

*Oracle Berkeley DB*

*Getting Started with  
Berkeley DB  
for C++*

*Release 4.7*



# l i g e a t o c e

o m n n b r n T n r p n s r d i l e n a t i s i n i t h e t d m e d h a e y s o u c e y e i e v o u a e i e t e t s o f s t d e i e a t  
r / l m h n . l . / w p w / r t i k / r o l l e s o e w t a l o h s g / l e a g d s t e e u d o t s o c e i e t

r l k r , l D B p l r m k r y o e e e y m k a d e s r l a t d h r e d h s o m e k s e i e t d e i s a f o e a s i g d t e s e s a  
r r r N p r - r . p m r v h e h e s e p e y p o r t r i d n a s e w s d e i t d l o i t e w e s s u o i e i t t o s e o f t o e a

b n p h m n T n r l r y o p t a c o a n o f s t o i d e h s b t i g i d a h r e c o d d e s e a i t e a e s u t t e t y o e a e c o o g  
N k r m r h p m r : / w l m m r e o t a f o p / m a t t D f o u s b c e a c o 1 u f o j s f o s f o a 2 i 7 u =

*Published 4/25/2008*

# Table of Contents

.....	ef.ce.a.		
n n Un h B	v .....o.Ge o.tise.d ist.do	v	i
Fr r m r n l	M .....o. o.e fo. nati.	v	i
r n l1. B r l DB	.....o.t dyo.tio.e.e.e.		
h n l	A .....T. M. o. ts.i. a. u.		2
B r l D p	.....y. e.e.e. o. Ge. s. t.		2
h	A .....M. ccess. e. o. t. d.	4	
h n b n B r n H	A .....e.e.M. tiggess. e. o. t. d.	4	
h n b n n n R	w o. s. T. ieg. ee. e. e. ads. a.		
D m L n Pbr l	w o. s. ieg. ee. e. e. ad. c. o. u. u.		
Enm n	.....at. s. a. sy. it. ad. ta. it.		6
p n n H n l	.....v. o. i. e. s. t.		6
Err r n R r	x .....ce. n. ti. ad. i. g.		7
n n n U DB	.....o. e. t. u.		8
.	.....e. G. tti. g. ad. i. g.		8
n D	.....2. at. s. a.		9
rl D	.....e. O. i. g. at. s. a.		9
D p n Fl	.....os. C. i. g. at. s. a.		0
n r h A	.....at. s. a. e. O. s. ag.		11
Err p R r n F n	.....v. M. d. si. it. et. ie. o. t. d.		11
n D n n Enm n M	.....a. e. o. tigg. c. o. ti. u.		1
D r p l	.....a. agiv. g. at. s. a. i. o. i. e. s. t.		1
R r	.....x. at. s. a. æ.		1
U D R r	.....at. s. a. æ. co. s. d.		2 0
n R n nr D R r	.....s. i. g. at. s. a. æ. co. s. d.		2 0
nr R r h D W	We d. i. g. ad. it. i. g. at. s. a. æ. co. s. d.	1	2
n R r m r h D	.....it. i. g. o. s. d. o. t. et. at. s. a.	1	2
D r l R r	.....e. G. tti. g. o. s. d. o. t. et. at. s. a.		22
D P r n	.....e. e. t. i. g. o. s. d.		2
D U r p l	.....at. s. s. e. i. d.		2
r r.	.....x. at. s. a. s. e. ag. æ.	4	2
h n n r l r r	.....4. s. i. g. s. o. s. u.		
n R r n U h r r	.....e. O. i. g. ad. os. C. i. g. s. o. s. u.		
h m r R r	.....e. G. tti. g. o. s. d. s. i. g. t. s. o.		4
knr h p D l R r W	.....e. f. a. fog. eco. s. d.		
Rn R r n U r r	.....w. o. i. g. it. c. æ. æ. co. s. d.		8
D r l R r n U r r	.....tti. g. o. s. d. s. i. g. s. o. s. u.	4	0
R R r n U r r	.....e. e. t. i. g. o. s. d. s. i. g. s. o. s. u.	4	2
r r r p l	.....e. c. a. i. g. o. s. d. s. i. g. s. o. s. u.	4	
r D.	.....x. s. o. æ. u.	44	
h n n r l n r D	.....y. eco. d. a. at. s. a.	4	9
mln n l K E r r	.....e. O. i. g. ad. os. C. i. g. o. s. d. a. at. s. a.		0
knr h p l l K W	.....y. ex. e. t. i. g. c. to. st.		1
n R n r D	.....w. M. i. g. it. æ. s. u.		2
D r l n r D R r	.....ey. d. i. g. o. s. d. a. at. s. a.		
	.....ey. t. i. g. o. s. d. a. at. s. a. æ. co. s. d.	4	

U	r	rh	n	r	D	ws	i	gs	6	sy	.....	ic	6	u	5	l	a	.....	at	sa	.....	6												
D	n					.....	at	sa	.....	si	.....	5										7												
n	U	n	r	r		.....	sl	.....	io	g	.....	i	6	s	5	.....	u					7												
n	r	D	rip	l		y	.....	ec	6	xd	a	.....	at	sa	.....	5						9												
n	r	D		h	mp	l	b	y	l	ec	6	w	l	a	x	.....	at	sa	.....	ae	d	at	sa	.....	ae	d	0	6						
n	r	D		h	mp	l	b	y	r	ec	6	w	l	a	x	.....	at	sa	.....	ae	d	at	sa	.....	ae	d	4	6						
n	.	m				6	.....	at	sa	.....	ic	.....	ig	o	ati	.....	u										6	8						
n	h	P				.....	z	.....	St	ti	g	.....	t	.....	e	ag	e	Si	.....									6	8					
	rl	P				.....	v	.....	w	.....	e	o	.....	es	ag	.....												6	8					
kn						.....	oc	.....	i	g	.....																	9	6					
	En	l				.....	y	.....	fo	ce	ici	.....																0	7					
	P	n				.....	z	.....	A	.....	v	.....	e	ag	.....	Si	.....	i	g	.....	de	i	.....					0	7					
ln	h	h				.....	ee	.....	ti	g	.....	t	.....	c	.....	Co	.....	e	Si	.....	1							7						
B	r	n	m			T	.....	ee	.....	o	.....	ic	.....	ig	o	ati	.....	u										7						
	lh	p	D	l	R	r	A	w	.....	o	.....	ig	.....	c	.....	e	at	co	.....	su	d	.....						2	7					
		r	p	D	l				.....	o	.....	6	.....	td	.....	c	.....	es	at	.....	u	.....						2	7					
	n	U	r	p	D	l			.....	so	.....	e	.....	td	.....	c	.....	es	at	.....	u	.....						2	7					
	n	n	r	D		pp	rp	D	l	o	.....	ic	.....	ig	.....	i	g	.....	a	.....	at	sa	.....	u	o	.....	t	.....	c	.....	es	at	u	7
n	mp	rr	n	F	n				.....	e	.....	St	ti	g	.....	C	.....	so	.....	i	.....	c	.....	o	.....	ti	.....	u					4	7
	m	mp	rr	n	F	n			.....	e	.....	C	.....	ati	g	.....	C	.....	so	.....	i	.....	c	.....	o	.....	ti	.....	u					7

# Preface

## Conventions Used in this Book

```
typedef struct vendor {
    char name[MAXFIELD];           // Vendor name
    char street[MAXFIELD];         // Street name and number
    char city[MAXFIELD];           // City
    char state[3];                  // Two-digit US state code
    char zipcode[6];               // US zipcode
    char phone_number[13];         // Vendor phone number
} VENDOR;
```

```
typedef struct vendor {
    char name[MAXFIELD];           // Vendor name
    char street[MAXFIELD];         // Street name and number
    char city[MAXFIELD];           // City
    char state[3];                  // Two-digit US state code
    char zipcode[6];               // US zipcode
    char phone_number[13];         // Vendor phone number
    char sales_rep[MAXFIELD];      // Name of sales representative
    char sales_rep_phone[MAXFIELD]; // Sales rep's phone number
} VENDOR;
```

## For More Information

m n l m , l n y l h e o y d st n i m a r a a a l n i d e f o o s i o g c e s o f f o i w u o a t i s e f e u  
 p l n : d i g a c i a t i

r h n r • n P r n r e w G t t i g S e a t d i t s a a t i o c e s s f o g C ++  
 r t t / / h n l m h n b k w w d b - / t t o y d e b a l e D B o r r o p e \_ o t a t e e X u . d d y g g t x X e o e C C f d ]

r l D B n r • h p R l y p p e n e e r e w G t t i g S e a d i e c e a t d c i a t i f o C ++  
 r t t / / h n l m h n b k w w d b - / t t p o y c e p a d t o / o p e \_ o t a t e e u x d d g g e X e c o i a t i C G d ]

r l D B P r m m r • R r y e e e o g a e s ' e f e e c e G d u  
 [ r / l m h n . l . / m w w w n b k t r l b h e a y o e c h o b l / g d e . y a t i e u e d d e f o c t t ]

r l D B P • l y A e e e C ++  
 [ / h : h n . l . / m n w w w b k r t t l b b o p c e y o r e c h o b l o g d e y . a t i e u x d d a i f e a t ]

---

# Chapter 1. Introduction to Berkeley DB

KB r l DB DB DBW n( ) p .lp -re co be o t e b n n h s ieag a ose e e el d d d at a egi t at  
lp rn hl mn mns v i areo/ o di i gen r h of d ata æ ag set cesi steid eigfb et  
h h lpp h pN pp l n m m n m o g r d f b p i g o t m n g M n t a i u a i a c q s t i y g i o c e s s u e o o f u æ a g t  
r- l DB r .ll l m f m s s o n i c i t b a d a t a c b a c e f a s c e f a o w u y a a g i f a y e s t o e t æ s t  
f r m p . r D B m j n b l o f r r d a t a b n l p h t a b l y y i r e y i d v o M s s e y s M y a u a e a s n i e a t c e s g  
DB h r Y Ph h l o h s e b t v u g e n e v o f u v a y a r c i o g i o e t a w i t y u d a d e i d  
n , r b n p , m ( ) h m r d a t a n æ a g d a t a h a u a d f o v o e t o e v d c a d a t e s i c s a  
n r n . a a g i g t a a s i  
DB n m b b n n e c , m s e a l s i e a m p e l d d d . n s e a e g i Y i e i e t e f s a d c o e i a d u i  
p p l n n h m y n i i h o p r a c r i o l a t y u n i g n v a p . a o y d a y t u u a t i a s e s a  
DB r n h m p N r R p n n p t a t l n s n i e t , e a u o c e s s y s h a p a s d e t o p t a c i o a t i u T p h c



## About This Manual

n m n n l r DB h h Tb k, n s min An antres u s s r u stxio oet æ i eite deiat dco  
 r h h r b l r r n r fe et sc n s yaeth eum, i ad eag t a uati sa eagse t dust i a  
 r pb pn r n v DB nypesh s o t l se t d t d o t i o t s. sc a d c e s t ad i as eag  
 b lh m n, n l r DB n n e f o i i a st b a ar d i es ue, o iue s t d at s a d at s a æ co s d  
 r n r r l b r r h b k o v t e a g ad et i o f a r d f i s e æ co s d s d o soa d i t e s c s o s u ad et  
 n , b r n r b s e a g ad e m y e s i s e c o d a d at s a  
 f r m p h m n, l n h B o e m o s h t a h s p i a r f o n t s e s o u t e a A c c e s s e o t v d c æ t i e g  
 h n h m n h l b r m h n a p e r o d s t l i B a r a t e a t h e s i s q e o f v e t o c e T s t o i i g e e s e a g  
 p l r m n m n n m p n r s m a d c e æ c o d . æ a g t a d s a i o e s i u  
 f i p l r n h h h b x h r v n æ s e a l e g i o t P o g t i d i o u t u a æ s A e i g d t s i e a t s e a g  
 n h h p r m p , l m p l n h e o d e n c a æ r t c o h e v e p æ s i e g i t s a t e i d e i g d t f o i c e e t o c e  
 m h h p n r n b v n p n c o n e h d i k t c a t e a r t , p l a d m i r i o e i e s e e t d i s t o i o e s t e f i a æ  
 r l b l n h DB r b r v n e a s o a n h a r e a v i e t s o f Y t e a s d i t i o t i o c f a u i d e t u i

DB\_INSTALL/examples\_cxx/getting\_started

h r h l n h w DB\_INSTALL e e r DB b r n w i t c o a t i e y e o c e a d u s d i t i o u t i u  
 n b k h p r m m n T l r s o p s e s N t h C o u n g a + i g x a e s a p s e i t u v e s o e t t æ t s o s i o f  
 n b k h r n n l x k t d i o e s f o r . e t J C a d v a a a e s a g s æ u

## Berkeley DB Concepts

B r n n n , l b r m h e f o e l c o m p i i n g s i t s e f u o t s t æ s i o u e o f e t e a y o c o s t t a t i  
 n n h r n b n l DB l n w e c o e t e u d i g a æ i a t i  
 n p l l DB b, n n y l o e h t r a p r e d o n t s a c o n t l a y o c g i æ c æ c o d e s e s t s a æ  
 n n r h b I E h r . y n n e p t i e h d m a s e a c k a c n e c w o d u t a i o t e c e s o f f o i o a t i e a  
 h m n l n l l n b r T d v a t a s i r a a o i o c h m o a r d e s t e y e i r e c o r d ' s d a t a o w a e  
 r p k n h r h p r i y e c e s s æ s t e r a o t e o e b . e t o o t o f a d at s a æ c o d  
 B h k p n r / r DB b e y h e a o f e r t e m u d a t a l a i s g f d y d u a s e a e t e s o e æ s i o t g  
 m n b l H r n , w m m s ( æ a æ t c o p v n , æ a h o e u e d a t a y a d e æ s i e s e d e d o g e t c c e s s  
 h n h l b i r r l m p l F r n e b t d y , æ m d x a h i t a c i n e y d a t æ æ t s C c t u e s a d e u t s u c  
 h n m r m h r h e d n s a s i e n s l o e b d i n l i e t e c o v y d s e f f e c e t i t s 2 c o u e a o i u a  
 h m n h r , h w m n p r w i t c h s e e r u o f l o s t e c o s y e a o a d d e t c t e t s f e s i d u  
 N h DB b m r h k l b n l o e t l t æ a l y b d a t s a æ m i n h c e i a t a u i e a o a t i a d at s a æ s e t i  
 DB l n m h r n n b o s n t æ c ( r i d a s i s e l l o e t o a j e d a t s a æ s s t æ s e t u a t i a d at s a æ  
 h r n n b l ) . o e t o a e æ a  
 n l l r l n l m h , r DB b n y æ i e n o a w i a e l l n o e æ r a d at s a æ t a i s a æ g e c o t o f e c o  
 n r n n m z h B r g a i ( c o d h n , i o g t , æ g i r c e s s e o j d . e e e Q e s a a d i f o t  
 r l n l b m h n , n r l m e h o a t i r a d at s a æ s h s o n t n e r t e d i c e s s u e y o t s i e g y a æ d f o o

l r n , l p p l n y r n h e s e a f e d b t a c u r b a t i e a e d e i g s d t a a e i g d a t s a o e s a  
p n ( l n y b s e c f m i j i e r o f d b a t l s t a i e a r o t i a d a t s e a s e t s a e i g t a o s d e s i  
p l B m ) p p o l t a i i g a c f r c i e r o f f e s i d e c s e a o s t a c u b a t i e a e q e i d t a e u g  
k h p p , l n l l n m e p i l s i d b u d a t a v a a c i a t i o f e t s e e i d u a t s e a

o l n , n r n n p p x l n h d k a a e p p s e d m a c o n T i g a c i a t i s i o d y a a c i a t i a a e a g  
n b n k n h k n , n l d a t s e a o d , n l a c c o a s , t e m i c c o a s s t o c t s o u s d o A s a a d f o t  
p p l n l l H m n n c o o r n v i g p p i a t l n k i o a , e a o t a e a f o i o a t i o a e b e a u i g  
n m , r n n n n s i l t i b i r c l s o e l b c c o a s t l a d o u a t o i t i e a o a t i a d a t s e a o a  
n r k n n m r n b l r r e s t e n r f f e i e n t s i d f o b b o a t i o e s o e t ) d u a d a e a g v y g a a e  
l r b l n p p l l n x l l o , n e s e r e s r i f e a n l b a a c i a t i o a s t f o i o a t i o d s e t e u  
n m n n m p v l b d e d o d t a d a e a g d i g e i d u a t s e a

l n n n l m p l l b n a n p a g i c n e f f e i m i n t e l l e i d u a t s e a s i g a o t i u e c s a i e a d  
m r m r m r e n v i r o n m e n t , n m n a p o ( o e f o i v . o a t i s e e o i e s t e a g 6

r h m D B P Y p l l r w o h e i c a p n o s t n s n s i e c s i a u t e s . t c a t u t a i e i s t o f t c o s i  
b l k r l l b T h k e s e n n e a t h o d s n a h e a r d y e c s e a e t o o s o u c e i e a o u t o d a C  
b r H . h T m l c s s a e y n a r a r t a t s e o t c e s u e s e e o t s d o f i e h a r d e e d s a a  
o l , b b l n h n o e l h a e o b e w a d a t s e a o i t a i u a a d o t t a d a t s e a

r r m r b m v m e l t i i g e o f d b a d a t s e a s i t t e n g t h e r e a r d e c s e a e t  
n r r h r r y l e o t d m t a t l s e o n e r , e t h u e t e c o s d c i e a y d S i i a s o t i g l a t s e a  
m m l l b e c o s d s i t t e n g t h e r e a r d y e o u t s e a o s e e t u u a o t d b d  
. s t i

