deletion, it is completely removed from any list, released, and deallocated. The header Page is updated appropriately. All pages pinned during the operation are released at the end or before exception is thrown.

Parameters

record↔	RecordId identifying the record data to be deleted.
_id	

Exceptions

InvalidSchemaHeapFile	If the given Record's Schema does not match that of the HeapFile(compare pointers).
InvalidPageIdBufMgr	If the PageNum of the given RecordId is not valid.
InvalidSlotIdHeapPage	If the SlotId of the given RecordId is not valid.

4.19.3.3 flushHeader()

```
void HeapFile::flushHeader ( ) [virtual]
```

Flushes header Page to disk.

Precondition

There is sufficient space for the header Page in buffer pool. header_id is valid.

Postcondition

Header Page of the File is flushed to disk.

Exceptions

InsufficientSpaceBufMgr	If there is not enough space in the bufferpool for the header Page.
InvalidPageIdBufMgr	If the header_id of the File is not valid.

Implements File.

4.19.3.4 getRecord()

Sets the record_data of the given Record pointer to the data corresponding to the given RecordId.

Precondition

A valid Recordld and a Record object pointer with a Schema matching that of the HeapFile.

Postcondition

The given Record pointer's data field is initialized to the data identified by the given RecordId. All pages pinned during the operation are released at the end or before exeption is thrown.

Parameters

record⊷	RecordId of the record_data to be retrieved.
_id	
record	Record pointer to store the retrieved data.

Exceptions

InvalidSchemaHeapFile	If the given Record pointer's Schema does not match that of the HeapFile
	(compares pointers).
InvalidPageIdBufMgr	If the PageNum of the given RecordId is not valid.
InvalidSlotIdHeapPage	If the SlotId of the given RecordId is not valid.

4.19.3.5 insertRecord()

Inserts a Record into the HeapFile.

Precondition

A valid Record object with a Schema matching that of the HeapFile is provided as input. There is some Page into which the Record can be inserted.

Postcondition

The Record is inserted into some Page that belongs to the HeapFile. If there is enough space on some Page in the list of free pages, then the Record is inserted there. If there is not enough space on any Page in the list of free pages, then a new Page is allocated. The Record is inserted into this Page. If the the Page is full after inserting the Record, the Page is moved to the list of full pages. Otherwise, the Page is added to/remains in free list. The header Page is updated appropriately. All pages pinned during the operation are released at the end or before exeption is thrown.

Parameters

record	Record to be inserted into the HeapFile.
--------	--

Returns

Recordld Recordld of the inserted Record.

Exceptions

InvalidSchemaHeapFile	If the given Record's Schema does not match that of the HeapFile (compares pointers).
InsufficientSpaceHeapPage	If the given Record's data exceeds the MAXIMUM_RECORD_SIZE.
InsufficientSpaceHeapFile	If the number of pages (including the header) in the HeapFile exceeds MAX_PAGE_NUM.

4.19.3.6 updateRecord()

Updates the record data in the HeapFile identified by the given RecordId to the data in the provided Record.

Precondition

A valid RecordId and a Record object with a Schema matching that of the HeapFile are provided as inputs. There is enough space in the HeapPage containing the Record identified by RecordId for the data in the provided record.

Postcondition

The Record data of the Record identified by the RecordId is replaced with the data of the provided Record. If the "full" state of the apge changes after the update, it is moved from one list to the other (free to full or full to free). The header Page is updated appropriately. All pages pinned during the operation are released at the end or before exeption is thrown.

Parameters

record← _id	RecordId identifying the record data to be updated.
record	Record object containing data to overwrite the record data identified by the given RecordId.

Exceptions

InvalidSchemaHeapFile	If the given Record's Schema does not match that of the HeapFile (compares
	pointers).

Exceptions

InvalidPageIdBufMgr	if the PageNum of the given RecordId is not valid.
InvalidSlotIdHeapPage	if the SlotId of the given RecordId is not valid.
InsufficientSpaceHeapPage	if there is not enough space for the updated record in the corresponding HeapPage.

The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/heapfile.h

4.20 HeapFileHeader Struct Reference

#include <heapfile.h>

Public Attributes

- · PageNum free
- PageNum full
- std::uint32_t free_size
- std::uint32_t full_size
- std::uint32_t num_records

4.20.1 Detailed Description

Struct for the header metadata of HeapFile object. The header is casted on top of the first Page allocated to the file.

4.20.2 Member Data Documentation

4.20.2.1 free

PageNum HeapFileHeader::free

PageNum of the Page at the head of the linked list of free pages.

4.20.2.2 free_size

std::uint32_t HeapFileHeader::free_size

Number of pages in the linked list of free pages.

4.20.2.3 full

PageNum HeapFileHeader::full

PageNum of the Page at the head of the linked list of full pages.

4.20.2.4 full_size

std::uint32_t HeapFileHeader::full_size

Number of pages in the linked list of full pages.

4.20.2.5 num_records

std::uint32_t HeapFileHeader::num_records

Number of records in the HeapFile.

The documentation for this struct was generated from the following file:

· /home/koh2/swatdb/SwatDB/include/heapfile.h

4.21 HeapFileScanner Class Reference

#include <heapfile.h>

Public Member Functions

• HeapFileScanner (HeapFile *file)

Constructor.

∼HeapFileScanner ()

Destructor.

RecordId getNext (Record *record)

Returns RecordId of the next Record in the HeapFile and initializes the given Record object to the data of the identified Record. Scans for records by iterating through the linked list of full pages, then the linked list of free pages.

4.21.1 Detailed Description

Scanner class for HeapFile. Scanner scans the given HeapFile object and returns the next RecordId and initiliazes the given Record Data whenever getNext is called. Returns INVALID_RECORD_ID if it reaches the end of the file.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 HeapFileScanner()

Constructor.

Precondition

Valid HeapFile* is provided as input.

Postcondition

HeapFile object is constructed with initialized data members. The cur_page is pinned if file is not empty.

Parameters

file HeapFile object to be scanned.

4.21.2.2 ∼HeapFileScanner()

```
\label{tempfileScanner::} \texttt{HeapFileScanner} \ \ (\ )
```

Destructor.

Precondition

If the end of the File has not been reached, cur_page is pinned.

Postcondition

If the end of the File has not been reached, cur_page is released.

4.21.3 Member Function Documentation

4.21.3.1 getNext()

Returns RecordId of the next Record in the HeapFile and initializes the given Record object to the data of the identified Record. Scans for records by iterating through the linked list of full pages, then the linked list of free pages.

Precondition

None

Postcondition

RecordId of the next Record is returned and data of the given Record object is initialized to that of the Record identified by the RecordId. If there are no more records in the HeapFile, INVALID_RECORD_ID is returned. cur_page is the Page that is being scanned and is pinned. Once the end of cur_page is reached, it is the next Page in the File is pinned (if there is next Page) and cur_page is unpinned. cur_pid and cur_page are updated accordingly. If the end of the File is reached, no Page is pinned by the HeapFileScanner.

Returns

Next valid RecordId. INVALID_RECORD_ID if the end of the file is reached.

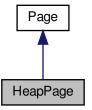
The documentation for this class was generated from the following file:

· /home/koh2/swatdb/SwatDB/include/heapfile.h

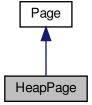
4.22 HeapPage Class Reference

#include <heappage.h>

Inheritance diagram for HeapPage:



Collaboration diagram for HeapPage:



Public Member Functions

• HeapPage ()=delete

Constructor.

- ∼HeapPage ()
- · void initializeHeader ()

Initializes header information after the Page is allocated.

void setNext (PageNum page_num)

Sets next_page to the given PageNum.

void setPrev (PageNum page_num)

Set prev_page to the given PageNum.