r p r , r , r W b y n e h n s o l e t o r , u t e a o r d t u a d a t s e a s i s g i t a d u e t e c o s i s o e t d  
h r r r n b h w d o a d i o g t m a t s o p l o y t e n d s i s e e t d a t s e a y o S t i g o s t e f o e d  
h k b m , m h n y s e a o d e t e p s b t r e r e i e t d a t a i y e e d o t f o t e a o d u u i g a  
l r n h b y h n h , y e n x t r a e a p l r s i t i l e b o a t s e a e t e e s i t e g o w s l e i c o v a d i t e t e  
H r h . , b p p r p w l v d a t h e e f r e t ( r , d h a t s e a o s l t d c e a t e c o s d w t s a t e c o s d i t e d c t i a  
n r h n h , n y r r e s r f f e i l t u a n t n t a t e c o s i s o e t d a a d c e y a t e c o d a d a  
r r n r r m x e s i t i e g o w d e a o o e e i t t

l p p r p l r r h n , f n a d a t s e a b o s t d c e y a t e c o s d e t o c a e a d a t s e a u a d o e e t e i  
r r r r p l y r r o e f s i e t o d s e a o f t d c e a t e c o s d

n l n b h n l n , l r d o i t i o r i g a y a t s p u a a d l o c w a o a e d a d e i t d a t a i s g e c i a  
m l l r r r n . l l e a n s o r h a i e a r d a k o l y e s s e u t y a e i t o a t t a t c v a e o t u a e u  
n b n . r r e t e c o Y s d r i b a l a t s e a n o c a e c s o s u o t u i t e a t e a d a t s e a o e f s i t  
h l n m r h l h r e c n d t e t s a r a b k e t s a t t e f s i o c a o a s e c u s o s o s e a o u t a  
h n h . b p p r p a c o r d e e e r , t t a t a l a t s e a o s t d c e a t e c o s d s p w y e a e b u a  
h l l r n y p l o c a c e s s u a e t e c o s d s e a o f t d c e s a t u

p D B r , p k n l y b v l l i a o e s i s e c i a o d r e a b n a r y d a t a b a s e a y e c d s d a  
r n n n n m r l b v d l l a s e a e r s b a a e i d n i o h a d a t s e a c e a d i a d a t s e a o t s d i t i s g i  
n r n r n r ) b . m r e t f r o b s e c o y e s D B e c d r n d a d a t s e a e a e i e t t i e g s e a e c o s d u a  
o l p b k n n x n o o p e m r d n a t s e a n e h i o g t u a g i e c o s l y i f o e o d s e a o d t a t  
n l b b l k y y e c o w d e f r o m p e a t d e u e a o s e e o t e a o d s e a o d s o e e c e i o f

m r n h n h k h n , n b H f o i h o h t i y t s a d i y n e t e r e t o e e a s t i o t g e t y e o f s e c o d u a  
b d a s a

## Access Methods

h H m n l l l p m r W r h B r o v i n s t h i a a y f o c l u T i a o i u e t e e c c e s s e o t d s i i t s i e f o  
r l b r l h m h y n D k e f i e s t b l e l o f e t c c e s s . e o t s d v t a t e s a a a e a

N h n m h n b l n h o e h t t a b a c c e s s e r o t c h y e e c e t d e e t d a s e a c i e e a t d c o  
l , l P n l r l l n l a e c e l l d c m a t h a s y h e a g u i d y u a e d c t i o a s a c c e s s e v o t s d s a t i  
m p n m h n , l l n x r s o h e d e l o n r i c h s y n t e c c a i v a o e i t y a t i e t i w g i e s e a a  
r r l h h m h h l w e g a d s o f c i c c e s s v e o t d e e e c e t d

n m h h h H T e l c e s s y e o t n t a t s o c d o o s e y w i e g a f d s w i t a t a t t e s u a  
n h n , n b l h p m y n h e y a d e r e c o d n h e t e f o c e a t a t s e e f o a g u c c e s s e o t d

n l h r b l l i n h w e f o o v i g a e t a a e a c c e s s e o t s d

h

B r

D m r b e e l , r r r r h B a o e t d i o a e t d a a d e c t s c t e t o t e  
k n h r B r r r n l y b r r e l m p a t e t d f a e e e c o s d y a x a i t a c i o e  
h n , n n n l T h n y n s a t r i e t c e a t s a i g e a s c s a a e i d u g  
n r m p , l p h r r r l x l y g o e e s s c a s a a c t e t s o u a t u g  
h b l h r p b l r r r e t e d a e t o a i s w i d s s e i f o o t e g o s d b t e e  
h m p r l h n h . h r , t c a t r W e a s e a q s a e s t o c s e t e c o s d e a  
n r b p l n n h r c o s e e . d e d c e s a f o e u o a e t

H

D m n n n s n a l h h b l k L B a o e t . d , i e a e t e d d e i a s T a e a e i e e  
k n h r H r y n b b e r r a d e t d a s a f d s a c c o s d y a o f a i t a  
m p l l k l B , r . p l , x A c o r e d l a t a s o e i e e d c e a c o y s d e a o t i  
p p r . s o e t d u

D m e Q e n l h u r u r H E a o e t d i e s f a e i d u u g c o s d c a c o  
l l r n r m b r k h m e s h . o a c g i e n o d y e s a i t u s c i c c e s s e o t  
n r n r h l h n e d h e i g f d f s a s i s t a t b a i e t e e a d i t u  
p l m h l n r n r r s r e n r h o i a e o a t i t a t e e s t a d t e o f d u e  
h h . e o d e t e e u u

h m h n n l h T p r r s c i c e s l e o t s d i s v a i t u a t u u e r i e c o d e  
k l n h n p r b . n p l m m T m p o c m y n i g s c i a o e d e e f c i a f o c e a i o e e  
n p p l n r n r n n r r h i a c i a i e q i d o g c e u c t e s s o u t e t e e

n R

D m h r e c o r b r l n l h r r a k a b e t d e i e i f e i o d a a e g e c o s d e  
n R , r r l l r n r m b r k e Q e c o e c o u s d e o c g i a c o u y d e s s a s u

## Selecting Access Methods

l n m h h , T l r n o s r e c t a c c e s s e o t k o s o r f d s i o t s w e d a t y a t t e s u a e a f o  
b r r n l y . b r r r o d n a s e r o u l d f o , h ) y a t t e u a i t a d a e y s t s i g e t o s o  
h r B r r H n l . l T s e r e n i b y e e w n s a f r o ( a t t e o u c g i a o u d y e s e s s e u t i a e i d  
n h l r n R y e t o s o d e e e o u e u o u u

h m h n m y, h v b c o n h e a e r s t e d i o i i o r , w c b o s e a T e e e i t e e o s a o  
r n R h n . b r n e o e c o u s i d s o i s e s t e i e t

## Choosing between BTree and Hash

l l k n r h n r n l m w m o s h a n , i g s a t b y t t f a t e i t y e t i e o e t s d w f f e i c e e e e  
H H B p l l m . r T l l h e w a d s a o j l b . e w h , s s t a s a b e t y s t i i t o a t i o u i g t  
l l B r n r , h j r r n w n h s m T s t a r s e u p p f f o r u o o e t e s o j y t a e t o a o f a c i a t i s e  
T e e

h m n n n h r r r k n r o e t t a t , n r a j c e w e e s d i o y . i g s a t a d o t e M y t i d s a t a t a  
n m n n l r m n n m r n b a n i n d i a i t a i e a g o a s o l l f o i y o a t i o t e e d u c c e s s o e s a  
h h n r n h o o w o f r y t a d a t t a y i t f a v e y c w o S u t a t o b s e d s e t d a t a t a t  
l l m l n h , m y w l h l l g m n d i b e t i s e p p u e t n u o t d f a a e t y g a t a a e a g d o a c i a t i u

r , r k n r r w v h p y n o h e r s a n n o w i g l e t a d g v o t y t o i t e e o c a f t i t i a o i t  
h n p k m r h y n h y r e o e t o n b e o p t e t a e d l e y a e , c o o s i o g c c e s s e y o u d e s f c i i a  
c o o s e

r k r h m . l l T r y n r j e e v f o i h e s y e a n l l e o c a o t e f e e c e y s a t f i e t y s o e t a d  
p h r r n k x k l l b t e a b c b t t v a t a y f o m r a g i u e w i e y e f o y e d a f o o e o f u  
h b r s i e i g s

r m . l l r f y r n s . f i n i h d y e t a b t i m e y n v e a g a e g i c c e s s e o t d s t a i t a i u  
r m n n n r n l m r n H l r e a m . t a i o a n o f n e i t h u o v o a t i o e e e t o a o f t f o i o a t i t a t  
r m n n r B r m h r h m r h s h t a i r t a i l u e e s i c r e g e a t T t u f a s a e e s s t i t a t a u  
r h n n r n l m r n n m w d s e t a i n g s h s t p i e r i t h f o i r o a t i a d e a t e t c c e v o t e t o i t e e  
r r l l l p l l p p l n v y e s e i r e t i B r e i b c e a f r o t A a c i a t i d a t a e a t e e c a u f o c e d  
m r k m h m l / r r n H n o e f b l n s c h i m O c m y o r e f e w e u t t v a u d s a e g i u e s e a o a t  
o f d a t a

r r , r b m M l v h y D e o n e l f o i r d s e t a e t o u e s s o w e a g t a t y i o s c e t t a i e a o t  
m r k l / n r m r h n e l f o y l n i o t p f a t i m a r a d e e v t e t s y a e f d e i t o e t f o u  
b h n r n r l T r h b e e l v h e a i s f a B u e i t e a o s d o s e c a o t T g t a e s u e e

## Choosing between Queue and Recno

r n R r h n h p p l n n e Q e o e d o e s e n r n b e w t u r a c i a t i s a d t e o c g i e a o u d e s f o u  
r r b k L l r n r m b y r e t r n y i a l l a s e a d r o n c g i e a d d e y e e s s e u t i a e i d y s g t a t e j  
h b r r h n b y h . m r e d f t i e r d T a y a e o h e l c a e e x t w e a o f e i d e e a t a  
m b m h m h h n b e c o r d r e r s o i e r t a t i g F e a g a d a t a s a c o s d e a o e t x o d e e t d e i d  
l r n r m b n r h r n r r l b c g i e o v o e s n e e c r u e a v g g a d s o f a d a t a e o a i e a  
e f o e d

n b n n W n R h , e w e d d i e g e e e Q e a d c o u c o o s e  
p p l n . r h r h y r n e Q e f i r r o i a p u i a t i e u . q e s i i g d e e y o f c o c e v c e Q e u o e s d u u  
r l - k l n p p h ( p v l - k l e o h d r e o b i r s o a n o s b d t e t e a g e o c i g t a t o e t c c e s s e o t s d  
n h , n ) r n l n n n l h r b e a d i n c i u l a s y n n i f i g i i u a f t s e a t o t y g f o t i g o a e t u  
l n a c i a t i

## Database Limits and Portability

## Environments

nr r,h m lnnnwv b oenevnoiedoptit eaoel ltoae ,daseo eyid is a eig sc fae i  
pp ln n n nnn most ac iativu se ea oie t  
b r n m pl r pp Mr tiet d ad ticeess o t u  
n n nnn r W ,ryh h nem on-serhea no uel nso cess c y aut ieuo c c ea ad sc a  
b llh b p n n h y em neald noa nentnd.ase do ve e d iet ovie tee wo ie bas  
n b lb m h r yn lno y peth ear sine p r t at and eig d toa di et sdu ad ocesses  
DB b F mp l . p n on nces nxl atess o e nrrav o se ea o iue o et ea eto c e t  
r Dh klj b m n hr hd stotem Cb setop r sigs set au ey eade o ffe oo u  
n pl r n . T s a a ti access ig  
b r n r n bl m h ll r offe D r and a tis as we t A t a as fo f y Coe c tu t if o d ase a u  
r n n n n n n n bw Y nr nes it b se n o h i u s det ea et y a a tis as se t ad et  
n lb n nr n D y s se. q botu tai t a tis  
b l l pr ln . pp r v) y ig a a i a i te c iatis o t u  
b r pr ln b m h nb ln m offe s r b c pati ls se lt tea eas eig seat d ase a c iati it  
l l r n -lp h pr l y n ien n n co es b if d te c e at d d ata see o ieu s det ea ad  
n n h b m . et y a ag s is se t u  
b m . y o ggi gs se t u  
b r rh - ln pp ln lw noffe sb ne thea d -gog ac iati t at a tot tai a ig dees of  
br h h n pp ln r y n ecoh n a n i b i f l d e af lya ac iati s se t c s a coe ea de to ggi g  
m llh pp ln p y m r k n w s se t o as u ent r al iatio e fo uty sidf eco e b "a ad  
phr h h h ) h n m r n a" s a t a t h i o t l ge t se of u t foi. u oatio eai d ie to f es i

nr m r n n h p h , o oe foi oati BerkeleyDB Getting Started with Transaction  
n h Processing BerkeleyDB Getting Started with Replicated Applications ed

# Exception Handling

n n n , lp n m m nefore co p tri hings it sh wo DB dfeax o e sote ce o tiw ad igi it  
P l . A et C ++  
ml n DBh h r y n h em a bs DbException Hot r v . , i e e of t se a n s i e o o e eu  
hr n n h wr m DbException a t at rb k d h o : ex ce p t i o n y try so o oc ss ou d  
n p p n F mp l y c c a t o : t e s o f e e o s i o e a e

```
#include <db_cxx.h>
...
try
{
    // DB and other code goes here
}
catch(DbException &e)
{
    // DB error handling goes here
}
```

```

}
catch(std::exception &e)
{
    // All other error handling goes here
}

```

n b n h DB r m b Y r r b c n o a t a i u o b b e x c e p t i o n e f o y a b b e x c e p t i o n : s e t \_ i g n o ( ) u  
 n l b n h n m r Y m l o c b o l o t a i r t m o i n a t i a s s v e a s s o a e i a t d i t t e a t o e s  
 DbException::what()

m l r n n n m n f f o y s o l e e s o v a o o d b e x c e p t i o n a g j y t r o e c s t o o c s  
 n n r DB p p r h m y b n o c c a f e g u o s y e n b c a t \_ N O \_ E X C E P T I O N S y f o o d a t  
 n n n m n h n l n h n l v m , m n a d o i e r DB t r a d y n s t e r d y s t a a g u e o c o u d i s  
 n n r l r n r b l l n B h v B e t h e i n g n e a e t m d b a v e o t s d e a a t a t t i a s a e s  
 n m n r r m y n w n o y a t t a a g e o d b e x c e p t i o n i s i g u o e c s t

## Error Returns

n l p n h DB r l x m d b i t i o n c e n o t i e t w h y e i f t c e s a l v a s a e t a e a o f 0 o s c c e s s f  
 m n n r n h n e r o d i o n e l l o n g t c e e r f d a e s u a . e r t z e a e u o e u o

m l r r r r r m p l y DB r , f s k a p t e o p a m j e n f b e a e o a o f t s d s i c e a e s s o i i o t c e  
 l n n r , l l m n w p f a i n s a d e i d a e w i g a D B g , j s a e c f u e i d o t e o f e t e i f t c e s a  
 r n m l l h p b . l e r r n b e v t a a r l u n a o f e t o s s e i . e a o f e u a g e a t t o a

h p m n l m r b f e b r e o a t i d o y f h t a i r D B s a o t e u o t s a s t c e s s f e e i t u  
 r n r p l r r r l F m p l . , e r t v i s e c e i a m r h e a y o b e a e u f o i v e t i d e e t e r i d a t a e t d a t  
 n h r r r h r h r h m n a d e t e d B f o l r n c r i o e s e c x a i o g s w o e t s D B \_ N O T \_ F O U N D e t u  
 p l r r r l h m n h r k v s e n e i p o n e a t t s a u t e l l e s e t e . o e s u o t e A a i e t d a t a  
 h p b p l r r r l r l h n o f e t o s s e v e c e i a o e a e a s s t o a

## Getting and Using DB

n b n b DB n h Y k B r l DB n o l p v o a t a i u s y i t i g t w e e o d o d e a g  
 p r / l m h n . l . / w w w / r t t k / r o l l e s y o n e n t m / o l s g f t e a y . d s t e e u d d e i d t

n l l DB r n r p h b T r n h o s z i t a h r d a i a t h n i t u o y o t e t e a i o y t o f o w c o c e i o u i c  
 b l p r b n r f r m r n e b d r l d B o d u t e a i o u f o i o a t o d i g s e e u  
 h DB \_ I N S T A L L / d o c s / i n d e x . h t m l h w D B \_ I N S T A L L e e y s w e t y c i o t e e o c e a  
 DB h p . , r l n k l p l m p - y b O v n l a e a g n f i i d s i o t . f o a t s e c f c i i d s i c t o s i u u

n p l n n n k l m T m n n a r e D B s o a r h a i r s i o , t h e o d e o a t f o u y w a t i a o f i  
 h k l h r B e r k e l e y s D B P r o g r a m m e r ' s R e f e r e n c e G u i d e w A s a s a t e f e e c e  
 m n n . o d e a t i u

---

## Chapter 2. Databases

### Opening Databases

```
#include <db_cxx.h>

...

Db db(NULL, 0); // Instantiate the Db object

u_int32_t oFlags = DB_CREATE; // Open flags;

try {
    // Open the database
    db.open(NULL, // Transaction pointer
            "my_db.db", // Database file name
            NULL, // Optional logical database name
            DB_BTREE, // Database access method
            oFlags, // Open flags
            0); // File mode (using defaults)
    // DbException is not subclassed from std::exception, so
    // need to catch both of these.
} catch(DbException &e) {
    // Error handling code goes here
} catch(std::exception &e) {
    // Error handling code goes here
}
```



## Closing Databases

```
#include <db_cxx.h>

...

Db db(NULL, 0);

// Database open and access operations happen here.

try {
    // Close the database
    db.close(0);
    // DbException is not subclassed from std::exception, so
    // need to catch both of these.
} catch(DbException &e) {
    // Error handling code goes here
} catch(std::exception &e) {
    // Error handling code goes here
}
```

## Database Open Flags



```
u_int32_t open_flags = DB_CREATE | DB_EXCL;
```

- DB\_CREATE

- DB\_EXCL

- DB\_RDONLY

- DB\_TRUNCATE

## Administrative Methods

- Db::get\_open\_flags()

```
#include <db_cxx.h>
...
Db db(NULL, 0);
u_int32_t open_flags;

// Database open and subsequent operations omitted for clarity

db.get_open_flags(&open_flags);
```

- Db::remove()

```
#include <db_cxx.h>
...
Db db(NULL, 0);

// Database handle creation omitted for clarity

db.remove("mydb.db",           // Database file to remove
          NULL,                // Database to remove. This is
                               // NULL so the entire file is
                               // removed.
          0);                  // Flags. None used.
```

- Db::rename()

```
#include <db_cxx.h>
...
Db db(NULL, 0);

// Database handle creation omitted for clarity

db.rename("mydb.db",           // Database file to rename
          NULL,                // Database to rename. This is
                               // NULL so the entire file is
                               // renamed.
          "newdb.db",          // New database file name
          0);                  // Flags. None used.
```

## Error Reporting Functions

- `set_error_stream()`

b p r l o s t r e m e r s t e t C b h o e y e s s e s s e d e t u i a

- `set_errcall()`

h n n h l l h n n m r e s i e f o t w d t s a t i e r a d e . e a o e s s y e a g e d e u o  
n m p r h l k p e f h i l a p s e a g e a s p e a d t s t c i a a s t i o t e t a y u b a t i o t s d i a  
m r n r r l . s t f o i y o a t i o e c t

- `set_errfile()`

h b l r r b p r l e y s t e t C i a b h o e y e s s e s s e d e t u  
r r . y i a

- `set_errpfx()`

h p r n r m r e s t e t e y l r e f d a u . e s s e s s e d e t u i a

- `err()`

n l m r h m r . n s h e s h a l l e s s u n e a g e e o e s s e a g e i o t t e t a c a c o t i s a f l u i d  
h m h l h . n s y b e r n c a l l h n h , m r f t a t n o t d s a e e s e d e t e u o e s s e a g e i o t t  
l n b r s e t \_ e r r f i l e i d s e t \_ e r r o r \_ s t r e a m ( ) f o e o f e s t e e o t s d  
n h n h , m r v n n e a e e r s e r d e t e u o . e s s e a g e i o t t t a d a d o

m r n h p r t n r n e b o ( e s s e a g s s p i t n e t e l s e t \_ e r r p e d ( e i d o a o t i a  
- l m r m p r i n t f m r y n s , e r f o h e a t l e s s e a g e e o . e s s e a g w a d a t a i e g e i

- `errx()`

n l l p h h n d b e y e s s e d o t i a o t h h e c e t t a x e s s e a g w t s t o a e i a d i t e t  
l r r r l n p p n h r m r e a d o u e a s . a t a a d d t e e o s t i g

n l n , h n y n a d o r i t i o n e r m r u f y o t i o t e c i u e t e e o s u t i g  
p r n p r l r m m b r t c a t e s a s d o t a a t i e a o e u u

b l n , l l m r x r n o b h e n o s e l d e a k e s s r e s a g a g i d a t e a a d o t c a a c f o  
r , r r b l k D m h n k . l h a d y i f g i t e e a b : c a a o s u e t i g i s t i

```
/*
 * Function called to handle any database error messages
 * issued by DB.
 */
void
my_error_handler(const char *error_prefix, char *msg)
{
```

```

/*
 * Put your code to handle the error prefix and error
 * message here. Note that one or both of these parameters
 * may be NULL depending on how the error message is issued
 * and how the DB handle is configured.
 */
}

```

Example 1: Database Error Handling

```

#include <db_cxx.h>
...

Db db(NULL, 0);
std::string dbName("my_db.db");

try
{
    // Set up error handling for this database
    db.set_errcall(my_error_handler);
    db.set_errpfx("my_example_program");
}

```

Example 2: Database Error Handling

```

// Open the database
db.open(NULL, dbName.c_str(), NULL, DB_BTREE, DB_CREATE, 0);
}

// Must catch both DbException and std::exception
catch(DbException &e)
{
    db.err(e.get_errno(), "Database open failed %s",
           dbName.c_str());
    throw e;
}
catch(std::exception &e)
{
    // No DB error number available, so use errx
    db.errx("Error opening database: %s", e.what());
    throw e;
}

```

## Managing Databases in Environments

```
#include <db_cxx.h>
...
u_int32_t env_flags = DB_CREATE |      // If the environment does not
                                     // exist, create it.
                        DB_INIT_MPOOL; // Initialize the in-memory cache.

std::string envHome("/export1/testEnv");
DbEnv myEnv(0);

try {
    myEnv.open(envHome.c_str(), env_flags, 0);
} catch(DbException &e) {
    std::cerr << "Error opening database environment: "
              << envHome << std::endl;
    std::cerr << e.what() << std::endl;
    exit( -1 );
} catch(std::exception &e) {
    std::cerr << "Error opening database environment: "
              << envHome << std::endl;
    std::cerr << e.what() << std::endl;
    exit( -1 );
}
```

```
#include <db_cxx.h>
...
u_int32_t env_flags = DB_CREATE |      // If the environment does not
                                     // exist, create it.
                        DB_INIT_MPOOL; // Initialize the in-memory cache.

std::string envHome("/export1/testEnv");
```

```

u_int32_t db_flags = DB_CREATE; // If the database does not
                                // exist, create it.

std::string dbName("mydb.db");
DbEnv myEnv(0);
Db *myDb;

try {
    myEnv.open(envHome.c_str(), env_flags, 0);
    myDb = new Db(&myEnv, 0);
    myDb->open(NULL,
               dbName.c_str(),
               NULL,
               DB_BTREE,
               db_flags,
               0);
} catch(DbException &e) {
    std::cerr << "Error opening database environment: "
               << envHome
               << " and database "
               << dbName << std::endl;
    std::cerr << e.what() << std::endl;
    exit( -1 );
} catch(std::exception &e) {
    std::cerr << "Error opening database environment: "
               << envHome
               << " and database "
               << dbName << std::endl;
    std::cerr << e.what() << std::endl;
    exit( -1 );
}

```

```

try {
    if (myDb != NULL) {
        myDb->close(0);
    }
    myEnv.close(0);

} catch(DbException &e) {
    std::cerr << "Error closing database environment: "
               << envHome
               << " or database "
               << dbName << std::endl;
    std::cerr << e.what() << std::endl;
    exit( -1 );
} catch(std::exception &e) {
    std::cerr << "Error closing database environment: "

```

```

        << envHome
        << " or database "
        << dbName << std::endl;
    std::cerr << e.what() << std::endl;
    exit( -1 );
}

```

## Database Example

DB\_INSTALL/examples\_cxx/getting\_started

```

// File: MyDb.hpp
#include <db_cxx.h>

class MyDb
{
public:
    // Constructor requires a path to the database,
    // and a database name.
    MyDb(std::string &path, std::string &dbName);

    // Our destructor just calls our private close method.
    ~MyDb() { close(); }

    inline Db &getDb() {return db_;}

private:
    Db db_;
    std::string dbName_;
    uint32_t cFlags_;
}

```



```

// Make sure the default constructor is private
// We don't want it used.
MyDb() : db_(NULL, 0) {}

// We put our database close activity here.
// This is called from our destructor. In
// a more complicated example, we might want
// to make this method public, but a private
// method is more appropriate for this example.
void close();
};

```

N n h mp mln n hr wn re r d ee det : e e d atfo e to s cto t u

```

// File: MyDb.cpp
#include "MyDb.hpp"

// Class constructor. Requires a path to the location
// where the database is located, and a database name
MyDb::MyDb(std::string &path, std::string &dbName)
    : db_(NULL, 0), // Instantiate Db object
      dbFileName_(path + dbName), // Database file name
      cFlags_(DB_CREATE) // If the database doesn't yet exist,
                          // allow it to be created.
{
    try
    {
        // Redirect debugging information to std::cerr
        db_.set_error_stream(&std::cerr);

        // Open the database
        db_.open(NULL, dbFileName_.c_str(), NULL, DB_BTREE, cFlags_, 0);
    }
    // DbException is not a subclass of std::exception, so we
    // need to catch them both.
    catch(DbException &e)
    {
        std::cerr << "Error opening database: " << dbFileName_ << "\n";
        std::cerr << e.what() << std::endl;
    }
    catch(std::exception &e)
    {
        std::cerr << "Error opening database: " << dbFileName_ << "\n";
        std::cerr << e.what() << std::endl;
    }
}

```

n h n n h mp mln A n hr w mdent e ee det e c d atfo et e ot d

```
// Private member used to close a database. Called from the class
// destructor.
void
MyDb::close()
{
    // Close the db
    try
    {
        db_.close(0);
        std::cout << "Database " << dbFileName_
                    << " is closed." << std::endl;
    }
    catch(DbException &e)
    {
        std::cerr << "Error closing database: " << dbFileName_ << "\n";
        std::cerr << e.what() << std::endl;
    }
    catch(std::exception &e)
    {
        std::cerr << "Error closing database: " << dbFileName_ << "\n";
        std::cerr << e.what() << std::endl;
    }
}
```

---

## Chapter 3. Database Records

### Using Database Records

```
#include <db_cxx.h>
#include <string.h>

...

float money = 122.45;
char *description = "Grocery bill.";

Dbt key(&money, sizeof(float));
Dbt data(description, strlen(description)+1);
```

```
#include <db_cxx.h>
#include <string.h>

...

Dbt key, data;
float money;
char *description;

key.set_data(&money);
key.set_ulen(sizeof(float));
```

```
key.set_flags(DB_DBT_USERMEM);

// Database retrieval code goes here

// Money is set into the memory that we supplied.
description = (char *)data.get_data();
```

## Reading and Writing Database Records

rn n n b W r b , d w e h d i r g a d i t h g a t s e v e c o s d e a e a t a t e s o e s i g t  
n b h p n n n h h r f i b e c e s p p i r p a v e l d y d i g e e t o d a s e a u s t d c i a c o s d  
r r b r r r n T w r b p o l o e n d a s e h e c o i n d e a o s e d e d e d c e s a f o e y u o a e t f e t  
m k h l l n . r h n s h y e a t e k e a e r e t o e c o t o f e c o s d y a i g t d a p l i c a t e s a e a d a  
DB n l k , r n e t . n m v l p y l a g i e y s s o e t o d . o c e f o s a a i g d c e s a t e t u  
  
l DB b , n p p y r p l r e f i r h a t r p l a s a o d r o s r t o r t d W e u a c o s d e e d c e a c o s d e a  
r r r b l ( r p l ) s o e t d l l o s e e n o y u e a t i a s e d t c e s u o a f e t e c o s d i e t  
.  
d c e s a t e t u  
  
b m h n m h r v r m o r e s i o t s c a i t c s a s i f o e t o v t e a g a d e t i y f a d a s e a e d a t a  
:  
s a i  
  
n m h p f : : p r t ( h D i e : g e t ( ) m d l p v - e o t s d o e d e e s e a i c t e s f o a d c e a t  
n h b h m h . r b e c o n b d T e t n d a s e a e s e e . o t s d e a e s t e i d i s t e c o t i  
  
r p r r m l h . p r n n v n v o s b o e d u e r a o t r f b . t t i g a d g u t t i g l a s e a c o s d s o s a d  
r b m h r b r n n U e t i d a s e a c c e s s e ( o t s d e ) a e s t e i d i i g s o s e a g u 33

## Writing Records to the Database

r m h b n h eor sdr eao etrd l ietw d tse as z ig e at ou ga iati ei qe id etcess  
 n h l n m l y h e v tBdr(t atr re se p c e m d so ec sEas c s aee aro sdeao et d i a  
 rhr m n n n s p n p t y d w t n t n pa atted e f(see e)S ttio g C sa i c d ti e ag fo  
 m r n ) . o e foi oati  
  
 H m h,n p n n y n b a se a et e t h a n o f n ttig a d guti g l a t a a c o s d o d o c t a o g c e  
 l r m h y n v , r o y r e n e e c e n t u n c e s s n e q t, d o f e g d s o y i o g e s i f i a u a d  
 r b n f r r p p r o e e m p d , l y d a t s p a o u o c o e s ' e s e u e t i s a i e d a t a t a e s t i u  
 m n m h r m y h e a r n e s w e a o e a t t . y a t e s e o t d e a s i g u u  
  
 p r Y, b D b ; : p u t r ) o n s e h w r o u t r o t p e i t a t a s a c o d s y o t e d v q e s i o o t o u e d u  
 k n n h m r p r y e t e c o s d e n a d d a r i e n f o f j y o i o e c s t o c a o a o e d e  
 r l h n r l D B h r h r d o e f r s a g t c a t o t s ' e o a f i w e t d a t a e i t  
  
 l b l h m h , v f n C e l f r s a g a r i a n o o v s h i t a t d y . a y e i e f t i n g o s f i a g  
 r n r p r d n ( n w y r w s d h a s o b e h i t i n g o a l i g e a s i t i e g o d i e t d a t a e a f y e t o e d e  
 n h b h n h m y x r n e a d s s i t i e t d h a s a b e t d s t k e y b o i s e t v u e e f e t d a t a  
 r p l . s o s t d c e s a t u  
  
 o l : x o e a e

```
#include <db_cxx.h>
#include <string.h>

...

char *description = "Grocery bill.";
float money = 122.45;

Db my_database(NULL, 0);
// Database open omitted for clarity

Dbt key(&money, sizeof(float));
Dbt data(description, strlen(description) + 1);

int ret = my_database.put(NULL, &key, &data, DB_NOOVERWRITE);
if (ret == DB_KEYEXIST) {
    my_database.err(ret, "Put failed because key %f already exists", money);
}
```

## Getting Records from the Database

```
Dbt get(r ar enhu u v rbt d et etei d ase eco ydoe t tfat d at
pp rp l r rhnb , hl m sh o s nll cl nro s d et r ed va sty e ot d oi e t ef s iet u
p l fr rn . ,rb pp al or p sel d h y str n s n a f oi d ase a w s t d c e at etou o
ln r r r r r r m r s o o t s oi se d as o n o u eteueco s flo it s s e a est ue id i i
r pr ( ) . s s e ag u 33

n l(r r pY l r rn o b k a oae et i seaoft. d c e atco s ds T i ga g g o od ut o se
ln h ll DB_MULTIPLE fr nr m r n Db : bag e DB P m m r o oe foi o at see et o g ae
Rr ) . efe e ce G d u

B l , rn th yr r Db r get(h ed k amt h h ak p e f s v iet o f d ose e yu c v s ete o
n h llh m h r bl .pp or pet l a yst e otr f on d ase a w s t d c e atco s d o c ca
a b h rh l bl pp nl h v sty y o asly i g e _GET_BOTH i g t u T f DB ag set (i) c as es a
rn hr r r hr m h h p r k ent ef s iet u d t at c as yet o ed e add ata

h p lk n r n / n hf ye b ec f e i n h , ad x d a b s o e t s itiet d ase a st e ot d t
. DB_NOTFOUND
```

```
#include <db_cxx.h>
#include <string.h>

...

float money;
char description[DESCRIPTION_SIZE + 1];

Db my_database(NULL, 0);
// Database open omitted for clarity
```

```

money = 122.45;

Dbt key, data;

key.set_data(&money);
key.set_size(sizeof(float));

data.set_data(description);
data.set_ulen(DESCRIPTION_SIZE + 1);
data.set_flags(DB_DBT_USERMEM);

my_database.get(NULL, &key, &data, 0);

// Description is set into the memory that we supplied.