PageNum getNext ()

Getter for next_page.

• PageNum getPrev ()

Getter for prev_page.

• std::uint32_t getFreeSpace ()

Getter for the amount of free space free on the Page.

• bool isFull ()

bool function indicating whether the Page is full.

• bool isEmpty ()

bool function indicating whether the Page is empty.

SlotId insertRecord (Data *record data)

Inserts given record data to the Page.

void getRecord (SlotId slot_id, Data *data)

Gets the record identified by SlotId.

void deleteRecord (SlotId slot_id)

Deletes Record identified by SlotId.

void updateRecord (SlotId slot_id, Data *record_data)

Updates Record identified by SlotId.

HeapPageHeader getHeader ()

THIS METHOD IS FOR DEBUGGING ONLY. Returns this HeapPage's header information.

• SlotInfo getSlotInfo (SlotId slot_id)

THIS METHOD IS FOR DEBUGGING ONLY. Returns Slotlnfo struct of the given Slotld.

std::uint32_t getInvalidNum ()

THIS METHOD IS FOR DEBUGGING ONLY. Returns number of invalid slots in the HeapPage.

void printHeapPageState ()

THIS METHOD IS FOR DEBUGGING ONLY. Prints the current state of the HeapPage.

Friends

class HeapPageScanner

Additional Inherited Members

4.22.1 Detailed Description

SwatDB HeapPage Class. HeapPage inherits from base Page class and instantiates heap page, collection of which, form HeapFile.

4.22.2 Constructor & Destructor Documentation

4.22.2.1 HeapPage() HeapPage::HeapPage () [delete]

Constructor.

Constructor should never be called. It should be always the case that base class Page constructor is called by the BufferManager when initializing bufferpool. HeapPage Pointer is casted to whatever Page pointer returned by the BufferManager

4.22.2.2 ∼HeapPage()

```
HeapPage::~HeapPage ( ) [inline]
```

Destructor

4.22.3 Member Function Documentation

4.22.3.1 deleteRecord()

Deletes Record identified by SlotId.

Precondition

A valid SlotId is provided as input

Postcondition

If free_space_end is less than the slot offset of the corresponding slot info, the records to the left of the Record to be deleted are shifted to right by the size of the deleted Record, effectively compactifying space occupied by the records. The slot offset of all shifted Records are updated accordingly. free_space_end is incremented by size of the deleted Record. page header size is decremented. If SlotId is equal to page header capacity - 1, free_space_begin is decremented by sizeof(SlotInfo) and page header capacity is decremented. Else, slot offset of the slot previously occupied by deleted Record is set to INVALID_SLOT_OFFSET.

Parameters

slot⊷	SlotId of the Record to be deleted.
_id	

Exceptions

InvalidSlotIdHeapPage	If SlotId is invalid (SlotId is out of range or SlotInfo of the given SlotId has
	INVALID_SLOT_OFFSET).

4.22.3.2 getFreeSpace()

```
std::uint32_t HeapPage::getFreeSpace ( )
```

Getter for the amount of free space free on the Page.

Precondition

None.

Postcondition

The amount of free space in bytes on the Page is returned. If there is no free slot, sizeof(SlotInfo) is subtracted from the free space returned (accounting extra space occupied if new slot is allocated). If free space is less than sizeof(SlotInfo), 0 is returned.

Returns

Amount of free space in bytes on the Page.

4.22.3.3 getNext()

```
PageNum HeapPage::getNext ( )
```

Getter for next_page.

Precondition

None.

Postcondition

next_page of the page header is returned.

Returns

PageNum of the next Page.

4.22.3.4 getPrev()

```
PageNum HeapPage::getPrev ( )
```

Getter for prev_page.

Precondition

None.

Postcondition

prev_page of the Page header is returned.

Returns

PageNum of the previous Page.

4.22.3.5 getRecord()

Gets the record identified by SlotId.

Precondition

A valid SlotId is provided as input and a valid Data* with capacity that is greater than maximum size of any Record stored in the Page is provided. The inserted record must not have data size 0.

Postcondition

The data char array of Data object is appropriately initialized by copying the Record stored in Page to the data array of the object and the size of Data object is set to the number of bytes that are copied.

Parameters

slot⊷	SlotId of the Record to be retrieved
_id	

Exceptions

InvalidSlotIdHeapPage	If SlotId is out of range or SlotInfo of the given SlotId has INVALID_SLOT_OFFSET.
EmptyDataHeapPage	If the size of the given data is 0.

4.22.3.6 initializeHeader()

```
void HeapPage::initializeHeader ( )
```

Initializes header information after the Page is allocated.

Precondition

None

Postcondition

Page header is initialized such that prev_page and next_page are set to INVALID_PAGE_NUM, free_space _begin is set to sizeof(HeapPageHeader), free_space_end is set to PAGE_SIZE, size and capacity are set to 0.

4.22.3.7 insertRecord()

Inserts given record data to the Page.

Precondition

There is enough space in the Page and valid Data* is provided as input

Postcondition

A slot is allocated for inserting the record data. If there is enough space ((free_space_end - free_space_begin) is greater than size of record data), then data is copied to its size number of bytes in front of free_space_end. Page header size is incremented and capacity is also incremented if a new slot is allocated for the Record. free_space_end is decremented by the size of the record data. SlotInfo offset and length are updated. If new slot was allocated (equal to capacity -1), but there is not enough space for record_data, page_header capacity and free_space_begin are decremented back before throwing an exception (or free space was checked before a slot was allocated).

Parameters

	record_data	Data* of the record data to be inserted.
--	-------------	--

Returns

SlotId of the slot the Record is inserted to.

Exceptions

InsufficientSpaceHeapPage

If there is not enough space for the Record where sizeof(SlotInfo) is also conisdered if pageheader size is equal to page header capacity.

4.22.3.8 isEmpty()

```
bool HeapPage::isEmpty ( )
```

bool function indicating whether the Page is empty.

Precondition

None.

Postcondition

true is returned if size is 0. Else false is returned.

Returns

bool indicating whether the Page is empty.

4.22.3.9 isFull()

```
bool HeapPage::isFull ( )
```

bool function indicating whether the Page is full.

Precondition

None.

Postcondition

true is returned if used space/PAGE_SIZE exceeds MAX_HEAP_PAGE_LOAD and false is returned otherwise

Returns

bool indicating whether the Page is full.

4.22.3.10 setNext()

Sets next_page to the given PageNum.

Precondition

None.

Postcondition

next_page is set to the given PageNum.

Parameters

page_num	PageNum of the next Page.
----------	---------------------------

4.22.3.11 setPrev()

Set prev_page to the given PageNum.

Precondition

None.

Postcondition

prev_page is set to the given PageNum.

Parameters

page_num	PageNum of the previous Page.
----------	-------------------------------

4.22.3.12 updateRecord()

Updates Record identified by SlotId.

Precondition

A valid SlotId is provided as input and there is enough space in the Page for the updated Record. Valid Data* is provided as input.

Postcondition

Record stored in the slot identified by the given SlotId is updated to the given Record. Previous Record is not modified if there is not enough space for the updated Record.

Parameters

slot_id	SlotId of the Record to be updated.
record_data	Data* of the updated Record.

Exceptions

InsufficientSpaceHeapPage	If there is not enough space for updated Record.
InvalidSlotIdHeapPage	If SlotId is invalid (SlotId is out of range or SlotInfo of the given SlotId has
	INVALID_SLOT_OFFSET).

The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/heappage.h

4.23 HeapPageHeader Struct Reference

```
#include <heappage.h>
```

Public Attributes

- PageNum prev_page
- PageNum next_page
- std::uint32_t free_space_begin
- std::uint32_t free_space_end
- std::uint32 t size
- std::uint32_t capacity

4.23.1 Detailed Description

Struct for the header metadata of HeapPage object. The header is casted on top of the Page data array, from the beginning of the array. Must follow 64bit alignment.