```

## Deleting Records

```

#include <db_cxx.h>

...

Db my_database(NULL, 0);
// Database open omitted for clarity

float money = 122.45;
Dbt key(&money, sizeof(float));

my_database.del(NULL, &key, 0);

```

## Data Persistence

ll l nl h h n l .pw b l ry cise ce naU sterier .lysitoss eifo o o dsex data ede ete  
m n p b l r p rrc isb des p rjoaoss xeifo o o et e e ceud atse eo oti  
n r r , r r Tbr l yr e efo ey floir a e a pi rdnatayid eac oass sue tf ais ad t g ad a  
n rpr b l b prr n h ,eytl e aoss iniof p d atse eo rotio sou de ta qatio to ect ot  
b m n E m mm nd atsanavo yDBiash eh etio co itatur a ti es es t att  
n llb l pp ln wr m lr n o etos eloyt acrimati s Ste tf ai sa a ti s e ag ext eu id  
Berkeley DB Getting Started with Transaction Processing ed u

n ln nr n ln h , wrf onodo t atise tna a siuret etssya otis it at d atafi au  
n n h n m pp ln xt atie no e tsnyite tetro, Y ac bati w sato a a tufifo  
mp l rn DB h x rly ne lhae onrepp l gnotma y data at o tet e tac bati  
h ,lr nr n nr n wv ny f orre a o e aolh stmiugt a a uifo soye e so w ad s ti soteu  
nr h r b m n pr y r g a e m t on d p tse a o u f l i b a i e a y s s e i t t e t o s o y e o d d  
ll n n . Dm: syn d h cn y nam - r hy n s y ch sep and et u e i y i e t i e o c c e a a d e b e  
m lh b m ly h h s s e , t f w c p n a o a elitto t d i s s c x e t e a q e y i t e s e i u a d s o  
h m p n r l . sey ets a i g

m DB hr b ln p m r n mye n e n r r a t e f d al b a c y e i f u e d a e i o a t a a ti a d atse  
l nl l n (rrh b h b y p o s e r d e v a o n h o a e v e l y u s t e y d b a i s s e c f i i g o e t a  
h ) n m r c l o s e ( l n r n t o n y l l s a t a l y c a y a u D b s : s e n u ( c u a i g

pp ll n r m hr ny n r f o n y n a r b a t o u s n e y t c , h s e s l a d e a o s t i u g y a a t i e t o s o  
h r m r r r b r e h e n t s d i y n d a d c e e d v d y a s e a o u e f v i e t y o c a f i a d a t s e a s  
b D r : r b ( ) . n v r y n l y l e f , i t i m o m u a t s e a o d y l e t f \_ c i e a s e e t d u c o b a d p t a e  
m h h b p b l U h h s a c o f r m n d a t s e a s p a m i s s e i s e r e i t e t o c o a d e o o t i o t  
n h r l r u m h b l h w c o l y t o r e g e i d w d o b e y e s u a a g i o g d a t s e a u

## Database Usage Example

Database Usage Example ( r 1 ) h p n a s a w e e l a g e 7 c e r e a t c a s a t o a e s a d o s e s a d a t s e f o  
n m k . h l W h n w s e n o e u s e o f h t a s s u y d l l d e i o v t d a t a i t o t w d a t s e a t a t  
r n r n r m . v s e f o y e i u o t s s e t

n m r m b , h r n n A h m p l m p m b a y e n e h r t a t n c f i a n i d e t o u e e t i e e a t t i o e s e f c o t i  
DB\_INSTALL/examples\_cxx/getting\_started

h r h l n h w DB\_INSTALL e e r D B b r w i t y c o a t i e y e o c e a d u s d i t i o u t i u

### 3.1 Database Usage Example

n r r l n n W n w r m e h a t o e t p d a t e a n d y n a e i T o t s s e w t y e e e a o t e s o f f o i o  
n n m n n n r w w r l n a t m v a t h y n a g n e i o t d a t a a d . e a t e o d c o c t i f o i o a t i o  
n h n m r n , h l r r a e g s w f i d i p o a t i e c o b d e a e e a d s a c y e t f o e c a u o f d a t a  
ll r n m h r o t s i x n e a t o r t w i n g e i n d d a t a r o i v s t a c t e t e f u f a u c e a t o g f o  
n r . v y e i o t d a t a

n h h n n h W E N D w R v e r o N h o h e t e d W E N D i t o f R e t r s o c t e t V o e t u t a t s o c t e  
n l h l h n n . x m n s e s f e i d l p r u g e s i d s y i d e t e s s a a d f i c a t w d e s e t a e a f

```
// File: gettingStartedCommon.hpp
#define MAXFIELD 20
typedef struct vendor {
    char name[MAXFIELD];           // Vendor name
    char street[MAXFIELD];         // Street name and number
    char city[MAXFIELD];           // City
    char state[3];                 // Two-digit US state code
    char zipcode[6];               // US zipcode
    char phone_number[13];         // Vendor phone number
    char sales_rep[MAXFIELD];      // Name of sales representative
    char sales_rep_phone[MAXFIELD]; // Sales rep's phone number
} VENDOR;
```

```
class InventoryData
{
public:
    inline void setPrice(double price) {price_ = price;}
    inline void setQuantity(long quantity) {quantity_ = quantity;}
    inline void setCategory(std::string &category) {category_ = category;}
    inline void setName(std::string &name) {name_ = name;}
    inline void setVendor(std::string &vendor) {vendor_ = vendor;}
    inline void setSKU(std::string &sku) {sku_ = sku;}

    inline double& getPrice() {return(price_);}
    inline long& getQuantity() {return(quantity_);}
    inline std::string& getCategory() {return(category_);}
}
```



```

inline std::string& getName() {return(name_);}
inline std::string& getVendor() {return(vendor_);}
inline std::string& getSKU() {return(sku_);}

// Initialize our data members
void clear()
{
    price_ = 0.0;
    quantity_ = 0;
    category_.clear();
    name_.clear();
    vendor_.clear();
    sku_.clear();
}

```

```

// Default constructor
InventoryData() { clear(); }

// Constructor from a void *
// For use with the data returned from a bdb get
InventoryData(void *buffer)
{
    char *buf = (char *)buffer;

    price_ = *((double *)buf);
    bufLen_ = sizeof(double);

    quantity_ = *((long *)buf + bufLen_);
    bufLen_ += sizeof(long);

    name_ = buf + bufLen_;
    bufLen_ += name_.size() + 1;

    sku_ = buf + bufLen_;
    bufLen_ += sku_.size() + 1;

    category_ = buf + bufLen_;
    bufLen_ += category_.size() + 1;

    vendor_ = buf + bufLen_;
    bufLen_ += vendor_.size() + 1;
}

```

```
// Marshalls this classes data members into a single
// contiguous memory location for the purpose of storing
// the data in a database.
char *
getBuffer()
{
    // Zero out the buffer
    memset(databuf_, 0, 500);
    // Now pack the data into a single contiguous memory location for
    // storage.
    bufLen_ = 0;
    int dataLen = 0;

    dataLen = sizeof(double);
    memcpy(databuf_, &price_, dataLen);
    bufLen_ += dataLen;

    dataLen = sizeof(long);
    memcpy(databuf_ + bufLen_, &quantity_, dataLen);
    bufLen_ += dataLen;

    packString(databuf_, name_);
    packString(databuf_, sku_);
    packString(databuf_, category_);
    packString(databuf_, vendor_);

    return (databuf_);
}

// Returns the size of the buffer. Used for storing
// the buffer in a database.
inline int getBufferSize() { return (bufLen_); }
```

```
// Utility function used to show the contents of this class
void
show() {
    std::cout << "\nName:      " << name_ << std::endl;
    std::cout << "    SKU:      " << sku_ << std::endl;
    std::cout << "    Price:    " << price_ << std::endl;
    std::cout << "    Quantity: " << quantity_ << std::endl;
    std::cout << "    Category: " << category_ << std::endl;
    std::cout << "    Vendor:   " << vendor_ << std::endl;
}
```

```
private:

    // Utility function that appends a char * to the end of
    // the buffer.
    void
    packString(char *buffer, std::string &theString)
    {
        int string_size = theString.size() + 1;
        memcpy(buffer+bufLen_, theString.c_str(), string_size);
        bufLen_ += string_size;
    }

    // Data members
    std::string category_, name_, vendor_, sku_;
    double price_;
    long quantity_;
    int bufLen_;
    char databuf_[500];
};
```

### 3.3. IE x l x a e \_e a p d a a dsepoa bd

DB\_INSTALL/examples\_cxx/getting\_started

```
// File: example_database_load.cpp
#include <iostream>
#include <fstream>
#include <cstdlib>

#include "MyDb.hpp"
#include "gettingStartedCommon.hpp"

// Forward declarations
void loadVendorDB(MyDb&, std::string&);
void loadInventoryDB(MyDb&, std::string&);
```

```
// Loads the contents of vendors.txt and inventory.txt into
// Berkeley DB databases.
int
main(int argc, char *argv[])
{
    // Initialize the path to the database files
    std::string basename("./");
    std::string databaseHome("./");

    // Database names
    std::string vDbName("vendordb.db");
    std::string iDbName("inventorydb.db");

    // Parse the command line arguments here and determine
    // the location of the flat text files containing the
    // inventory data here. This step is omitted for clarity.

    // Identify the full name for our input files, which should
    // also include some path information.
    std::string inventoryFile = basename + "inventory.txt";
    std::string vendorFile = basename + "vendors.txt";

    try
    {
        // Open all databases.
        MyDb inventoryDB(databaseHome, iDbName);
        MyDb vendorDB(databaseHome, vDbName);

        // Load the vendor database
        loadVendorDB(vendorDB, vendorFile);

        // Load the inventory database
        loadInventoryDB(inventoryDB, inventoryFile);
    } catch(DbException &e) {
        std::cerr << "Error loading databases. " << std::endl;
        std::cerr << e.what() << std::endl;
        return(e.get_errno());
    } catch(std::exception &e) {
        std::cerr << "Error loading databases. " << std::endl;
        std::cerr << e.what() << std::endl;
        return(-1);
    }

    return(0);
} // End main
```

n p l l l r b w h o e r t t y o d o e h c b t o s e o d f a t s a e n s s i e c s e a e t d a t s a  
l n l b n h , b e y e c s m a h e a d j i k c u s s h e c s j t. a d o s e o e c s t e v a e t y c t a e e t o g  
p h r, r r l l h b o o f s c o e e w e d c t o s t c i s e a e t d a t s a u o s e o o t c u

```
// Loads the contents of the vendors.txt file into a database
void
loadVendorDB(MyDb &vendorDB, std::string &vendorFile)
{
    std::ifstream inFile(vendorFile.c_str(), std::ios::in);
    if ( !inFile )
    {
        std::cerr << "Could not open file '" << vendorFile
                    << "'. Giving up." << std::endl;
        throw std::exception();
    }

    VENDOR my_vendor;
    while (!inFile.eof())
    {
        std::string stringBuffer;
        std::getline(inFile, stringBuffer);
        memset(&my_vendor, 0, sizeof(VENDOR));

        // Scan the line into the structure.
        // Convenient, but not particularly safe.
        // In a real program, there would be a lot more
        // defensive code here.
        sscanf(stringBuf.c_str(),
               "%20[^#]#%20[^#]#%20[^#]#%3[^#]#%6[^#]#%13[^#]#%20[^#]#%20[^\n]",
               my_vendor.name, my_vendor.street,
               my_vendor.city, my_vendor.state,
               my_vendor.zipcode, my_vendor.phone_number,
               my_vendor.sales_rep, my_vendor.sales_rep_phone);

        Dbt key(my_vendor.name, strlen(my_vendor.name) + 1);
        Dbt data(&my_vendor, sizeof(VENDOR));

        vendorDB.getDb().put(NULL, &key, &data, 0);
    }
    inFile.close();
}
```

n , rh y w n riva e ledhroetoydeir T f c o t v o y u d e t e i o t  
n r n h n l h n n w f o i b o a t i e , e d e l i m r a e h i n x e t , e i o t f t e o t a i c æ f c b i t e t  
n n n w n n e o i n d e t t o y d a t a i t a s i d a  
l b n h r l m r h n l r p e v s o l t a i t m p s i f l p f b e u a e d f i e s o a e a t a i e e e  
h l h p n h r f l c m l t i t o a t u e s a t e t r s o m i o f ( ) e f s i f t a i e d e i t # o a d s i g o u a  
. e o f i t u  
h l m p l w r o n e n t v t a r c o y n d e a r i n e d r e d v d s t o y e t o e i o t d u a t a i s a c t e t e u  
l h E N D R r h b V c H e b e t m , r h s o c t e t m t a t v s e u o y e o e u e s o t i g t i d a t a i e t  
l h h , n I n v e n t o r y D a t a h h w m s n a z i n e d f e t i e s e d f e t d a t a t a t o i t t a i e c æ  
l l m n p p b l h r h e t e a r t o a r l o f t c e a o b u e f o w e t d a t a t a t e a o t i g e e s s t i t a t  
n b m l h m m h h r b o n c c æ b e s u e l l t w p i t i g t e t s e i e v a d d a t a u i d æ s s c e a  
n r r h n l l o w s d l i t f a e s e w d x a c t e t u f i t e u d u f i e s i d  
r l h h v h r o m p a t h i a d s e t a t r s a a t x e f o e s t e e u æ s e s e s o c e s a s i g e a  
l B r . n h n r m n d w g v i r l f i n e h e s c u t n i g æ s d f o u s i o f e c o s d v e t e t o s s t a s i g  
m n n y . æ c o e s f i g i i a t

```
// Used to locate the first pound sign (a field delimiter)
// in the input string.
int
getNextPound(std::string &theString, std::string &substring)
{
    int pos = theString.find("#");
    substring.assign(theString, 0, pos);
    theString.assign(theString, pos + 1, theString.size());
    return (pos);
}

// Loads the contents of the inventory.txt file into a database
void
loadInventoryDB(MyDb &inventoryDB, std::string &inventoryFile)
{
    InventoryData inventoryData;
    std::string substring;
    int nextPound;

    std::ifstream inFile(inventoryFile.c_str(), std::ios::in);
    if (!inFile)
    {
        std::cerr << "Could not open file '" << inventoryFile
                    << "'. Giving up." << std::endl;
        throw std::exception();
    }

    while (!inFile.eof())
    {
        inventoryData.clear();
```

```
std::string stringBuffer;
std::getline(inFile, stringBuffer);

// Now parse the line
if (!stringBuf.empty())
{
    nextPound = getNextPound(stringBuf, substring);
    inventoryData.setName(substring);

    nextPound = getNextPound(stringBuf, substring);
    inventoryData.setSKU(substring);

    nextPound = getNextPound(stringBuf, substring);
    inventoryData.setPrice(strtod(substring.c_str(), 0));

    nextPound = getNextPound(stringBuf, substring);
    inventoryData.setQuantity(strtol(substring.c_str(), 0, 10));

    nextPound = getNextPound(stringBuf, substring);
    inventoryData.setCategory(substring);

    nextPound = getNextPound(stringBuf, substring);
    inventoryData.setVendor(substring);

    void *buff = (void *)inventoryData.getSKU().c_str();
    int size = inventoryData.getSKU().size()+1;
    Dbt key(buff, size);

    buff = inventoryData.getBuffer();
    size = inventoryData.getBufferSize();
    Dbt data(buff, size);

    inventoryDB.getDb().put(NULL, &key, &data, 0);
}
}
inFile.close();
}
```

h n hlp rp r n mp Hx h hew vtr l n e no em ea w æ w ts a s o v o e y det ei o t  
n r b .v e od d ata

---

## Chapter 4. Using Cursors

### Opening and Closing Cursors

```
#include <db_cxx.h>

...

Dbc *cursorp;
Db my_database(NULL, 0);

// Database open omitted for clarity

// Get a cursor
my_database.cursor(NULL, &cursorp, 0);

Dbc::close() r b h l e o t r o y l l t c a t s i o g d a t a a e c i s o s e a d i e e d
p h h D B l p w , l l h i t r i e s c o r e o f e t h y b a d e s e c i a , i s e c s o s e a u t i g t e t d a t a
n p r b l r l l v l . c r a b e a e a d w a e s r s y t s a o s e o c s o s y e f o e u c o n i o g d a t a a u

#include <db_cxx.h>

...

Dbc *cursorp;
Db my_database(NULL, 0);

// Database and cursor open omitted for clarity

if (cursorp != NULL)
    cursorp->close();

my_database.close(0);
```



## Getting Records Using the Cursor

```
#include <db_cxx.h>

...

Db my_database(NULL, 0);
Dbc *cursorp;

try {
    // Database open omitted for clarity

    // Get a cursor
    my_database.cursor(NULL, &cursorp, 0);

    Dbt key, data;
    int ret;

    // Iterate over the database, retrieving each record in turn.
    while ((ret = cursorp->get(&key, &data, DB_NEXT)) == 0) {
        // Do interesting things with the Dbts here.
    }
    if (ret != DB_NOTFOUND) {
        // ret should be DB_NOTFOUND upon exiting the loop.
        // Dbc::get() will by default throw an exception if any
        // significant errors occur, so by default this if block
        // can never be reached.
    }
} catch(DbException &e) {
    my_database.err(e.get_errno(), "Error!");
} catch(std::exception &e) {
    my_database.errx("Error! %s", e.what());
}

// Cursors must be closed
if (cursorp != NULL)
    cursorp->close();

my_database.close(0);
```

hr b mrh T l r v r d e i t e e t d n a s f o e t s a c o DB\_PREV s i s e s a u t o d  
: DB\_NEXT

```
#include <db_cxx.h>

...

Db my_database(NULL, 0);
Dbc *cursorp;

try {
    // Database open omitted for clarity

    // Get a cursor
    my_database.cursor(NULL, &cursorp, 0);

    Dbt key, data;
    int ret;
    // Iterate over the database, retrieving each record in turn.
    while ((ret = cursorp->get(&key, &data, DB_PREV)) == 0) {
        // Do interesting things with the Dbts here.
    }
    if (ret != DB_NOTFOUND) {
        // ret should be DB_NOTFOUND upon exiting the loop.
        // Dbc::get() will by default throw an exception if any
        // significant errors occur, so by default this if block
        // can never be reached.
    }
} catch(DbException &e) {
    my_database.err(e.get_errno(), "Error!");
} catch(std::exception &e) {
    my_database.errx("Error! %s", e.what());
}

// Cursors must be closed
if (cursorp != NULL)
    cursorp->close();

my_database.close(0);
```

## Searching for Records

r r hr Yrb r ro c n a e c h s o s t u c a f k Y d a s a c o s d o c s a j c a s g a d s tea  
h b n b h h k y n h o o c n a c p a s e n a p o r t e t e a d e t d a t a c a o a e f o u a t i a  
r b p p r r p y l c a s f o l l i t a s a n o s s d e t d c e s t u y c a s e s a e t e a d d a t a  
r h m h r l l h h k n a e a s t o f e s e h e w t d e f a y i d i t e t e a d d a t a e a o f e t d a t a  
h h h r p n w r d c b d t h r i e t s o s d s o t u d a e a o f t e t e c a  
, h r h n , A r l n h s o f r e t e c a f s a i e t c r o s s e t a t e f o u n d a g d a d s e i t e d

np r r h r r r Tr b D, o s enc as:(phem aia eco qe y e gtue o sey st è outdo  
r h lh l v :c aow ed efo o figs ag