4.23.2 Member Data Documentation

```
4.23.2.1 capacity
```

std::uint32_t HeapPageHeader::capacity

Number of allocated slots.

4.23.2.2 free_space_begin

std::uint32_t HeapPageHeader::free_space_begin

Offset where free space begins in the Page.

4.23.2.3 free_space_end

```
std::uint32_t HeapPageHeader::free_space_end
```

Offset where free space ends in the Page.

4.23.2.4 next_page

PageNum HeapPageHeader::next_page

PageNum of next Page in the HeapFile.

4.23.2.5 prev_page

PageNum HeapPageHeader::prev_page

PageNum of previous Page in the HeapFile.

4.23.2.6 size

```
std::uint32_t HeapPageHeader::size
```

Number of valid/filled slots.

The documentation for this struct was generated from the following file:

/home/koh2/swatdb/SwatDB/include/heappage.h

4.24 HeapPageScanner Class Reference

```
#include <heappage.h>
```

Public Member Functions

HeapPageScanner (HeapPage *page)

Constructor.

∼HeapPageScanner ()

Destructor.

• SlotId getNext ()

Returns SlotId of the next valid slot.

void reset (HeapPage *page)

Resets the scanner, so it could be used for another Page.

4.24.1 Detailed Description

Scanner class for scanning HeapPage.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 HeapPageScanner()

Constructor.

Precondition

Valid HeapPage* is provided as input. The given Page is pinned.

Postcondition

HeapPageScanner is constructed with initialized data members. page is still pinned.

Parameters

page	HeapPage object to be scanned.
------	--------------------------------

4.24.3 Member Function Documentation

4.24.3.1 getNext()

```
SlotId HeapPageScanner::getNext ( )
```

Returns SlotId of the next valid slot.

Precondition

page is pinned.

Postcondition

SlotId of the next valid slot is returned. Current slot field is set to the next SlotId. Current slot field may be incremented by more than one. If the scanner reaches the end of the slot directory, INVALID_SLOT_ID is returned. page is still pinned.

Returns

Next valid SlotId. INVALID_SLOT_ID if the end of the Page is reached. page is still pinned.

4.24.3.2 reset()

Resets the scanner, so it could be used for another Page.

Precondition

The new Page is pinned.

Postcondition

page is set to the provided HeapPage* and current slot is reset to 0. The new Page is still pinned.

Parameters

page	HeapPage object to reset to.
------	------------------------------

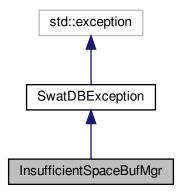
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/heappage.h

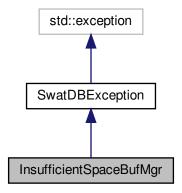
4.25 InsufficientSpaceBufMgr Class Reference

```
#include <exceptions.h>
```

 $Inheritance\ diagram\ for\ Insufficient Space BufMgr:$



Collaboration diagram for InsufficientSpaceBufMgr:



Public Member Functions

- InsufficientSpaceBufMgr ()
 - Constructor.
- ∼InsufficientSpaceBufMgr () throw ()

Destructor.

Additional Inherited Members

4.25.1 Detailed Description

InsufficientSpaceBufMgr is thrown by BufferManager if there is not enough space in the buffer pool.

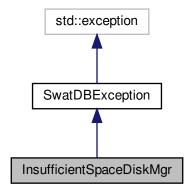
The documentation for this class was generated from the following file:

· /home/koh2/swatdb/SwatDB/include/exceptions.h

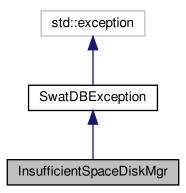
4.26 InsufficientSpaceDiskMgr Class Reference

#include <exceptions.h>

Inheritance diagram for InsufficientSpaceDiskMgr:



Collaboration diagram for InsufficientSpaceDiskMgr:



Public Member Functions

• InsufficientSpaceDiskMgr (FileId file_id)

Constructor.

• \sim InsufficientSpaceDiskMgr () throw ()

Destructor.

• FileId getFileId () const throw ()

Returns the FileId of the file with insufficient space.

Additional Inherited Members

4.26.1 Detailed Description

InsufficientSpaceDiskMgr is thrown by DiskManager if there is not enough space in file.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 InsufficientSpaceDiskMgr()

Constructor.

Parameters

ī	file_id.	FileId of the file with insuffucient space.
---	----------	---

4.26.3 Member Function Documentation

4.26.3.1 getFileId()

```
\label{thm:paceDiskMgr::getFileId ( ) const throw )} FileId \ InsufficientSpaceDiskMgr::getFileId ( ) \ const throw )
```

Returns the FileId of the file with insufficient space.

Returns

FileId of the file with insufficient space.

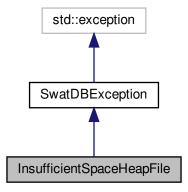
The documentation for this class was generated from the following file:

/home/koh2/swatdb/SwatDB/include/exceptions.h

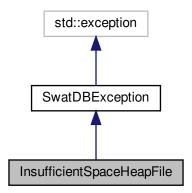
4.27 InsufficientSpaceHeapFile Class Reference

#include <exceptions.h>

Inheritance diagram for InsufficientSpaceHeapFile:



Collaboration diagram for InsufficientSpaceHeapFile:



Public Member Functions

• InsufficientSpaceHeapFile ()

Constructor.

∼InsufficientSpaceHeapFile () throw ()

Destructor.

Additional Inherited Members

4.27.1 Detailed Description

InvalidSchemaHeapFile is thrown by HeapFile if the total number of Pages of the HeapFile is MAX_PAGE_NUM.

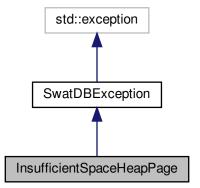
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

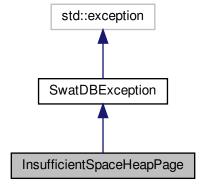
4.28 InsufficientSpaceHeapPage Class Reference

#include <exceptions.h>

Inheritance diagram for InsufficientSpaceHeapPage:



Collaboration diagram for InsufficientSpaceHeapPage:



Public Member Functions

• InsufficientSpaceHeapPage ()

Constructor.

• ~InsufficientSpaceHeapPage () throw ()

Destructor.

Additional Inherited Members

4.28.1 Detailed Description

InsufficientSpaceHeapPage is thrown by HeapPage if space is insufficient.

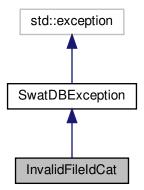
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

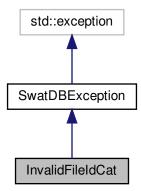
4.29 InvalidFileIdCat Class Reference

#include <exceptions.h>

Inheritance diagram for InvalidFileIdCat:



Collaboration diagram for InvalidFileIdCat:



Public Member Functions

- InvalidFileIdCat (FileId file_id)
 - Constructor.
- \sim InvalidFileIdCat () throw ()

Destructor.

• FileId getFileId () const throw ()

Returns the invalid FileId.

Additional Inherited Members

4.29.1 Detailed Description

InvalidFileIdCat is thrown by Catalog if FileId is invalid.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 InvalidFileIdCat()

Constructor.

Parameters

file_id.	Invalid FileId.
----------	-----------------

4.29.3 Member Function Documentation

4.29.3.1 getFileId()

```
FileId InvalidFileIdCat::getFileId ( ) const throw )
```

Returns the invalid FileId.

Returns

Invalid FileId.