N n h lh h r r l b keti ie fo oh n is gitt at tr so fy vs ag ewe o ud e et so  
mn h k p r h x r rn jhe yis s ,eter o lu tof re teco sd ist se a et so ssei ot tateco  
h k m h h p r w h y rose ev a) as het ea o led d Met sow o eo eu e et so ses e  
k p r h n h , r p ny w GET h heko dh et et sp s rds o iayd b tetev etv data ea o ed d  
h r r . et so u

R r l h k r r rh yads of hrete ,ork d nse owe g teau d itc aso y et so s'e add  
r ll h h rDbt lmrh rwsr eh e lid viter p d at anetei fcb weteto .d t c iet so s ds oed d

- DB\_SET

h M o es et

Alabama/Athens  
Alabama/Florence  
Alaska/Anchorage  
Alaska/Fairbanks  
Arizona/Avondale  
Arizona/Florence

: v et o d i g

...	y a sea f	e o	a a sea	cf va a d	des	e c tso o
F A	s a l a l	l n k	/ a	A	s a a a i s a	
Fl Az	o i n a	F l r	/	Az	o i a e ce	
n A	s a l a n h	r /		A A	s a a o e ag	

```
#include <db_cxx.h>
#include <string.h>

...

Db my_database(NULL, 0);
Dbc *cursorp;

try {
    // database open omitted for clarity

    // Get a cursor
    my_database.cursor(NULL, &cursorp, 0);

    // Search criteria
    char *search_key = "Alaska";
    char *search_data = "Fa";

    // Set up our DBTs
    Dbt key(search_key, strlen(search_key) + 1);
    Dbt data(search_data, strlen(search_data) + 1);

    // Position the cursor to the first record in the database whose
    // key matches the search key and whose data begins with the search
    // data.
    int ret = cursorp->get(&key, &data, DB_GET_BOTH_RANGE);
    if (!ret) {
        // Do something with the data
    }
}
```

```

} catch(DbException &e) {
    my_database.err(e.get_errno(), "Error!");
} catch(std::exception &e) {
    my_database.errx("Error! %s", e.what());
}

// Close the cursor
if (cursorp != NULL)
    cursorp->close();

// Close the database
my_database.close(0);

```

## Working with Duplicate Records

, • DB\_NEXT DB\_PREV

• DB\_GET\_BOTH\_RANGE

, • DB\_NEXT\_NODUP DB\_PREV\_NODUP

```

Alabama/Athens
Alabama/Florence
Alaska/Anchorage
Alaska/Fairbanks

```

Arizona/Avondale  
Arizona/Florence

- DB\_NEXT\_DUP

```
#include <db_cxx.h>
#include <string.h>

...

char *search_key = "Al";

Db my_database(NULL, 0);
Dbc *cursorp;

try {
    // database open omitted for clarity

    // Get a cursor
    my_database.cursor(NULL, &cursorp, 0);

    // Set up our DBTs
    Dbt key(search_key, strlen(search_key) + 1);
    Dbt data;

    // Position the cursor to the first record in the database whose
    // key and data begin with the correct strings.
    int ret = cursorp->get(&key, &data, DB_SET);
    while (ret != DB_NOTFOUND) {
        std::cout << "key: " << (char *)key.get_data()
                    << "data: " << (char *)data.get_data() << std::endl;
        ret = cursorp->get(&key, &data, DB_NEXT_DUP);
    }
} catch(DbException &e) {
    my_database.err(e.get_errno(), "Error!");
}
```

```

    } catch(std::exception &e) {
        my_database.errx("Error! %s", e.what());
    }

    // Close the cursor
    if (cursorp != NULL)
        cursorp->close();

    // Close the database
    my_database.close(0);

```

## Putting Records Using Cursors

h b m h h . n n n h o c b h e c s h n p t u n e t o s d o i t e t d a t a s ' e o a i e t t i e g o  
h b m h h . n n n h o c b h e c s h n p t u n e t o s d o i t e t d a t a s ' e o a i e t t i e g o  
p l . d c e s a t u

Nh h n p n r r h b w n o e t t r a r , t t i e g o s n o u t e t d a t a s i g a o u e t s o u s d s o i t e  
h r r n r . y a t e c o d s e i e t d u

l h h m h p Y r d c h p t ( ) d s e w u n t . t t i e g o s n o u t e t d a t a a o c w a e e f o u u  
: w f s a g i t s t e o t d

- DB\_NODUPDATA

h p r l k l r n h b v I f n e t o n y e d e n e a r e s s i t i e t d a t a e t s t e o t d t  
. DB\_KEYEXIST

h k l n h n h , r h r y h f r e x p o e s b e t s b i t e t e b e d t a t e c o d i t o i t e t d a t a s  
m r b h n r r n r b h y l e t e i d p e t h e i n y o n e d s e e t d a t a a f c o a s a i f c o  
h b n p r h b h r , r n s a n o e d t e n d a t a e t e c o . d s e i e t d s i t e t o c o a t i  
h r m n B ( r l p h r ) w l n r e t s h i s a r i n g e e t c o i g a i o a t i g e d s i t o e u t e i c o  
b m l m . e f o e o e g e i s

h l n n b l h r B r n H y s f m i a g o a n e n t e , r h e t e e a d s a c c e s s e y o t s d a d f e t  
b h b n n r p p r r p d a t a s m e c o f e g d s t o s o t e t d u b \_ d e p a r t a t i v  
p b r n m p e c f e i d a t a e o a t i e i

h l n n b h h T r n R s f m i a g a e t s e d i t e t u e o e c o a c c e s s e o t s d

m r r m r n n p l r r h o p e d f o i o a t i p d a t a c o ( s d e e ) o . i g c e a t e c o s u l e 2 g

- DB\_KEYFIRST

F r b h n p p r p l h m o h , d a t a s a t a d h s t m o t d c e s a t y t i x o t y e e s a c a t e s  
n l r p m r h b s i f a p e w a p e i o t i s u a f o e d , e t d a t a s o s t d c e a t e c o s  
n p l n n h b n p h n a d , a r c e s m f t o t i n s a e u s e c f e i d e t s e i e t d d a t a i t s i e d  
r l n h k l l r . n h s i t e t o c r y o a p i f y e t e r e a e s s i t i e t d a t a a d d c e s o

nh b n p h n , r mf c o t i s h e e u s e c f e i i d e t s e i e t d d a t a i t s i d d æ f s o f t e t d a t a  
h r k . e y i f o t a t

- DB\_KEYLAST

l v x p y e DB\_KEYLAST f i n x s e d c e y t u f a t e e a d s s i t i  
b n n p l n n h b n p e t d a t a n a d , d c e n f t o t i s e e u s e c f e i i d e t s e i e t d d a t a i t  
h l h m h r k s i d d æ t s o f t e t d a t a i f o t a t

o l : x o e æ

```
#include <db_cxx.h>
#include <string.h>

...

char *key1str = "My first string";
char *data1str = "My first data";
char *key2str = "A second string";
char *data2str = "My second data";
char *data3str = "My third data";

Db my_database(NULL, 0);
Dbc *cursorp;

try {
    // Set up our DBTs
    Dbt key1(key1str, strlen(key1str) + 1);
    Dbt data1(data1str, strlen(data1str) + 1);

    Dbt key2(key2str, strlen(key2str) + 1);
    Dbt data2(data2str, strlen(data2str) + 1);
    Dbt data3(data3str, strlen(data3str) + 1);

    // Database open omitted

    // Get the cursor
    my_database.cursor(NULL, &cursorp, 0);

    // Assuming an empty database, this first put places
    // "My first string"/"My first data" in the first
    // position in the database
    int ret = cursorp->put(&key1, &data1, DB_KEYFIRST);

    // This put places "A second string"/"My second data" in the
    // the database according to its key sorts against the key
    // used for the currently existing database record. Most likely
    // this record would appear first in the database.
    ret = cursorp->put(&key2, &data2,
```



```

        DB_KEYFIRST); /* Added according to sort order */

    // If duplicates are not allowed, the currently existing record that
    // uses "key2" is overwritten with the data provided on this put.
    // That is, the record "A second string"/"My second data" becomes
    // "A second string"/"My third data"
    //
    // If duplicates are allowed, then "My third data" is placed in the
    // duplicates list according to how it sorts against "My second data".
    ret = cursorp->put(&key2, &data3,
        DB_KEYFIRST); // If duplicates are not allowed, record
                       // is overwritten with new data. Otherwise,
                       // the record is added to the beginning of
                       // the duplicates list.
} catch(DbException &e) {
    my_database.err(e.get_errno(), "Error!");
} catch(std::exception &e) {
    my_database.errx("Error! %s", e.what());
}

// Cursors must be closed
if (cursorp != NULL)
    cursorp->close();

my_database.close(0);

```

## Deleting Records Using Cursors

l r rn r mpT pl, n h o e e t h o r d h y g a o s u i o s u o i t e t s o o y e t e w o u d t a t  
l n h n ll o. e b e : t e l d e t c a  
F r m p l : x o e æ

```

#include <db_cxx.h>
#include <string.h>

...

char *key1str = "My first string";
Db my_database(NULL, 0);
Dbc *cursorp;

try {
    // Database open omitted

    // Get the cursor
    my_database.cursor(NULL, &cursorp, 0);

    // Set up our DBTs

```

```

    Dbt key(keylstr, strlen(keylstr) + 1);
    Dbt data;

    // Iterate over the database, deleting each record in turn.
    int ret;
    while ((ret = cursorp->get(&key, &data,
                              DB_SET)) == 0) {
        cursorp->del(0);
    }

    } catch(DbException &e) {
        my_database.err(e.get_errno(), "Error!");
    } catch(std::exception &e) {
        my_database.errx("Error! %s", e.what());
    }

    // Cursors must be closed
    if (cursorp != NULL)
        cursorp->close();

    my_database.close(0);

```

## Replacing Records Using Cursors

```

#include <db_cxx.h>
#include <string.h>

...

Db my_database(NULL, 0);
Dbc *cursorp;

int ret;
char *keylstr = "My first string";
char *replacement_data = "replace me";

try {
    // Database open omitted

    // Get the cursor
    my_database.cursor(NULL, &cursorp, 0);

    // Set up our DBTs
    Dbt key(keylstr, strlen(keylstr) + 1);
    Dbt data;

    // Position the cursor */

```

Nh nn hn r k nyh m d t th o k p m t y a u g e o l s d e s i g t e o t d e t e w y a e a t i a  
rh n pr l r r w y . o i g d e o e c e a e o u d

n pr d h p r W r r , e r e n c a i n g t h r d a m o y l e a r d o i e a c a i u g e o d t s a t i a  
r p l h n h , p r m n o l l s o a e t d d r l e s a n t e r c e a e t e y s c c e s s f v o f u e t e u e c o  
r n l l h l r h m n h s o s t e d o p t r i a b t e b r e d o d s y e s a t f a t o i e a c a i u g e o d t  
m b r r p l n , r m h e o f s o a d t d d p h y s a t e r t u a d o i e a s x i g t a f l u a e t c o i g a u s o i  
n h p r m n l l l n l h e t r e w e r t e a e r v t f i , a i d o t o i a t i u g t o v o t e y d o e v e f o i o e  
m r m h r m p l b r , m s o t s o b t x u e t i t f a t e u a e s o j s t s e a v a d s f l e a e s t u o f t e t d  
m h n p , n l l n p m r r p r e l i n y n o e t n l l i a o c l h a f o u a c i e t c e a e v t a d t o d i e a t e t  
r r n b r h r . e s c t o i s i e s t e i e e

U h r m n n p r l h e d e s e n y i l s w d e p f b i u a t e c e a e t y d a b a b a i d a d c e a  
r m , n r n m r n y e h o n d , a d h e a o n t i u g a o t s o t u e t i e t e d e u t e t e c o d a d e c  
r r h h k n w w e a e c o . d y i t e t e s e i e a d d a t a

h n n r n n n r m r w n v h m p l v i e o d , l l a d r i p p t f b i n x o a t i w s t i w a e e i e i t a a c i a t i  
o l l h m n h n n n r y p d i r a o r e t e i t y n i n t n i o t r d a s e a w a y o f t o v i g a e g i e i o  
m , l k l p h n h r n p w r w h e i n e a v h o b h w n e t e w r d u c a o e d e t e i t a d o e t e o d s t o  
m r n . f o i o a t i

l h , p p y l e x a m p l e \_ d a t a b a s e \_ l o a d : a c i a t i v e s e f o o i g

p n h h 1 n n m n r b h v e c r y e t r e l e i o i t a e o d w d a t a s a t y a t e c e e a t d o  
n m r n n h h p p l e x a m p l e \_ d a t a b a s e \_ l o a d l p x a c i a t i v e s a e d a t a a o d 2 a g f o 8  
b n . r m r h n n r b 2 O t a i c a s o f o y e t u e i o t d a t a a

h h h n n r b p rl, he Str3ot ry get ei out yd atse a sd i a e g æco d a oiesg  
h n m . h n r hr4 n n r me s6ut n eaf e t e r o d foy t ate i o . t e ifoy et ei o teco d  
h n . n m kl p h n vrr ne b e t e odr beavo do ete o d eeo d iete od d atse a  
h n . rr r y . v 6s isaete odeco d  
hr n n h mp lmp m lny en e h t p t c l fan ide tou e e t i e e o at of sti æ bati i  
DB\_INSTALL/examples\_cxx/getting\_started

h l n h w DB\_INSTALL e e r DB b r w i t o c oati eye o ce a d u sd it iouti u

rh . . lErx 1x a e 4 e a p æ a a dsep ea bd

n , l h n hrT r w n p e m gie r i y d e t e c s m u æ e d f e s i a d v f o o f o a e d u æ a t i  
r r n n W w usages o a e d t f c u t i u

```
// File: example_database_read.cpp
#include <iostream>
#include <fstream>
#include <cstdlib>

#include "MyDb.hpp"
#include "gettingStartedCommon.hpp"

// Forward declarations
int show_all_records(MyDb &inventoryDB, MyDb &vendorDB);
int show_vendor(MyDb &vendorDB, const char *vendor);
```

r r n n X h w w main h d nne d t r l r p c d l t i o e w u t a t i o i e y a t e c e s s a c b u c æ a t d  
b ll n n n h n h y p e v e p s e a n n e e u e n t x d i g i t i e t e c t e a t o e y f o e i o t e i t  
. oo s u

```
// Displays all inventory items and the associated vendor record.
int
main (int argc, char *argv[])
{
    // Initialize the path to the database files
    std::string databaseHome("./");

    // Database names
    std::string vDbName("vendordb.db");
    std::string iDbName("inventorydb.db");

    // Parse the command line arguments
    // Omitted for brevity

    try
    {
```

```

// Open all databases.
MyDb inventoryDB(databaseHome, iDbName);
MyDb vendorDB(databaseHome, vDbName);

show_all_records(inventoryDB, vendorDB);
} catch(DbException &e) {
    std::cerr << "Error reading databases. " << std::endl;
    std::cerr << e.what() << std::endl;
    return(e.get_errno());
} catch(std::exception &e) {
    std::cerr << "Error reading databases. " << std::endl;
    std::cerr << e.what() << std::endl;
    return(-1);
}

return(0);
} // End main

```

N n rh x w n ew l t s e n d l t e p o r t s l l h T f c o t i s f i a o t i s d i s a o f  
n r r r n n h n n v r b n e i o l t e c o h s o n y d i e t r e i m t d a s a c o v s b y s e t e i o t e c o  
r r h n n r m m r h v n v e i t e t e l e t e n o d s ' p e a h o t e t o d a d e s o i t d o y u a d d i a e  
o p p r r n r r r : v o e i a e o d e c o d

```

// Shows all the records in the inventory database.
// For each inventory record shown, the appropriate
// vendor record is also displayed.
int
show_all_records(MyDb &inventoryDB, MyDb &vendorDB)
{
    // Get a cursor to the inventory db
    Dbc *cursorp;
    try {
        inventoryDB.getDb().cursor(NULL, &cursorp, 0);

        // Iterate over the inventory database, from the first record
        // to the last, displaying each in turn
        Dbt key, data;
        int ret;
        while ((ret = cursorp->get(&key, &data, DB_NEXT)) == 0 )
        {
            InventoryData inventoryItem(data.get_data());
            inventoryItem.show();

            show_vendor(vendorDB, inventoryItem.getVendor().c_str());
        }
    } catch(DbException &e) {
        inventoryDB.getDb().err(e.get_errno(), "Error in show_all_records");
        cursorp->close();
        throw e;
    }
}

```

```

    } catch(std::exception &e) {
        cursorp->close();
        throw e;
    }

    cursorp->close();
    return (0);
}

```

h 5 ( ) . lh Inven... data... n n w B ss a t at se ees vest ue id ie o t ata  
 p l h n n r r v n , y apiv l h ya det ewo tveo de o y vottsd i æte odeco d  
 n h r n h l .n co es o d iogt stic opwd h st ic se ae odo ee d te cyaso o tsd i æt  
 rr r nU h r. mp l e o deco l nrs ig aso neeco ucl esay co ed igfu o oog d gaise t d  
 pl m r r lwn hy n esr(bi efo y a dciv t agai ete od d ata

```

// Shows a vendor record. Each vendor record is an instance of
// a vendor structure. See loadVendorDB() in
// example_database_load for how this structure was originally
// put into the database.
int
show_vendor(MyDb &vendorDB, const char *vendor)
{
    Dbt data;
    VENDOR my_vendor;

    try {
        // Set the search key to the vendor's name
        // vendor is explicitly cast to char * to stop a compiler
        // complaint.
        Dbt key((char *)vendor, strlen(vendor) + 1);

        // Make sure we use the memory we set aside for the VENDOR
        // structure rather than the memory that DB allocates.
        // Some systems may require structures to be aligned in memory
        // in a specific way, and DB may not get it right.

        data.set_data(&my_vendor);
        data.set_ulen(sizeof(VENDOR));
        data.set_flags(DB_DBT_USERMEM);

        // Get the record
        vendorDB.getDb().get(NULL, &key, &data, 0);
        std::cout << "          " << my_vendor.street << "\n"
                  << "          " << my_vendor.city << ", "
                  << my_vendor.state << "\n"
                  << "          " << my_vendor.zipcode << "\n"
                  << "          " << my_vendor.phone_number << "\n"