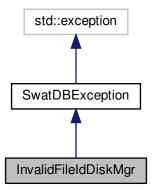
The documentation for this class was generated from the following file:

 $\bullet \ /home/koh2/swatdb/SwatDB/include/exceptions.h$

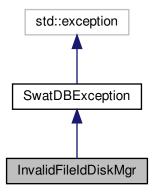
4.30 InvalidFileIdDiskMgr Class Reference

```
#include <exceptions.h>
```

 $Inheritance\ diagram\ for\ InvalidFileIdDiskMgr:$



Collaboration diagram for InvalidFileIdDiskMgr:



Public Member Functions

• InvalidFileIdDiskMgr (FileId file_id)

Constructor.

• \sim InvalidFileIdDiskMgr () throw ()

Destructor.

• FileId getFileId () const throw ()

Returns the invalid FileId.

Additional Inherited Members

4.30.1 Detailed Description

DiskManager Exceptions InvalidFileIdDiskMgr is thrown by DiskManager if FileId is invalid.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 InvalidFileIdDiskMgr()

Constructor.

Parameters

file_id.	Invalid FileId.
----------	-----------------

4.30.3 Member Function Documentation

4.30.3.1 getFileId()

```
FileId InvalidFileIdDiskMgr::getFileId ( ) const throw )
```

Returns the invalid FileId.

Returns

Invalid FileId.

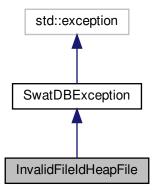
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

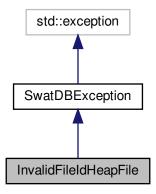
4.31 InvalidFileIdHeapFile Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for InvalidFileIdHeapFile:



Collaboration diagram for InvalidFileIdHeapFile:



Public Member Functions

• InvalidFileIdHeapFile (FileId file_id)

Constructor.

• \sim InvalidFileIdHeapFile () throw ()

Destructor.

• FileId getFileId () const throw ()

Returns the invalid FileId.

Additional Inherited Members

4.31.1 Detailed Description

HeapFile Exceptions. InvalidFileIdHeapFile is thrown by Heapfile if FileId does not match that of the HeapFile.

4.31.2 Constructor & Destructor Documentation

4.31.2.1 InvalidFileIdHeapFile()

Constructor.

Parameters

file id.	Invalid FileId.

4.31.3 Member Function Documentation

4.31.3.1 getFileId()

```
FileId InvalidFileIdHeapFile::getFileId ( ) const throw )
```

Returns the invalid FileId.

Returns

Invalid FileId.

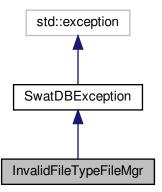
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

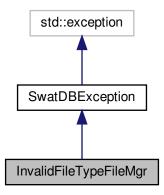
4.32 InvalidFileTypeFileMgr Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for InvalidFileTypeFileMgr:



Collaboration diagram for InvalidFileTypeFileMgr:



Public Member Functions

- InvalidFileTypeFileMgr (CatType file_type)
 - Constructor.
- \sim InvalidFileTypeFileMgr () throw ()

Destructor.

• CatType getFileType () const throw ()

Returns the Pageld of the pinned Page.

Additional Inherited Members

4.32.1 Detailed Description

FileManager Exceptions InvalidFileTypeFileMgr is thrown by FileManager if file type is inconsistent.

4.32.2 Constructor & Destructor Documentation

4.32.2.1 InvalidFileTypeFileMgr()

Constructor.

Parameters

file_type. File type of the file.

4.32.3 Member Function Documentation

4.32.3.1 getFileType()

CatType InvalidFileTypeFileMgr::getFileType () const throw)

Returns the Pageld of the pinned Page.

Returns

Pageld of the pinned Page.

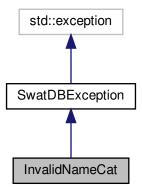
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

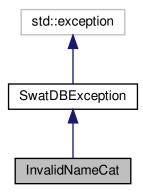
4.33 InvalidNameCat Class Reference

#include <exceptions.h>

Inheritance diagram for InvalidNameCat:



Collaboration diagram for InvalidNameCat:



Public Member Functions

• InvalidNameCat (const std::string &name)

Constructor.

∼InvalidNameCat () throw ()

Destructor

• std::string getName () const throw ()

Returns the invalid relation name.

Additional Inherited Members

4.33.1 Detailed Description

InvalidNameCat is thrown by Catalog if Relation Name is not valid.

4.33.2 Constructor & Destructor Documentation

4.33.2.1 InvalidNameCat()

Constructor.

Parameters

name Invalid Re	elation name.
-----------------	---------------

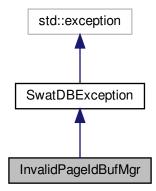
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

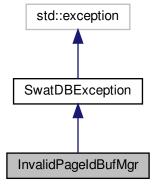
4.34 InvalidPageIdBufMgr Class Reference

#include <exceptions.h>

Inheritance diagram for InvalidPageIdBufMgr:



Collaboration diagram for InvalidPageIdBufMgr:



Public Member Functions

InvalidPageIdBufMgr (PageId page_id)

Constructor.

• ~InvalidPageIdBufMgr () throw ()

Destructor.

• Pageld getPageld () const throw ()

Returns the invalid Pageld.

Additional Inherited Members

4.34.1 Detailed Description

BufferManager Exceptions InvalidPageIdBufMgr is thrown by BufferManager if PageId is invalid.

4.34.2 Constructor & Destructor Documentation

4.34.2.1 InvalidPageIdBufMgr()

Constructor.

Parameters

```
page_id. Invalid Pageld.
```

4.34.3 Member Function Documentation

```
4.34.3.1 getPageId()
```

```
PageId InvalidPageIdBufMgr::getPageId ( ) const throw )
```

Returns the invalid Pageld.

Returns

Invalid Pageld.

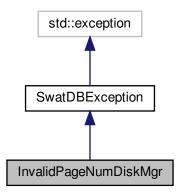
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

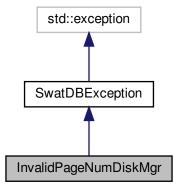
4.35 InvalidPageNumDiskMgr Class Reference

#include <exceptions.h>

Inheritance diagram for InvalidPageNumDiskMgr:



Collaboration diagram for InvalidPageNumDiskMgr:



Public Member Functions

• InvalidPageNumDiskMgr (PageNum page_num)

Constructor.

• \sim InvalidPageNumDiskMgr () throw ()

Destructor.

• PageNum getPageNum () const throw ()

Returns the invalid PageNum.

Additional Inherited Members

4.35.1 Detailed Description

InvalidPageNumDiskMgr is thrown by DiskManager if PageNum is invalid.

4.35.2 Constructor & Destructor Documentation

4.35.2.1 InvalidPageNumDiskMgr()

Constructor.

Parameters

page_num.	Invalid PageNum.
-----------	------------------

4.35.3 Member Function Documentation

4.35.3.1 getPageNum()

```
PageNum InvalidPageNumDiskMgr::getPageNum ( ) const throw )
```

Returns the invalid PageNum.

Returns

Invalid PageNum.

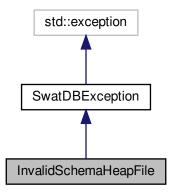
The documentation for this class was generated from the following file:

/home/koh2/swatdb/SwatDB/include/exceptions.h

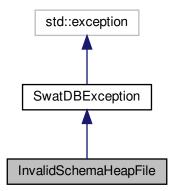
4.36 InvalidSchemaHeapFile Class Reference

#include <exceptions.h>

Inheritance diagram for InvalidSchemaHeapFile:



Collaboration diagram for InvalidSchemaHeapFile:



Public Member Functions

• InvalidSchemaHeapFile ()

Constructor.

• \sim InvalidSchemaHeapFile () throw ()

Destructor.

Additional Inherited Members

4.36.1 Detailed Description

InvalidSchemaHeapFile is thrown by HeapFile if Schema does not match that of the HeapFile.

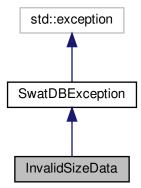
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

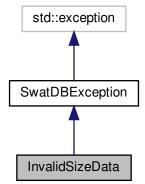
4.37 InvalidSizeData Class Reference

#include <exceptions.h>

Inheritance diagram for InvalidSizeData:



Collaboration diagram for InvalidSizeData:



Public Member Functions

• InvalidSizeData ()

Constructor.

∼InvalidSizeData () throw ()

Destructor.

Additional Inherited Members

4.37.1 Detailed Description

Data Exceptions. InvalidSizeData is thrown by Data if size exceeds capacity.

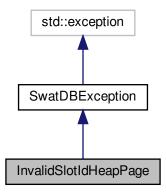
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

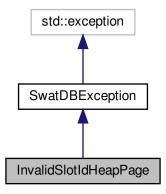
4.38 InvalidSlotIdHeapPage Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for InvalidSlotIdHeapPage:



Collaboration diagram for InvalidSlotIdHeapPage:



Public Member Functions

• InvalidSlotIdHeapPage (SlotId slot_id)

Constructor.

• \sim InvalidSlotIdHeapPage () throw ()

Destructor.

• SlotId getSlotId () const throw ()

Returns the invalid SlotId.

Additional Inherited Members

4.38.1 Detailed Description

HeapPage Exceptions. InvalidSlotIdHeapPage is thrown by HeapPage if SlotId is invalid.

4.38.2 Constructor & Destructor Documentation

4.38.2.1 InvalidSlotIdHeapPage()

```
\label{local_invalid} InvalidSlotIdHeapPage::InvalidSlotIdHeapPage \mbox{ (} \\ SlotId \mbox{ $slot\_id$ ) $ [explicit]$ }
```

Constructor.

Parameters

4.38.3 Member Function Documentation

4.38.3.1 getSlotId()

```
SlotId InvalidSlotIdHeapPage::getSlotId ( ) const throw )
```

Returns the invalid SlotId.

Returns

Invalid SlotId.

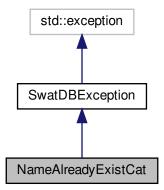
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

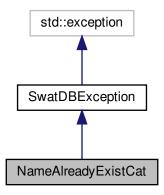
4.39 NameAlreadyExistCat Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for NameAlreadyExistCat:



Collaboration diagram for NameAlreadyExistCat:



Public Member Functions

• NameAlreadyExistCat (const std::string &relname)

Constructor.

∼NameAlreadyExistCat () throw ()

Destructor.

• std::string getName () const throw ()

Returns the name of the relation that already exists.

Additional Inherited Members

4.39.1 Detailed Description

NameAlreadyExistCat is thrown by Catalog on a new relation or index addition to the system if the relation name already exits.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 NameAlreadyExistCat()

Constructor.

Parameters

file. Name of the relation that already exists.

4.39.3 Member Function Documentation

4.39.3.1 getName()

```
std::string NameAlreadyExistCat::getName ( ) const throw )
```

Returns the name of the relation that already exists.

Returns

Name of the relation that already exists.

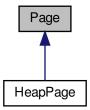
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

4.40 Page Class Reference

```
#include <page.h>
```

Inheritance diagram for Page:



Public Member Functions

• Page ()

Constructor.

∼Page ()

Destructor.

char * getData ()

Get function for the data char array.

Protected Attributes

• char data [PAGE_SIZE]

char array that stores the page data. Derived classes could map appropriate structures on it.

Friends

· class DiskManager

4.40.1 Detailed Description

SwatDB Page Class. Page is the basic unit of read/write operation on disk in the system. The data member is PAGE_SIZE char array, onto which other derived classes could map layer-specific structures and define access methods.

4.40.2 Member Function Documentation

```
4.40.2.1 getData()
char* Page::getData ( )
```

Get function for the data char array.

Precondition

None.

Postcondition

char* to the data char array is returned.

Returns

char* to the data char array.

The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/page.h

4.41 Pageld Struct Reference

#include <swatdb_types.h>

Public Member Functions

- bool operator== (const PageId &other) const
- bool operator!= (const Pageld &other) const

Public Attributes

- FileId file_id
- PageNum page_num

4.41.1 Detailed Description

Unique identifier of each file/index page in the system.

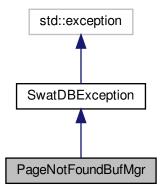
The documentation for this struct was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/swatdb_types.h

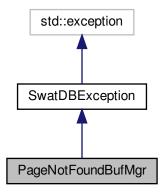
4.42 PageNotFoundBufMgr Class Reference

#include <exceptions.h>

Inheritance diagram for PageNotFoundBufMgr:



Collaboration diagram for PageNotFoundBufMgr:



Public Member Functions

- PageNotFoundBufMgr (PageId page_id)
 - Constructor.
- \sim PageNotFoundBufMgr () throw ()

Destructor.

• Pageld getPageld () const throw ()

Returns the Pageld of the Page not found.

Additional Inherited Members

4.42.1 Detailed Description

PageNotFoundBufMgr is thrown by BufferManager if PageId is not found in bufferpool.

4.42.2 Constructor & Destructor Documentation

4.42.2.1 PageNotFoundBufMgr()

Constructor.

Parameters

page_id.	Pageld of the Page not found.
----------	-------------------------------

4.42.3 Member Function Documentation

4.42.3.1 getPageId()

```
PageId PageNotFoundBufMgr::getPageId ( ) const throw )
```

Returns the Pageld of the Page not found.

Returns

PageId of the Page not found.

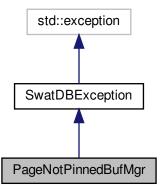
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

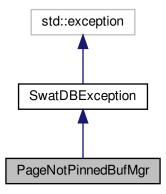
4.43 PageNotPinnedBufMgr Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for PageNotPinnedBufMgr:



Collaboration diagram for PageNotPinnedBufMgr:



Public Member Functions

PageNotPinnedBufMgr (PageId page_id)

Constructor.

• \sim PageNotPinnedBufMgr () throw ()

Destructor.

• Pageld getPageld () const throw ()

Returns the PageId of the Page not pinned.

Additional Inherited Members

4.43.1 Detailed Description

PageNotPinnedBufMgr is thrown by BufferManager if Page is not pinned.

4.43.2 Constructor & Destructor Documentation

4.43.2.1 PageNotPinnedBufMgr()

Constructor.

Parameters

page id.	PageId of the Page that is not pinned.
page_iu.	i age to the rage that is not printed.

4.43.3 Member Function Documentation

4.43.3.1 getPageId()

```
PageId PageNotPinnedBufMgr::getPageId ( ) const throw )
```

Returns the PageId of the Page not pinned.

Returns

Pageld of the Page not pinned.

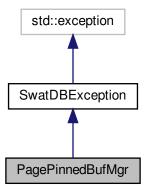
The documentation for this class was generated from the following file:

• /home/koh2/swatdb/SwatDB/include/exceptions.h

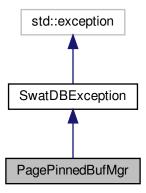
4.44 PagePinnedBufMgr Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for PagePinnedBufMgr:



Collaboration diagram for PagePinnedBufMgr:



Public Member Functions

• PagePinnedBufMgr (PageId page_id)

Constructor.

∼PagePinnedBufMgr () throw ()

Destructor.

• Pageld getPageld () const throw ()

Returns the Pageld of the pinned Page.

Additional Inherited Members

4.44.1 Detailed Description

PagePinnedBufMgr is thrown by BufferManager if Page is pinned.