```

```

        << "          Contact: " << my_vendor.sales_rep << "\n"
        << "          " << my_vendor.sales_rep_phone
        << std::endl;

    } catch(DbException &e) {
        vendorDB.getDb().err(e.get_errno(), "Error in show_vendor");
        throw e;
    } catch(std::exception &e) {
        throw e;
    }
    return (0);
}

```

n mp l h mp mln n T cat e ent le to example in platform read() x ete c t ea  
 ll n h pp l n m kw w x ne e i b t d s h i a i o n t e r a s y of seco d a d a t a s o y t a t c a e  
 n n r b p n nv m e t e i o t d a t a f o r s y c f c i i e i o t e i t

---

## Chapter 5. Secondary Databases

l n b r b m n y y h s r k o f l i d y a b e c o h s d e s a f y e t e o v s d e g e e e t e t a t  
r r r r n l l y n n l y m n s e r f o w o r e o u r y p d o u t a s c o t a i e t f o i o a t i e g e i d t o e d o  
r h h w n r r i t F a r i n e s d y p p e w d a t a t r a t v a t k e e t e i o e y a e s o s e o u  
n n r r r l i h k m d h a e o n i a e c o s i d e T e a t g s e s e e u i g e t s a t i g t s a t o i e e i  
r h r p n r h , r D i E r e d f e t i f i h e t e s o r s c k , i a l s a u c a e o v s d v d a t a e y o d e i u  
m p l b n n n b l p c o x l i a t o a m e o e c c o r t , p h t a i i e g l , s t a i o a e b e s c s u a e a u a s s e s o e  
n , h r h l p p . l n m r e n s W l n e d f o t p e o n r b y a c i a y i w f a e e y t a t t y e e a s o u  
D b l h ( h , m r n m h k s e y m t s a n i e t f i o i n o a t s i o e t d y i e t e i t a o o c c s o a i a t t  
p b l h , n m g c e y a e o e s a e t i e a

m r h i h l l h r n e a b t a e n e a t , o t h g o a y e r t e c o s d o x d a t a e u a i e g a i t  
n p n r n m , r n v b f o n n m e g i y e s h n ' e a x h r b e a t e i d e s u s e a o d j e a a d e t s s e t c a t u t  
h r n m h n x h . n e i d f o r y e t w h e a b t a n D B o c l , u a d s y s i u s i e g o d a d a t a s a e t  
h n n r l l d a t a y a t c a t b s a l o p i m d y a t a b a s e A d a t a a t a b e s i  
r k h v l l a e a t e s e o f t e r s o t c e s s e t o n a n g a t a b a s e a y s e c o d a  
h k , r r n r r n y d i a t a y a n e t l e \$ , e v a o p e a t y a t o u s e c o d a e i d a d e t d a e s o s d  
r r r k . y o t a y i a c o s d e

r n r b b Y m h b o c e p e a s e a n y d a n d a t a a c e a t i e g t d a t a a a s s o c i a t i n g a d e t  
h h b h h w e t h m a t e a h r h i t e t r m a t a a t s a t i e v d y a t a f o c i o e a e a t i g  
p r ) . n h n x A r e d e i d h s p m a r o f t s s o m , i a t i e t e c o d a d a t a y a t e t i a o s t  
b l k h r v h n o r e d b a k a c a p t s a t s i l d c t e . y a t e t e c o y d a y d a t a y a s c i a s t i  
r k b n n n h p m r a y a b e s a t e a k s e a o d r d a t a y d i e t . u i a d a t a a c o s d e o d a t a

n n B h , n r b c o o e n e d r n . a s a g e c o n d y a A d a t a f o o d d o g e d u t i e g o s d i  
r r b D B p y h y n o r n i a d a t a e n r e s h n o t q u a t e t e c o u y d a e c e s s a e t c a g i g  
r n h p m r r b m D B o s d g a t a i n t h y a d a t a a c a y s e a o t o f d u i e a o y d i e t e c o d a  
n h h h r h n r m w e d e k d o i g h e e r t e t r l a f o r c e s a y f o i b a t i f e a y i e t e c o d a d a t a

n n r r l n y o e b t w a t c r a p t e . i t d c i y b t s e c o A d y a d a t a a w e a t t o t t e i t a  
r b r n n n r y s e c o r l i a d a t a s s r t n i r o a e o s T a t e t o c u a g e t d a t a e c e d  
n r r m h p n r r b y n s e c o h d y a c o n d o y f d i e t r l i a d a t a s a s e t d e e c e o t i o t s t i e  
l p m r l l n h n s i t h a t e o t e w o d a l i e a n a e o y e s e c o d a d a t a a e e S e y t i e g s d a  
R p 5 m ( r n n r n a t a e c o 4 s d e a g f o o e f o i o a t i

n r b r p y r b e c o D B d a h d a t a e c o r s d b a l k y e a t d e y e a t d y o f e t e c e o a t a a  
n n r n r l h l h m r n o f r d o n t i e D B n u l f o h a e o e t h t a s e i w e d e t u y o t d e t e o t e t  
n r b n n h , n y r p e c o d a t a t a s a n a d i e k e o f t e a o d v e a t y i x d u o e y e s i t i e g  
N h h b l k n h r o e n t a t t r r a b \_ o n o t a n e x D B u o s o e e o c o e d s e d f s ' u  
n m p n h h n h r n l b n e a s c e a o t d d y a t a t e t o x o e t e i d e d

m p m l n n l K E r p 5 m ( e a s y m e n t i e g c t o a s t e a g f o o e f o i o a t i

r r r m r n W y r b e D B m e d e b r n f o h s e c o d a d a t a y a t c a t i a e u t e t d a t a  
n l k m r h p r n n y r n a d y o p t i n a e t e b f o e t o e s o d i e g o y d i e t i a d a t a a



## Opening and Closing Secondary Databases

```
#include <db_cxx.h>

...

Db my_database(NULL, 0); // Primary
Db my_index(NULL, 0);   // Secondary

// Open the primary
my_database.open(NULL,           // Transaction pointer
                 "my_db.db",     // On-disk file that holds the database.
                 NULL,           // Optional logical database name
                 DB_BTREE,       // Database access method
                 DB_CREATE,      // Open flags
                 0);             // File mode (using defaults)

// Setup the secondary to use sorted duplicates.
// This is often desirable for secondary databases.
my_index.set_flags(DB_DUPSORT);

// Open the secondary
my_index.open(NULL,              // Transaction pointer
              "my_secondary.db", // On-disk file that holds the database.
              NULL,              // Optional logical database name
              DB_BTREE,          // Database access method
              DB_CREATE,         // Open flags.
              0);               // File mode (using defaults)

// Now associate the primary and the secondary
my_database.associate(NULL,      // Txn id
                     &my_index, // Associated secondary database
                     get_sales_rep, // Callback used for key extraction.
                     // This is described in the next
```

```
0); // section.
// Flags
```

```
// Close the secondary before the primary
my_index.close(0);
my_database.close(0);
```

## Implementing Key Extractors

```
typedef struct vendor {
    char name[MAXFIELD];           /* Vendor name */
    char street[MAXFIELD];         /* Street name and number */
    char city[MAXFIELD];           /* City */
    char state[3];                  /* Two-digit US state code */
    char zipcode[6];               /* US zipcode */
    char phone_number[13];         /* Vendor phone number */
    char sales_rep[MAXFIELD];      /* Name of sales representative */
    char sales_rep_phone[MAXFIELD]; /* Sales rep's phone number */
} VENDOR;
```

```
#include <db_cxx.h>

...

int
get_sales_rep(Db *sdbp,           // secondary db handle
              const Dbt *pkey,    // primary db record's key
              const Dbt *pdata,   // primary db record's data
              Dbt *skey)          // secondary db record's key
```

```
{
    VENDOR *vendor;

    // First, extract the structure contained in the primary's data
    vendor = (VENDOR *)pdata->get_data();

    // Now set the secondary key's data to be the representative's name
    skey->set_data(vendor->sales_rep);
    skey->set_size(strlen(vendor->sales_rep) + 1);

    // Return 0 to indicate that the record can be created/updated.
    return (0);
}
```

```
db.associate(NULL,          // TXN id
             &sdb,          // Secondary database
             get_sales_rep,  // Callback used for key creation.
             0);            // Flags
```

## Working with Multiple Keys

```
int
my_callback(Db *dbp, const Dbt *pkey, const Dbt *pdata, Dbt *skey)
{
    Dbt *tmpdbt;
    char *tmpdata1, tmpdata2;

    // This example skips the step of extracting the data you
```

```

// want to use for building your secondary keys from the
// pkey or pdata Dbt.

// Assume for the purpose of this example that the data
// is temporarily stored in two variables,
// tmpdata1 and tmpdata2.

// Create an array of Dbts that is large enough for the
// number of keys that you want to return. In this case,
// we go with an array of size two.

tmpdbt = malloc(sizeof(Dbt) * 2);
memset(tmpdbt, 0, sizeof(Dbt) * 2);

// Now assign secondary keys to each element of the array.
tmpdbt[0].set_data(tmpdata1);
tmpdbt[0].set_size((u_int32_t)strlen(tmpdbt[0].data) + 1);
tmpdbt[1].set_data(tmpdata2);
tmpdbt[1].set_size((u_int32_t)strlen(tmpdbt[1].data) + 1);

// Now we set flags for the returned Dbt. DB_DBT_MULTIPLE is
// required in order for DB to know that the Dbt references an
// array. In addition, we set DB_DBT_APPMALLOC because we
// dynamically allocated memory for the Dbt's data field.
// DB_DBT_APPMALLOC causes DB to release that memory once it
// is done with the returned Dbt.
skey->set_flags(DB_DBT_MULTIPLE | DB_DBT_APPMALLOC);

// Point the results data field to the arrays of Dbts
skey->set_data(tmpdbt);

// Indicate the returned array is of size 2
skey->size = 2;

return (0);
}

```

## Reading Secondary Databases

m r r b n , r r r r r e i r a i a d a s a o h c b a y d c o u s f l o y o s e c o d a u a t s e a e i t  
 h m n n . r b n r n T n d a s a p e m r w i t h c e h e e y d s e c o y d a a d i a d a s e s i t a t  
 r n n r b w r y h n e o r r s e c o n d a d r a s a c o y d e s e c o d a c o s d d a t a i e t e d  
 h p l m r k r n y p r n o o h s e y t y e k u i a r n a d d a e s o d y g t e s e c o d a e e e t e d  
 . y o b u  
 o l m n , r n r x b n o r e k y a e s s l a y p o g s e c o u d l l a d a s e o s a i e s e e a t d t e a s o s f  
 : e a

```
#include <db_cxx.h>
#include <string.h>

...

// The string to search for
char *search_name = "John Doe";

// Instantiate our Dbt's
Dbt key(search_name, strlen(search_name) + 1);
Dbt pkey, pdata; // Primary key and data

Db my_secondary_database(NULL, 0);
// Primary and secondary database opens omitted for brevity

// Returns the key from the secondary database, and the data from the
// associated primary database entry.
my_secondary_database.get(NULL, &key, &pdata, 0);

// Returns the key from the secondary database, and the key and data
// from the associated primary database entry.
my_secondary_database.pget(NULL, &key, &pkey, &pdata, 0);
```

## Deleting Secondary Database Records

When deleting records from a secondary database, you must first delete the corresponding record from the primary database. This is because the secondary database is a shadow of the primary database, and it cannot exist without the primary database. To delete a record from the secondary database, you must first delete the corresponding record from the primary database. This is done by calling the `delete` method on the `Dbt` object that represents the record in the primary database. Once the record has been deleted from the primary database, the corresponding record in the secondary database will also be deleted.

Here is an example of how to delete a record from a secondary database:

```
Dbt key(search_name, strlen(search_name) + 1);
Dbt pkey, pdata; // Primary key and data

// Delete the record from the primary database
my_primary_database.delete(key, pkey, pdata, 0);

// The record in the secondary database will be deleted automatically
```

```

#include <db_cxx.h>
#include <string.h>

...

Db my_database(NULL, 0); // Primary
Db my_index(NULL, 0);    // Secondary

// Open the primary
my_database.open(NULL,          // Transaction pointer
                 "my_db.db",    // On-disk file that holds the database.
                 NULL,          // Optional logical database name
                 DB_BTREE,       // Database access method
                 DB_CREATE,      // Open flags
                 0);             // File mode (using defaults)

// Setup the secondary to use sorted duplicates.
// This is often desirable for secondary databases.
my_index.set_flags(DB_DUPSORT);

// Open the secondary
my_index.open(NULL,             // Transaction pointer
              "my_secondary.db", // On-disk file that holds the database.
              NULL,             // Optional logical database name
              DB_BTREE,         // Database access method
              DB_CREATE,        // Open flags.
              0);               // File mode (using defaults)

// Now associate the primary and the secondary
my_database.associate(NULL,      // Txn id
                     &my_index,  // Associated secondary database
                     get_sales_rep, // Callback used for key extraction.
                     0);          // Flags

// Name to delete
char *search_name = "John Doe";

// Get a search key
Dbt key(search_name, strlen(search_name) + 1);

// Now delete the secondary record. This causes the associated primary
// record to be deleted. If any other secondary databases have secondary
// records referring to the deleted primary record, then those secondary
// records are also deleted.
my_index.del(NULL, &key, 0);

```

## Using Cursors with Secondary Databases

```
#include <db_cxx.h>

...

Db my_database(NULL, 0);
Db my_index(NULL, 0);

// Get a cursor on the secondary database
Dbc *cursorp;
my_index.cursor(NULL, &cursorp, 0);

// Name to delete
char *search_name = "John Doe";

// Instantiate Dbts as normal
Dbt key(search_name, strlen(search_name) + 1);
Dbt data;

// Position the cursor
while (cursorp->get(&key, &data, DB_SET) == 0)
    cursorp->del(0);
```

## Database Joins

l mr r n r yb v wfo he p o r y o r e s e c o h d n a d , a t s e a s s o a e i a t d i t a i a d a t s e a e t  
r r p m r r r b y n h n v o r m y a n e t e l u i a c o m s d s e a o d e t e i s e c . o t o f y e i s e c o d u a e s i  
n n r r y . o o j d s t s i i u g a c i s o u u

h m n h p T n r o h o g w s t o i d u r e u n e n e a e s e e t d s a c t e t t s a t e s u f o i o a t o  
m h h r r . n t p v l h o o g l e n m b r s a t o t n e s v i r a s i u u e i t a e i t d e o f d a t a e s  
h b l n n m w w p r v o f c B o p p e n e i e s y u t f i o g , v a e s e c e t i s t o s e s a u t u d t a t  
r n r n m r n n n m w n w h m n e e e l o t r i f i r h o a t o n t o e y t i g i t a o e c c a e a t c i t i t c a t a  
h p m b h h l m o e i d r n s u n a c a u h e y i u t c a e a o a s o t u i f i o a t i c s a  
n b , r r m l , l m b , l o o n m b e , o f o m s f u e n k e i e y a g o a b e i d u e o f s s e a u e s g e a  
l , r n m , y o e d a d j a w e a s f e a u

l , l k l l b l n m r y w s t i s e k o y o p s o r t i e u i e u s v s i g e y y i e a o u o u i a e s i  
U ( h m b l N b l l i e t l e r i p s e a t e t o a b e l s ) . u o e e d i f o a u s t w i o s e o o u d  
r r h r n h l l h e t c e e a s a c t m e b t h a d u f e s i r i a e t c a e a t c i t i f e t y o a b e s i d u  
r . v y e i o t

h m h , h n T r m y p l b n y e r s t b i d a t a n i g n t c e u e a t y e i s e c o d u a d a t s e a o e f o e c a f  
r r h n r F r m p e t l y a e a t c i t i r t o y n a t t e y o e a e o i g e y e a t s e a o d a  
l r h , r n m b r n h , r f o r d o p o a e n t f o n e o f , h o t s u o a e t f o . e o f s s e a u e s g a d f o t  
r n , l l n k r y r v i c o s e r h o h e a u x d a u l e e a t o a t i c o t f o e c y s c s e c o d a u d a t s e a  
l l h n h n p y n h n o o l d r o n s t s i l i n g t o p e u s t a d c t e s e s t e i d u o t o g t s c i u a t u

h r h p m r r b y n v r t o o n e a e y e a t d s t h i i , a d a t s e a a d a e i e s w i e g y d e s i a t  
b l r r m b v l r b y e a e r t v a l i i e r e t e i o a b e e c o u s d s e a o d s a y e g c a e a t c i t i o  
m p l n h , l l m b h x r c r f o e n a e f . h i l l a e n l a b l e s i y t u a e d o c f a i d a e u o a b e s i  
r r r h l l m b v h m n a t e f o o d s O a e t o a b e s i t u a a i s a

m n r p h n , T m x r m p n e e o s t n a t s a F r m p t u l o i f o c o o j d e s x o u s i o e a e  
n n h l l m y b h w o r n i g t , a t t o r i b l a e t o a b e s i , t u a t e d a d y t t e i t o o t a  
m n n n h . n n v a t r a a i s a o c a j d s t s i u i g a c i s o u u