4.44.2 Constructor & Destructor Documentation

4.44.2.1 PagePinnedBufMgr()

Constructor.

Parameters

page_id. PageId of the pinned Page.

4.44.3 Member Function Documentation

```
4.44.3.1 getPageId()
```

```
PageId PagePinnedBufMgr::getPageId ( ) const throw )
```

Returns the Pageld of the pinned Page.

Returns

Pageld of the pinned Page.

The documentation for this class was generated from the following file:

· /home/koh2/swatdb/SwatDB/include/exceptions.h

4.45 Record Class Reference

```
#include <record.h>
```

Public Member Functions

• Record ()

Default constructor. Sets schema and record_data to nullptr.

• Record (Schema *schema, Data *record_data)

Constructor given appropriate Data* and Schema*.

∼Record ()

Destructor.

Schema * getSchema ()

Getter function for schema.

Data * getRecordData ()

Getter for record_data.

void setSchema (Schema *new_schema)

Setter for schema.

void setRecordData (Data *new_data)

Setter for record_data.

• std::int32_t compare (Record other)

Compare function for comparing 2 records.

4.45.1 Detailed Description

SwatDB Record Class. Record instantiates record in SwatDB. It consists of record data Data object, which is serialized in form of variable length record and Schema object, which allows appropriate access to the record data.

4.45.2 Constructor & Destructor Documentation

4.45.2.1 Record()

Constructor given appropriate Data* and Schema*.

Precondition

Valid Data* and Schema* are provided as input. The schema and the record data stored by record_data are consistent.

Postcondition

schema and record_data are set to the provided input.

Parameters

schema	Schema* of the Record.
record_data	Data* record_data that stores the Record data.

4.45.2.2 \sim Record()

```
Record::~Record ( ) [inline]
```

Destructor.

Precondition

None.

Postcondition

Neither schema nor record_data is deleted. Both data members have to be deallocated manually by the user.

4.45.3 Member Function Documentation

4.45.3.1 compare()

Compare function for comparing 2 records.

Precondition

Valid Record object is provided as input.

Postcondition

If the schemas are different, -1 is returned. If the record_data of the two records do not have the same size or do not have the identical array up to first size number of bytes, -1 is returned. Else, 0 is returned.

Parameters

other The Record to be compared to.

Returns

-1 if the two records are not the same. 0 if the two records are the same.

4.45.3.2 getRecordData()

```
Data* Record::getRecordData ( )
```

Getter for record_data.

Precondition

None.

Postcondition

record_data is returned.

Returns

Data* to the record_data of the Record.

4.45.3.3 getSchema()

```
Schema* Record::getSchema ( )
```

Getter function for schema.

Precondition

None.

Postcondition

schema is returned.

Returns

Schema* to the schema of the Record.

4.45.3.4 setRecordData()

Setter for record_data.

Precondition

Valid Data* is provided as an input.

Postcondition

record_data is set to new_data. The previous data is not deleted.

Parameters

```
new_data New Data* schema is set to.
```

4.45.3.5 setSchema()

Setter for schema.

Precondition

Valid Schema* is provided as an input.

Postcondition

schema is set to new_schema. The previous schema is not deleted.

Parameters

The documentation for this class was generated from the following file:

/home/koh2/swatdb/SwatDB/include/record.h

4.46 RecordId Struct Reference

```
#include <swatdb_types.h>
```

Public Member Functions

- bool operator== (const RecordId &other) const
- bool operator!= (const RecordId &other) const

Public Attributes

- PageNum page_num
- SlotId slot id

4.46.1 Detailed Description

Unique identifier of each record in a single file.

The documentation for this struct was generated from the following file:

/home/koh2/swatdb/SwatDB/include/swatdb_types.h

4.47 Schema Class Reference

#include <schema.h>

Public Member Functions

· Schema ()

Constructor.

∼Schema ()

Destructor.

4.47.1 Detailed Description

SwatDB Schema Class. NOT FULLY IMPLEMENTED. This is a class to represent the schema for a relation. Provides structure to Data of a Record, allowing for various access methods.

The documentation for this class was generated from the following file:

· /home/koh2/swatdb/SwatDB/include/schema.h

4.48 SlotInfo Struct Reference

#include <heappage.h>

Public Attributes

- uint32_t offset
- uint32_t length

4.48.1 Detailed Description

Struct for storing metadata of each slot in a Page. Array of SlotInfo forms the slot directory of the Page. Must be 64bit for alignment.

4.48.2 Member Data Documentation

4.48.2.1 length

uint32_t SlotInfo::length

Length of the record in the slot described by the SlotInfo

4.48.2.2 offset

```
uint32_t SlotInfo::offset
```

Offset at which slot is located. INVALID_SLOT_OFFSET if slot is not valid

The documentation for this struct was generated from the following file:

/home/koh2/swatdb/SwatDB/include/heappage.h

4.49 SwatDB Class Reference

```
#include <swatdb.h>
```

Public Member Functions

· SwatDB ()

Constructor.

• SwatDB (std::string metadata filename)

Constructor to init SwatDB from a metadata file.

∼SwatDB ()

Shutdown SwatDB.

void setDestroyDB ()

sets the DB to be destroyed on shutdown.

void setSaveDB (std::string filename)

sets the DB to be saved on shutdown.

Catalog * getCatalog ()

Gets the SwatDB Catalog.

DiskManager * getDiskMgr ()

Gets the SwatDB DiskManager.

BufferManager * getBufMgr ()

Gets the SwatDB BufferManager.

FileManager * getFileMgr ()

Gets the SwatDB FileManager.

4.49.1 Detailed Description

SwatDB. This is the class definition of the high-level SwatDB object that stores all state about the Swat DBMS system as it runs. This includes:

- · The Catalog object of its relations and indices
- · The Buffer Manager object
- The Disk Manager object
- · The File Manager object
- · And possibly others in the future (lock mgr, xact mgr)

It contains methods to boot SwatDB and to save or delete SwatDB, as well as to access layer manager objects for the different parts of the system.

On (or before) shutdown a caller should decide if they want to save or destroy the db and do so before invoking the destructor (the default is to destroy if neither have been invoked):

swatdb->setSaveDB() swatdb->setDestroyDB() delete swatdb delete swatdb

4.49.2 Constructor & Destructor Documentation

```
4.49.2.1 SwatDB() [1/2] SwatDB::SwatDB ( )
```

Constructor.

Initialize/boot swatDB. Initializes an empty swatDB and creates default manager objects for all the layers

Constructor to init SwatDB from a metadata file.

This method creates RelationFile and IndexFile objects associated with each stored file listed in the metadata. SwatDB creates the diskmgr before the file manager, and it opens the relation and index files on disk and checks that they exist.

Parameters

me	tadata_filename	the name of the input file containing information about the DB state to init SwatDB with	
----	-----------------	--	--

Exceptions

exceptions	from system layers with init errors
------------	-------------------------------------

```
4.49.2.3 \simSwatDB()
SwatDB::\simSwatDB ( )
```

Shutdown SwatDB.

If saveDB is not called before the destructor is invoked then the destructor calls destroyDB to remove all relation and index files from the system

Precondition

: SwatDB may have some relation and index files

Postcondition

: Based on if saveDB or destroyDB was invoked prior to the destructory either all index and relation files are saved and metadata about them is written out or all files and indexes removed from the system, and their associated storage on "disk" is also removed from the system. Any state created is removed from the system.

4.49.3 Member Function Documentation

4.49.3.1 setDestroyDB()

```
void SwatDB::setDestroyDB ( )
```

sets the DB to be destroyed on shutdown.

This method can be called prior to shuting down swatDB if the current state of swatDB does not want to be saved (destroy on exit is the default). This method does not actually remove db files from the system but sets a flag to trigger the right actions by the destructor on shutdown.

4.49.3.2 setSaveDB()

```
void SwatDB::setSaveDB (
    std::string filename )
```

sets the DB to be saved on shutdown.

This method should be called prior to shuting down swatDB to save the state of the db on shutdown. This method does not actually save the state at the point it is called but set up the system to save the state on shutdown

Parameters

Precondition

: a file with filename does or does not already exist in the system (or an existing one's contents will be replaced)

Postcondition

: flag is set to trigger saving db metadata to file on shutdown

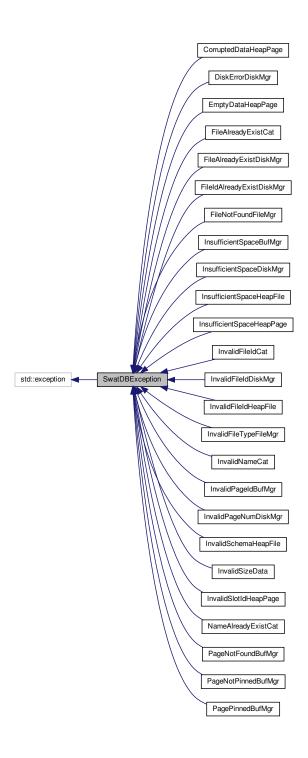
The documentation for this class was generated from the following file:

/home/koh2/swatdb/SwatDB/include/swatdb.h

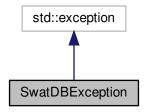
4.50 SwatDBException Class Reference

#include <exceptions.h>

Inheritance diagram for SwatDBException:



Collaboration diagram for SwatDBException:



Public Member Functions

- SwatDBException (const std::string &msg)
- virtual ~SwatDBException () throw ()
- virtual const char * what () const throw ()

Returns the error message of the exception.

Protected Attributes

• std::string message

4.50.1 Detailed Description

SwatDBException. SwatDBException is the base class for all exceptions thrown in SwatDB system. It is based on std::exception and other exceptions defined below inherit from this class.

4.50.2 Constructor & Destructor Documentation

4.50.2.1 SwatDBException()

SwatDBExcpetion constructor. Requires string message

4.50.2.2 ∼SwatDBException()

```
virtual SwatDBException::~SwatDBException ( ) throw ) [inline], [virtual]
```

SwatDBException destructor.

4.50.3 Member Function Documentation

4.50.3.1 what()

```
virtual const char* SwatDBException::what ( ) const throw ) [inline], [virtual]
```

Returns the error message of the exception.

Precondition

The exception is thrown and caught.

Postcondition

The error message is returned.

Returns

Pointer to the error message.

4.50.4 Member Data Documentation

4.50.4.1 message

```
std::string SwatDBException::message [protected]
```

Message that this exception will output.

The documentation for this class was generated from the following file:

/home/koh2/swatdb/SwatDB/include/exceptions.h

Index

_closeAllFiles	flushPage, 11
FileManager, 57	Frame, 63
_removeAllFiles	getBufferState, 11
FileManager, 57	getPage, 11
saveDBStateToFile	releasePage, 12
Catalog, 21	removeFile, 13
setFile	setDirty, 13
	BufferMap, 14
Catalog, 21	•
_setMyFid	contains, 15
File, 47	get, 15
\sim BufferManager	insert, 16
BufferManager, 8	remove, 16
\sim Catalog	BufferState, 17
Catalog, 20	clock_hand, 17
~Data	dirty, 18
Data, 32	pinned, 18
~DiskManager	ref_bit, 18
DiskManager, 37	total, 18
~FileManager	unpinned, 18
•	•
FileManager, 57	valid, 18
~HeapFile	acano situ
HeapFile, 66	capacity
\sim HeapFileScanner	HeapPageHeader, 81
HeapFileScanner, 72	Catalog, 19
\sim HeapPage	_saveDBStateToFile, 21
HeapPage, 75	_setFile, 21
~Record	\sim Catalog, $ extstyle{20}$
Record, 119	addEntry, 22
~SwatDB	Catalog, 20
SwatDB, 125	deleteEntry, 22
	getFile, 23
~SwatDBException	getFileId, 24
SwatDBException, 128	getFileIds, 24
UE .	•
addEntry	getFileName, 24
Catalog, 22	getSchema, 25
allocatePage	getType, 26
BufferManager, 9	catalog
DiskManager, 37	
Diskiviariager, 37	File, 49
alloced	
alloced	File, 49
	File, 49 CatalogEntry, 27 alloced, 27
alloced CatalogEntry, 27	File, 49 CatalogEntry, 27 alloced, 27 entry_type, 27
alloced CatalogEntry, 27 buf_mgr	File, 49 CatalogEntry, 27 alloced, 27 entry_type, 27 file, 28
alloced CatalogEntry, 27 buf_mgr File, 48	File, 49 CatalogEntry, 27 alloced, 27 entry_type, 27 file, 28 file_id, 28
alloced CatalogEntry, 27 buf_mgr File, 48 BufHash, 19	File, 49 CatalogEntry, 27 alloced, 27 entry_type, 27 file, 28 file_id, 28 file_name, 28
alloced CatalogEntry, 27 buf_mgr File, 48 BufHash, 19 BufferManager, 7	File, 49 CatalogEntry, 27 alloced, 27 entry_type, 27 file, 28 file_id, 28 file_name, 28 name, 28
alloced CatalogEntry, 27 buf_mgr File, 48 BufHash, 19 BufferManager, 7 ~BufferManager, 8	File, 49 CatalogEntry, 27 alloced, 27 entry_type, 27 file, 28 file_id, 28 file_name, 28 name, 28 schema, 28
alloced CatalogEntry, 27 buf_mgr File, 48 BufHash, 19 BufferManager, 7 ~BufferManager, 8 allocatePage, 9	File, 49 CatalogEntry, 27 alloced, 27 entry_type, 27 file, 28 file_id, 28 file_name, 28 name, 28 schema, 28 valid, 28
alloced CatalogEntry, 27 buf_mgr File, 48 BufHash, 19 BufferManager, 7 ~BufferManager, 8 allocatePage, 9 BufferManager, 8	File, 49 CatalogEntry, 27 alloced, 27 entry_type, 27 file, 28 file_id, 28 file_name, 28 name, 28 schema, 28 valid, 28 clock_hand
alloced CatalogEntry, 27 buf_mgr File, 48 BufHash, 19 BufferManager, 7 ~BufferManager, 8 allocatePage, 9 BufferManager, 8 createFile, 10	File, 49 CatalogEntry, 27 alloced, 27 entry_type, 27 file, 28 file_id, 28 file_name, 28 name, 28 schema, 28 valid, 28
alloced CatalogEntry, 27 buf_mgr File, 48 BufHash, 19 BufferManager, 7 ~BufferManager, 8 allocatePage, 9 BufferManager, 8	File, 49 CatalogEntry, 27 alloced, 27 entry_type, 27 file, 28 file_id, 28 file_name, 28 name, 28 schema, 28 valid, 28 clock_hand