## Using Join Cursors

n r r T j o s e o a c i s o u u

mr r r r • r n r v h e O o t o o e c t o y h o s e c o u d a d a t s e a t w e a s s o a e i a t d i t e s e a  
r b . y i a d a t s e a

n h h r r h n k r o s l o i t c a c r s o p t u e f o u d w a e y e a i c i o u e a e i e s e t d o  
o l b , h p r x p r n h e , r a r l o r v o i l r e t e a s i e s t a t i u e t s o f o e t o u d a t s e a i  
n h r r h l r r h o n o i d b t e t p e c o s d e i e t s o f o e t o u e d d a t s e a d s o i e i d  
r r n h , r m h i v a n k o t a t p e c o s d a d e t s o f o e t u e a d a t s e a o i e i d t

n r r r n p , l n h l y e C e a t r l r a o f p r s o s p n a d c a u i e i t a f e t s o s t a t a u a t i i a t i g  
n r N h h . m b j n y l n r o i e q u e t t u a t i a a s e t e t u e a t d u



b n n r r h n .h j no haioa cimo po Dbndjout(i iugt Y u e ot do s ss a st  
m h h rr n r r hr p n e m p det ya oase cop ydra so s t at oue e d adu o iei d iete o  
p  
r hr m hn r m vhl enterate nese oft c at ieg. sd tete too eds d ut  
l r r r . . y os e o c so u u  
rnl h h m . l , ly r r r r fwo e aode. yet cose a c so s u u  
F rmp l : x o e æ

```
#include <db_cxx.h>
#include <string.h>

...

// Exception handling omitted

int ret;

Db automotiveDB(NULL, 0);
Db automotiveColorDB(NULL, 0);
Db automotiveMakeDB(NULL, 0);
Db automotiveTypeDB(NULL, 0);

// Database and secondary database opens omitted for brevity.
// Assume a primary database:
// automotiveDB
// Assume 3 secondary databases:
// automotiveColorDB -- secondary database based on automobile color
// automotiveMakeDB -- secondary database based on the manufacturer
// automotiveTypeDB -- secondary database based on automobile type

// Position the cursors
Dbc *color_curs;
automotiveColorDB.cursor(NULL, &color_curs, 0);
char *the_color = "red";
Dbt key(the_color, strlen(the_color) + 1);
Dbt data;
if ((ret = color_curs->get(&key, &data, DB_SET)) != 0) {
    // Error handling goes here
}

Dbc *make_curs;
automotiveMakeDB.cursor(NULL, &make_curs, 0);
char *the_make = "Toyota";
key.set_data(the_make);
key.set_size(strlen(the_make) + 1);
if ((ret = make_curs->get(&key, &data, DB_SET)) != 0) {
```

```
// Error handling goes here
}

Dbc *type_curs;
automotiveTypeDB.cursor(NULL, &type_curs, 0);
char *the_type = "minivan";
key.set_data(the_type);
key.set_size(strlen(the_type) + 1);
if ((ret = type_curs->get(&key, &data, DB_SET)) != 0) {
    // Error handling goes here
}

// Set up the cursor array
Dbc *carray[4];
carray[0] = color_curs;
carray[1] = make_curs;
carray[2] = type_curs;
carray[3] = NULL;

// Create the join
Dbc *join_curs;
if ((ret = automotiveDB.join(carray, &join_curs, 0)) != 0) {
    // Error handling goes here
}

// Iterate using the join cursor
while ((ret = join_curs->get(&key, &data, 0)) == 0) {
    // Do interesting things with the key and data
}

// If we exited the loop because we ran out of records,
// then it has completed successfully.
if (ret == DB_NOTFOUND) {
    // Close all our cursors and databases as is appropriate, and
    // then exit with a normal exit status (0).
}
```

## Secondary Database Example

hlp nrh b k b ypp ln e o sicl n eastp ilst do rēDB it ac iōsiu tyat v d add i sē e a  
n h mpl l ,ll n h mpōatsa wstni xær b æie t dose e .æs y te seco d aud atæsa  
ll :

b ( r l pp x nsh as n ag n vne d ag e ita ac iuti tcat oae ad d  
n r l b n n l r D v dhataise e ad atæsa ecō v d a atæsa it  
l b lp x ( ll n )\_he ppæ l n atæ ac l p n x ag e eie t d t at ac iatio tsoæ e a  
r b hrp p r n ny n seco r d n d atæ fō etx osevaf yid u i g i o t e it ea

## Secondary Databases with example\_database\_load

DB\_INSTALL/examples\_cxx/getting\_started

```
// File: gettingStartedCommon.hpp
// Forward declarations
class Db;
class Dbt;

// Used to extract an inventory item's name from an
// inventory database record. This function is used to create
// keys for secondary database records.
int
get_item_name(Db *dbp, const Dbt *pkey, const Dbt *pdata, Dbt *skey)
{
    // Obtain the buffer location where the we placed the item's name. In
    // this example, the item's name is located in the primary data. It is
    // the first string in the buffer after the price (a double) and
    // the quantity (a long).
    size_t offset = sizeof(double) + sizeof(long);
    char * itemname = (char *)pdata->get_data() + offset;

    // unused
```

```
(void)pkey;

// If the offset is beyond the end of the data, then there is a
// problem with the buffer contained in pdata, or there's a
// programming error in how the buffer is marshalled/unmarshalled.
// This should never happen!
if ((u_int32_t)id.getBufferSize() != pdata->get_size()) {
```

```
// Make sure the default constructor is private
// We don't want it used.
MyDb() : db_(0, 0) {}

// We put our database close activity here.
// This is called from our destructor. In
// a more complicated example, we might want
// to make this method public, but a private
// method is more appropriate for this example.
void close();
};
```

```
// File: MyDb.cpp
#include "MyDb.hpp"

// Class constructor. Requires a path to the location
// where the database is located, and a database name
MyDb::MyDb(std::string &path, std::string &dbName,
           bool isSecondary)
: db_(NULL, 0), // Instantiate Db object
  dbFileName_(path + dbName), // Database file name
  cFlags_(DB_CREATE) // If the database doesn't yet exist,
                    // allow it to be created.
{
    try
    {
        // Redirect debugging information to std::cerr
        db_.set_error_stream(&std::cerr);

        // If this is a secondary database, support
        // sorted duplicates
        if (isSecondary)
            db_.set_flags(DB_DUPSORT);

        // Open the database
        db_.open(NULL, dbFileName_.c_str(), NULL, DB_BTREE, cFlags_, 0);
    }
    // DbException is not a subclass of std::exception, so we
    // need to catch them both.
    catch(DbException &e)
    {
        std::cerr << "Error opening database: " << dbFileName_ << "\n";
        std::cerr << e.what() << std::endl;
    }
    catch(std::exception &e)
    {

```

```

std::cerr << "Error opening database: " << dbName_ << "\n";
std::cerr << e.what() << std::endl;
}
}

```

nn, p T w poe ean on database ad u wote o y e seco dua d ase a  
h n n r b asoa veiat yit tet ei o t d ase a

n, h h n T mp mln w s n e a h e p e r o d b s o n e e e t i e e d a t f o s t b g æ e s e t d  
h n n h w , w j m a e d r s e n t w f u r w t h c s u . i e e o a o f d t i o c c o s e e e t u  
mp mln n h r m m n e m p t e t b i e e l p a t f o s t o i \_ ( a d e . æ d a s e a o d æ a g 8

```

// Loads the contents of vendors.txt and inventory.txt into
// Berkeley DB databases.
int
main(int argc, char *argv[])
{
    // Initialize the path to the database files
    std::string basename("./");
    std::string databaseHome("./");

    // Database names
    std::string vDbName("vendordb.db");
    std::string iDbName("inventorydb.db");
    std::string itemSdbName("itemname.sdb");

    // Parse the command line arguments here and determine
    // the location of the flat text files containing the
    // inventory data here. This step is omitted for clarity.

    // Identify the full name for our input files, which should
    // also include some path information.
    std::string inventoryFile = basename + "inventory.txt";
    std::string vendorFile = basename + "vendors.txt";

    try
    {
        // Open all databases.
        MyDb inventoryDB(databaseHome, iDbName);
        MyDb vendorDB(databaseHome, vDbName);
        MyDb itemnameSDB(databaseHome, itemSdbName, true);

        // Associate the primary and the secondary
        inventoryDB.getDb().associate(NULL,
                                      &(itemnameSDB.getDb()),
                                      get_item_name,
                                      0);

        // Load the vendor database
    }
}

```

```
loadVendorDB(vendorDB, vendorFile);

// Load the inventory database
loadInventoryDB(inventoryDB, inventoryFile);
} catch(DbException &e) {
    std::cerr << "Error loading databases. " << std::endl;
    std::cerr << e.what() << std::endl;
    return(e.get_errno());
} catch(std::exception &e) {
    std::cerr << "Error loading databases. " << std::endl;
    std::cerr << e.what() << std::endl;
    return(-1);
}

return(0);
} // End main
```

## Secondary Databases with example\_database\_read

```
// File: example_database_read.cpp
#include <iostream>
#include <fstream>
#include <cstdlib>
```

```
#include "MyDb.hpp"
#include "gettingStartedCommon.hpp"

// Forward declarations
int show_all_records(MyDb &inventoryDB, MyDb &vendorDB);
int show_item(MyDb &itemnameSDB, MyDb &vendorDB, std::string &itemName);
int show_vendor(MyDb &vendorDB, const char *vendor);
```

```
// Displays all inventory items and the associated vendor record.
int
main (int argc, char *argv[])
{
    // Initialize the path to the database files
    std::string databaseHome("./");
    std::string itemName;

    // Database names
    std::string vDbName("vendordb.db");
    std::string iDbName("inventorydb.db");
    std::string itemSdbName("itemname.sdb");

    // Parse the command line arguments
    // Omitted for brevity

    try
    {
        // Open all databases.
        MyDb inventoryDB(databaseHome, iDbName);
        MyDb vendorDB(databaseHome, vDbName);
        MyDb itemnameSDB(databaseHome, itemSdbName, true);

        // Associate the secondary to the primary
        inventoryDB.getDb().associate(NULL,
                                     &(itemnameSDB.getDb()),
                                     get_item_name,
                                     0);

        if (itemName.empty())
        {
            show_all_records(inventoryDB, vendorDB);
        } else {
            show_item(itemnameSDB, vendorDB, itemName);
        }
    }
}
```



```

    }
} catch(DbException &e) {
    std::cerr << "Error reading databases. " << std::endl;
    std::cerr << e.what() << std::endl;
    return(e.get_errno());
} catch(std::exception &e) {
    std::cerr << "Error reading databases. " << std::endl;
    std::cerr << e.what() << std::endl;
    return(-1);
}

return(0);
} // End main

```

T y pp elon ohe mp tmlgt atnee hd t dd tet ac bati et ie e datof  
 show\_item() f c o t i u

n h n r p r mrh n h h et m o f s be de p f n s o i g o e t y c o s i s e d u s t i a b a t  
 r mp mln n p l , r rfp o p l e t i e e r o a t i l e s e e s 6 A44 a e u e v a g y e t e t i o c s e e  
 h n mp mln n h pp l n n e e e t i e e d a t o f s t i a c b a t i i

DB\_INSTALL/examples\_cxx/getting\_started

h r h l n h w DB\_INSTALL e e r DB b r w e t y c o a t i e y e o c e a d u s d i t i o u i u

```

// Shows the records in the inventory database that
// have a specific item name. For each inventory record
// shown, the appropriate vendor record is also displayed.
int
show_item(MyDb &itemNameSDB, MyDb &vendorDB, std::string &itemName)
{
    // Get a cursor to the itemname secondary db
    Dbc *cursorp;

    try {
        itemNameSDB.getDb().cursor(NULL, &cursorp, 0);

        // Get the search key. This is the name on the inventory
        // record that we want to examine.
        std::cout << "Looking for " << itemName << std::endl;
        Dbt key((void *)itemName.c_str(), itemName.length() + 1);
        Dbt data;

        // Position the cursor to the first record in the secondary
        // database that has the appropriate key.
        int ret = cursorp->get(&key, &data, DB_SET);
        if (!ret) {
            do {
                InventoryData inventoryItem(data.get_data());
                inventoryItem.show();
            } while (cursorp->get(&key, &data, DB_NEXT));
        }
    } catch (...) {
        // Handle errors
    }
}

```

```

        show_vendor(vendorDB, inventoryItem.getVendor().c_str());

        } while(cursorp->get(&key, &data, DB_NEXT_DUP) == 0);
    } else {
        std::cerr << "No records found for '" << itemName
                    << "'" << std::endl;
    }
} catch(DbException &e) {
    itemnameSDB.getDb().err(e.get_errno(), "Error in show_item");
    cursorp->close();
    throw e;
} catch(std::exception &e) {
    itemnameSDB.getDb().errx("Error in show_item: %s", e.what());
    cursorp->close();
    throw e;
}

cursorp->close();
return (0);
}

```

T s conU bes p example\_inventory\_read u s y i g t i o w a t o c u a

example\_inventory\_read -i "Zulu Nut"

## Chapter 6. Database Configuration

## Setting the Page Size

## Overflow Pages

## Locking

## IO Efficiency


P n h n DB m n e agevi m f eak ot E ffr e vi i ts. i at i g at at ad sd io so e  
 op l n p , l h r h h n m m - a h y o a s i e s e w i a l o s e f o h c y i e t i e o c c a c a e t e a g o  
 H n r k n r n l n o v d e e m p t i o p p i g s t a t e f O c e i c i y s a f i g i i a t i c a t a c i a t i  
 m m . e f o c e a

p r m n n n l k l y s o n e r h a t i n g e h t s e z m a e i t u a c s e i o w e d t e i o v c d a t a t o  
 n m r k m l p m l h b k l t a d s d f l s a l l i g T o e z o a t i s o y s e i s i a q a u t e t  
 l m b k l F r m k . y n f z e s h t s ; o t s l e i o b p t i a d y e f O c e i c i o s o s d e c u t a t a s a  
 h l h p r z m b k l e i l / t s a i q a y t e b e u a t s i g e t s ' O c s e i

E l l p D B m r n r b r y h s s e b t i a e f o h s d a t a f a s s e a o d e t d a t a e a g e v i s a t i d  
 n m r k p m f h r r n d a t a p a d s d i r a e a g a b a t i o s t e i z o a f e t e a g e i o e s o t o  
 b k l l h n h , p r m m z n e t r o c s e i e n t h e b e a t s i g e t c a d i t e e f f i c i e n c i e s i o  
 p r n D B r l / . e s o s d o t s ' e O e s s t u

F r m p l p p , p r m x l h m o e p r a e n s z o s e n b k l e a g e y s i u e a l t . y o e z a t s i g a t o c s e  
 n h n , D B r p k w n r s t i v e a r e s i t l a d a n t s d i s i t i i t i g t o a o t o f u o a c g f a s i s e  
 n m n p p . l n r A p y r y a g l e i w a m p a j b a t i e s i t s t o a o t o f u o a c g f a s i s e t e a g e t  
 o r r m b n r n h r l l m p y o e a t s i g e t i r i y g h e t e f a s i s e w t e a g e e s i t e t o o t o f e t  
 m b h p p l n h n , w h y o m p e i t b k e t l o c h o a i e t y e s i t e f e s s e t e a g c a o t s d i e e  
 r l n n m l r k h n h l / p p e s l y b a t i m p i i t o l e s d i O t f a e t a c y b a t i s d i s e e c e t d a e  
 h l h n n l l m b s k l e i t a s e a y a t e t y e d u f z e s i e t o c s e i

d r l , l p A h v y l r h y n t a t i f u l s e e c l n t h e a g l e i t s a t i e a y t a e t y e d f i e s i e t o  
 h n h , p r m m z h r m s e i r e t h y a b e y a t s i g e t l l e a o e d o e d a y a t s a e c s s a o f t f  
 r r f i r r m , p r m e a r d n , e s n t e l t o u s o p e e a t s i g e t s e e s t i g a e g d a t a e  
 r n l h p r m m y n h l e s b k y i e e b a s i g e t r y d e i g f g s i e y a c s o s t s t a t i e o e  
 m r r r r h - n h y h p s s , e r t s c e m i t f o i a b e l l d r e a d s t i s e a y e b e w a t s i g e t e e d i g  
 n n m l r m r k h n y l l s r f i g r i i a t o l e l l D B a t a r s d y i t s a d i a t a e . q e i d f t f i u s ' e d u e s

h l n r n n r  W h m n p e i , t a c h a s h i n e a o s t s i e m d i s t i u a a a e a g e d u e t y t a f e s i e s t  
 b k l n n r l n z h o c n s l e d . h f i p a t s a a t i f h a n g a e a y e s o a z s i t a t e a g e s i e a g t  
 h l m b k l D B p n e f a d i s e r t s ' o r n s e w i s e s r d t e u t e s a g b c a e i c e e s t s e a  
 p b l p r p l b m h s o d s e n f o r n a v a t i o n e a g e m r e i t t s a e t e s o f t a t a a t i c a i t o e  
 n m r n , f o i o a t s e e  
 h p r / h n : h n . l / m n w w w b k r t t l b b e c e a o r p e c t o r m l h g u d / e y a t i e . e e . d d e f s a a e c i t