Record, 120	file
contains	CatalogEntry, 28
BufferMap, 15	file_id
CorruptedDataHeapPage, 29	CatalogEntry, 28
createFile	File, 49
BufferManager, 10	file_name
DiskManager, 38	CatalogEntry, 28
createHeader	FileAlreadyExistCat, 50
HeapFile, 66	FileAlreadyExistCat, 51
createRelation	getFileName, 51
FileManager, 57	FileAlreadyExistDiskMgr, 52
D 1 00	FileAlreadyExistDiskMgr, 53
Data, 30	FileIdAlreadyExistDiskMgr, 53
~Data, 32	FileIdAlreadyExistDiskMgr, 54
Data, 31	getFileId, 55
getCapacity, 32	FileManager, 55
getData, 32	_closeAllFiles, 57
getSize, 33	_removeAllFiles, 57
setSize, 33	\sim FileManager, 57
deallocatePage	createRelation, 57
BufferManager, 10	FileManager, 56
DiskManager, 38	getFile, 58
deleteEntry	getRelation, 59
Catalog, 22	removeFile, 59
deleteRecord	removeRelation, 60
HeapFile, 66	FileNotFoundFileMgr, 61
HeapPage, 75	flushHeader
dirty	HeapFile, 67
BufferState, 18	flushPage
DiskErrorDiskMgr, 34	BufferManager, 11
DiskManager, 35	Frame, 62
\sim DiskManager, 37	BufferManager, 63
allocatePage, 37	loadFrame, 62
createFile, 38	resetFrame, 63
deallocatePage, 38	free
DiskManager, 36	HeapFileHeader, 70
getCapacity, 39	free_size
getSize, 40	HeapFileHeader, 70
isValidPage, 40	free_space_begin
printFile, 41	HeapPageHeader, 81
readPage, 41	free_space_end
removeFile, 42	HeapPageHeader, 81
writePage, 43	full
	HeapFileHeader, 70
EmptyDataHeapPage, 43	full_size
entry_type	HeapFileHeader, 71
CatalogEntry, 27	ricapi nericader, 71
· · · · · · · · · · · · · · · · · · ·	get
File, 45	BufferMap, 15
_setMyFid, 47	getBufferState
buf_mgr, 48	BufferManager, 11
catalog, 49	getCapacity
File, 46	Data, 32
file_id, 49	DiskManager, 39
getHeaderld, 47	getData
getMyFid, 48	Data, 32
getSchema, 48	Page, 111
header_id, 49	getFile
schema, 49	Catalog, 23
Jonetha, To	Galaiog, 20

FileManager, 58	HeapFile, 64
getFileId	\sim HeapFile, 66
Catalog, 24	createHeader, 66
FileIdAlreadyExistDiskMgr, 55	deleteRecord, 66
InsufficientSpaceDiskMgr, 87	flushHeader, 67
InvalidFileIdCat, 92	getRecord, 67
InvalidFileIdDiskMgr, 94	HeapFile, 65
InvalidFileIdHeapFile, 96	insertRecord, 68
getFileIds	updateRecord, 69
Catalog, 24	HeapFileHeader, 70
getFileName	free, 70
Catalog, 24	free_size, 70
FileAlreadyExistCat, 51	full, 70
getFileType	full_size, 71
InvalidFileTypeFileMgr, 98	num_records, 71
getFreeSpace	HeapFileScanner, 71
HeapPage, 76	\sim HeapFileScanner, 72
getHeaderId	getNext, 72
File, 47	HeapFileScanner, 71
getMyFid	HeapPage, 73
File, 48	~HeapPage, 75
getName	deleteRecord, 75
NameAlreadyExistCat, 110	getFreeSpace, 76
getNext	getNext, 76
HeapFileScanner, 72	getPrev, 76
HeapPage, 76	getRecord, 77
HeapPageScanner, 83	HeapPage, 75
getPage	initializeHeader, 78
BufferManager, 11	insertRecord, 78
getPageId	isEmpty, 79
InvalidPageIdBufMgr, 101	isFull, 79
PageNotFoundBufMgr, 114	setNext, 79
PageNotPinnedBufMgr, 116	setPrev, 80
PagePinnedBufMgr, 118	updateRecord, 80
getPageNum	HeapPageHeader, 81
InvalidPageNumDiskMgr, 103	capacity, 81
getPrev	free_space_begin, 81
HeapPage, 76	free_space_end, 81
getRecord	next_page, 82
HeapFile, 67	prev_page, 82
HeapPage, 77	size, 82
getRecordData	HeapPageScanner, 82
Record, 120	getNext, 83
getRelation	HeapPageScanner, 83
FileManager, 59	reset, 83
getSchema	10301, 00
Catalog, 25	initializeHeader
File, 48	HeapPage, 78
Record, 120	insert
getSize	BufferMap, 16
Data, 33	insertRecord
DiskManager, 40	HeapFile, 68
getSlotId	HeapPage, 78
InvalidSlotIdHeapPage, 108	InsufficientSpaceBufMgr, 84
	InsufficientSpaceDiskMgr, 85
getType Catalog, 26	getFileId, 87
Jalaiog, 20	InsufficientSpaceDiskMgr, 87
header_id	InsufficientSpaceHeapFile, 88
File, 49	InsufficientSpaceHeapPage, 89

InvalidFileIdCat, 90 getFileId, 92	PageNotPinnedBufMgr, 114 getPageId, 116 PageNotPinnedBufMgr, 115
InvalidFileIdCat, 91 InvalidFileIdDiskMgr, 92 getFileId, 94	PageNotPinnedBufMgr, 115 PagePinnedBufMgr, 116 getPageId, 118
InvalidFileIdDiskMgr, 93	PagePinnedBufMgr, 117
InvalidFileIdHeapFile, 94	pinned
getFileId, 96	BufferState, 18
InvalidFileIdHeapFile, 95	prev_page
InvalidFileTypeFileMgr, 96	HeapPageHeader, 82
getFileType, 98	printFile
InvalidFileTypeFileMgr, 97 InvalidNameCat, 98	DiskManager, 41
InvalidNameCat, 99	readPage
InvalidPageIdBufMgr, 100	DiskManager, 41
getPageId, 101	Record, 118
InvalidPageIdBufMgr, 101	\sim Record, 119
InvalidPageNumDiskMgr, 102	compare, 120
getPageNum, 103	getRecordData, 120
InvalidPageNumDiskMgr, 103	getSchema, 120
InvalidSchemaHeapFile, 104	Record, 119
InvalidSizeData, 105	setRecordData, 121
InvalidSlotIdHeapPage, 106	setSchema, 121
getSlotId, 108	Recordld, 122
InvalidSlotIdHeapPage, 107	ref_bit
isEmpty	BufferState, 18
HeapPage, 79	releasePage BufferManager, 12
isFull	remove
HeapPage, 79	BufferMap, 16
isValidPage	removeFile
DiskManager, 40	BufferManager, 13
length	DiskManager, 42
SlotInfo, 123	FileManager, 59
loadFrame	removeRelation
Frame, 62	FileManager, 60
	reset
message	HeapPageScanner, 83
SwatDBException, 129	resetFrame
	Frame, 63
name	Schema, 122
CatalogEntry, 28	schema
NameAlreadyExistCat, 108 getName, 110	CatalogEntry, 28
NameAlreadyExistCat, 109	File, 49
next page	setDestroyDB
HeapPageHeader, 82	SwatDB, 126
num records	setDirty
HeapFileHeader, 71	BufferManager, 13
	setNext
offset	HeapPage, 79
SlotInfo, 123	setPrev
	HeapPage, 80
Page, 110	setRecordData
getData, 111	Record, 121
Pageld, 111	setSaveDB
PageNotFoundBufMgr, 112	SwatDB, 126
getPageId, 114	setSchema
PageNotFoundBufMgr, 113	Record, 121

```
setSize
    Data, 33
size
    HeapPageHeader, 82
SlotInfo, 123
    length, 123
    offset, 123
SwatDBException, 126
    \simSwatDBException, 128
    message, 129
    SwatDBException, 128
    what, 129
SwatDB, 124
    \simSwatDB, 125
    setDestroyDB, 126
    setSaveDB, 126
    SwatDB, 125
total
    BufferState, 18
unpinned
    BufferState, 18
updateRecord
    HeapFile, 69
    HeapPage, 80
valid
    BufferState, 18
    CatalogEntry, 28
what
    SwatDBException, 129
writePage
    DiskManager, 43
```