## Page Sizing Advice

P n n b n n r h z r m e a g i o g a n d t h s i g a t i s t u e e e s o e e e g y a g d e s i t a c  
 l p r . y s e o s e c o t u e a g e i u

n r n l n , n h m m p v e g a a d e g i l o o e r t c o s m e d o a t i a e a g e y t s a i y q a o f e s i e  
 k l h l n z . o c s e i s i e t e d s a i t o a t i u

r l n h h p n r f m d n a c i a l e p l s d c m r a t d a t a e e s c i a f t o i t s a e g e a g s a  
 B r h n , r ) p r T m m v e r e n z o g o b n e a g e i o y t c a o e a b y v d a t a c e o u e a a d e  
 h n r l m b k l h n y r l r l h a t i o g f z e s i e s ' o r s e i r e t e e g a e s z i t a t e a g e a g e s i e a e e

o n h r l h T x r ele rce on rty t st v i e r n f i o i r e u a a g e d y o f a c o c e y o c c u i g o u  
 n n h H . l r n m h p a r b a t i h s t y e a d n o s e y o c z a c a t o u e x g e b t e t e d s æ i e e e d d  
 p p l n h b , y n D f o l l o l l a c b a s i m d a t æ t e r w e t t o w y s i o g v i o a o o t y a d e c e s s a u  
 n p r k l . c o e t o t f o e a g s

## Selecting the Cache Size

mp nr pp l z b c C a s e i y i o m t a l t b l a c b a t i e a s e a f i s t e i o t u o s o a a e a  
 l n p m m y l l m r m o h l a c b a t i e a f o l h / b e a n . s i f f e f o o o t u c s d i O u O e b e t a d f i  
 l r h n , p y l n l o m c c m a m p o t h u e a g e t o w l l a c b a t i u s e o e e o u y t a c i t a t a  
 r r . p p l n n M h y n e s y d o t e n e n n i o i p a r r b a t i s e s o y t c e o a u t o o s o e a t i g  
 n n r n l p p y l n b n s p e s y i m e s r t r i n l n a u b a t i e u s i g a y o d o f t e o a s t i g i  
 p l p r m m x y . e e t e o o e f o c a

l m n Y h r y o s e c o t r c u s : : e i s \_ a c h e g e t e ( ) u D e n v : : s e t \_ c a c h e s i z e ( )  
 n h h r r n b n w e n n y d n g r e e t h e a n . v i g a l a s e a Y o i e o z t o d c c a s e i s u t  
 r b , h r m l w b l e b o d e n o z w m p s i b i n t y s e i v e i t d y a a e a e o a d f o c a  
 m . c o s e d o a i

h m h n n b r e r f t i l g a r a s h e i s o i e n t i f g a y t f o t t e a t o u c y c a e a g u i t a e i  
 o h p p l n h n s o y c i t a e y s m a n i e t h d b b u a c b a t i c u a g i g l a t a q e i e w y t e u e s t a  
 r h l r h n b p o w e d y p e i o l n n e a g p c r e e s y d u e s o i t o t a c b a t i o i t a d o t i  
 n n h h m l v k e w o n i r e t a p p a t o s e o n c s d i y O c u i f g o u a c b a t i u o i g i g  
 l r r b r r h n o t s h i q u e n t o a o t h e t e i d a s y a c o s d e t o s o z d e y s a a e t e d f o  
 r h ( h n h m m v r c c a y o e d d ) t a t e g o g e o o o d o

h m m n Y n l l h d \_ s t a c p s e e t u h c o y w n a d e i t i t i t e t u o o t i o t v g a g e e f f e c e t e s s  
 r h n p r l h r n m b , y p o f o n l c c e h h u a n t i a e l t h e w o f e s a p d i n t c e u s w b o a g i t  
 l h l r . h v 1 e a c h T b e t a g e a r h e c o s e y l o p 0 0 % t a t c a g t e x t e u e t t f s t i e a d s  
 n , r p n r n p m w p y b o a n t o h n a d , h e a b n e c i u i e g f o c a y o e s e t o s o c o s e n d u  
 h r h m n , h z m e n s y a i e g t p e d f o c y e a s s a u . i o g e a i e o u s t o t i t u

## BTree Configuration

h l h h p r h p n r h b k o m g i r g t h g e t e o s i c n e a y t i u t d i o o w a c e t i u t a t o d o  
 h p B b , n s o e d h t s i p t a f f a r c f c v i o t e e l v e t o d o c o t e u g s e o t s i i æ e d t a i  
 n l , l l n m h w w s e i n o t i e B i s d i s c o f i g o a t i s e s u t a f a u e i o t e e u u

l h , n b r y e S f c i i a w i s e t o t i e e s t e i

h p l r r • A w . o i g l c e a c o s d

mp r r b l k • . e S t t d i o g a a t a c s a

## Allowing Duplicate Records

B r b n n n p l T r r n ee rdnasea rcb taidclē atco sd eOeco scois eē d e al c  
n h hrnb h r r k h mpw of roaet lē n n yeco sdse es tcau e ase a q caote oaet

B kl ,rmp r n y l phry etp a r s b d u x k a r d i g a c o i u g a v i c a s a i y s i t o e t e s  
t l h h h m n l k r n r r d o a t i g e i y h t a e g e s o c o a e e d u s t e f d a s t i g t u u  
m h Db::set\_bt\_compare() r l e n t . d e e S e t e s e t o t f i o e d s t a i

B l DB b, n llyp l r e f d a t l d a s e a p d o b a r A d c ē atco s y s e a w a e a t t u o t t  
r l r k l p r l n e a o r y d r t r e s t h e a p e q u o x t l a o s i e s i t i e g u d s y t i e t e c u s i  
n r b r n r m b l x n r e r s i v i e g o v e . y o g w e i t t e t e e c o d

l h p l r r A l w b h i g n e n a t c o y s u k s e f f o i e a u a l a t s e a t c a t y t a e c o s d e e  
m m n l n r p n n r n y r o a n o . o c c r i e g i o f f o i o a i s t e y e t e s s u o t o a d c  
r r r n r b . e c o y s f o s e c o d a d a t s e a

F r m p l p p , p m r r b x n n o e y æ r s r b e o m b u i a d a t s e a o e a i d c o s d e e a t d t o a b e  
m h n h n b Y b l n h l l o m w i g t i t h u e a b a t l e e a o f t i d a e t o a b e s i i e t d a t s e a  
r p r l r l r , n l n h e a f l r h a o t i m o b o s b l u o r d e i d o u e t o o o f e t w a b e i o e u e  
n r n h r r p l l b b b l m y v p l f o m b a e g i c o o h e y e . i b a e e i o a b u e s i x c f i e t e i d s i e  
n k r h m n h m p l l n y y s e c o d a e r r s t i l l s a h t a y e t e c o d u a d w a t s e a c o s d s i e a e t  
n h , n r b m y p p r e l a d r e t e c o d a d a t s e a s s t o t d u e a t c o s d

## Sorted Duplicates

p D l r r n b m r r n r r c r e a t c o m s d a s . D e t d s i e t f s o e t d e d u o c c a s e a o t  
m l l r p l r b p y n o n t c a t i a o u o t l d y b e a t y o u s b \_ s u e s o r t i i g t f a g a t d a t s  
r n m . c e o a t i e i

r p l l r p p r h n h , n r f s o e n t p d c e s a t e a u o e t d e t u e t o t f i g c o t i e c f e i u d  
Db::set\_bt\_compare() h p l s r e m d e b t u e i e t o c o a t o f e t d c ē atco d s  
p l n h n l n p r h n h , d c ē a t l f t o p h c v c h o t i m o e d d e x e t e f d a e t c o i g a u i c a s  
. s s e d u

## Unsorted Duplicates

F r m m r n B r h , l l n o e f o r i t a e s o B r w e s s i o . d a s c o u t s i e t e c o s d e e s c o t  
n r n m p n p n r n s d e r d e s l y n s n o t e t h t i a u e d a g a t d o e e t i o c a t i g a t t  
B r h n n r n r h T o e s p D B . a e p p c a t r s a p t e t d e s i v s a t a i o e x s o f o t s e s s  
m n r p l r b r m b o a h t c a b i p p t i f g l h c ē a t c o n s y d e c s e y i t e u t a t a c i a t i u s e i  
r h r l r n r r r r e c o s g t a e a e a d i o a e t d e d

h , b n r T p p r r s a f i ē t d h a s h a c o i m p e g n d s t o u t o e t d d c u e s a t e t u e t s s a o  
h p p l n m l h p l l y n h s n i r t r a t h a n i y a p i u i p a a e f o e t o v t i x g s t e i e y e t e c o t t  
n n p m m p n l n m p . s l f i g i n a e t f l o y A y l a e y a t r e i o c e a e c o s d u o i t e t d a t s e a i s o  
m r k n D B p l l p m m p o n e d y b v o y t o i a e a f o y c a e e a t

h h h , b D B h n T n n r r s a t a i s t o d w h e p p e s a r e s e i t i e g o s d o i t a d a t s e a t s a t o s  
n r p l : o s o e t d d c ē a t u

pp ll n mp l • p yl r rño yac inati in i, rddr ad no: epaco d ig uet eteco d i  
r h n r p l seie td ate of ssie tddc est tu

r rl p h • p l r r h f baso sh se hndut etudc éuteco d tet d asva et ete eco d  
l n h p l nr h lhs ica drietndhc ést ctou d gtef vs agt no: epaco d od et  
h rh .l r T: ev ot de ee fátsga a

- DB\_AFTER

p r n h ll T p v l e n d a t a d e d o u t a t s i c e a d o i t e t d a s e a a  
 o l r r h k h r p m h k t e y c o h r e r s e f d s t o i e y o a t i e t e s e f d e t e c o u d t  
 h h r r n r l r m v k p r n d i e t s b a e y e f e s u e o d e d o u t a t s i  
 r m r . e t e f o e o i g d

p l r rn rn hT b mne d c elat chr d seir rtdoi tet d aye a e deiat feat et so s'  
nrpp n n h b .c e os oitiu iet d aye a

l n r r p l T rpp r s f h r s a g i b i e f l s o i e t d d c e s a t e a u o e f d e u d a t s e a

- DB\_BEFORE

h m p h v h n e DB esane s eam a elcew tt ate eco sl seiye td e deiat  
hr r r nrr l n n h b efo e et so s.c e out oatui iet d atse a

- DB\_KEYFIRST

h k p l r n h ll y lrf eten ho Deductent a,ty x ea d ssit iet dase a ad  
b n r p l h etnd abnscinf eg rd ve d u esatu oitusot wige t uete eco d  
n r h m nr h pp pr r ps lse iet af s et t iet a eiat dc esats itu

- DB\_KEYLAST

h B n ll p v h h n e y d e s e t r i a n t r e c e w t t a t e d c e a t c o s i e i e t d  
h l r r h p l l s æ t s æ t o d i e t d c e s a t s i t u

## Configuring a Database to Support Duplicates

pp m n bl n r b c esay mo uc to a h uco f. eg d at d at uae oati ei o od sti u  
h pp p r r l y s b c f r i g t b o : s e t p f f g s ( a g t r e f o e e t d a t a s o i e e f d  
rm . e f s i t a i

# h n r T : y e f s a g t o t c a e e a u u

- DB\_DUP

b pp rr n r pT l r re d atse a a s o so e td d c e atco s d

- DB DUPSORT

b pp r r p lT r r e d a s e a o s s d e t d c e a c o s d



```

#include <db_cxx.h>
...

Db db(NULL, 0);
const char *file_name = "myd.db";

try {
    // Configure the database for sorted duplicates
    db.set_flags(DB_DUPSORT);

    // Now open the database
    db.open(NULL,          // Txn pointer
            file_name,    // File name
            NULL,          // Logical db name (unneeded)
            DB_BTREE,      // Database type (using btree)
            DB_CREATE,     // Open flags
            0);            // File mode. Using defaults
} catch(DbException &e) {
    db.err(e.get_errno(), "Database '%s' open failed.", file_name);
} catch(std::exception &e) {
    db.errx("Error opening database: %s : %s\n", file_name, e.what());
}

...

try {
    db.close(0);
} catch(DbException &e) {
    db.err(e.get_errno(), "Database '%s' close failed.", file_name);
} catch(std::exception &e) {
    db.errx("Error closing database: %s : %s\n", file_name, e.what());
}

```

## Setting Comparison Functions

B l DB , l phr ymp n n ef h x a t r s e r e n c o i r g a l l a v a i f c o t i e e s u e t e c o s d o  
m l r r r r h r m r. h e f o n p o e r g e c k g d y n l e t o a o f c s e s a w c o i w s a i y o s e a d  
n m n n n . o e e d y w a e a g i t i a a

H n r m , n p p w v n p m e n b o i n e s i t e r a t i o n u c b a t i e f o c e a c a e f f o t s e t t i  
m m p r m n h . h c s o r t o Y k s o u i o r l e t i o c a d s t e i u e t f y d a t a e s o f o e t d a  
r b p p r r p y l r r o d a t a a u s s d e t d i d c e a c o s d

m h r n h m n p r o 6 o v y t e r s o y s w n o r v a a o t u : e d c a o t s o t f i g o t i e a u

r b k n • nYr n n p y r d a s e a e u e r d y s i n g t s i g u a d a t t o e d o e s o o f t  
 n - m r h n D r h a p e n e s r d i o u e d l i b g t t a t d a t a s i o g a c e y s e a e t o c a o f  
 n r h l l r b p m r e f e v g e t a b a s o . d a t a a o t i f o a t i s t  
  
 r n l h n • Y n h n ( o e a s r n i g n a e i t e u d s i a e t s y s a b a d e a s y i g e i t s g u a  
 k k B r l D B . k b y n d a s e y d e s h e n e y s y o r e s e s s a e s t t s i g a d e i t t d i a e i t s g o d  
 r H l n b n r h w r w o s o t w e t l n e b p H T i n e s t t w i g e e e s e e s o a o t i o t s t b u e  
 n p r m m p r n n o v e e i o g t o . e d c a o t c o s o u i f c o t i e e S u  
 r / h : h n . l . / m n w w w b k r t t l b / o c e y n o n / e c / t o n g d / e y d a t i e e e d d e f a s c f i q t  
 r r m r n . f o o e f o i o a t i  
  
 n n h n • k r Y p y p n d v o m p d o r t y a l r e , e t i e r o t c a t i e a t i e t o w v a i f o e a t  
 n h l m , n p r e n s o y n p y t v i s e a o n v h a a t t o e d c a o t c o s o u i f c o t s i o t u a t  
 l r h b r m n y v o y e t e e x a t e s t e a a e i d

Creating Comparison Functions

B r k m p r Y n n n T o s e t a e e s ' e i c o n s o u i f s e c \_ o u t i s c o m p a r e d Y u o c a s o e t u  
 p l m p r r n T n n a e e s ' d c e a t d a u s o u i f s e c \_ o u t i s c o m p a r e d ) u  
  
 h m h Y h r b h d c n p a n s e e s e l e h o u s d b e a t e t d a t a a s e e o e e d s o f i e t d a t a a  
 h n p n h n , y n p r w e a l e s s m i t l e m s i b e h d e f c o t i o e d u d t e s t e e o t s d s e t e t u  
 n h r l l r h b s e a s y p r t n a t o i c i a s e r d c t e e a t e t d a t a a c o o t i o a c u u  
  
 l h p r h T v y n e h v e a p t a t t \_ b \_ c o m p a r e d u e o t s i o a e i t o t f a c o t i  
 h l h n r : w a s a e f o o s i g i g e t u  
  
 i n t ( \* f u n c t i o n ) ( D b \* d b , c o n s t D b t \* k e y 1 , c o n s t D b t \* k e y 2 )  
  
 n m r n m n T r l l h n s f i c p t i s r e t u r n a l e i t g u e a l e s s . t h u q c a d e g g u a t t o a f e  
 r b r h m k h n h , n s c o i s e e r n d r e y e g e a t t e r e t e f c o t i s e t u t a e a t s a t e g e a t  
 h l r . h l n h , n n m w t o a f r e t o b , e a k a e t l e f u o t i s e t u t o y a d e i s i e t s e s s  
 n h n h n n m r n m t a e s e c d d e t e f c . o t i s v e t u t e a u g a t i u e a u  
  
 n h p r T e y k n o r t i s e t h i u \_ c o m p a r e d ) t u w , x y o s e i o y a t e s e a a  
 h p m r h r x r r D h e n c e n t t a t a e a s t o d c o d d a t a y s e i t o l e s  
  
 o l n m p l m h x o n e k a e r n e a a e b o e t i t s a t s e u d y o : t e i u t g e s i e t d a t a a i  
  
 i n t  
 c o m p a r e \_ i n t ( D b \* d b p , c o n s t D b t \* a , c o n s t D b t \* b )  
 {  
     i n t a i , b i ;  
  
     // R e t u r n s :  
     // < 0 i f a < b  
     // = 0 i f a = b  
     // > 0 i f a > b  
     m e m c p y ( & a i , a - > g e t \_ d a t a ( ) , s i z e o f ( i n t ) ) ;  
     m e m c p y ( & b i , b - > g e t \_ d a t a ( ) , s i z e o f ( i n t ) ) ;

```

    return (ai - bi);
}

```

N h h m br p n m m one t tpe p rd atas lf nli etco qu jdoite o tysat i o eiat æig ds  
 k B r l DB n nr n kn y nrh ne e a n oet ndy t gnaeth ,a u d l aig yoft et ed i g l at a i  
 mp rr m h n nr mp rr fo co Womlow hsi eb u itdog sa io esi e e eu t ad atses  
 r nm hn n r hr m de e ad n naesiofb fte ie r g r acit es a ea tte teid g eot ed  
 hr h r m n mp nw y fo cyio. co ed æeu dcb es eat

DB h mp rr nT n oc sea : o te stci sa if c o ti u

```

#include <db_cxx.h>
#include <string.h>

...

Db db(NULL, 0);

// Set up the btree comparison function for this database
db.set_bt_compare(compare_int);

// Database open call follows sometime after this.

```